

Introduction to Representations and Estimation in Geometry

Klas Nordberg

Computer Vision Laboratory
Department of Electrical Engineering
Linköping University

Version: 0.40– September 18, 2018

Contents

Preface	11
1 Background and Overview	15
1.1 Euclidean geometry	15
1.2 Perspective	15
1.3 Projective geometry	16
1.4 Photogrammetry	17
1.5 Computer vision	18
I Representations	19
2 Cartesian Representations	21
2.1 Points in \mathbb{E}^2	22
2.2 Lines in \mathbb{E}^2	22
2.2.1 Slope and Intercept	23
2.2.2 Hesse Normal Form	24
2.2.3 Parametric Representations of a Line	25
2.3 Points, planes and lines in \mathbb{E}^3	26
2.4 Basic geometric operations	27
2.4.1 The line that intersects two points in \mathbb{E}^2	27
2.4.2 The point of intersection between two lines in \mathbb{E}^2	28
2.4.3 Distance between a point and a line in \mathbb{E}^2	29
2.5 Before We Continue	30
3 Homogeneous Representations in 2D	31
3.1 Homogeneous coordinates of 2D points	31
3.1.1 P-normalization	32
3.2 Dual homogeneous coordinates of 2D lines	34
3.2.1 D-normalization	35
3.3 Relations between points and lines	36
3.3.1 The line that intersects two points	36
3.3.2 The point of intersection between two lines	37
3.3.3 Points on a line, or not	37
3.3.4 Lines through a point, or not	39
3.4 Operations on normalized homogeneous coordinates	39
3.4.1 Distance between two points	39
3.4.2 Distance between a point and a line	40
3.4.3 Area of a triangle	42
3.4.4 Concluding remarks	44
3.5 Points and lines at infinity	45
3.5.1 Points at infinity	45
3.5.2 The line at infinity	47

3.5.3	Projective geometry and the projective plane	48
3.6	Parametric representations of lines	49
3.7	Plücker coordinates	50
3.8	Duality Principle in the Projective Plane	52
4	Transformations in 2D	53
4.1	How to characterize a transformation	53
4.1.1	Degrees of freedom	53
4.2	Rigid transformations	54
4.3	Similarity transformations	58
4.4	Affine transformations	59
4.4.1	Non-uniform scaling	60
4.4.2	Reflections	60
4.4.3	Shearing	62
4.4.4	Decomposition of an affine transformation	63
4.5	Dual transformations	63
4.5.1	Geometric characterization of affine transformations	64
4.6	Projections	65
4.7	Concluding remarks	65
5	Homogeneous Representations in 3D	67
5.1	Homogeneous coordinates of 3D points	67
5.2	Dual homogeneous coordinates of 3D planes	68
5.2.1	Parametric representation of a plane	69
5.3	Homogeneous representations of 3D lines	70
5.3.1	Parametric representation of a 3D line	70
5.3.2	Plücker coordinates of 3D lines	71
5.3.3	L-normalization	72
5.3.4	Dual Plücker coordinates	73
5.3.5	The duality mapping	74
5.3.6	Internal constraint and degrees of freedom	74
5.3.7	DL-normalization	75
5.4	Incidence relations between points, planes, and lines	75
5.4.1	The intersection of a line with a plane	76
5.4.2	The plane that intersects a line and a point	76
5.4.3	The point that intersects three planes	77
5.4.4	The plane that intersects three points	78
5.4.5	Points on a plane, or not	79
5.4.6	Planes through a point, or not	80
5.4.7	The intersection of two lines	80
5.5	Distances	81
5.5.1	Distance between two points	81
5.5.2	Signed distance between a point and a plane	81
5.5.3	Distance between a point and a line	81
5.6	The duality principle in \mathbb{E}^3	82
6	Transformations in 3D	83
6.1	Degrees of freedom	83
6.2	Rigid transformations	83
6.3	Similarity transformations	84
6.4	Affine transformations	84
6.5	Dual transformations	86
6.6	Line transformations	86

7 Homographies	87
7.1 Homographies in 2D	87
7.1.1 Applications of homographies	90
7.1.2 Points at infinity revisited	90
7.2 Homographies in 3D	92
7.3 Canonical set of homogeneous coordinates	92
7.3.1 Canonical set homogeneous coordinates in 2D	92
7.3.2 Canonical set of homogeneous coordinates in 3D	93
7.3.3 Concluding remarks	93
8 The Pinhole Camera	95
8.1 Central projection	95
8.1.1 Camera obscura	97
8.2 Camera centered coordinate systems	98
8.2.1 The normalized camera	101
8.3 The general pinhole camera	105
8.3.1 Camera transformations	105
8.3.2 The camera at infinity	106
8.3.3 Internal and external camera parameters	107
8.3.4 Camera resectioning	110
8.4 The digital image	112
8.5 The geometry of a pinhole camera	115
8.5.1 Projection lines	115
8.5.2 The image of a line	116
8.5.3 Projection planes	117
8.5.4 Camera projection constraints	118
8.5.5 Geometric interpretation of \mathbf{C}	119
8.5.6 Field of view	119
9 Prelude to Two-View Geometry	121
9.1 Correspondences	121
9.1.1 Interest points	122
9.1.2 The correspondence problem	124
9.2 Planar homography	126
9.2.1 Simple derivation of a planar homography	126
9.2.2 The calibrated case	127
9.2.3 Solving for the plane and relative camera pose	127
9.2.4 Applications	130
9.3 Rotational homography	131
9.3.1 Derivation of a rotational homography	131
9.3.2 A constraint on rotational homographies	133
9.3.3 Solving for \mathbf{R} and \mathbf{K}	134
9.3.4 Before we continue	135
9.3.5 Applications	136
10 Epipolar Geometry	137
10.1 Stereo Cameras	139
10.1.1 Epipolar points	140
10.1.2 Epipolar line	141
10.1.3 Epipolar plane	142
10.2 The fundamental matrix and the epipolar constraint	143
10.2.1 The fundamental matrix	143
10.2.2 The epipolar constraint	144
10.2.3 Symmetry	144
10.2.4 The epipolar constraint in practice	146

10.2.5 Transfer	146
10.3 More properties of the fundamental matrix	147
10.3.1 Invariance to homography transformations of 3D space	147
10.3.2 Transformations of the image coordinates	148
10.3.3 Parameterization of \mathbf{F}	149
10.3.4 Camera matrices from \mathbf{F}	149
10.3.5 Factorization of \mathbf{F}	151
10.3.6 Summary of properties related to \mathbf{F}	152
10.4 Triangulation	153
10.4.1 Ambiguities of the world coordinate system	154
10.5 Calibrated epipolar geometry	155
10.5.1 Normalized stereo cameras	155
10.5.2 The essential matrix	156
10.5.3 Camera poses from \mathbf{E}	157
10.5.4 Transformations of the image coordinates	159
10.5.5 Minimal parameterization of \mathbf{E}	161
10.5.6 Summary of properties related to \mathbf{E}	161
11 Representations of 3D Rotations	163
11.1 Rotation matrices	164
11.1.1 Representations of $SO(3)$	165
11.2 Axis-angle representation	166
11.2.1 Rodrigues' rotation formula	167
11.2.2 Axis-angle from $SO(3)$	169
11.2.3 Summary	170
11.3 $so(3)$	170
11.3.1 Using the matrix exponential function	170
11.3.2 The Cayley transform	172
11.3.3 Summary	174
11.4 Unit quaternions	174
11.4.1 The quaternionic embedding of $SO(3)$	175
11.4.2 \mathbf{R} from unit quaternion	176
11.4.3 Unit quaternion from \mathbf{R}	177
11.4.4 Relation to the Cayley representation	178
11.4.5 Summary	178
11.5 Three-angle representations	179
11.5.1 Finding the Euler angles for \mathbf{R}	180
11.5.2 Summary	181
11.6 Which representation do I choose?	181
11.7 Related topics	183
11.7.1 Uniformly sampled random rotations	183
11.7.2 Twisted rotations	183
11.7.3 Singularities and gimbal lock	187
II Estimation	189
12 Introduction to Estimation in Geometry	191
12.1 Least squares formulation	191
12.2 Total least squares formulation	194
12.3 What causes the errors? (Part I)	196
12.3.1 Measurement errors	196
12.3.2 Model errors	197
12.3.3 Outliers and inliers	199
12.4 Geometric errors	200

12.4.1 Examples of geometric errors	201
12.4.2 Before we continue	204
12.5 Algebraic errors	204
12.5.1 Introductory example	205
12.5.2 Model parameters, data matrix and residual	205
12.5.3 Internal constraints	206
12.5.4 The size of \mathbf{A}	206
12.5.5 General solution strategy for the over-determined case	208
12.5.6 The inhomogeneous method	209
12.5.7 The homogeneous method	212
12.6 Probability based estimation	213
12.6.1 Back to points and a line	213
12.6.2 Maximum a posteriori estimation	216
12.6.3 Maximum likelihood estimation	216
13 Estimation of Transformations	219
13.1 Homography estimation	219
13.1.1 Geometric errors	220
13.1.2 Direct Linear Transformation (DLT)	220
13.1.3 The correspondence problem	223
13.1.4 Minimal case estimation	223
13.1.5 Degeneracies	224
13.2 The homogeneous method revisited	225
13.2.1 SVD profile	225
13.2.2 Data normalization	228
13.2.3 SVD profile, again	235
13.2.4 Closing remarks	237
13.3 Camera matrix estimation	239
14 Non-linear Estimation	241
14.1 Non-linear estimation techniques	241
14.1.1 Initial solutions	242
14.1.2 Parameterizations	242
14.2 Re-parameterization	245
14.2.1 An example	246
14.3 The residual	247
14.3.1 Re-mapping of the residual	247
14.3.2 Choosing residual	248
14.4 An example: estimation of a line	250
14.4.1 Using a homogeneous representation as parameters	250
14.4.2 Minimal parameterizations	252
14.5 An example: estimation of a homography	254
14.6 Practical issues	256
14.6.1 Computing the Jacobian	256
14.6.2 Sparse Jacobian	256
15 Estimation Involving Rotations	259
15.1 Estimation of 3D rotations	260
15.1.1 Using OPP	260
15.1.2 Using SOPP	261
15.1.3 Using quaternions	261
15.1.4 OK, so which algorithm do I choose?	263
15.2 Estimation of rigid transformations	263
15.2.1 Iterative Closest Point (ICP)	264
15.3 Non-linear least squares estimation involving rotations	266

15.3.1	Quaternions	266
15.3.2	Axis-angle representation	267
15.3.3	Euler angles	269
15.4	The Perspective n -Point Problem (PnP)	270
15.4.1	Algebraic estimation	270
15.4.2	Geometric minimization	271
15.4.3	Minimal case estimation (P3P)	272
16	Estimation for Two-View Geometry	275
16.1	Triangulation	275
16.1.1	The mid-point method	275
16.1.2	Algebraic methods	277
16.1.3	Optimal triangulation	277
16.1.4	Closing remarks	280
16.2	Estimation of \mathbf{F}	282
16.2.1	The 8-point algorithm	282
16.2.2	The 7-point algorithm	285
16.2.3	Degeneracies	286
16.2.4	Minimization of geometric errors	287
16.3	Estimation of \mathbf{E}	290
16.3.1	Algebraic estimation	290
16.3.2	Minimal case estimation of \mathbf{E}	290
16.4	Estimation of ω from a rotational homography	291
16.4.1	Algebraic estimation	292
17	Robust Estimation	295
17.1	What causes the errors? (Part II)	296
17.1.1	Inliers and outliers	296
17.2	Robust errors	297
17.3	RANSAC	298
17.3.1	An indeterministic approach	299
17.3.2	Robust estimation of a 2D line	300
17.3.3	Probabilities and number of trials	302
17.3.4	Variants of the basic RANSAC algorithm	302
17.4	Revisiting the correspondence problem	303
17.4.1	RANSAC and the correspondence problem	304
17.4.2	Preprocessing of data	305
17.4.3	Closing remarks	305
17.5	Robust estimation in practice	306
17.5.1	Robust estimation of \mathbf{H}	307
17.5.2	Robust estimation of \mathbf{F}	307
17.5.3	Robust estimation of \mathbf{E}	308
17.5.4	Robust PnP	310
III	Applications	313
18	Camera Calibration	315
18.1	Automatic camera calibration	316
18.1.1	Mathematical foundation	316
18.2	Lens distortion	317
18.2.1	Radial lens distortion	319
18.2.2	General distortion models	322
18.2.3	General coordinate systems	323
18.2.4	OK, so which distortion model do I choose?	323

18.2.5	Calibration of lens distortion	324
18.2.6	Compensating for the lens distortion	324
18.3	Zhang's calibration method	325
18.3.1	Geometric errors	328
18.3.2	Lens distortion	328
18.3.3	Summary	328
19	Image Mosaics	331
19.1	Introduction	331
19.2	Using homographies	333
19.2.1	A mosaic from two images	333
19.2.2	Blending	338
19.2.3	A mosaic from several images	340
19.3	Using spherical coordinates for panoramas	340
19.3.1	A panorama from two images	345
19.4	Further reading	346
20	Rectified Stereo	349
20.1	Rectified stereo cameras	349
20.1.1	Preliminary results	350
20.1.2	Fully rectified stereo	351
20.2	Rectified epipolar geometry	353
20.2.1	Fundamental matrix	354
20.2.2	Triangulation	354
20.3	Synthetic rectification	357
20.3.1	Preliminary results	358
20.3.2	Hartley's rectification method	359
21	Structure from Motion	365
21.1	Simplifying assumptions	365
21.2	Reconstruction from normalized cameras	367
21.2.1	Accumulative reconstruction from triplets of normalized cameras	368
21.2.2	Analysis	369
21.3	Bundle adjustment	369
21.4	Incremental bundle adjustment	370
21.4.1	Bookkeeping	371
21.4.2	Initialization and first bundle adjustment	372
21.4.3	Adding an image	373
21.5	Practical issues	378
21.5.1	Parameterization of the camera poses	378
21.5.2	Flexible scheme for adding cameras	381
21.5.3	Image coordinate system	381
21.5.4	The residual vector \mathbf{r}	381
21.5.5	Sparsity of \mathbf{J}	382
21.5.6	The Schur complement trick	382
21.5.7	Scale ambiguity	384
21.5.8	The first camera pose	384
21.6	Further reading	385
Bibliography		387
Index		391
List of Algorithms		396

Preface

This book contains material for an introductory course on homogeneous representations for geometry in 2 and 3 dimensions, camera projections, representations of 3D rotations, epipolar geometry, and estimation of various type of geometric objects. Based on these results, a set of applications are presented. It also contains a toolbox of general results that are useful for the presented material. The book is intended for undergraduate studies at advanced level in master programs, or in PhD-courses at introductory level.

Toolbox

The reader is assumed to be familiar with basic concepts in geometry, linear algebra and calculus. At various places in this presentation, there are references to the definitions of some of these basic concepts, more specifically to the compendium [54], here referred to as *Toolbox*. The readers who are unfamiliar with, or need to refresh, certain concept that appear in this presentation are encouraged to consult the first part of the *Toolbox* compendium. Additional mathematical theory that most readers may not be familiar with is presented in the second part of the *Toolbox* compendium.

Organization

The material is organized as follows:

- **Chapter 1** Gives a brief overview and contains a historic survey, which aims to providing a context in terms of mathematical and technical developments, as well as applications.

Part I presents a collection of algebraic representations that are used for various types of geometric objects. These objects are mainly from Euclidean geometry such as points, lines, or planes. Additional types of objects, in the form of geometric transformations and constraints, are also presented here.

- **Chapter 2** presents an overview of some basic problems in geometry. This chapter uses Euclidean geometry and Cartesian coordinates for formulate problems and solutions. In the subsequent chapters, we will see that these problems can be solved in a simpler and also more general way based on homogeneous representations.
- **Chapter 3** contains an introduction to homogeneous representations of the geometry in 2D space. The main geometric objects discussed here are points and lines, and their homogeneous representations.
- **Chapter 4** makes an overview of different types of transformations in 2D space. More precisely, it covers transformations that can be represented as linear mappings on the homogeneous representations introduced in Chapter 3, except for homographies that are presented in Chapter 7.
- **Chapter 5** presents homogeneous representation in 3D space, mainly by extending the homogeneous representations in Chapter 3. The main geometric objects discussed here are points and planes, but also lines, and their homogeneous representations.
- **Chapter 6** makes an overview of various classes of transformations on 3D space, mainly by extending the transformations in Chapter 4 to the 3D case.
- **Chapter 7** closes the presentation of transformations in 2D and 3D space, by defining the most general class of transformations that can be represented as linear transformations on the homogeneous representations: homographies.

- **Chapter 8** presents the pinhole camera model, mapping the 3D space to an image. A homogeneous representation of this mapping is also defined, in terms of the camera matrix. The latter is given a more practical form as a normalized camera in combinations with internal camera parameters.
- **Chapter 9** introduces concepts that are important when two images of the same scene are analyzed, e.g., corresponding points. Two special cases of two-view geometry are also presented: planar homographies and rotational homographies.
- **Chapter 10** describes general relations that occur when two cameras are observing the same scene, or epipolar geometry. One basic issue is the correspondence problem: how can we know if image points in two images correspond to the same 3D point? This leads to the epipolar constraint defined by the fundamental matrix or, in the calibrated case, the essential matrix. A related problem is triangulation: given a pair of corresponding image points, where is the 3D point?
- **Chapter 11** contains an overview of various types of algebraic representations for rotations in 3D space. These are important whenever a rotation appears in the mathematical formulation of some geometric object, for example when it is estimated.

Part II addresses the problem of how to estimate various types of geometric object from measurements. It covers both linear and non-linear methods, and discusses different ways to define the errors that are minimized by these methods.

- **Chapter 12** contains a first introduction to estimation of various types of geometric objects from observed data. Here, we consider estimation problems that lead to linear solution methods, such as the homogeneous methods and the inhomogeneous method, and also define the concepts of algebraic and geometric errors.
- **Chapter 13** continues the presentation on estimation started in Chapter 12, by considering the estimation of transformations, for example homographies and cameras. An important tool to deal with such problems is direct linear transformation. A second issue discussed here is data normalization, which can have a significant effect of the resulting estimate.
- **Chapter 14** takes the discussion on estimation one step further, by considering non-linear estimation problems. In this case, iterative methods are used to refine an initial solution of the estimation problem. Various aspects of this topic are discussed in the context of estimation in geometry.
- **Chapter 15** considers a particular class estimation problems, where a 3D rotation is involved among the free parameters to be estimated. Several estimation problems of this type are presented, often with solution methods that are dedicated to the special case that the free parameter is an $SO(3)$ rotation.
- **Chapter 16** considers various estimation problem in relation to epipolar geometry, in particular how to do triangulation, and how to estimate the fundamental matrix or the essential matrix.
- **Chapter 17** discusses robust estimation: estimation of geometric objects from datasets that contains a significant amount of outliers. The main result is the RANSAC algorithms, which can be applied to a range of estimation problems. In particular, it is used to find correspondences in stereo images.

Part III combines representations and estimation techniques and discusses a set of applications that use this combination for solving practical problems.

- **Chapter 18** makes an introduction to the specific problem of estimation the parameters of a camera, in particular the internal parameters and the lens distortion. This is called camera calibration.
- **Chapter 19** presents methods for building a larger image out of a set of images, a so-called mosaic. The original images can either be from a camera that looks in different directions from a single point, creating a panorama image, or from a camera that looks at a planar surface from different viewpoints.
- **Chapter 20** investigates rectified stereo, a special case of epipolar geometry that occurs when the optical axes of the stereo cameras are parallel and perpendicular to the baseline. In this case, all epipolar lines, in both images, are parallel. This implies a significant simplification to problems such as finding corresponding image points, and to determine where the corresponding 3D point is located.

- **Chapter 21** combines several of the last chapters into a method for reconstructing a 3D scene from observed points in multiple views. This includes an extensive use of epipolar geometry and estimation techniques.

Acknowledgments

Several people have been involved in the discussion, organization, and proof reading of the material in this book. In particular, I would like to thank Martin Danelljan, Michael Felsberg, Per-Erik Forssén, Johan Hedborg, Fahad Khan, Jan-Åke Larsson, Rudolf Mester, Mikael Persson, and Andreas Robinson for participating in this process.

Alternative sources

Much of the material that is covered in this presentation can also be found in the following publications, which are devoted to geometry for computer vision:

- *Multiple View Geometry for Computer Vision* by Hartley & Zisserman (2nd ed. 2004) [30]. This book is the standard reference in this area, and contains lots of material beyond what is presented here.
- *Epipolar Geometry in Stereo, Motion and Object Recognition* by Xu & Zhang (1996) [65].
- *Three-dimensional computer vision : a geometric viewpoint* by Faugeras (1993) [18].

In addition, there are some publications that cover geometry but also other aspects of computer vision. Here is a short list of some more recent books of this type:

- *Computer Vision for Visual Effects* by Radke (2013). [56].
- *Computer Vision : Models, Learning and Inference* by Prince (2012) [55].
- *Computer Vision : Algorithms and Applications* by Szeliski (2011) [62].
- *Image Processing, Analysis, and Machine Vision* by Sonka, Hlavac & Boyle (3rd ed. 2008) [60].

Various representations of 3D rotations are discussed in the book by Hartley & Zisserman, but also in

- *Robotics, Vision and Control* by Corke (2011) [14].

Observations

Along the way a number of observations are made and are presented in boxes, like this one. They are often general in nature and do not apply only to the particular example that is used to illustrate the observation.

Figures and images

All figures and images are produced by the author, with the following exceptions:

- Figure 8.2 on page 98, by User:Pbroks13 on Wikimedia Commons.

Errata

Errata for this book is published on the following web address:

<http://www.cvl.isy.liu.se/research/publications/IREG>

Chapter 1

Background and Overview

This chapter presents some historical background to the results that will be introduced in later chapters.

1.1 Euclidean geometry



Euclid was a Greek mathematician who lived around 300 BC and is famous, among other things, for his seminal work the *Elements*. It is a treatment of geometry in two and three dimensions, based on a set of axioms from which he could deduce a large set of theorems. Euclid did not invent geometry: other mathematicians had already introduced many results presented in his book. His contribution lies mainly in the mathematical rigor he used to derive these results. Euclid was also able to extend the application of these results beyond what was known before. His work in geometry was later expanded by other mathematicians of Antiquity, until circa 350 AD. In terms of new results in geometry, not much happened for about 1300 years after that.

In this presentation, we will use *Euclidean geometry* as a general label for the geometry that was developed during Antiquity. Although this includes Euclid's own work, it is a misnomer since it also refers to contributions from other mathematicians, both before and after Euclid. In fact, parts of what we here call Euclidean geometry cannot even be attributed with a Greek origin. Some results were developed first, or in parallel, in Egypt, Babylonia, India, or in China. That aside, and while the literature offers no reasonable alternative label, Euclid's work has had a profound impact even on today's view of geometry and mathematics.

Euclidean geometry includes the two-dimensional plane, with points and lines, circles, ellipses, and an assortment of curves such as parabolas, hyperbolas, and spirals. Typical properties related to these geometric objects are lengths or distances, angles, and areas. In Euclidean geometry we also study operations between different objects, such as the intersections of lines or curves, and incidence relations between points, or point-sets, and lines, or more general curves. Euclidean geometry also extends to 3D space, where the geometric objects are points, lines, and planes, but also various curves, surfaces, and solids. In 3D space, the properties related to such objects also include volumes and solid angles. Trigonometry can be seen as a special field of geometry dealing with angles, and as such becomes a part of Euclidean geometry.

1.2 Perspective

The idea of perspective in art has been around for a long time. For example, Euclid tried to explain how our eyes see the 3D world in his book *Optics*, and he was not first to do so. So ever since Antiquity, mathematicians have tried to formulate principles of how perspective works, although these ideas have been tentative and sometimes even incorrect. In parallel, perspective seems to have been known among many painters, who sometimes even used perspective effects in their work. But this was not done in a systematic way and, when it happened, it was often not based on correct principles. Sometimes it did not go further than depicting an object that is closer to the viewer with larger relative size than the same object when it is further away. For a long time, art was instead governed by traditions and of reproducing religious motifs and symbols.

All this changed in the early 1400s, when linear perspective came into fashion in European art. Filippo Brunelleschi, an Italian multi-talent, for example in architecture, art, engineering and mathematics, is often given

credit for introducing linear perspective in art. He was certainly not first to use perspective, but he was able to make a convincing argument why the principles of linear perspective give realistic images. In (about) 1413 at the Dome of Florence, he set up an experiment where a viewer could compare a panel, on which was painted the Baptistry next to the Dome, with a real view of the same building. The panel was facing the Baptistry, and by looking through a hole in the panel, it was possible to observe the building from behind the panel. Through this hole it was also possible to observe the painting on the panel, using a mirror in front of it. The viewer could compare the two views and see the striking resemblance between the perspective based painting and the real Baptistry, by moving the mirror in and out of view.

Brunelleschi's experiment appears to have sparked a revolution. It was not long before many of Florence's famous artists began to use linear perspective in their work. In 1435 Leon Battista Alberti published *De pictura*, the first book that formulated the proper mathematical principles of linear perspective in art. From then on, and for a long time, linear perspective had a central role in European art. It was not until more than 400 years later that the impressionist painters of the late 1800s gave perspective a less prominent position and some decades later the cubists rid themselves completely of perspective.

To summarize, the artists have had a pretty good idea about how the 3D world should be realistically depicted, from the early 1400s and on. Yet, a formal study of the mathematical consequences of these results was not made until some 200 years later.

1.3 Projective geometry

As a precursor to projective geometry, the astronomer Johannes Kepler, who also worked on mathematics and geometry, had described points at infinite distance from any ordinary point. He was studying ellipses, and what happens when one of the two focal points is fixed and the other one is moved further and further away from the first. Kepler noticed that when the second focal point is placed at infinite distance from the first one, the ellipse turns into a parabola. He was the first mathematician to describe parabolas as special cases of ellipses, where one of the two focal points lies at infinity [19]. For Kepler, an accomplished astronomer, points at infinity were not an abstraction. In his practical work he observed light rays from distant stars, and must have noticed that they appear as parallel lines. The idea of a point at infinity (a star) acting as the intersection of these lines is then close at hand.

One of the first who studied the mathematics behind linear perspective was Girard Desargues, a French mathematician, engineer and architect. This led him to formulate some of the earliest results of what we today know as *projective geometry*, a geometry based only on points, lines, and (in 3D) planes. Projective geometry can also be seen as the study of properties in an image which remain unaltered (invariant) in projections.

In projective geometry there are no angles, distances, areas, or volumes. There is not even a concept of an object lying in front of, behind, or between something else. The focus is instead on relations like incidence, and operations like intersections and projections. A main result is the formulation of the *projective plane*: the usual Euclidean plane extended by Kepler's *ideal points*, or *points at infinity*. This idea allows us to use various results that are known from Euclidean geometry, but without having to make specific assumptions, e.g., about the configuration of intersecting lines.

When we are using projective geometry, it is safe to say that two distinct lines *always* intersect at a unique point, and this is true even if the lines are parallel. In this case, the intersection is simply a point at infinity. In the projective plane, there is no distinction between the usual points and those lying at infinity. Any statement that is true for Euclidean points, or any operation that can be applied to them, is true or can be applied also to points at infinity.

Desargues formulated his work in the book *Brouillon projet*, published in 1639. Some of the most famous mathematicians at the time, such as Pierre de Fermat and Blaise Pascal, expressed their appreciation of this work. However, it did not have a significant impact on the mathematical community as a whole, and there are many theories as to why. Some claim that Desargues' book was difficult to read, others that he focused on formulating known results using projective geometry rather than providing new practical methods that were useful for solving problems. [12]

There is another popular theory about why Desargues' projective geometry had little impact at his own time. Only a few years earlier, in 1637, René Descartes had published *La Géométrie*. This book presented what we now call *analytic geometry*, which includes the concept of a coordinate system, or a reference frame, and the idea that points can be represented in terms of coordinates.

Analytic geometry enabled mathematicians to reformulate the well-known results from Euclidean geometry, now using coordinates. It became possible to connect geometry and algebra in a much more explicit way than had been possible before. This invention in geometry had a profound impact on mathematics. It also laid a mathematical foundation for physics as we know it today. So, while the mathematicians were busy extending Descartes' work on analytic geometry and, eventually, inventing calculus, Desargues' projective geometry fell into oblivion for some centuries.

At the end of the 1700s geometry had been extended to include also differential geometry. Curves or surfaces are here defined based on functions. Length and area are now determined by integration, and things like curvature and normals are defined using derivatives. A century later, algebraic geometry had been established, where curves and surfaces instead are seen as the solutions of (typically) polynomial equations. In the development of both differential geometry and algebraic geometry, mathematicians had rediscovered many of the original ideas that Desargues and others had formulated some centuries earlier in projective geometry. This topic was now established as an independent domain in geometry.

Two novel ideas were presented in this later stage of projective geometry: *homogeneous coordinates* and *projective spaces*. These steps, finally, made an explicit connection between geometry and linear algebra. Various geometric objects, such as points, lines, and planes, are now represented by vectors or matrices. But they do not appear as proper vectors or matrices: they are elements of projective spaces, i.e., equivalence classes of vectors. They are said to be equivalent when they differ only by a scalar multiplier. As we will see, this idea applies also to a large class of transformations. There are also new types of geometric objects in the form of geometric constraints that can appear in homogeneous form.

1.4 Photogrammetry

With the invention of photography in the first half of the 1800s, projective geometry combined with the disciplines of optics and land survey led to the field of *photogrammetry*. The objective is here to make measurements of geometric objects in the 3D world, often of points, but it could also be of lines, or the position and orientation of an object. In photogrammetry, all these measurements are based on photographs, 2D projections of 3D space.

For example, assuming that we know certain physical parameters of the camera, photogrammetry uses the parallax, the displacement of a point in two stereo photographs, to determine the location of a point in the 3D world using a technique called *triangulation*. Another example is to determine the camera pose; the position and orientation of the camera that has produced an image. One of the main applications for photogrammetry is to produce topographical maps based on aerial images. In the early days, kites or balloons carried the cameras. Later they were brought to the skies by airplanes and satellites. [10]

A significant part of photogrammetry is about engineering. For example, the construction of mechanical and optical devices that produce accurate projections, or special types of projections of a 3D scene, such as the ground below an airborne camera. Other devices can take a pair of aerial stereo images and trace elevation contours to generate topographical maps.

Photogrammetry also led to new results and methods in mathematics. Photogrammeters broke new land both in geometry and in methods for estimating geometric quantities. For example, the basics of epipolar geometry, relating corresponding features in two images, was formulated already in 1883 by Gudio Hauck [34]. Other problems that were studied included triangulation of 3D points from stereo images, and a range of problems that allowed them to determine the position or orientation of points, lines, and of cameras. By formulating them as estimation problems, it became possible to take into account measurement inaccuracies. The interesting quantity is then found by minimizing some type of error, e.g., using least squares methods.

Computers became available to photogrammetrists in the 1950s. The numerical problems related to estimation in geometry could now have much more complex formulations and still be solved in practice. For example: given a set of images from cameras at multiple positions, which include some 3D points that are visible in all images, how can we determine both the positions of the 3D points and the poses of the cameras? Duane Brown formulated this so-called *bundle adjustment* problem already in 1958 [7], together with a method for its solution. Due to the limited capacity of the early computers, bundle adjustment could initially only be applied to small problems, or to problems for which computation time is not critical.

1.5 Computer vision

The first images were digitized in 1957, and it now became possible to write computer programs that process digital images. This lead to the areas which we know today as digital image processing, machine vision, and computer vision. In these relatively intertwined fields, the goal is often to extract information from digital images. For example, we want to detect if a specific object is present or not in an image, or determine the position or the motion of objects in the image. In other applications, images are processed or transformed, e.g., to reduce the impact of image distortion in the form of noise or blur.

Initially, the processing of digital images did not take geometry explicitly into account, other than in the form of simple 2D transformations that can be defined in the image plane. Although the image often is a projection of the 3D world, this observation was for a long time usually ignored or simplified. Partly, this was because other and more fundamental problems required attention. For example, detection and characterization of low level features, such as lines, edges, and corners, and analysis of motion in an image, were hot research topics in computer vision till the end of the 1980s.

Another reason for its initial lack of geometric reasoning is that computer vision developed more or less isolated from the relatively mature area of photogrammetry. This field already had a toolbox for geometric analysis of images, and it was familiar to complex computations for solving problems. But it also had, and still has, a relatively limited application range, mainly to produce topographical maps from aerial images.

Computer vision, instead, has had a much wider range of applications, for example in medicine and in manufacturing. It has always been aiming at systems that can handle dynamic situations, and produce a real-time response when whatever is observed changes its state. As a consequence, it took a while for computer vision researchers to become interested in geometry and the mathematics that describes the projection of a 3D space into 2D images. The two fields, initially, simply had few areas of common ground.

Eventually, computer vision began treating geometry more seriously. But the lack of interaction with photogrammetry led to reinvention of methods and to introduction of parallel terminology, sometimes making the interaction even less effective [32]. On top of this, a large body of the original results in photogrammetry is published in German, making it less accessible to many researchers in computer vision. A historical survey of geometric computer vision and its connection to photogrammetry is presented in [61].

From the late 1980s and during a period of about 15 years, computer vision devoted much attention to the geometric relations that occur in two, or more, images of the same scene. Techniques for dealing with epipolar geometry, camera calibration, or pose and motion estimation were established within computer vision. As has been mentioned, some of these results were then already known for a long time in photogrammetry. The so-called structure from motion problem (SfM), in which 3D structure is derived from multiple views, became a main attraction to researchers. Partly this was because it includes a large range of sub-problems which each can be individually honed to improve the overall performance. Partly, it was because the basic methods of bundle adjustment, conceived by Brown some 40 years earlier, now could run in close to real-time on modern computers.

This geometric interest within the computer vision community has led to a range of practical applications. For example, systems for combining real and computer animated images in the film industry [56], and systems for combining aerial photos, ground photos, maps, and annotations into complex map applications such as Google Maps [50].

Part I

Representations

Chapter 2

Cartesian Representations

One of the main topics of this presentation is geometry, and we start with a quick recapitulation of the most basic objects in Euclidean geometry. These are points and lines in both 2D and 3D, as well as planes in 3D. We will also consider operations defined on these objects, such as finding intersections or determine distances. The results form a foundation for the rest of the presentation in the following chapters.

In this discussion we make use of the Euclidean spaces \mathbb{E}^2 and \mathbb{E}^3 , and their connection to the vector spaces¹ \mathbb{R}^2 or \mathbb{R}^3 , respectively. When necessary, we assume to have defined a Cartesian coordinate system for each of the Euclidean spaces. These allow us to represent a point in \mathbb{E}^2 or \mathbb{E}^3 as a vector in \mathbb{R}^2 or \mathbb{R}^3 . The elements of the vector then hold the Cartesian coordinates of the point relative to the coordinate system.

By a *Cartesian coordinate system*, we mean that the two axes use the same length unit and that they are perpendicular. These assumptions are not strict requirements of a general coordinate system, but they simplify many calculations. For example, the expressions for length, distance, and scalar product in Section 2.1 become simpler in this case. Another aspect of a coordinate system is the so-called handedness of the axes. The handedness² makes concepts such as clockwise and counter-clockwise rotations well-defined. For most of the derivations made in the initial chapters the handedness is not made explicit, since it does not affect the results.

What coordinate system we use is often not very important, and it is simply assumed that one exists, and that it can define coordinates. In these cases, the coordinate system is implicit, and may not appear in the illustrations. We will also see that in some cases there may be a particular coordinate system that makes it easier to derive certain results. In some situations, there may even be two or more coordinate systems involved, which means that every point in \mathbb{E}^2 then has a set of coordinates, one for each coordinate systems. See Figure 2.1 for an illustration.

¹Euclidean spaces and their relation to \mathbb{R}^n are described in Toolbox Section 3.6.

²Handedness is defined in Toolbox Section 3.7.1.

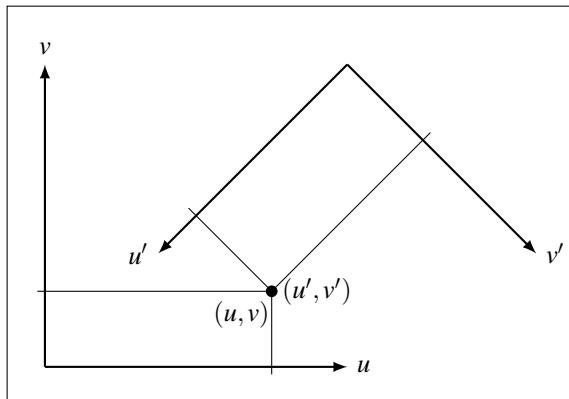


Figure 2.1: Two Cartesian coordinate systems, and a point in \mathbb{E}^2 . The point has coordinates (u, v) relative the left coordinate system, and coordinates (u', v') relative the right coordinate system.

2.1 Points in \mathbb{E}^2

Given a coordinate system³, any point in \mathbb{E}^2 can conveniently be represented as a vector $\bar{\mathbf{y}} \in \mathbb{R}^2$,

$$\bar{\mathbf{y}} = \begin{pmatrix} u \\ v \end{pmatrix}, \quad (2.1)$$

where u and v are the coordinates of the point relative to a Cartesian coordinate system. The scalar product between two vectors $\bar{\mathbf{y}}_1 = (u_1, v_1)$ and $\bar{\mathbf{y}}_2 = (u_2, v_2)$ in \mathbb{R}^2 is given by

$$\bar{\mathbf{y}}_1 \cdot \bar{\mathbf{y}}_2 = \bar{\mathbf{y}}_1^\top \bar{\mathbf{y}}_2 = u_1 u_2 + v_1 v_2. \quad (2.2)$$

Using the scalar product we can introduce a norm, or length, of vectors in \mathbb{R}^2 . For example, for $\bar{\mathbf{y}}$ in Equation (2.1) we define

$$\|\bar{\mathbf{y}}\| = (\bar{\mathbf{y}} \cdot \bar{\mathbf{y}})^{\frac{1}{2}} = \sqrt{u^2 + v^2}, \quad (2.3)$$

and define the distance between the two points $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$ as

$$d(\bar{\mathbf{y}}_1, \bar{\mathbf{y}}_2) = \|\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2\| = \sqrt{(u_1 - u_2)^2 + (v_1 - v_2)^2}. \quad (2.4)$$

The scalar product also allows us to define the concept of orthogonality: two vectors $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$ are orthogonal if and only if $\bar{\mathbf{y}}_1 \cdot \bar{\mathbf{y}}_2 = 0$.

About notation

The vector $\bar{\mathbf{y}}$ in Equation (2.1) is a *Cartesian representation*. Later chapters introduce an alternative *homogeneous representation*, not only of points, but of other geometric objects as well. The homogeneous representation is one of the main features of Part I, and it is used quite a lot throughout this book, so it should have a simple notation. To distinguish the two types of representations, we use the bar to denote the “standard” Cartesian coordinate representation of points. Vectors without the bar instead refer to homogeneous coordinates. The bar notation is also applied to functions of points in the Euclidean representation, e.g., the distance function in Equation (2.4). Normalized vectors in \mathbb{R}^n , with unit norm, will often have a hat instead of a bar, e.g., $\hat{\mathbf{l}}$ or $\hat{\mathbf{p}}$.

2.2 Lines in \mathbb{E}^2

Algebraically, we may think of a line in \mathbb{E}^2 as a set of points $\bar{\mathbf{y}} = (u, v)$ that satisfy the *equation of the line*:

$$u l_1 + v l_2 = \Delta, \quad (2.5)$$

for some real numbers l_1, l_2 , and Δ that characterize the line. A set of points that all lie on a common line are *co-linear*⁴. We can multiply both sides of Equation (2.5) by any non-zero number and the resulting equation is still satisfied for the same (u, v) as in Equation (2.5). This means that the parameters l_1, l_2 , and Δ do not form a unique representation of a specific line.

Although Equation (2.5) provides an excellent algebraic representation of a line, it gives only a few clues to the geometric interpretation of exactly which line it is. For example, the vector (l_1, l_2) is a normal to the line, but there also are many lines with the same normal. To make a geometric interpretation more explicit, there are a few options and we will discuss two of the more common approaches. After this, a few alternative representations of a line that use a parametric approach are presented.

³In the literature, *reference frame* is sometimes used instead of coordinate system. This term is occasionally used also in this presentation.

⁴The term co-linear includes the case when all points are identical, and there is no unique line that includes all points.

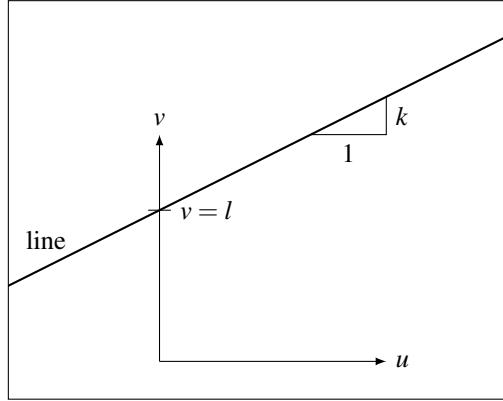


Figure 2.2: A line in \mathbb{E}^2 represented by its slope k and intercept l relative to a coordinate system.

2.2.1 Slope and Intercept

In general, we can set normalize the three parameters in Equation (2.5) such that one of l_1 or l_2 is equal to one. For example, setting $l_2 = 1$ gives

$$v = -l_1 u + \Delta. \quad (2.6)$$

This is a common way to specify a line: as a functional expression of how one of the two coordinates depends on the other for any point on the line. In Equation (2.6), u is a free variable and v depends on u . This dependency between the two coordinates can be formulated more compactly as

$$v(u) = k u + l. \quad (2.7)$$

In this form, the parameters (k, l) have a direct geometric interpretation.

The parameter k specifies how much the v coordinate increases for a point on the line, when its coordinate u increase by one unit. This is the same as the derivative of the function $v(u)$. If the coordinate system is defined such that u points right in the horizontal direction, and v points up, we can also describe k as the *slope* of the line, or its steepness. In the following presentation, we will refer to k as the slope of the line, even though this term is appropriate only when the directions of the coordinate axes follow the specification made above.

The parameter l appears in Equation (2.7) as $l = v(0)$. This means that l is the vertical coordinate of the point where the line intersects the (vertical) v -axis. This coordinate is commonly referred to as the *intercept* of the line. The parameters (k, l) in relation to a line are illustrated in Figure 2.2.

Degenerate Case

Although the representation of a line in \mathbb{E}^2 relative to a coordinate system by its slope and intercept is quite common, there is a problem. Equation (2.7) cannot represent a line which algebraic expression is of the type $u = \Delta$. In short, such a line does not have a well-defined slope and intercept. Obviously, it helps if we instead set $l_1 = 1$ in this case and express u as a function of v . But this also change the geometric interpretation of what we mean by slope and intercept, and we do not want to have different interpretations for different cases.

It may be argued that this issue is not very common, in practice we may never even observe lines that are exactly vertical. But even if this is true, the problem appears already for lines that are approximately vertical. If we try to determine (or estimate) the slope of an approximately vertical line, it should have a very large magnitude. But, apart from being very large, its numerical value can be almost random. In fact, also its sign can be random. This observation is true also for the intercept.⁵

In the general case, this type of degeneracy in the representation of a line can lead to various problems in the subsequent numerical calculations, problems that we want to avoid. Therefore, we will instead use the more general Hesse normal form, described in the next section, and employ the slope-intercept representation only for simple examples.

⁵By random, we mean here that the numerical value cannot be predicted by an intuitive analysis of the data from which the line is determined.

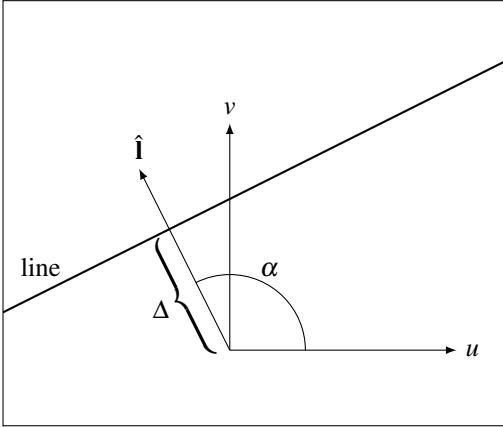


Figure 2.3: A line in \mathbb{E}^2 and its Hesse parameters.

2.2.2 Hesse Normal Form

If we return to Equation (2.5), we can make the observation that in order to form a representation of a line, it must be the case that l_1 and l_2 are not both equal to zero. It is then always possible to apply a normalization that leads to $l_1^2 + l_2^2 = 1$, and we can set

$$\begin{pmatrix} l_1 \\ l_2 \end{pmatrix} = \begin{pmatrix} \cos \alpha \\ \sin \alpha \end{pmatrix} = \hat{\mathbf{l}}, \quad (2.8)$$

for some angle α or normalized vector $\hat{\mathbf{l}} \in \mathbb{R}^2$. With this normalization, Equation (2.5) becomes:

$$u \cos \alpha + v \sin \alpha = \Delta, \quad \text{or} \quad \bar{\mathbf{y}} \cdot \hat{\mathbf{l}} = \Delta. \quad (2.9)$$

This normalization does not make $\hat{\mathbf{l}}$ and Δ unique. We can change the sign of $\hat{\mathbf{l}}$, corresponding to adding (or subtracting) π to α , and at the same time also change the sign of Δ , and they will still represent the same line. To reduce the ambiguity in $\hat{\mathbf{l}}$ and Δ , we can choose to always use $\Delta > 0$ whenever possible, which then makes $\hat{\mathbf{l}}$ unique. When $\Delta = 0$ there are still two possible choices for $\hat{\mathbf{l}}$.

The equation of the line presented in Equation (2.9), where $\|\hat{\mathbf{l}}\| = 1$ and $\Delta \geq 0$, is called⁶ the *Hesse normal form*, and $(\hat{\mathbf{l}}, \Delta)$ are the *Hesse parameters* of the line. In contrast to the slope and intercept parameters in Section 2.2.1, the Hesse parameters do not have any degenerate cases. As will see shortly, they also extend in a straight-forward way to the representation of a plane in \mathbb{E}^3 . Therefore, the slope-intercept parameters (k, l) will be used only occasionally, and we will instead use $(\hat{\mathbf{l}}, \Delta)$ as the preferred parameters for lines in \mathbb{E}^2 .

The Hesse parameters have intuitive geometric interpretations. The vector $\hat{\mathbf{l}}$ is a normal to the line. When $\Delta > 0$, $\hat{\mathbf{l}}$ points from the origin towards the line. The real number Δ is the distance from the origin to the line, measured in the direction of $\hat{\mathbf{l}}$, i.e., perpendicular to the line. When $\Delta = 0$, the line passes through the origin and $\hat{\mathbf{l}}$ can point in either of two directions. Figure 2.3 illustrates the Hesse parameters of a line in \mathbb{E}^2 .

Definition 2.1: Hesse parameters of a line in \mathbb{E}^2

Any line in \mathbb{E}^2 have a set of Hesse parameters, $(\hat{\mathbf{l}}, \Delta)$, which in general are unique. They refer to a particular coordinate system, where $\hat{\mathbf{l}} \in \mathbb{R}^2$ and $\|\hat{\mathbf{l}}\| = 1$. It is a normal vector that points from the origin in the perpendicular direction to the line. Δ is the distance from the origin to the line, in the direction of $\hat{\mathbf{l}}$. When $\Delta = 0$, the line passes through the origin and $\hat{\mathbf{l}}$ can point in either of two directions. In all other case is $\hat{\mathbf{l}}$ unique.

⁶After the German mathematician Otto Hesse, 1811–1874.

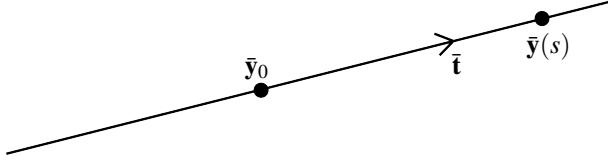


Figure 2.4: A line in \mathbb{E}^2 represented in parametric form, in accordance with Equation (2.10). \bar{y}_0 is a point on the line, \bar{t} is a tangent vector of the line, and $\bar{y}(s)$ is a parameterization of any point on the line.

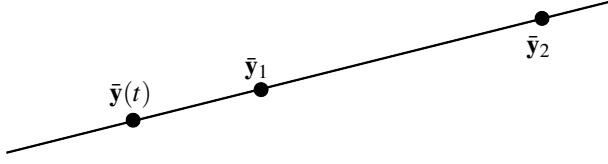


Figure 2.5: A line in \mathbb{E}^2 represented in parametric form, in accordance with Equation (2.13). \bar{y}_1 and \bar{y}_2 are two distinct points on the line, and $\bar{y}(t)$ is a parameterization of any point on the line.

2.2.3 Parametric Representations of a Line

The equation of a line, Equation (2.5), is an implicit representation of points on a specific line; it specifies a constraint that must be satisfied by any point on the line. An alternative is to explicitly describe the set of points that solve the equation, i.e., which satisfy the constraint. For example, as

$$\bar{y}(s) = \begin{pmatrix} u \\ v \end{pmatrix} = s \bar{t} + \bar{y}_0, \quad (2.10)$$

where \bar{t} is a tangent vector of the line, i.e., $\bar{t} \cdot \hat{1} = 0$, and \bar{y}_0 is the Cartesian coordinates of some point on the line. For each unique $s \in \mathbb{R}$, Equation (2.10) gives a unique point along the line. See Figure 2.4 for an illustration of these parameters.

Since Equation (2.10) involves not only parameters that specify the line, here \bar{t} and \bar{y}_0 , but also an additional free parameter, s , this specification of the line is called a *parametric representation*. It is useful, e.g., when we want to explicitly describe an arbitrary point on the line. But is it also more ambiguous than the Hesse parameters or using the slope and intercept.

To make the tangent vector \bar{t} less ambiguous, we can, for example, normalize it as $\hat{t} = \pm(-l_2, l_1)$, but even then there is ambiguity in the sign of \hat{t} . Also the point \bar{y}_0 is ambiguous, as it can lie anywhere on the line. We can solve this issue, for example, by choosing \bar{y}_0 as the point on the line closest to the origin. This point is given by $(\Delta \cdot l_1, \Delta \cdot l_2)$, and inserted in Equation (2.11) it gives

$$\bar{y}(s) = \begin{pmatrix} u \\ v \end{pmatrix} = s \begin{pmatrix} l_2 \\ -l_1 \end{pmatrix} + \Delta \begin{pmatrix} l_1 \\ l_2 \end{pmatrix}. \quad (2.11)$$

This last expression can be reformulated in a more compact form as

$$\bar{y}(s) = \begin{pmatrix} l_2 & \Delta \cdot l_1 \\ -l_1 & \Delta \cdot l_2 \end{pmatrix} \begin{pmatrix} s \\ 1 \end{pmatrix}. \quad (2.12)$$

Alternative Parametric Form

As an alternative to the parametric form described above, based on a point on the line and a tangent vector, we can instead use a parametric form based on two distinct points on the line, \bar{y}_1 and \bar{y}_2 . Any point \bar{y} on the line must then satisfy

$$\bar{y} = \bar{y}(t) = t \bar{y}_1 + (1 - t) \bar{y}_2, \quad (2.13)$$

for a unique $t \in \mathbb{R}$. Notice that $t \in [0, 1]$ gives a point on the line segment between $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$. See Figure 2.5 for an illustration.

2.3 Points, planes and lines in \mathbb{E}^3

Now, when notations and representations for point and lines in \mathbb{E}^2 are established, it is easy to extend these to \mathbb{E}^3 .

Points

A point in this space is represented as the vector

$$\bar{\mathbf{x}} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad (2.14)$$

where (x_1, x_2, x_3) are the coordinates of the point relative to a Cartesian coordinate system. The scalar product between two vectors $\bar{\mathbf{x}}_1 = (x_{11}, x_{21}, x_{31})$ and $\bar{\mathbf{x}}_2 = (x_{12}, x_{22}, x_{32})$ in \mathbb{R}^3 is given as

$$\bar{\mathbf{x}}_1 \cdot \bar{\mathbf{x}}_2 = \bar{\mathbf{x}}_1^\top \bar{\mathbf{x}}_2 = x_{11}x_{12} + x_{21}x_{22} + x_{31}x_{32}. \quad (2.15)$$

Using the scalar product we can introduce a norm, or length, of vectors in \mathbb{R}^3 :

$$\|\bar{\mathbf{x}}\| = (\bar{\mathbf{x}} \cdot \bar{\mathbf{x}})^{\frac{1}{2}} = \sqrt{x_1^2 + x_2^2 + x_3^2}, \quad (2.16)$$

and define the distance between two points $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$ as

$$d(\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2) = \|\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2\| = \sqrt{(x_{11} - x_{12})^2 + (x_{21} - x_{22})^2 + (x_{31} - x_{32})^2}. \quad (2.17)$$

The scalar product also allows us to define the concept of orthogonality: two vectors $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$ are orthogonal if and only if $\bar{\mathbf{x}}_1 \cdot \bar{\mathbf{x}}_2 = 0$.

Planes

Equation (2.5) defines a set of points in \mathbb{E}^2 that form a line. The straightforward extension of this equation to \mathbb{E}^3 is not a line, but instead a two-dimensional surface, a plane. It is defined by all points $\bar{\mathbf{x}} = (x_1, x_2, x_3)$ that satisfy the *equation of the plane*:

$$x_1 p_1 + x_2 p_2 + x_3 p_3 = \Delta, \quad (2.18)$$

where p_1, p_2, p_3 , and Δ are parameters that characterize the plane. A set of points that all lie in the same plane are referred to as *co-planar*⁷.

In the same way as for the 2D case, we can normalize the parameters of this equation such that $\Delta \geq 0$ and $p_1^2 + p_2^2 + p_3^2 = 1$. Furthermore, with this normalization we can interpret Δ as the distance from the origin to the plane, measured in a direction perpendicular to the plane. The vector $\hat{\mathbf{p}} = (p_1, p_2, p_3)$ is then a normal of the plane. If $\Delta > 0$, the vector $\hat{\mathbf{p}}$ is unique and points from the origin to the plane. When $\Delta = 0$, $\hat{\mathbf{p}}$ can point in either of two directions, with opposite signs. Figure 2.6 illustrates the parameters of a plane in \mathbb{E}^3 .

Alternatively, the plane can be explicitly represented as the set of points $\bar{\mathbf{x}}$ that solve Equation (2.18):

$$\bar{\mathbf{x}}(s, t) = s \bar{\mathbf{t}}_1 + t \bar{\mathbf{t}}_2 + \bar{\mathbf{x}}_0 = (\bar{\mathbf{t}}_1 \quad \bar{\mathbf{t}}_2 \quad \bar{\mathbf{x}}_0) \begin{pmatrix} s \\ t \\ 1 \end{pmatrix}, \quad (2.19)$$

where $\bar{\mathbf{t}}_1$ and $\bar{\mathbf{t}}_2$ are two linearly independent tangent vectors of the plane, i.e., $\bar{\mathbf{t}} \cdot \hat{\mathbf{p}} = \bar{\mathbf{t}}_2 \cdot \hat{\mathbf{p}} = 0$, $\bar{\mathbf{x}}_0$ is an arbitrary point in the plane, and s, t are any real values.

⁷The term co-planar includes the cases when the points are co-linear, as well as when all points are identical.

Lines

To get a line in \mathbb{E}^3 , we extend instead the parametric representation of a line in Equation (2.10) to the three-dimensional case:

$$\bar{\mathbf{x}}(s) = s \bar{\mathbf{t}} + \bar{\mathbf{x}}_0, \quad (2.20)$$

where s assumes any value in \mathbb{R} , $\bar{\mathbf{t}}$ is a tangent vector of the line, and $\bar{\mathbf{x}}_0$ is an arbitrary point on the line. Alternatively, we can extend the parametric form in Equation (2.13) and write the set of points on a line in \mathbb{E}^3 as

$$\bar{\mathbf{x}} = \bar{\mathbf{x}}(t) = t \bar{\mathbf{x}}_1 + (1 - t) \bar{\mathbf{x}}_2, \quad (2.21)$$

where $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$ are two distinct points on the line, and $t \in \mathbb{R}$. Both these parametric representations suffer from the same type of ambiguities that has been discussed for a 2D line. Without additional assumptions, we cannot uniquely determine the points $\bar{\mathbf{x}}_0$, or $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$ for a specific line.

We can also see a 3D line as the intersection of two planes. This observation becomes important in Section 5.3.2, which describes an even more useful representation of 3D lines. In the same way as for the 2D case, a set of points that all lie on the same line are *co-linear*. A set of planes that intersect along a common line are said to be *co-linear*, too. Finally, a set of points or lines in \mathbb{E}^3 that all lie in a common plane are *co-planar*.

2.4 Basic geometric operations

We have now established the basic geometric objects in \mathbb{E}^2 and \mathbb{E}^3 . We have also discussed how to represent them, using Cartesian coordinates for points, and various parameters for lines or planes. Next, we look at basic operations that can be applied to these objects. We will not go through the whole catalog, it suffices with some examples. They illustrate the type of computations that result when we ask simple questions about relations in geometry.

2.4.1 The line that intersects two points in \mathbb{E}^2

Let $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$ be two points in 2D space. What is the line that intersects both points, as illustrated in Figure 2.7? Clearly, $\bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2$ is a tangent vector of this line, and by rotating this line 90° we obtain a normal vector, from which we can derive the parameters l_1 and l_2 of the line. With

$$\bar{\mathbf{y}}_1 = \begin{pmatrix} u_1 \\ v_1 \end{pmatrix} \quad \bar{\mathbf{y}}_2 = \begin{pmatrix} u_2 \\ v_2 \end{pmatrix}, \quad (2.22)$$

we have

$$\text{tangent vector} = \bar{\mathbf{y}}_1 - \bar{\mathbf{y}}_2 = \begin{pmatrix} u_1 - u_2 \\ v_1 - v_2 \end{pmatrix}, \quad \Rightarrow \quad \text{normal vector} = \begin{pmatrix} v_2 - v_1 \\ u_1 - u_2 \end{pmatrix}. \quad (2.23)$$

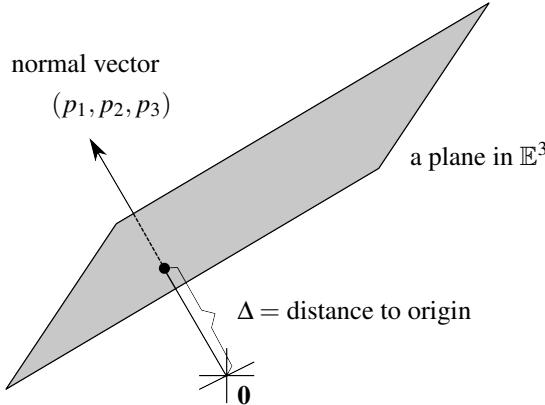


Figure 2.6: A plane in \mathbb{E}^3 and its corresponding parameters.

If we swap the two points, the sign of the normal vector changes, so the normal vector is not fully determined regarding the sign in accordance with the rule described in Section 2.2. By properly normalizing the normal vector of the line, we get

$$\hat{\mathbf{l}} = \begin{pmatrix} l_1 \\ l_2 \end{pmatrix} = \frac{\pm 1}{\sqrt{(u_1 - u_2)^2 + (v_1 - v_2)^2}} \begin{pmatrix} v_2 - v_1 \\ u_1 - u_2 \end{pmatrix}. \quad (2.24)$$

To get Δ , we can insert this normal vector together with either of $\bar{\mathbf{y}}_1$ or $\bar{\mathbf{y}}_2$ into Equation (2.5). Using $\bar{\mathbf{y}}_1$ in Equation (2.5), Δ is given as

$$\Delta = \frac{\pm 1}{\sqrt{(u_1 - u_2)^2 + (v_1 - v_2)^2}} (u_1(v_2 - v_1) + v_1(u_1 - u_2)) = \frac{\pm(u_1 v_2 - u_2 v_1)}{\sqrt{(u_1 - u_2)^2 + (v_1 - v_2)^2}}. \quad (2.25)$$

We get the same Δ also when we use $\bar{\mathbf{y}}_2$ in Equation (2.5). To summarize: given two distinct points $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$, their common intersecting line has parameters $\hat{\mathbf{l}} = (l_1, l_2)$ and Δ given by Equation (2.24) and Equation (2.25). The sign in Equation (2.25) becomes well-defined if we assume $\Delta \geq 0$.

In terms of the defining equation of a line, Equation (2.5), the normalization of the parameters (l_1, l_2) and Δ is arbitrary, and we can equally well use the simpler expressions:

$$\begin{pmatrix} l_1 \\ l_2 \end{pmatrix} = \begin{pmatrix} v_2 - v_1 \\ u_1 - u_2 \end{pmatrix} \quad \text{and} \quad \Delta = u_1 v_2 - u_2 v_1, \quad (2.26)$$

but then we cannot interpret Δ as the distance to the plane from the origin.

As a concluding remark, we can compute a unique line that intersects $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$ only if $\bar{\mathbf{y}}_1 \neq \bar{\mathbf{y}}_2$. Otherwise, there is an infinite set of lines intersecting both points. If $\bar{\mathbf{y}}_1 = \bar{\mathbf{y}}_2$ we get $(l_1, l_2) = (0, 0)$, and this cannot be a line normal. Consequently, Equations (2.24) and (2.25) must be used with some care.

2.4.2 The point of intersection between two lines in \mathbb{E}^2

Given two lines in \mathbb{E}^2 we can determine their point of intersection. Let the two lines be parameterized as

$$\begin{pmatrix} l_1 \\ l_2 \end{pmatrix} \quad \text{in combination with} \quad \Delta_1, \quad \text{and} \quad \begin{pmatrix} m_1 \\ m_2 \end{pmatrix} \quad \text{in combination with} \quad \Delta_2. \quad (2.27)$$

We assume here that the line parameters are normalized according to the discussion in Section 2.2. We can write a point on either of the two lines in parametric form, Equation (2.12), as

$$\bar{\mathbf{y}}_1(s) = \begin{pmatrix} l_2 & \Delta_1 l_1 \\ -l_1 & \Delta_1 l_2 \end{pmatrix} \begin{pmatrix} s \\ 1 \end{pmatrix}, \quad \bar{\mathbf{y}}_2(t) = \begin{pmatrix} m_2 & \Delta_2 m_1 \\ -m_1 & \Delta_2 m_2 \end{pmatrix} \begin{pmatrix} t \\ 1 \end{pmatrix}. \quad (2.28)$$

Each of the two lines is parameterized by an independent parameter, s and t , respectively, and we want to determine values for these parameters, here denoted s_0 and t_0 , such that the two parameterizations produce the same point, i.e., $\bar{\mathbf{y}}_1(s_0) = \bar{\mathbf{y}}_2(t_0)$. This gives two linear equations in s_0 and t_0 :

$$\begin{pmatrix} l_2 s_0 + \Delta_1 l_1 \\ -l_1 s_0 + \Delta_1 l_2 \end{pmatrix} = \begin{pmatrix} m_2 t_0 + \Delta_2 m_1 \\ -m_1 t_0 + \Delta_2 m_2 \end{pmatrix}. \quad (2.29)$$

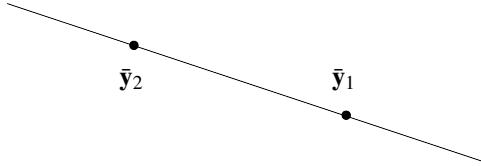


Figure 2.7: Two points in \mathbb{E}^2 and their intersecting line.

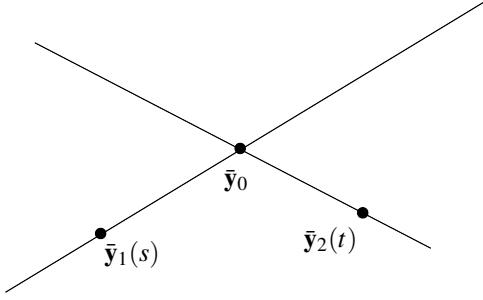


Figure 2.8: Two lines in \mathbb{E}^2 and their point of intersection, \bar{y}_0 .

If we take into account that $l_1^2 + l_2^2 = m_1^2 + m_2^2 = 1$ and solve for s_0 and t_0 , the result is

$$\begin{pmatrix} s_0 \\ t_0 \end{pmatrix} = \frac{1}{l_2 m_1 - l_1 m_2} \begin{pmatrix} -\Delta_1 l_1 m_1 - \Delta_1 l_2 m_2 + \Delta_2 \\ -\Delta_1 + \Delta_2 l_1 m_1 + \Delta_2 l_2 m_2 \end{pmatrix}. \quad (2.30)$$

We obtain the point of intersection, \bar{y}_0 illustrated in Figure 2.8, by inserting either s_0 or t_0 in Equation (2.30) into $\bar{y}_1(s_0)$ or $\bar{y}_2(t_0)$, respectively, described in Equation (2.28):

$$\bar{y}_0 = \bar{y}_1(s_0) = \bar{y}_2(t_0) = \frac{1}{l_2 m_1 - l_1 m_2} \begin{pmatrix} \Delta_2 l_2 - \Delta_1 m_2 \\ \Delta_1 m_1 - \Delta_2 l_1 \end{pmatrix}. \quad (2.31)$$

As a concluding observation, we can compute a unique point as the intersection of two lines only when the two lines are distinct. Otherwise, there is an infinite set of points that lie on both lines. Moreover, to get a reasonable result from Equation (2.31), the two lines must not be parallel. As a consequence, we have to observe some care when using also this equation. We will see later on that the assumption of non-parallel lines does not apply when using homogeneous representations.

2.4.3 Distance between a point and a line in \mathbb{E}^2

Consider a point \bar{y} and a line in 2D, where the line has a normal vector $\hat{\mathbf{l}} = (l_1, l_2)$, of unit norm, and distance to the origin $\Delta \geq 0$. What is the distance, d , between the point and the line, measured perpendicularly to the line?

Since $\hat{\mathbf{l}}$ has unit length we can construct an orthogonal vector $\hat{\mathbf{t}} = (-l_2, l_1)$, a tangent to the line, such that $\hat{\mathbf{l}}, \hat{\mathbf{t}}$ form an ON-basis of \mathbb{R}^2 . This means we can expand \bar{y} as⁸

$$\bar{y} = \underbrace{\hat{\mathbf{l}}(\hat{\mathbf{l}} \cdot \bar{y})}_{:=\bar{y}_1} + \underbrace{\hat{\mathbf{t}}(\hat{\mathbf{t}} \cdot \bar{y})}_{:=\bar{y}_2} = \bar{y}_1 + \bar{y}_2. \quad (2.32)$$

Consequently, we see \bar{y} as a sum of two orthogonal vectors. One vector, $\bar{y}_1 = \hat{\mathbf{l}}(\hat{\mathbf{l}} \cdot \bar{y})$, is normal to the line, and the other vector, $\bar{y}_2 = \hat{\mathbf{t}}(\hat{\mathbf{t}} \cdot \bar{y})$, is parallel to the line. The vector \bar{y}_1 is the orthogonal projection of \bar{y} onto the normal vector $\hat{\mathbf{l}}$. The different points and vectors are illustrated in Figure 2.9.

The point on the line closest to the origin is given by $\bar{y}_0 = \Delta \hat{\mathbf{l}}$. Both \bar{y}_1 and \bar{y}_0 lie on the line that intersects the origin and is perpendicular to the original line. The quantity we want to determine, d , is the distance between these two points:

$$d = \|\bar{y}_1 - \bar{y}_0\|. \quad (2.33)$$

By inserting the expressions, derived for \bar{y}_1 and \bar{y}_0 , we get

$$d = \|\hat{\mathbf{l}}(\hat{\mathbf{l}} \cdot \bar{y}) - \Delta \hat{\mathbf{l}}\| = \|\hat{\mathbf{l}}(\hat{\mathbf{l}} \cdot \bar{y} - \Delta)\| / \text{since } \hat{\mathbf{l}} \text{ has unit norm} / = |\hat{\mathbf{l}} \cdot \bar{y} - \Delta|. \quad (2.34)$$

⁸Here we are expanding a vector using an ON-basis in accordance with Toolbox Section 3.1.4.

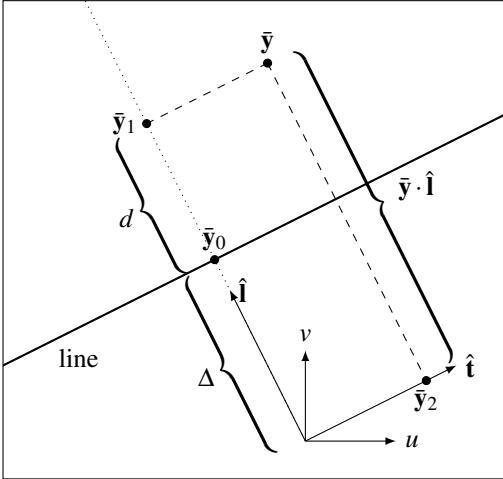


Figure 2.9: A point \bar{y} and a line in \mathbb{E}^2 , with distance d between them.

2.5 Before We Continue

In this chapter we have derived explicit expressions for the line that intersects two points, and for the point of intersection relative to two lines. They are based on Cartesian coordinates for points, represented as vectors in \mathbb{R}^2 , and the usual parameterization of a line. We could have made similar derivations for the 3D plane that intersects three points, the common point of intersection to three planes, or the intersecting point of a 3D line with a plane, and so on. The derivations included here are not motivated by the resulting expressions alone. Instead, they illustrate that rather simple geometric questions sometimes lead to complex computations. For example, if we ask “what is the common point of intersection for two specific lines in 2D?”, we find that several computational steps are needed to determine the answer. First, we need to solve one equation and then insert the solution into another. To be sure, we can determine the resulting expression once and for all, as in Equation (2.31). But the complexity of this equation is enough to prevent most reader from remembering it by heart.

The following chapters introduce an alternative representation of geometric objects, based on *homogeneous coordinates*. They make way for simpler derivations of the above results and also make the resulting expressions less complicated. Homogeneous coordinates are elements of *projective spaces*.

Chapter 3

Homogeneous Representations in 2D

Before you read this chapter, you should be familiar with the basic results presented in Chapter 2. You should also have a look at Toolbox Section 7.1, which explains the concept of projective spaces and projective elements. The cross product operator is defined in Toolbox Section 3.7.3. It may also help to have a quick look at the singular value decomposition, presented in Toolbox Section 8.2.

In this and in the following chapters, you will be introduced to homogeneous representations of various types of *geometric objects*. More precisely, these geometric objects are:

- **Euclidean objects.** The usual objects that you are familiar with from Euclidean geometry. They include points and lines in \mathbb{E}^2 , and point, planes, and lines in \mathbb{E}^3 and its extension to the projective plane. Other examples are circles, cones, and more or less exotic surfaces, but these are not covered in this compendium.
- **Constraints.** Euclidean objects are not very exciting one at the time. But as soon as you have at least a pair of them, it is possible to talk about how they relate to one another. For example, we can ask if a point lies on a specific line in \mathbb{E}^2 , or if two lines in \mathbb{E}^3 intersect or not. As you will see shortly, homogeneous representations allow us to encode such relations as algebraic constraints. They often take the form of homogeneous equations that involve vectors or matrices.
- **Transformations.** We will often have reasons to transform to Euclidean objects. Common examples of transformations are rotations, translations, and scaling operations. Other examples that we will encounter include shearing, reflections, and something called homographies.

The notion of constraints or transformations as geometric objects may not be immediately clear. One reason for this interpretation is that they have homogeneous representations derived from, and compatible with, the homogeneous representations defined for Euclidean objects. This leads to the second reason: estimation of geometric objects from observed data. We can estimate a line that passes through a set of observed points. Based on the homogeneous representations, estimation extends also to constraints and transformations. As we will see, there is a common toolbox of methods which applies to almost any type of geometric object.

In this chapter we will introduce homogeneous representations for Euclidean objects in \mathbb{E}^2 . This space will be extended to include objects at “infinite distance”. We will also discuss how to describe geometric relations between these objects in an algebraic form. Later chapters extend these results to \mathbb{E}^3 , and to homogeneous representations of transformations.

3.1 Homogeneous coordinates of 2D points

Let $\bar{\mathbf{y}}$ be a point in \mathbb{E}^2 with Cartesian coordinates in \mathbb{R}^2 given as:

$$\bar{\mathbf{y}} = \begin{pmatrix} u \\ v \end{pmatrix}. \quad (3.1)$$

The *canonical form* of the homogeneous coordinates of the point $\bar{\mathbf{y}}$ is a vector in \mathbb{R}^3 :

$$\begin{pmatrix} \bar{\mathbf{y}} \\ 1 \end{pmatrix} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}. \quad (3.2)$$

We define the *homogeneous coordinates* of $\bar{\mathbf{y}}$ as the projective element $\mathbf{y} \in P(\mathbb{R}^3)$ generated by the canonical form in Equation (3.2):

$$\mathbf{y} \sim \text{Eq} \left[\begin{pmatrix} \bar{\mathbf{y}} \\ 1 \end{pmatrix} \right] = \text{Eq} \left[\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \right]. \quad (3.3)$$

Here, $\text{Eq}(\mathbf{v})$ is the equivalence class generated by \mathbf{v} and the relation \sim , described in Toolbox Section 7.1. Equation (3.3) defines a mapping $\mathbb{R}^2 \rightarrow P(\mathbb{R}^3)$. It takes the 2-dimensional vector $\bar{\mathbf{y}}$ and concatenates it with an extra dimension. The third element is set = 1. The result is a vector in \mathbb{R}^3 , the canonical form in Equation (3.2). This vector is a representative of a projective element in $P(\mathbb{R}^3)$, here denoted \mathbf{y} . Formally, this projective element is what we mean by the homogeneous coordinates of $\bar{\mathbf{y}}$. This means that any vector in \mathbb{R}^3 that is equivalent to the canonical form in Equation (3.2) is a valid representative of the homogeneous coordinates of $\bar{\mathbf{y}}$. In short, we can generate a representative of the homogeneous coordinates of $\bar{\mathbf{y}} \in \mathbb{R}^2$ by adding an extra dimension set to 1. We are also allowed to multiply the resulting vector in \mathbb{R}^3 by an arbitrary non-zero scalar.

We can, in fact, define homogeneous coordinates various ways. Appending the extra dimension at the end is an arbitrary choice, as is setting to 1. Equation (3.3) shows the standard formulation used in most textbooks, and this is the formulation that we use in the rest of this compendium.

About notation

Before we continue, a comment on notation is helpful. We will use \mathbf{y} to denote a projective element in $P(\mathbb{R}^3)$, and also to denote a specific vector in \mathbb{R}^3 . The vector is a representative of the projective element, as described in Toolbox Section 7.1.1. However, we will not make a sharp distinction between $\text{Eq}(\mathbf{v})$ and \mathbf{v} , when $\mathbf{v} \in \mathbb{R}^3$. Since \mathbf{y} is the homogeneous coordinates of a specific point in \mathbb{E}^2 , we will also use \mathbf{y} to refer to this point. The context often makes it clear which interpretation is the correct one when we use \mathbf{y} as a notation. But sometimes it is necessary to make a clear distinction between these interpretations: do \mathbf{y} mean a projective element in $P(\mathbb{R}^3)$, a representative of a projective element, or a point in \mathbb{E}^2 ? In these cases, we have to write “the projective element”, “the vector” or “the point” to clarify what \mathbf{y} stands for. It is now time to present our first observation:

3.1 The notation does not make a distinction between projective elements, in $P(\mathbb{R}^3)$, representatives of these projective elements as vectors in \mathbb{R}^3 , and the points in \mathbb{E}^2 that have Cartesian coordinates $\bar{\mathbf{y}}$ when \mathbf{y} are the homogeneous coordinates of $\bar{\mathbf{y}}$.

An example

Consider the 2D-point $\bar{\mathbf{y}} = (1, 2)$ that has a homogeneous representation, for example as the vectors

$$\mathbf{y} \sim \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} \sim \begin{pmatrix} 2 \\ 4 \\ 2 \end{pmatrix} \sim \begin{pmatrix} -1 \\ -2 \\ -1 \end{pmatrix}. \quad (3.4)$$

All three vectors are representatives of the same projective element, this projective element consists of the homogeneous coordinates of a point in \mathbb{E}^2 with Cartesian coordinates $(1, 2)$ relative to the chosen coordinate system.

3.1.1 P-normalization

The mapping $\mathbb{R}^2 \rightarrow P(\mathbb{R}^3)$ defined by homogeneous coordinates in Equation (3.3) has an inverse, mapping a projective element in $P(\mathbb{R}^3)$ back to \mathbb{R}^2 . Let \mathbf{y} be a vector in \mathbb{R}^3 that represents the homogeneous coordinates of

some point $\bar{\mathbf{y}} \in \mathbb{R}^2$. We know that \mathbf{y} is equal to some non-zero scalar times the canonical form in Equation (3.2), so with numerical values for \mathbf{y} given by

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}, \quad (3.5)$$

we see that

$$\mathbf{y} \sim \begin{pmatrix} y_1/y_3 \\ y_2/y_3 \\ 1 \end{pmatrix}. \quad (3.6)$$

By identifying elements in Equation (3.3) and Equation (3.6) it follows that

$$\bar{\mathbf{y}} = \begin{pmatrix} y_1/y_3 \\ y_2/y_3 \end{pmatrix}. \quad (3.7)$$

The process of normalizing a representative of a homogeneous element in $P(\mathbb{R}^3)$, in this case the vector $\mathbf{y} \in \mathbb{R}^3$ with specific numerical values given by Equation (3.5), to the canonical form in Equation (3.2), is here referred to as *point normalization*, or *P-normalization* for short. It allows us to take a general vector in \mathbb{R}^3 and determine which point in \mathbb{E}^2 it represents as its homogeneous coordinates. Clearly, this process assumes that $y_3 \neq 0$, otherwise the P-normalization gives an undefined result. As we will see in Section 3.5, it is possible to give an interpretation of \mathbf{y} also for the case $y_3 = 0$.

Three useful functions

In the following presentation we use `normp` to denote the general mapping $\mathbb{R}^n \rightarrow \mathbb{R}^n$ that divides a vector by its last element. It produces the canonical form for a vector that represents homogeneous coordinates of a point. The function `normp` : $\mathbb{R}^n \rightarrow \mathbb{R}^n$ is defined as:

$$\text{normp} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} v_1/v_n \\ v_2/v_n \\ \vdots \\ 1 \end{pmatrix}, \quad \text{where } v_n \neq 0. \quad (3.8)$$

This function is well-defined for arbitrary dimensions of \mathbb{R}^n , except for vectors where the last element = 0. We also need a function `cart` : $\mathbb{R}^n \rightarrow \mathbb{R}^{n-1}$ producing the Cartesian coordinates $\bar{\mathbf{v}} \in \mathbb{R}^{n-1}$ of a point with homogeneous coordinates $\mathbf{v} \in \mathbb{R}^n$:

$$\text{cart} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} v_1/v_n \\ v_2/v_n \\ \vdots \\ v_{n-1}/v_n \end{pmatrix}, \quad \text{where } v_n \neq 0. \quad (3.9)$$

Finally, we define a function `hom` : $\mathbb{R}^{n-1} \rightarrow \mathbb{R}^n$ that produces the canonical form of the homogeneous coordinates from Cartesian coordinates $\bar{\mathbf{u}} \in \mathbb{R}^{n-1}$:

$$\text{hom} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \\ 1 \end{pmatrix}. \quad (3.10)$$

These three functions are related in the following way for vectors $\bar{\mathbf{u}} \in \mathbb{R}^{n-1}$ and $\mathbf{v} \in \mathbb{R}^n$, where $v_n \neq 0$:

$$\bar{\mathbf{u}} = \text{cart}(\text{hom}(\bar{\mathbf{u}})), \quad \mathbf{v} \sim \text{hom}(\text{cart}(\mathbf{v})), \quad \text{normp}(\mathbf{v}) = \text{hom}(\text{cart}(\mathbf{v})). \quad (3.11)$$

3.2 Dual homogeneous coordinates of 2D lines

Recall the algebraic definition of a line given by Equation (2.5) in Section 2.2, which we here rewrite as

$$ul_1 + vl_2 - \Delta = 0. \quad (3.12)$$

A specific line is characterized by the parameters l_1, l_2, Δ , and any point in \mathbb{E}^2 with Cartesian coordinates $\bar{\mathbf{y}} = (u, v)$ lies on the line if and only if Equation (3.12) is satisfied. The line parameters are ambiguous unless we normalize them, e.g., such that $l_1^2 + l_2^2 = 1$ and $\Delta \geq 0$. In this case, $\hat{\mathbf{l}} = (l_1, l_2)$ is a normal vector of the line, in general pointing “away” from the origin, and Δ as the closest distance from the origin to the line. See Figure 2.3 for an illustration of the line parameters.

We rewrite Equation (3.12) in the following form

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \cdot \begin{pmatrix} l_1 \\ l_2 \\ -\Delta \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{y}} \\ 1 \end{pmatrix} \cdot \begin{pmatrix} \hat{\mathbf{l}} \\ -\Delta \end{pmatrix} = 0. \quad (3.13)$$

In the light of Section 3.1, the last expression is a scalar product between two vectors where the first vector represents the homogeneous coordinates of the point $\bar{\mathbf{y}}$. The second vector depends only on the parameters of the line:

$$\begin{pmatrix} l_1 \\ l_2 \\ -\Delta \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{l}} \\ -\Delta \end{pmatrix}. \quad (3.14)$$

This is the *canonical form* of a vector that represents a line in \mathbb{E}^2 . As before, we assume that $l_1^2 + l_2^2 = 1$ and $\Delta \geq 0$. The homogeneous representation of a line in \mathbb{E}^2 is the projective element in $P(\mathbb{R}^3)$ corresponding to the canonical form¹:

$$\mathbf{l} \sim \begin{pmatrix} l_1 \\ l_2 \\ -\Delta \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{l}} \\ -\Delta \end{pmatrix}. \quad (3.15)$$

Consequently, Equation (3.13) can be written in the more compact form as

$$\mathbf{y} \cdot \mathbf{l} = \mathbf{y}^\top \mathbf{l} = 0. \quad (3.16)$$

To see what Equation (3.16) means, recall that \mathbf{y} represents the homogeneous coordinates of the 2D point $\bar{\mathbf{y}}$, as defined in Equation (3.3). In Equation (3.16), we can multiply the vector \mathbf{y} by any non-zero scalar and still it will be orthogonal to the vector \mathbf{l} . This observation applies also to \mathbf{l} , it can be multiplied by any non-zero scalar and still be orthogonal to \mathbf{y} . As a consequence, Equation (3.16) defines an orthogonality relation between two projective elements of $P(\mathbb{R}^3)$. One element represents a 2D point lying on the line, and one represents the line itself.

The projective element represented by \mathbf{l} in Equation (3.15) is here referred to as the *dual homogeneous coordinates* of the line with parameters l_1, l_2, Δ . At this point it is not obvious in what sense these coordinates are “dual”, and we will return to this issue in Section 4.5. At least it should be evident that a given vector $\mathbf{v} \in \mathbb{R}^3$ can represent either the homogeneous coordinates of a 2D point or the dual homogeneous coordinates of a line. Information about its type is not available from the vector’s elements. We have to know the definition of \mathbf{v} , or how to compute it, to determine its true character. Once this is done, we have to treat \mathbf{v} exclusively as either the homogeneous coordinates of a point, or as the dual homogeneous coordinates of a line.

To summarize: we have defined a new type of projective element, a homogeneous representation of a 2D-line in the form of its dual homogeneous coordinates. The geometric statement that a point $\bar{\mathbf{y}}$ lies on a line is equivalent to the algebraic statement that the homogeneous coordinates of the point, \mathbf{y} , and the dual homogeneous coordinates of the line, \mathbf{l} , are orthogonal: $\mathbf{y} \cdot \mathbf{l} = 0$. In the same way as for \mathbf{y} , the symbol \mathbf{l} refers either to the dual homogeneous coordinates of a line, a projective element in $P(\mathbb{R}^3)$, or to a specific vector in \mathbb{R}^3 that is a representative of the

¹Remember observation 3.1 on page 32

projective element. It can also refer to the corresponding line in \mathbb{E}^2 . The context should make it clear which interpretation is the correct one.

3.2 There are two distinct types of homogeneous representations for Euclidean objects in \mathbb{E}^2 : the homogeneous coordinates that refer to points, and the dual homogeneous coordinates that refer to lines. They should not be confused.

3.2.1 D-normalization

Let $\mathbf{l} \in \mathbb{R}^3$ be a representative of the dual homogeneous coordinates of some 2D-line, where

$$\mathbf{l} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix}. \quad (3.17)$$

What line is it? We know that \mathbf{l} is equal to some non-zero scalar times the canonical form in Equation (3.14), and if we assume that the parameters of the line satisfy $l_1^2 + l_2^2 = 1$ and $\Delta \geq 0$ we see that

$$\mathbf{l} \sim \frac{-\text{sign}(\beta_3)}{\sqrt{\beta_1^2 + \beta_2^2}} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} l_1 \\ l_2 \\ -\Delta \end{pmatrix}, \quad (3.18)$$

where²

$$\text{sign}(x) = \begin{cases} -1 & x < 0, \\ \pm 1 & x = 0, \\ 1 & x > 0. \end{cases} \quad (3.19)$$

With references to the discussion in Section 2.2, where α denotes the angle of the line, and Δ is the orthogonal distance from the origin to the line, we see that

$$l_1 = \cos \alpha = -\frac{\text{sign}(\beta_3)\beta_1}{\sqrt{\beta_1^2 + \beta_2^2}}, \quad (3.20)$$

$$l_2 = \sin \alpha = -\frac{\text{sign}(\beta_3)\beta_2}{\sqrt{\beta_1^2 + \beta_2^2}}, \quad (3.21)$$

$$\Delta = \frac{\text{sign}(\beta_3)\beta_3}{\sqrt{\beta_1^2 + \beta_2^2}}. \quad (3.22)$$

The normalization of the vector \mathbf{l} described in Equation (3.18) is here referred to as *dual line normalization*, or *D-normalization* for short. It makes the parameters of the line explicit, as the canonical form of dual homogeneous coordinates in Equation (3.14). For the general case, we define the function $\text{norm}_D : \mathbb{R}^n \rightarrow \mathbb{R}^n$ as

$$\text{norm}_D \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix} = \frac{-\text{sign}(\beta_n)}{\sqrt{\beta_1^2 + \dots + \beta_{n-1}^2}} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix}. \quad (3.23)$$

²The sign function used here returns either +1 or -1 when applied to 0.

Clearly, this normalization requires that $\beta_1, \dots, \beta_{n-1}$ are not all zero. In Section 3.5, we discuss possible interpretations of \mathbf{l} when this assumption does not apply.

3.3 Each of the two homogeneous representations, for points and for lines, have a specific form of normalization operation that can be applied to a vector in \mathbb{R}^3 to produce the corresponding canonical form. You need to know what type of geometric objects you are working with in order to apply the correct normalization. Applying the wrong one is like making tea from coffee beans: you will not get the expected result.

3.3 Relations between points and lines

In this section we get a first glimpse of practical applications that use homogeneous representation of points and lines. The topic is so-called *incidence relations*. They talk about whether some points lie on a line, or some lines intersect at a single point, and conditions for these events. We base these geometric relations on the algebraic relation $\mathbf{y} \cdot \mathbf{l} = 0$, where \mathbf{y} and \mathbf{l} are the homogeneous representations of a point and a line that intersect, as described in Section 3.2.

3.3.1 The line that intersects two points

Let \mathbf{y}_1 and \mathbf{y}_2 be the homogeneous coordinates of two 2D points. Which line intersects both points? Or, more precisely, what are the dual homogeneous coordinates \mathbf{l} of this line? We know that both \mathbf{y}_1 and \mathbf{y}_2 intersect \mathbf{l} and, consequently, it must be the case that

$$\mathbf{y}_1 \cdot \mathbf{l} = \mathbf{y}_2 \cdot \mathbf{l} = 0. \quad (3.24)$$

This means that we seek \mathbf{l} that is orthogonal to both \mathbf{y}_1 and \mathbf{y}_2 . Based on the properties of the cross product in \mathbb{R}^3 , this allows us to construct \mathbf{l} as

$$\mathbf{l} \sim \mathbf{y}_1 \times \mathbf{y}_2 \sim \mathbf{y}_2 \times \mathbf{y}_1. \quad (3.25)$$

There are some points worth mentioning here. One is the minimal effort that we used here to derive a homogeneous representation of the intersecting line of two points. Compare this to the “traditional” derivation presented in Section 2.4.1. Another point is the simple formulation of the result: as a cross product between two vectors. This is in contrast to the rather elaborate expressions in Equation (2.24) and Equation (2.25). We see that the unnormalized line parameters in Equation (2.26) are exactly the elements of the vector $\mathbf{l} = \mathbf{y}_1 \times \mathbf{y}_2$, for the case that both \mathbf{y}_1 and \mathbf{y}_2 are P-normalized.

3.4 In general, homogeneous representations give simple geometric relations a simple algebraic form, and these forms are often simple to derive. This is in contrast to the corresponding forms based on Cartesian coordinates, which are tedious to derive and produce more complex expressions as results.

A degenerate case

What happens in the case that $\mathbf{y}_1 \sim \mathbf{y}_2$, i.e., \mathbf{y}_1 and \mathbf{y}_2 refer to the same point in \mathbb{E}^2 ? In this case, there is an infinite set of lines that intersect this single point, and we get

$$\mathbf{l} \sim \mathbf{y}_1 \times \mathbf{y}_2 = \mathbf{0}. \quad (3.26)$$

This is an important result that deserves some discussion. The vector $\mathbf{0} \in \mathbb{R}^3$ cannot be D-normalized into line parameters, using the procedure described in Section 3.2.1. Instead, we use the zero vector as a “flag”, which indicates that the result is not a specific line. In this case the result is instead ambiguous since there are many solutions to the problem of determining the line.

This observation means that it is not necessary to first check if \mathbf{y}_1 and \mathbf{y}_2 are distinct points. If they are, their cross product gives \mathbf{l} , the intersecting line. If the points are not distinct, and only then, the cross product is instead the zero vector that flags an ambiguous result. This is in contrast to the procedure in Section 2.4.1, where we are

required to check that $\bar{\mathbf{y}}_1 \neq \bar{\mathbf{y}}_2$ before making the calculations. In general, the zero vector flags that we have applied an operation to *degenerate data*, e.g., data for which the result is ambiguous.

3.5 The zero vector is often used as a flag to signal that some operation has been applied to degenerate data.

3.3.2 The point of intersection between two lines

Finding the homogeneous representation of the point of intersection between two lines is similar to the case of finding the line that intersects two points. We do this by interchanging points and lines. Let \mathbf{l}_1 and \mathbf{l}_2 be the dual homogeneous coordinates of two 2D lines. We want to find the point of intersection between \mathbf{l}_1 and \mathbf{l}_2 . More precisely, we seek the homogeneous coordinates \mathbf{y} of this point. Since both \mathbf{l}_1 and \mathbf{l}_2 intersect \mathbf{y} , it must be the case that

$$\mathbf{l}_1 \cdot \mathbf{y} = \mathbf{l}_2 \cdot \mathbf{y} = 0. \quad (3.27)$$

This means that we seek \mathbf{y} that is orthogonal to both \mathbf{l}_1 and \mathbf{l}_2 , and this allows us to construct \mathbf{y} as

$$\mathbf{y} \sim \mathbf{l}_1 \times \mathbf{l}_2 \sim \mathbf{l}_2 \times \mathbf{l}_1. \quad (3.28)$$

Again, we see that homogeneous representations lead to a simple expression for the intersecting point, as well as a simple way to derive it. Compare this to the “traditional” derivation in Section 2.4.2. We also notice that if the two lines \mathbf{l}_1 and \mathbf{l}_2 have the parameters in Equation (2.27), then $\mathbf{y} = \mathbf{l}_1 \times \mathbf{l}_2$ represents the homogeneous coordinates of the point $\bar{\mathbf{y}}_0$ in Equation (2.31). We get its Cartesian coordinates by a P-normalization of \mathbf{y} .

From the outset we have to assume the two lines \mathbf{l}_1 and \mathbf{l}_2 to be distinct. If not, there are infinitely many points that intersect both \mathbf{l}_1 and \mathbf{l}_2 . But we also see that when two lines are not distinct, it follows that $\mathbf{y} = \mathbf{l}_1 \times \mathbf{l}_2 = \mathbf{0}$, and we can use $\mathbf{0}$ as a flag for this special case, in a similar way as we did in Section 3.3.1. In summary, if the two lines, \mathbf{l}_1 and \mathbf{l}_2 , are distinct, $\mathbf{l}_1 \times \mathbf{l}_2$ gives the homogeneous coordinates of their point of intersection. If, and only if, the two lines coincide this cross product instead vanishes, indicating a degenerate case.

Looking ahead

Here you may argue that also when \mathbf{l}_1 and \mathbf{l}_2 are distinct but parallel lines the result is degenerate. Since such lines never intersect, $\mathbf{l}_1 \times \mathbf{l}_2$ cannot represent a point. In Euclidean geometry, this statement is correct. However, the homogeneous representation that is introduced in this chapter leads to an extension of the usual set of points, to include points at infinity. This enables us to state that two distinct lines *always* intersect at a unique point, either at a “normal” Euclidean point or at a point at infinity. This idea will be further developed in Section 3.5.

3.3.3 Points on a line, or not

Two distinct points always intersect with a line in \mathbb{E}^2 , but what if we have three or more points? How can we test if they are co-linear? And if they are co-linear, what line do they lie on? Let $\mathbf{y}_1, \dots, \mathbf{y}_m$ be the homogeneous coordinates of $m \geq 2$ points in \mathbb{E}^2 . If they all are lying on one and the same line \mathbf{l} , it must be the case that $\mathbf{y}_k \cdot \mathbf{l} = 0$ for $k = 1, \dots, m$. A more compact way of describing this statement is to first form the $3 \times m$ matrix $\mathbf{Y} = (\mathbf{y}_1 \dots \mathbf{y}_m)$, holding the homogeneous coordinates of the points in its columns. If all the points lie on the line \mathbf{l} , this is equivalent to

$$\begin{pmatrix} \mathbf{y}_1^\top \mathbf{l} \\ \vdots \\ \mathbf{y}_m^\top \mathbf{l} \end{pmatrix} = \begin{pmatrix} \mathbf{y}_1^\top \\ \vdots \\ \mathbf{y}_m^\top \end{pmatrix} \mathbf{l} = \mathbf{Y}^\top \mathbf{l} = \mathbf{0}. \quad (3.29)$$

This means that \mathbf{l} lies in the null space of \mathbf{Y}^\top , but it also means that we can describe a necessary and sufficient condition for the existence of a line that intersects all m points as: \mathbf{Y} is rank deficient. An equivalent condition is: the columns of \mathbf{Y} are linearly dependent. Assuming that the points are distinct and co-linear, the null space of \mathbf{Y}^\top is 1-dimensional and corresponds to the homogeneous coordinates of \mathbf{l} . In the special case that $m = 3$, we can formulate the necessary and sufficient condition the existence of an intersecting line as: $\det(\mathbf{Y}) = 0$. In the general case, a necessary and sufficient condition for the existence of an intersecting plane is $\det(\mathbf{Y} \mathbf{Y}^\top) = 0$.

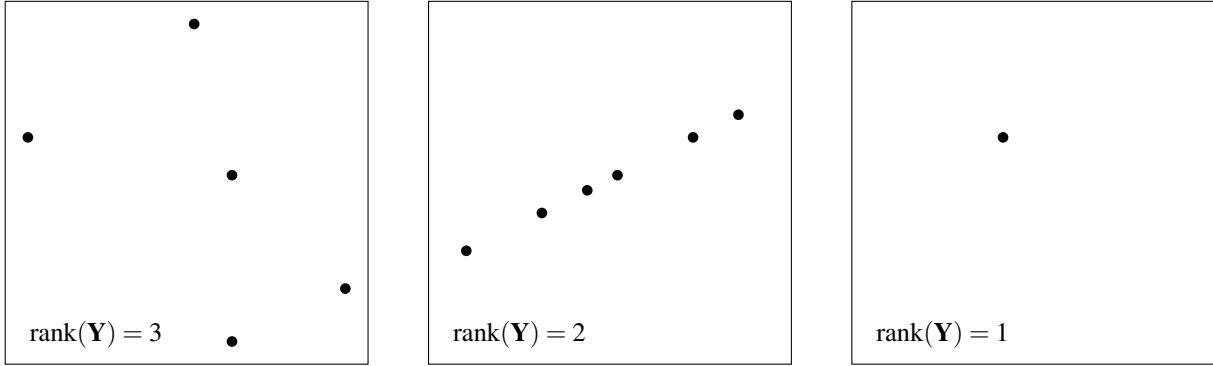


Figure 3.1: Different configuration of points correspond to different rank of the data matrix \mathbf{Y} , defined in Section 3.3.3. Left: The points are in general positions, corresponding to $\text{rank}(\mathbf{Y}) = 3$. Center: The points are on a line, corresponding to $\text{rank}(\mathbf{Y}) = 2$. Right: The points are identical, corresponding to $\text{rank}(\mathbf{Y}) = 1$.

A computational approach for solving the problem discussed here is to do a singular value decomposition³ of \mathbf{Y}^\top . The rank of \mathbf{Y} must be 1, 2, or 3, and the solution can be described by the following three cases:

- **Rank 3:** $\text{Null}(\mathbf{Y}^\top)$ is trivial, i.e., it only contains the null vector, and does not include the dual homogeneous coordinates of any line. This indicates that the points are not co-linear.
- **Rank 2:** $\text{Null}(\mathbf{Y}^\top)$ is 1-dimensional and corresponds to a unique projective element \mathbf{l} that solves Equation (3.29). This implies that the points are co-linear, and \mathbf{l} is a right singular vector of \mathbf{Y}^\top corresponding to the singular value = 0.
- **Rank 1:** $\text{Null}(\mathbf{Y}^\top)$ is 2-dimensional. This indicates that the points are not distinct, they refer to one and the same point. Consequently, there are infinitely many lines that intersect that single point. Their dual homogeneous coordinates span $\text{Null}(\mathbf{Y}^\top)$: any linear combination of two linearly independent null vectors of \mathbf{Y}^\top represents a valid line.

The three cases are illustrated in Figure 3.1. Instead of considering the right singular vectors of \mathbf{Y}^\top , we can use the left singular vectors of \mathbf{Y} .

For this particular problem, the solution is found in the null space of \mathbf{Y}^\top . This is a general observation.

3.6 The solution to many problems in geometry can be found in the null space of some matrix. This matrix is formed from information that usually is known, it will later be known as the *data matrix*.

As elements of a null space, a solution's existence and uniqueness are determined by the dimension of this space.

3.7 The solution is not only described by a set of parameters, it is also characterized by whether or not it exists, and if it is unique. This characterization is provided by the dimension of the null space of the data matrix.

Looking ahead

We need to apply these results with some care. In Section 12.3 we will make the conclusion that coordinates of points in an image cannot be determined with arbitrary accuracy. The same applies to the parameters of any geometric object that we get from measurements in an image. As a consequence, we need to accept the notion that co-linear points lie *approximately* on a line. If the points lie approximately on the line \mathbf{l} , then \mathbf{Y}^\top has a singular value that is *approximately* equal to zero. The corresponding right singular vector is *approximately* equal to \mathbf{l} .

³Singular value decomposition (SVD) is described in Toolbox Section 8.2.

Can we get away by replacing “equal to” with “approximately equal to”? The bad news, and you better get it already now, is that this may work, or maybe not. The problem is that in practice, inexact coordinates cause singular values that should be zero to become non-zero. Even if these non-zero values are small, they can be of the same magnitude as other singular values that have non-zero values in the ideal case. This confusion, in turn, can cause large errors in the estimated parameters, and makes the idea of using “approximately equal to” risky.

Being able to distinguish between singular values that are “non-zero and small” and those that are “equal to zero but disturbed by a small error” is non-trivial. It is one of the main issues when we look at the problem of estimating the parameters of various geometric objects in Part II. This is further discussed in Section 13.2.

3.3.4 Lines through a point, or not

The results of the previous section have a similar algebraic form if we instead consider a set of $m \geq 2$ lines: $\mathbf{l}_1, \dots, \mathbf{l}_m$ and ask the question: do they all intersect at a common point? And if they do, which point is it? The intersection point \mathbf{y} must satisfy

$$\begin{pmatrix} \mathbf{l}_1^\top \mathbf{y} \\ \vdots \\ \mathbf{l}_m^\top \mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{l}_1^\top \\ \vdots \\ \mathbf{l}_m^\top \end{pmatrix} \mathbf{y} = \tilde{\mathbf{Y}}^\top \mathbf{y} = \mathbf{0}, \quad (3.30)$$

where $3 \times m$ matrix $\tilde{\mathbf{Y}} = (\mathbf{l}_1 \dots \mathbf{l}_m)$ holds the dual homogeneous coordinates of the lines in its columns. A necessary and sufficient condition for the existence of a common point of intersection is that $\tilde{\mathbf{Y}}$ is rank deficient or, equivalently, the columns of $\tilde{\mathbf{Y}}$ are linearly dependent. In the case that $m = 3$, this condition is equivalent to $\det(\tilde{\mathbf{Y}}) = 0$, and in the general case to $\det(\tilde{\mathbf{Y}} \tilde{\mathbf{Y}}^\top) = 0$.

In the case that \mathbf{y} exists, it is found in $\text{Null}(\tilde{\mathbf{Y}})$ which, in turn, can be determined from an SVD of $\tilde{\mathbf{Y}}^\top$. There are 3 possible cases, depending on the rank of $\tilde{\mathbf{Y}}^\top$:

- **Rank 3:** $\text{Null}(\tilde{\mathbf{Y}}^\top)$ is trivial and does not contain the homogeneous coordinates of any point. This indicates that the lines do not intersect at a common point.
- **Rank 2:** $\text{Null}(\tilde{\mathbf{Y}}^\top)$ is 1-dimensional and corresponds to a unique projective element \mathbf{y} that satisfies Equation (3.30). This implies that the lines intersect at a common point, and the projective element \mathbf{y} is unique and represented by a right singular vector of $\tilde{\mathbf{Y}}^\top$ corresponding to the singular value = 0.
- **Rank 1:** $\text{Null}(\tilde{\mathbf{Y}}^\top)$ is 2-dimensional. This indicates that the lines are not distinct, they refer to one and the same line. Consequently, there are infinitely many points that lie on that single line, and their homogeneous coordinates span $\text{Null}(\tilde{\mathbf{Y}}^\top)$: any linear combination of two linearly independent null vectors of $\tilde{\mathbf{Y}}^\top$ represents a valid point.

The three cases are illustrated in Figure 3.2. The discussion at the end of Section 3.3.3 applies also here: we can consider the left singular vectors of $\tilde{\mathbf{Y}}$ instead of the right singular vectors of $\tilde{\mathbf{Y}}^\top$, and in practice it is not always clear what we mean by a singular value equal to zero when the line parameters can have inaccuracies.

3.4 Operations on normalized homogeneous coordinates

One of the most basic operations in geometry is to determine the distance between different objects. Another is to determine the area of a region. We can determine both distances and areas of basic geometric objects in a simple way, when points and lines appear in normalized homogeneous forms. Also, the order in which triplets of points appear, as well as the problem of determining whether a point lies inside or outside a specific region can be solved using these homogeneous forms. How to do that is the topic of this section.

3.4.1 Distance between two points

Let $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$ be the Cartesian coordinates of two points in \mathbb{E}^2 . How do we determine the distance between them if instead only their homogeneous coordinates \mathbf{y}_1 and \mathbf{y}_2 are known? To answer this question, let us first make some observations. Recall that \mathbf{y}_1 and \mathbf{y}_2 are representatives of projective elements, and there is no notion of distance between elements of a projective space. Furthermore, each of \mathbf{y}_1 and \mathbf{y}_2 equals the corresponding canonical form

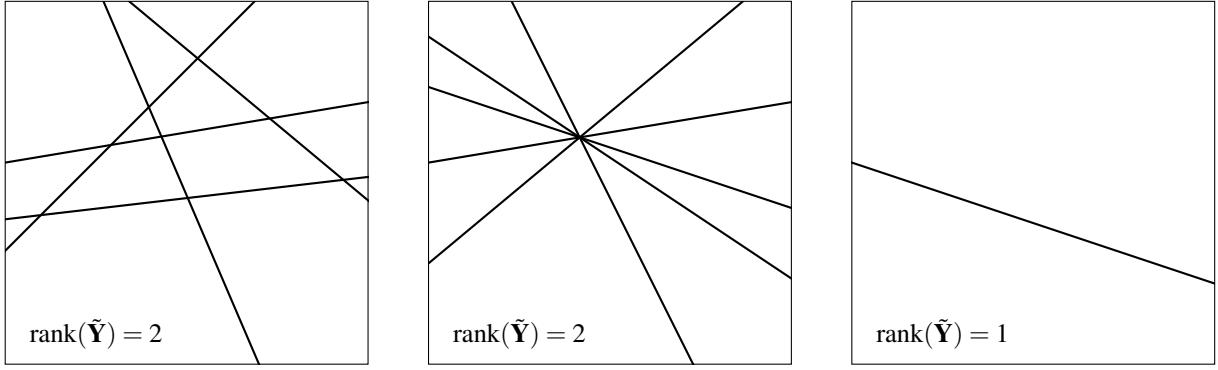


Figure 3.2: Different configuration of lines correspond to different rank of the data matrix $\tilde{\mathbf{Y}}$, defined in Section 3.3.4. Left: The lines have general positions and orientations, corresponding to $\text{rank}(\tilde{\mathbf{Y}}) = 3$. Center: The lines are intersecting at a single point, corresponding to $\text{rank}(\tilde{\mathbf{Y}}) = 2$. Right: The lines are identical, corresponding to $\text{rank}(\tilde{\mathbf{Y}}) = 1$.

multiplied by an individual scalar. This means that it is not meaningful to consider the distance between \mathbf{y}_1 and \mathbf{y}_2 in \mathbb{R}^3 since it depends also on these individual scalings. But we can bring \mathbf{y}_1 and \mathbf{y}_2 to its canonical form by applying a P-normalization on each of them, which assures that the first two elements are $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$, respectively, and the last element in both vectors are = 1.

The last observation implies that we can extract $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$ using the cart function, Equation (3.9), and then compute the distance in \mathbb{E}^2 as in Equation (2.4). It should also be clear that this distance, in \mathbb{R}^2 , is the same as the distance between P-normalized vectors \mathbf{y}_1 and \mathbf{y}_2 , in \mathbb{R}^3 . This gives the same distance since the normalization set both of the third elements = 1. To summarize: we can determine the distance between two points in \mathbb{E}^2 , represented by homogeneous coordinates \mathbf{y}_1 and \mathbf{y}_2 , as the “point-to-point” distance function, denoted d_{PP} :

$$d_{\text{PP}}(\mathbf{y}_1, \mathbf{y}_2) = \|\text{cart}(\mathbf{y}_1) - \text{cart}(\mathbf{y}_2)\| = \|\text{norm}_{\text{P}}(\mathbf{y}_1) - \text{norm}_{\text{P}}(\mathbf{y}_2)\|. \quad (3.31)$$

3.4.2 Distance between a point and a line

In Section 2.4.3 we considered a point in \mathbb{E}^2 , represented by the vector $\bar{\mathbf{y}} \in \mathbb{R}^2$, and a line with parameters $(\hat{\mathbf{l}}, \Delta)$. We derived an expression for the distance between the point and the line, as in Equation (2.34). If the point and the line instead have homogeneous representations, \mathbf{y} and \mathbf{l} , we can apply P-normalization of \mathbf{y} and D-normalization of \mathbf{l} . This produces the proper coordinates of the point and parameters of the line, and plugged into Equation (2.34) they give us the distance.

Based on homogeneous representations, it is possible to go one step further. After the normalizations, we get

$$\text{norm}_{\text{P}}(\mathbf{y}) = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}, \quad \text{norm}_{\text{D}}(\mathbf{l}) = \begin{pmatrix} \hat{\mathbf{l}} \\ -\Delta \end{pmatrix} = \begin{pmatrix} l_1 \\ l_2 \\ -\Delta \end{pmatrix}. \quad (3.32)$$

We assume that the normal vector of the line, $\hat{\mathbf{l}} = (l_1, l_2)$, has unit norm, and $\Delta \geq 0$, which leads to

$$\text{norm}_{\text{P}}(\mathbf{y}) \cdot \text{norm}_{\text{D}}(\mathbf{l}) = l_1 u + l_2 v - \Delta = \bar{\mathbf{y}} \cdot \hat{\mathbf{l}} - \Delta. \quad (3.33)$$

In combination with Equation (2.34), this allows us to express the distance between a point \mathbf{y} and a line \mathbf{l} in \mathbb{E}^2 as the scalar product between the normalized versions of these two vectors. We sometimes refer to this function as the “point-to-line” distance, denoted d_{PD} , defined as

$$d_{\text{PD}}(\mathbf{y}, \mathbf{l}) = |\text{norm}_{\text{P}}(\mathbf{y}) \cdot \text{norm}_{\text{D}}(\mathbf{l})|. \quad (3.34)$$

Hence, a convenient way to compute the distance between a point and a line is to first normalize the respective homogeneous representations. Then we compute the scalar product of the resulting vectors and, finally, take the absolute value of the result.

The function d_{PD} provides the distance between a *point* and its *dual* counterpart, in this case a line. As we will see in Chapter 5, the dual counterpart of a point in \mathbb{E}^3 is instead a plane, and the corresponding function gives the distance between a point and a plane.

3.8 It is essential to apply appropriate normalization of homogeneous representations in order to determine Euclidean distances.

Signed distance between a point and a line

In the case that $\Delta \neq 0$, it must be the case that

$$s(\mathbf{y}, \mathbf{l}) = \text{norm}_P(\mathbf{y}) \cdot \text{norm}_D(\mathbf{l}) = \bar{\mathbf{y}} \cdot \hat{\mathbf{l}} - \Delta \begin{cases} < 0 & \text{when } \bar{\mathbf{y}} \text{ and } \mathbf{0} \text{ lie on the same side of the line,} \\ = 0 & \text{when } \bar{\mathbf{y}} \text{ lie on the line,} \\ > 0 & \text{when } \bar{\mathbf{y}} \text{ and } \mathbf{0} \text{ lie on different sides of the line.} \end{cases} \quad (3.35)$$

These different cases are illustrated in Figure 3.3. In the case that $\Delta = 0$, we leave the question of how $\bar{\mathbf{y}}$ relates to the origin and the line undetermined. We refer to the function s in Equation (3.35) as the *signed distance* between a point and a line in \mathbb{E}^2 .

Based on this relation between a point and a line, for two points \mathbf{y}_1 and \mathbf{y}_2 , and a line \mathbf{l} , we can make the following two statements:

$$\begin{aligned} \text{sign}(s(\mathbf{y}_1, \mathbf{l})) = \text{sign}(s(\mathbf{y}_2, \mathbf{l})) &\Leftrightarrow \bar{\mathbf{y}}_1 \text{ and } \bar{\mathbf{y}}_2 \text{ lie on the same side of the line,} \\ &\quad \text{and} \end{aligned} \quad (3.36)$$

$$\text{sign}(s(\mathbf{y}_1, \mathbf{l})) = -\text{sign}(s(\mathbf{y}_2, \mathbf{l})) \Leftrightarrow \bar{\mathbf{y}}_1 \text{ and } \bar{\mathbf{y}}_2 \text{ lie on different sides of the line.}$$

This relation between two points and a line, without any reference to the origin, is useful in certain applications when we need to determine how pairs of points relate to one another and to a line. See Figure 3.4 for an illustration.

Looking ahead

Being able to determine the distance between a point and a line becomes handy in Part II, for example when we want to estimate a line that lies as close as possible to a fixed set of points. In this case, we can try to determine the line \mathbf{l} such that $d_{PD}(\mathbf{y}_k, \mathbf{l})$ is as small as possible for all points \mathbf{y}_k . We have seen that these distances correspond to scalar products, $\mathbf{y}_k \cdot \mathbf{l}$. But to assure that we get the Euclidean distance, \mathbf{y} must be P-normalized and \mathbf{l} be D-normalized. These normalizations amount to an individual scalar multiplied onto each of the vectors \mathbf{y}_k and \mathbf{l} , corresponding to a scalar multiplied onto $\mathbf{y}_k \cdot \mathbf{l}$. So, if the Euclidean distances are scalar multiples of $\mathbf{y}_k \cdot \mathbf{l}$, why not minimize these scalar products, without taking the normalizations into account? Skipping normalization simplifies the computations, and could still lead to the same result.

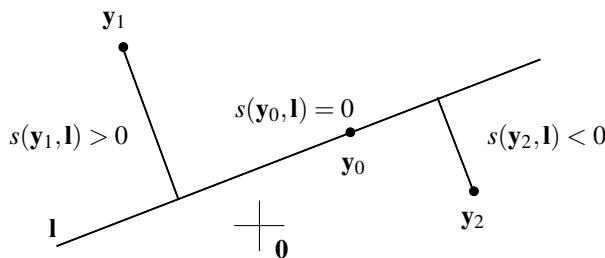


Figure 3.3: The signed distance, s , between a point and a line \mathbf{l} has a sign that depends on whether the point and the origin $\mathbf{0}$ are on the same side of the line, or not.

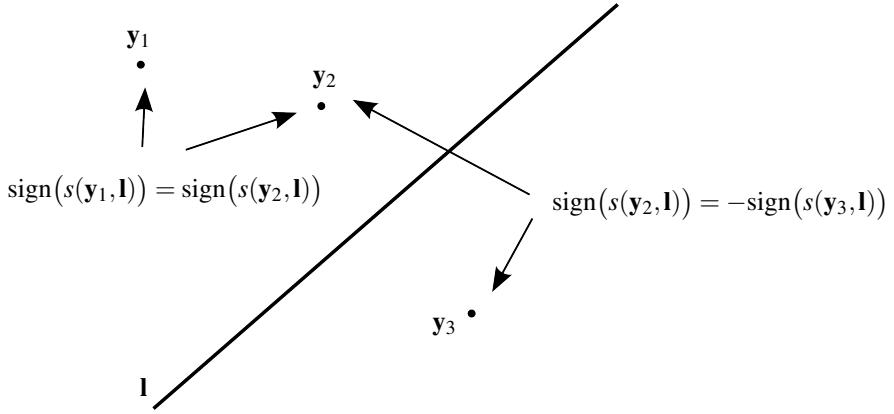


Figure 3.4: The signed distance between a fixed line and various points, s , has the same sign as long as the point stays on the same side of the line. When the point crosses the line, s changes sign.

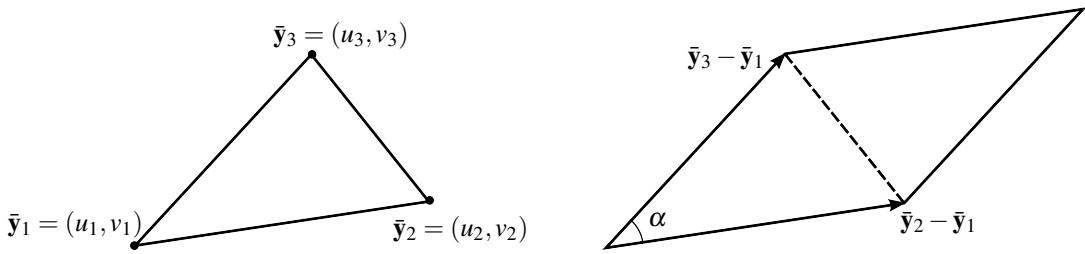


Figure 3.5: Left: a triangle defined by the three points $\bar{y}_1, \bar{y}_2, \bar{y}_3$. Right: a parallelepiped with twice the area of the triangle.

Measuring the closeness between a point \mathbf{y}_k and a line \mathbf{l} as $\mathbf{y}_k \cdot \mathbf{l}$, without normalizations, is not wrong. As it turns out, plain scalar products form the basis for a wide range of estimation method in geometry. But when we minimize these scalar products we must always keep in mind that we are *not* minimizing Euclidean distances. Since $\mathbf{y} \cdot \mathbf{l}$ corresponds to some scalar multiplied with a Euclidean distance, it is easy to make it smaller, or larger, by a scaling of either \mathbf{y} or \mathbf{l} , without changing the Euclidean distance. This means that we should not, in general, treat $\mathbf{y} \cdot \mathbf{l}$ as a distance, at least in a geometric meaning. Instead, it can be described as an *algebraic distance*, or *algebraic error* in the context of estimation. We will return to this issue in Chapter 12.

3.4.3 Area of a triangle

Consider the triangle illustrated in Figure 3.5, left, defined by the three corner points $\bar{y}_1, \bar{y}_2, \bar{y}_3$. We want to determine the area of this triangle, denoted A , equal to half the area of the parallelogram to the right in the same figure. Basic trigonometry allows us to express this latter area A' as

$$A' = \|\bar{y}_2 - \bar{y}_1\| \|\bar{y}_3 - \bar{y}_1\| \sin \alpha, \quad (3.37)$$

where α is the angle at corner point \bar{y}_1 . Here, we must assume that α is defined such that $\sin \alpha \geq 0$, i.e., $0 \leq \alpha \leq \pi$. Assuming P-normalized homogeneous coordinates, it then follows that

$$A' = \|\mathbf{y}_2 - \mathbf{y}_1\| \|\mathbf{y}_3 - \mathbf{y}_1\| \sin \alpha. \quad (3.38)$$

• See Toolbox Section 3.7.2 for a presentation of the cross product.

Based on the properties of the cross product defined for vectors in \mathbb{R}^3 , the right-hand side of Equation (3.38) can also be expressed as:

$$A' = \|(\mathbf{y}_2 - \mathbf{y}_1) \times (\mathbf{y}_3 - \mathbf{y}_1)\| = \|\mathbf{y}_1 \times \mathbf{y}_2 + \mathbf{y}_2 \times \mathbf{y}_3 + \mathbf{y}_3 \times \mathbf{y}_1\|, \quad (3.39)$$

Thus, a general expression for the area A is given as

$$A = \frac{1}{2} \|\text{norm}_P(\mathbf{y}_1) \times \text{norm}_P(\mathbf{y}_2) + \text{norm}_P(\mathbf{y}_2) \times \text{norm}_P(\mathbf{y}_3) + \text{norm}_P(\mathbf{y}_3) \times \text{norm}_P(\mathbf{y}_1)\|. \quad (3.40)$$

An alternative to using cross products to determine A is to compute the determinant of a 3×3 matrix that holds the P-normalized homogeneous coordinates of the points in its columns:

$$A = \frac{1}{2} \left| \det \begin{pmatrix} \text{norm}_P(\mathbf{y}_1) & \text{norm}_P(\mathbf{y}_2) & \text{norm}_P(\mathbf{y}_3) \end{pmatrix} \right|. \quad (3.41)$$

A third option is to express A directly in the Cartesian coordinates of the three points:

$$A = \frac{1}{2} |(u_1 - u_3)(v_2 - v_1) - (u_1 - u_2)(v_3 - v_1)|. \quad (3.42)$$

The equivalence of Equations (3.40), (3.41) and (3.42) is left as an exercise to the reader.

Order of corner points and directed lines

The results derived above imply that

$$\text{norm}_P(\mathbf{y}_1) \times \text{norm}_P(\mathbf{y}_2) + \text{norm}_P(\mathbf{y}_2) \times \text{norm}_P(\mathbf{y}_3) + \text{norm}_P(\mathbf{y}_3) \times \text{norm}_P(\mathbf{y}_1) = \begin{pmatrix} 0 \\ 0 \\ \pm 2A \end{pmatrix}. \quad (3.43)$$

The important issue in the following discussion is not the area A , but rather the sign that appears in the above equation. Recall that the cross product $\mathbf{a} \times \mathbf{b}$ of two vectors \mathbf{a} and \mathbf{b} is defined in accordance with the right-hand rule. Hence, if the three corner points $\bar{\mathbf{y}}_1, \bar{\mathbf{y}}_2, \bar{\mathbf{y}}_3$ of the triangle in Figure 3.5 appear in counter-clockwise order, then $(\mathbf{y}_2 - \mathbf{y}_1) \times (\mathbf{y}_3 - \mathbf{y}_1)$ points in the same direction as the third axis in \mathbb{R}^3 . Vice versa, if the three points appear in clockwise order, then the two vectors point in opposite directions. Consequently, the sign of

$$\det \begin{pmatrix} \text{norm}_P(\mathbf{y}_1) & \text{norm}_P(\mathbf{y}_2) & \text{norm}_P(\mathbf{y}_3) \end{pmatrix} = (\text{norm}_P(\mathbf{y}_1) \times \text{norm}_P(\mathbf{y}_2)) \cdot \text{norm}_P(\mathbf{y}_3) \quad (3.44)$$

reflects the order of the three points.

Based on this results, and given two points \mathbf{y}_1 and \mathbf{y}_2 , it makes sense to define a vector $\mathbf{l} \in \mathbb{R}^3$ (for example) as

$$\mathbf{l} = \text{norm}_P(\mathbf{y}_1) \times \text{norm}_P(\mathbf{y}_2) \quad (3.45)$$

and use the sign of $\mathbf{l} \cdot \text{norm}_P(\mathbf{y}_3)$ to check the order of the three points. \mathbf{l} is a representative of the dual homogeneous coordinates of the line passing through \mathbf{y}_1 and \mathbf{y}_2 , but as vector it is defined in such a way that

$$\mathbf{l} \cdot \text{norm}_P(\mathbf{y}_3) \begin{cases} < 0, & \text{if } \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3 \text{ appear in clockwise order,} \\ = 0, & \text{if the three points are collinear,} \\ > 0, & \text{if } \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3 \text{ appear in counter-clockwise order.} \end{cases} \quad (3.46)$$

Intuitively, we can think of \mathbf{l} in Equation (3.45) as describing a *directed line* in \mathbb{E}^2 , going from \mathbf{y}_1 to \mathbf{y}_2 . With this description in mind, an alternative formulation of Equation (3.46) is

$$\mathbf{l} \cdot \text{norm}_P(\mathbf{y}_3) \begin{cases} < 0, & \text{if } \mathbf{y}_3 \text{ lies on the right side of the line,} \\ = 0, & \text{if } \mathbf{y}_3 \text{ lies on the line,} \\ > 0, & \text{if } \mathbf{y}_3 \text{ lies on the left side of the line,} \end{cases} \quad (3.47)$$

where “right” and “left” are relative to the direction of the line. See Figure 3.6.

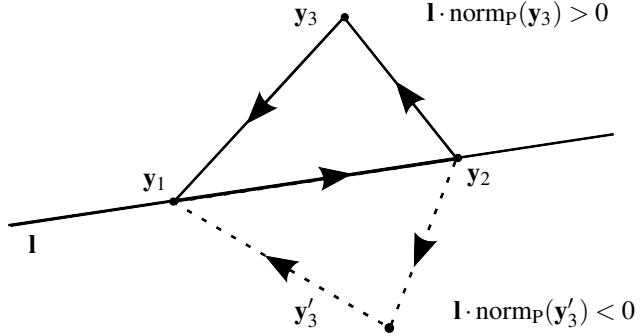


Figure 3.6: An illustration of the directed line \mathbf{l} , from point \mathbf{y}_1 to \mathbf{y}_2 .

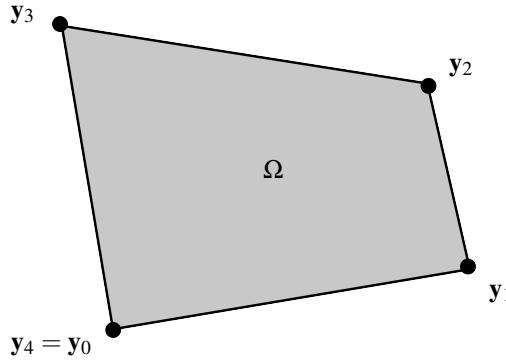


Figure 3.7: A convex region Ω bounded by a polygon chain. It is defined in terms of a sequence of vertex points, here: $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4 = \mathbf{y}_0$. The points are enumerated in counter-clockwise order around the boundary.

Inside or outside?

A practical application of these results is to determine if a point \mathbf{y} lies inside or outside a region Ω bounded by a polygon chain. With the additional assumption that Ω is convex, i.e., any line intersects the boundary of Ω at most twice, a simple solution exists to this problem. In this context, we assume that Ω contains its boundary points.

Let Ω be a convex region bounded by a polygon chain. It consists of a sequence of n vertex points $\mathbf{y}_k, k = 1, \dots, m$. We have here enumerated the points in the order they appear along the boundary in counter-clockwise order. We also add an extra point \mathbf{y}_0 equal to the last point in the sequence: $\mathbf{y}_0 = \mathbf{y}_m$. See Figure 3.7. From the earlier results follows that

$$\bar{\mathbf{y}} \in \Omega \Leftrightarrow (\text{norm}_P(\mathbf{y}_{k-1}) \times \text{norm}_P(\mathbf{y}_k)) \cdot \mathbf{y} \geq 0 \quad \text{for all } k = 1, \dots, m. \quad (3.48)$$

This means that \mathbf{y} lies on the boundary of Ω if and only if the scalar product vanishes for some k . This method is summarized in Algorithm 3.1.

3.4.4 Concluding remarks

This section has highlighted some applications based on the normalized forms of homogeneous coordinates. These normalizations are important for certain operations, but, unless necessary, we should avoid these normalizations. As will be illustrated in the rest of Part I, it is possible to do quite a lot of geometric operations using homogeneous representations without applying normalizations. At some point in the computations we may need some Euclidean quantity, such as a distance or an area. Only at that point does normalization make sense. Hence, a common approach is to first transfer the problem at hand to a homogeneous representation. By applying suitable algebraic manipulations, we can determine some interesting geometric object, e.g., a point or a line. Then, and only then, do

Algorithm 3.1: Determine if a 2D point \mathbf{y} lies inside a convex region Ω bounded by a polygonal chain.

Input: A sequence of m vertex points $\mathbf{y}_k, k = 1, \dots, m$, which define the boundary of Ω .

Input: A point \mathbf{y} that is to be tested if $\bar{\mathbf{y}} \in \Omega$.

Output: A boolean (*TRUE* or *FALSE*) variable: RESULT.

```

1 Set  $\mathbf{y}_0 = \mathbf{y}_n$ 
2 Set RESULT = TRUE
3 foreach  $k = 1, \dots, m$  do
4     Set  $\mathbf{l} = \text{normp}(\mathbf{y}_{k-1}) \times \text{normp}(\mathbf{y}_k)$ . // Directed line from  $\mathbf{y}_{k-1}$  to  $\mathbf{y}_k$ 
5     if  $\mathbf{l} \cdot \text{normp}(\mathbf{y}) < 0$  then
6         Set RESULT = FALSE
7     end
8 end
```

we apply the proper normalizations to get the interesting quantity.

3.9 Do not apply normalizing operations on homogeneous representations of point or line, unless it is required in order to obtain some Euclidean quantity, such as a distance or an area.

One reason for not normalizing after every step, is that some intermediate results may be in the form of points or lines at infinity. These are defined in the following section.

3.5 Points and lines at infinity

In Euclidean geometry, all points and lines lie at finite distance from each other. In particular, they lie at finite distance from any point chosen as the origin. In the following presentation, we refer to these points as *proper points* and the line are *proper lines*. As we will see in this section, the homogeneous representations introduced in this chapter open for the possibility of defining what we mean by a point or a line that lie at infinite distance from the origin. They appear from the usual incidence relations discussed above, as special cases of homogeneous coordinates. An advantage is that we can treat them as any other homogeneous coordinates, except that normalization does not apply. Once points or lines at infinity become familiar concepts, we can even start to use these objects in the same way as the proper ones. Both types of homogeneous coordinates can be plugged into the same expressions.

Even though we refer to these new objects as “points” or “lines”, they have little in common with their Euclidean counterparts. You could think of them as extensions of Euclidean geometry, based on homogeneous representations.

3.5.1 Points at infinity

In the case that two distinct lines are parallel, their point of intersection is undefined. To confirm this fact, let us investigate two lines with D-normalized dual homogeneous coordinates,

$$\mathbf{l}_1 = \begin{pmatrix} l_1 \\ l_2 \\ -\Delta_1 \end{pmatrix}, \quad \mathbf{l}_2 = \begin{pmatrix} l_1 \\ l_2 \\ -\Delta_2 \end{pmatrix}, \quad (3.49)$$

and their point of intersection. The two lines have a common normal vector $\hat{\mathbf{l}} = (l_1, l_2)$, and distances Δ_1, Δ_2 to the origin. If we insist $\Delta_1 \geq 0$ and $\Delta_2 \geq 0$, this excludes the possibility that the two lines can lie on each side of the origin. To allow also this possibility, Δ_1 and Δ_2 can be either positive or negative, the important fact is that the two lines share a common normal vector $\hat{\mathbf{l}}$. In accordance with Section 3.3.2, the point of intersection is given as:

$$\mathbf{y}_0 \sim \mathbf{l}_1 \times \mathbf{l}_2 = (\Delta_1 - \Delta_2) \begin{pmatrix} -l_2 \\ l_1 \\ 0 \end{pmatrix}. \quad (3.50)$$

The vector \mathbf{y}_0 cannot be P-normalized to represent any point in \mathbb{E}^2 . With this observation in mind, it is still clear that \mathbf{y}_0 in Equation (3.50) is a well-defined element of $P(\mathbb{R}^3)$ as long as $\Delta_1 \neq \Delta_2$. Can we give it some useful interpretation? Let us start with two non-parallel lines, where the angles α_1 and α_2 define the corresponding normal vectors:

$$\mathbf{l}_1 = \begin{pmatrix} \cos \alpha_1 \\ \sin \alpha_1 \\ -\Delta_1 \end{pmatrix}, \quad \mathbf{l}_2 = \begin{pmatrix} \cos \alpha_2 \\ \sin \alpha_2 \\ -\Delta_2 \end{pmatrix}. \quad (3.51)$$

The intersecting point is

$$\mathbf{y}_0 \sim \mathbf{l}_1 \times \mathbf{l}_2 = \begin{pmatrix} -\Delta_2 \sin \alpha_1 + \Delta_1 \sin \alpha_2 \\ \Delta_2 \cos \alpha_1 - \Delta_1 \cos \alpha_2 \\ \sin(\alpha_2 - \alpha_1) \end{pmatrix}. \quad (3.52)$$

We now let one of the two lines, say \mathbf{l}_1 , be fixed, while \mathbf{l}_2 rotates to become more and more parallel to \mathbf{l}_1 , without actually making it happen. In this process the first two elements of the right-hand side of Equation (3.52) are relatively stable, they will approach the value $(\Delta_1 - \Delta_2)(\sin \alpha_1, -\cos \alpha_1)$. But the third element goes toward zero. After the final P-normalization to get the Cartesian coordinates of the point of intersection, $\bar{\mathbf{y}}_0$, this limit process implies that $\bar{\mathbf{y}}_0$ will move along the line \mathbf{l}_1 , further and further away from the origin. For the case that the lines really become parallel we can intuitively think of the intersection $\bar{\mathbf{y}}_0$ as moved all the way to an infinite distance from the origin, but still lying on the line \mathbf{l}_1 . See Figure 3.8 for an illustration. As a limit, it is then natural to interpret Equation (3.50) in terms of a *point at infinity*, which lies at infinite distance from the origin.

Intuition fails

This intuitive interpretation of Equation (3.50) is not rock solid. If \mathbf{l}_1 is fixed and we rotate \mathbf{l}_2 to make it more and more parallel to \mathbf{l}_1 , we can make $\bar{\mathbf{y}}_0$ disappear to infinite distance in two opposite directions. These directions are given by the tangent of the first line, which can be either $(l_2, -l_1)$ or $(-l_2, l_1)$. This means that when we think of Equation (3.50) as a representation of a point at infinite distance, we cannot distinguish between the case that it lies at infinite distance in the direction given by $(l_2, -l_1)$ or by $(-l_2, l_1)$. At infinity, points at “opposite directions” are equal. Instead, we can identify a point at infinity with the *orientation of a line*. The point corresponds to a tangent vector of the line, disregarding the tangent’s sign.

We can take two distinct and parallel lines and follow one of them in either direction, all the way to “infinity” to find the intersection, a point at infinity. But the projective element which represents this point, Equation (3.50), is independent of the parameters Δ_1 or Δ_2 , as long as they are distinct. Hence, we get same point at infinity regardless of which line we follow, or in which direction we go. Moving the two lines closer or further apart still give the same point at infinity.

3.10 Intuitively, it may help to think of a point at infinity not as a point, but as an orientation of a line in \mathbb{E}^2 .

We can extend this result to three distinct and parallel lines, $\mathbf{l}_1, \mathbf{l}_2$, and \mathbf{l}_3 . In accordance with the discussion

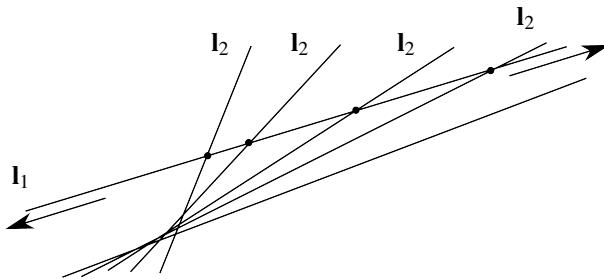


Figure 3.8: A fixed line, \mathbf{l}_1 , and a second line \mathbf{l}_2 , which is becoming more and more parallel to \mathbf{l}_1 . As a consequence, the intersection points move to an infinite distance from all other points in \mathbb{E}^2 . The same point at infinity is obtained if the intersection points lie to the left side and is moved further and further to the left.

above, the intersection of \mathbf{l}_1 and \mathbf{l}_2 is a point at infinity. Also \mathbf{l}_2 and \mathbf{l}_3 intersect at a point at infinity. Less obvious from an intuitive perspective is that all three lines intersect at the same point at infinity. See Figure 3.9 for an illustration. A more general statement can be formulated as:

3.11 Any set of distinct and parallel lines intersect at a single point, a point at infinity.

We summarize the discussion so far. The homogeneous representations of points and lines suggest in a natural way that we can interpret the intersection of two distinct and parallel lines as a point at infinite distance from the origin, in fact at infinite distance from any point in \mathbb{E}^2 . In doing so, we must equate points at opposite directions, and rather think of a point at infinity as the orientation of a line. The corresponding homogeneous representation for such a point is given in Equation (3.50).

In the literature, a proper point is sometimes called an *affine point*, while a point at infinity is referred to as an *ideal point*, an *affine vector*, or a *direction*.

3.5.2 The line at infinity

In the previous section we said that also points at infinity give correct results when used in computations. Let us verify this statement by determining the intersecting line of two points, \mathbf{y}_1 and \mathbf{y}_2 , where one of them is a point at infinity:

$$\mathbf{y}_1 \sim \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix}, \quad \mathbf{y}_2 \sim \begin{pmatrix} u_2 \\ v_2 \\ 0 \end{pmatrix}. \quad (3.53)$$

Here we must assume that $(u_2, v_2) \neq (0, 0)$, otherwise \mathbf{y}_2 is not a representative of a proper projective element. If we do some intuitive reasoning about the expected result, it should be a line that passes through the proper point \mathbf{y}_1 and then continues in a direction to make it intersect with \mathbf{y}_2 , at infinity. To do so, (u_2, v_2) must be a tangent vector of the line, but disregarding the sign. Disregarding the sign means that the resulting line must continue in two opposite directions from \mathbf{y}_1 .

Let us now compute the intersecting line:

$$\mathbf{l} \sim \mathbf{y}_1 \times \mathbf{y}_2 = \begin{pmatrix} -v_2 \\ u_2 \\ u_1 v_2 - u_2 v_1 \end{pmatrix}. \quad (3.54)$$

This is a line with a normal vector given by $(-v_2, u_2)$, i.e., (u_2, v_2) is a tangent vector. It is a straightforward exercise to verify that the line intersects \mathbf{y}_1 . This means that the resulting line meets our intuitive expectations. It also means that we can choose at least one of the two points to be at infinity in Equation (3.25).

Another aspect of this result is that any proper line, with D-normalized parameters $\mathbf{l} = (l_1, l_2, \Delta)$, must intersect

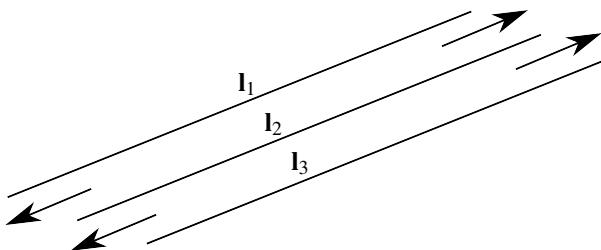


Figure 3.9: Three parallel lines, $\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3$. All three lines intersect at a common point at infinity, in the directions indicated by the arrows.

exactly one point at infinity. More precisely, consider the point

$$\mathbf{y} = \begin{pmatrix} l_2 \\ -l_1 \\ 0 \end{pmatrix}. \quad (3.55)$$

This is a point at infinity, and it is also the case that $\mathbf{y} \cdot \mathbf{l} = 0$, so this point intersects with the line \mathbf{l} . As we have already described, \mathbf{y} acts as the intersecting point between the line \mathbf{l} and any other line that is parallel to \mathbf{l} .

What about the intersecting line of two points at infinity? As usual we assume that the two points are distinct, but now both are points at infinity:

$$\mathbf{y}_1 = \begin{pmatrix} u_1 \\ v_1 \\ 0 \end{pmatrix}, \quad \mathbf{y}_2 = \begin{pmatrix} u_2 \\ v_2 \\ 0 \end{pmatrix}. \quad (3.56)$$

Again, we allow ourselves some intuitive speculations about the nature of the line that intersects two points at infinity. It seems reasonable to expect that a line which passes through points that are at “infinite” distance from the origin, itself is a line at infinite distance. We do the math:

$$\mathbf{l} \sim \mathbf{y}_1 \times \mathbf{y}_2 = \begin{pmatrix} 0 \\ 0 \\ u_1 v_2 - u_2 v_1 \end{pmatrix}. \quad (3.57)$$

This is a well-defined projective element of $P(\mathbb{R}^3)$, but it cannot be D-normalized to give the corresponding line parameters. The normalization would scale \mathbf{l} to make the norm of the first two elements = 1, but that implies an infinite scaling in this case. This, in turn, makes Δ , the distance to the origin, take an infinite value. This fits well with our intuitive characterization of the line \mathbf{l} , and means that there is only one such *line at infinity*. It intersects with *all* points at infinity, and has a canonical dual homogeneous representation as

$$\mathbf{l}_\infty = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \quad (3.58)$$

Based on this observation, we extend the set of proper lines in \mathbb{E}^2 with the single line at infinity.

As a result of this extension, we can now interpret every non-zero vector in \mathbb{R}^3 as the homogeneous representation of a point or of a line. Some of them will be at infinity, others are proper points or lines. Since neither the line at infinity, nor points at infinity, can be properly normalized, it is not meaningful to consider distances or other Euclidean quantities related to them.

3.5.3 Projective geometry and the projective plane

The geometry that we learned up to secondary school is based on Euclidean spaces. In the 2D case, it forms a plane that extends without limits. This limitless, however, does not imply that it contains points at infinity. In a Euclidean space, there is a finite distance between any two points and, in particular, parallel lines do not intersect.

The proper points and the ideal points at infinity are distinct sets of points. A new type of geometry, *projective geometry*, can be formed by considering the union of the two point-sets.

Definition 3.1: Projective Plane

The *projective plane* is the union of \mathbb{E}^2 , consisting of the proper Euclidean points, and the set of ideal points at infinity. The projective plane is denoted as \mathbb{P}^2 .

We will not make presentation of projective geometry here, for a formal treatment of the subject see [15]. There, the projective plane is constructed using an axiomatic approach rather than as the union of proper and ideal points. Here, we can often take a similar position: a point is simply any point in the projective plane, and we need not specify if it is a proper or an ideal point. Only when Cartesian coordinates are required, do we need to worry about not having to deal with ideal points.

Not only are the axioms of projective spaces blind to proper points versus points at infinity. Many familiar concepts in Euclidean geometry, e.g., distances, angles, or one point lying between two other points on a line, vanish completely in projective geometry. The main concepts in projective geometry is instead intersection of lines and co-linearity of points, projections, and transformations under which these concepts are invariant.

At the cost of ridding ourselves of the familiar Euclidean concepts, projective geometry offers an advantage in the form of *completeness*: all theorems can be presented without “special cases”. For example, in projective geometry it is always true that two distinct lines intersect in a one unique point. In Euclidean geometry this is true only if the lines are not parallel. This may seem like a small feat, but it becomes much more relevant when we extend projective geometry to the 3D case, and start to look at projections from 3D space to images.

In a similar way as we use Cartesian coordinates to represent points in the Euclidean plane, homogeneous coordinates are used to represent points in the projective plane. A practical detail in this matter is that homogeneous coordinates allow us to distinguish between proper points and points at infinity (by looking at the third element). This means that we can still apply the familiar methods and concepts from Euclidean geometry to proper points, using P-normalization, while we have to be more careful as soon as points at infinity are involved. However, as long as we are using homogeneous coordinates in our computations to determine a point or a line, or any other geometric object, we need not distinguish between the two classes of points.

3.6 Parametric representations of lines

Section 2.2 presents two alternative parametric representations of a line in \mathbb{E}^2 , described in Equations (2.10) and (2.13). We will now put these representations in the context of homogeneous coordinates and extend them to \mathbb{P}^2 . Starting with the first form, Equation (2.10), the homogeneous coordinates of any point $\bar{\mathbf{y}}$ on the line become

$$\mathbf{y}(s) \sim \begin{pmatrix} \bar{\mathbf{y}}(s) \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} \bar{\mathbf{y}}_0 + s\bar{\mathbf{t}} \\ 1 \end{pmatrix}}_{:=\mathbf{y}_0} = \underbrace{\begin{pmatrix} \bar{\mathbf{y}}_0 \\ 1 \end{pmatrix}}_{:=\mathbf{y}_0} + s \underbrace{\begin{pmatrix} \bar{\mathbf{t}} \\ 0 \end{pmatrix}}_{:=\mathbf{t}} = \mathbf{y}_0 + s\mathbf{t}. \quad (3.59)$$

Here, $\bar{\mathbf{y}}_0 \in \mathbb{R}^2$ is a fixed point on the line, $\bar{\mathbf{t}} \in \mathbb{R}^2$ is a tangent vector of the line, and $s \in \mathbb{R}$. The right-hand side of Equation (3.59) is a linear combination of \mathbf{y}_0 and \mathbf{t} , as vectors in \mathbb{R}^3 . The former represents the homogeneous coordinates of the proper point $\bar{\mathbf{y}}_0$, and the latter a point at infinity, with an orientation given by $\bar{\mathbf{t}}$.

An alternative is to form a linear combination of the vectors \mathbf{y}_0 and \mathbf{t} :

$$\lambda_1 \mathbf{y}_0 + \lambda_2 \mathbf{t} = \lambda_1 \left(\mathbf{y}_0 + \frac{\lambda_2}{\lambda_1} \mathbf{t} \right) \sim \mathbf{y} \left(\frac{\lambda_2}{\lambda_1} \right). \quad (3.60)$$

The right-hand side of this expression are the homogeneous coordinates of a point on the line, given by the parameter $s = \lambda_2/\lambda_1$ in Equation (3.59).

We can now make some conclusions. First of all, if we form the homogeneous coordinates of any point on the line, the corresponding vector in \mathbb{R}^3 is a linear combination of \mathbf{y}_0 and \mathbf{t} . We also see that if we form an arbitrary linear combination of \mathbf{y}_0 and \mathbf{t} , the result is the homogeneous coordinates of a point on the line. This means that \mathbf{y} and \mathbf{t} , as vectors in \mathbb{R}^3 , form a basis of a 2-dimensional subspace $S \subset \mathbb{R}^3$ that contains the homogeneous coordinates of any point on the line. Finally, the subspace S must be orthogonal to $\mathbf{l} \in \mathbb{R}^3$, the homogeneous coordinates of the line. This follows from $\mathbf{y}(s) \cdot \mathbf{l} = 0$ for all points $\mathbf{y}(s)$ on the line.

Formally, we should avoid $\lambda_1 = 0$ in Equation (3.59), since this leads to an undetermined value of the parameter s . But it is also the case that making λ_1 smaller and smaller simply makes s larger and larger. As a limit value when $\lambda_1 \rightarrow 0$, the result is a point on the line with homogeneous coordinates \mathbf{t} , i.e., the point at infinity with an orientation given by \mathbf{t} . Hence, $\lambda_1 = 0$ is in fact a valid choice in the linear combination. The only linear combination that is not allowed appears for $(\lambda_1, \lambda_2) = (0, 0)$, since this produces a zero vector in Equation (3.60).

With this last result in mind, we now turn to an extension of the parametric representation of a line in \mathbb{E}^2 , described in Equation (2.13). Here, we choose two distinct points on the line, $\mathbf{y}_1, \mathbf{y}_2$, points in the projective plane \mathbb{P}^2 . This includes the case that one or both points are at infinity. Any other point on the line that passes through the points can be expressed as a linear combination of the two points. The homogeneous form of this parametric representation is

$$\mathbf{y}(\lambda) = \lambda \mathbf{y}_1 + (1 - \lambda) \mathbf{y}_2. \quad (3.61)$$

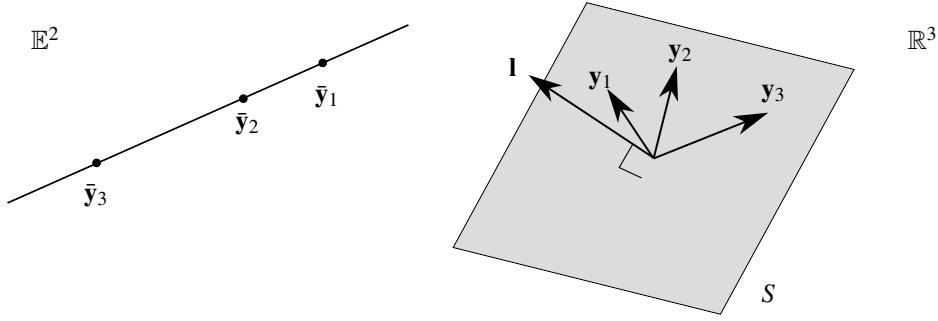


Figure 3.10: Left: three points, \bar{y}_1 , \bar{y}_2 and \bar{y}_3 , on a line in \mathbb{E}^2 . Right: the corresponding homogeneous coordinates lie in a 2-dimensional subspace $S \subset \mathbb{R}^3$. The subspace S is orthogonal to \mathbf{l} .

We have already seen that the homogeneous coordinates of any point on the line can be written as a linear combination of \mathbf{y} and \mathbf{t} . This means that \mathbf{y}_1 and \mathbf{y}_2 , the homogeneous coordinates of \bar{y}_1 and \bar{y}_2 , respectively, both lie in the subspace S . Furthermore, since $\bar{y}_1 \neq \bar{y}_2$ then \mathbf{y}_1 and \mathbf{y}_2 are linearly independent and form a basis of S , and S is orthogonal to \mathbf{l} . See Figure 3.10 for an illustration.

We summarize the observations made here.

3.12 The homogeneous coordinates of all points on a line in \mathbb{E}^2 form to a 2-dimensional subspace S in \mathbb{R}^3 . The subspace S is orthogonal to \mathbf{l} , the dual homogeneous coordinates of the line.

3.13 The homogeneous coordinates of any point on the line can be written as a linear combination of the homogeneous coordinates of two distinct points on the line, where one of the two points may be a point at infinity.

The fact that we can identify a line in \mathbb{E}^2 with a 2-dimensional subspace of \mathbb{R}^3 is further developed in the following section, which introduces a homogeneous representation of the subspace itself.

3.7 Plücker coordinates

A point in \mathbb{E}^2 with Cartesian coordinates $\bar{\mathbf{y}}$ can be represented by its homogeneous coordinates \mathbf{y} , a projective element of $P(\mathbb{R}^3)$, Equation (3.3). This projective element corresponds, in principle⁴, to a 1-dimensional subspace of \mathbb{R}^3 . A line in \mathbb{E}^2 has dual homogeneous coordinates \mathbf{l} , again a projective element of $P(\mathbb{R}^3)$, Equation (3.15). This projective element, too, corresponds to a 1-dimensional subspace of \mathbb{R}^3 .

In the case that the line intersects with the point we know that $\mathbf{y} \cdot \mathbf{l} = 0$. Hence, the two 1-dimensional subspaces in \mathbb{R}^3 , corresponding to the point and to the line, are orthogonal in this case. More generally, all lines intersecting with a specific point \mathbf{y} correspond to a 2-dimensional subspace of \mathbb{R}^3 . This subspace consists of dual homogeneous coordinates of all lines \mathbf{l} such that $\mathbf{y} \cdot \mathbf{l} = 0$, and it forms the orthogonal complement⁵ of \mathbf{y} in \mathbb{R}^3 .

In the same way: all points on a specific line \mathbf{l} correspond to a 2-dimensional subspace of \mathbb{R}^3 . It consists of homogeneous coordinates of all points \mathbf{y} such that $\mathbf{y} \cdot \mathbf{l} = 0$, and it forms the orthogonal complement of the vector \mathbf{l} .

This suggests a one-to-one correspondence between a subspace and its orthogonal complement. One is the orthogonal complement of the other. Given a subspace S , we can determine S^\perp as the orthogonal complement of S , and vice versa. Hence, an alternative representation to the homogeneous coordinates \mathbf{y} is to consider the 2-dimensional subspace $\mathbf{y}^\perp \subset \mathbb{R}^3$, corresponding to all lines that intersect the point. In a similar way, a line \mathbf{l}

⁴Remember that $\mathbf{0}$, which is an element of the subspace, is not included in a projective element.

⁵The orthogonal complement of a subspace is defined in Toolbox Section 3.1.4.

can be represented as the 2-dimensional subspace $\mathbf{l}_\perp \subset \mathbb{R}^3$, consisting of all points that lie on the line. This last correspondence is the one discussed in the previous section.

Can we give a 2-dimensional subspace a homogeneous representation? The answer is: yes, based on Plücker coordinates of the subspace. In what follows, we will construct the Plücker coordinates of the 2-dimensional subspace \mathbf{l}_\perp . To simplify the construction, we assume \mathbf{l} to be a proper line, although the result applies also the general case. The line has dual homogeneous coordinates in accordance with Equation (3.15), and we select one arbitrary proper point on the line with homogeneous coordinates \mathbf{y} . As a vector in \mathbb{R}^3 , \mathbf{y} then satisfies $\mathbf{y} \in \mathbf{l}_\perp$. Another vector in \mathbf{l}_\perp is given as

$$\mathbf{t} = \begin{pmatrix} l_2 \\ -l_1 \\ 0 \end{pmatrix}. \quad (3.62)$$

This vector represents a point at infinity in the orientation given by the line \mathbf{l} . In accordance with the result of Section 3.6, we can then express the homogeneous coordinates of two proper points, \mathbf{y}_1 and \mathbf{y}_2 , lying on \mathbf{l} , as

$$\mathbf{y}_1 \sim \mathbf{y} + s\mathbf{t}, \quad \mathbf{y}_2 \sim \mathbf{y} + t\mathbf{t}, \quad (3.63)$$

where s and t are arbitrary scalars. We then form the matrix $\mathbf{L} \in so(3)$ as

$$\mathbf{L} \sim \mathbf{y}_1 \mathbf{y}_2^\top - \mathbf{y}_2 \mathbf{y}_1^\top. \quad (3.64)$$

• $so(3)$ is defined in Toolbox Section 3.3.4.1.

The anti-symmetric 3×3 matrix \mathbf{L} is referred to as the *Plücker coordinates* of the 2-dimensional subspace $\mathbf{l}_\perp \in \mathbb{R}^3$. Since \mathbf{l}_\perp acts as a representation of the line \mathbf{l} , we may as well refer to \mathbf{L} as the Plücker coordinates of the line \mathbf{l} .

At this point, it may not be obvious that Plücker coordinates can be useful. For example, is not \mathbf{L} dependent on the choice of the point \mathbf{y}_1 and \mathbf{y}_2 ? What happens if we choose two other points on the line \mathbf{l} ? To answer this question, we expand \mathbf{L} in Equation (3.64) into the linear combinations given in Equation (3.63):

$$\mathbf{L} \sim (\mathbf{y} + s\mathbf{t})(\mathbf{y} + t\mathbf{t})^\top - (\mathbf{y} + t\mathbf{t})(\mathbf{y} + s\mathbf{t})^\top = (s-t)(\mathbf{t}\mathbf{y}^\top - \mathbf{y}\mathbf{t}^\top). \quad (3.65)$$

This means that if we choose to see \mathbf{L} as a projective element in $P(\mathbb{R}^{3 \times 3})$, then \mathbf{L} is independent of the choice of \mathbf{y}_1 and \mathbf{y}_2 as long as they are on the line. Clearly, we must assume that \mathbf{y}_1 and \mathbf{y}_2 are distinct points, otherwise $\mathbf{L} = \mathbf{0}$. Hence, \mathbf{L} in Equation (3.64) is a homogeneous representation of the 2-dimensional subspace spanned by all $\mathbf{y} \in \mathbb{R}^3$, which are the homogeneous coordinates of points on the line \mathbf{l} .

3.14 As a projective element, the Plücker coordinates in Equation (3.64) are independent of the choice of \mathbf{y}_1 and \mathbf{y}_2 , as long as they are distinct and lie on the line \mathbf{l} .

\mathbf{L} is a 3×3 anti-symmetric matrix, corresponding to a cross product operator⁶ $[\mathbf{v}]_\times$ of some vector $\mathbf{v} \in \mathbb{R}^3$. If $\mathbf{L} = [\mathbf{v}]_\times$, what can be said about \mathbf{v} ? It must be the case that \mathbf{v} is a null vector of \mathbf{L} , since $\mathbf{L}\mathbf{v} = [\mathbf{v}]_\times\mathbf{v} = \mathbf{0}$, and we see that

$$\mathbf{L}\mathbf{l} = (\mathbf{y}_1 \mathbf{y}_2^\top - \mathbf{y}_2 \mathbf{y}_1^\top)\mathbf{l} = \mathbf{y}_1(\mathbf{y}_2 \cdot \mathbf{l}) - \mathbf{y}_2(\mathbf{y}_1 \cdot \mathbf{l}) = \mathbf{0}, \quad (3.66)$$

since both points \mathbf{y}_1 and \mathbf{y}_2 lie on the line \mathbf{l} . Consequently, we can identify $\mathbf{v} = \mathbf{l}$ and write⁷

$$\mathbf{L} \sim [\mathbf{l}]_\times, \quad \text{or} \quad \mathbf{l} \sim [\mathbf{L}]^\times \sim [\mathbf{y}_1 \mathbf{y}_2^\top - \mathbf{y}_2 \mathbf{y}_1^\top]^\times. \quad (3.67)$$

Dual Plücker coordinates

Now that we have constructed Plücker coordinates for a line in \mathbb{E}^2 , it should not come as a surprise that we can do the same exercise instead for a point \mathbf{y} . Let \mathbf{l}_1 and \mathbf{l}_2 be any two distinct lines that intersect at \mathbf{y} , and form the *dual Plücker coordinates* of the point \mathbf{y} as

$$\tilde{\mathbf{L}} \sim \mathbf{l}_1 \mathbf{l}_2^\top - \mathbf{l}_2 \mathbf{l}_1^\top. \quad (3.68)$$

⁶The cross product operator is described in Toolbox Section 3.7.3.

⁷Here we are using the inverse cross product operator, defined in Toolbox Section 3.7.3.3.

The 3×3 anti-symmetric matrix $\tilde{\mathbf{L}}$ is a representative for the projective element in $P(\mathbb{R}^{3 \times 3})$ that corresponds to the 2-dimensional subspace \mathbf{y}_\perp . As a projective element, it is independent of which lines \mathbf{l}_1 and \mathbf{l}_2 we choose, as long as they are distinct and intersect at \mathbf{y} . Furthermore

$$\tilde{\mathbf{L}} = [\mathbf{y}]_\times, \quad \text{or} \quad \mathbf{y} \sim [\tilde{\mathbf{L}}]^\times \sim [\mathbf{l}_1 \mathbf{l}_2^\top - \mathbf{l}_2 \mathbf{l}_1^\top]^\times. \quad (3.69)$$

Plücker coordinates are seldom useful for dealing with geometric problems in \mathbb{E}^2 . Instead, we have the dual homogeneous coordinates of lines and can use them solve various problems. The situation is different in \mathbb{E}^3 , where the algebraic extension of a line is a plane, rather than a line. As we will see in Section 5.3.2, Plücker coordinates are then a useful representation of a line in \mathbb{E}^3 .

Looking ahead

The Plücker coordinates introduced here have limited practical applications. It is often easier to represent a 2D line with its dual homogeneous coordinates \mathbf{l} , rather than the matrices \mathbf{L} or $\tilde{\mathbf{L}}$. As we will see in Chapter 5, the dual homogeneous representation of lines in 2D extends to planes in the 3D case, not to lines. This means that we need some other representation for 3D lines, and Plücker coordinates are precisely the solution to this problem.

3.8 Duality Principle in the Projective Plane

We have now introduced the basic homogeneous representations for the geometry in the projective plane, and can make a general observation. The canonical forms for the homogeneous representations of a point and of a line are distinct, but there is symmetry. We can often replace the word “point” with “line” in a correct geometric statement, if we also replace “line” with “point”. In algebraic terms, this means that we change \mathbf{y} to \mathbf{l} , and vice versa. The result of this swap is again a correct geometric statement or algebraic relation. An example of this idea is the similarity in the text of Sections 3.3.3 and 3.3.4. We refer to this observation as follow:

Definition 3.2: Duality principle

There is a symmetry between points and lines in the projective plane. If an algebraic expression that describes a geometric relation between points or lines, there is a corresponding algebraic expression that describes a geometric relation between the dual entities. This symmetry is referred to as *duality of points and lines*.

There is a similar duality principle also for the extended space based on \mathbb{E}^3 , formulated in Section 5.6. In that case, points are not in a dual relation to lines, but to planes.

Still, the different parameters are encoded in distinct ways for the vectors \mathbf{y} and \mathbf{l} . It then becomes important to distinguish between points and lines when we want to make geometric interpretations of these algebraic expressions. In particular, this is the case when points or lines are being transformed. This is further expanded in Section 4.5, where we give the dual relation between points and lines a more precise interpretation.

Chapter 4

Transformations in 2D

Before you read this chapter, you need to be familiar with the homogeneous representations of points and lines that are presented in Chapter 3. You may also want to have a look at Toolbox Section 7.1.3, describing how linear transformations between vector spaces generate transformations between the corresponding projective spaces, and at the special form of singular value decomposition described in Toolbox Section 8.2.7.

In this chapter we present different types of transformations in 2D space that appear in a large range of applications. We start with the simplest ones and move on to more complex and general transformations. They can all be represented as linear transformations, applied on the homogeneous representation of points or of lines. In this chapter, we will not go all the way to the most general type of transformations in this category: homographies. They are instead presented separately in Chapter 7, when additional material is available.

There are two specific applications of transformations as they are described here. In the first case, they are used to describe the change in position, or the motion, of points relative to a fixed coordinate system. In the second case, it is the coordinate system that changes, which in turn means that the coordinates of all points in the space change accordingly. In general, changes in the coordinates of a point can be an effect caused by variations both in the position of the point in the space as well as of the coordinate system. In practice, the resulting change in the coordinates may appear identical for the two cases. As a consequence, we do not distinguish between changes in positions of points from changes of the coordinate system.

4.1 How to characterize a transformation

In the following sections, we will define and describe different types of transformation in \mathbb{E}^2 , starting with the simplest ones to moving on to complex types. Each class of transformations is characterized in three equivalent ways. First geometrically, in terms of operations in \mathbb{E}^2 that change points in a particular way. A transformation is also characterized in terms of it is represented algebraically, as a particular type of matrix group¹. Finally, we will also give a geometric characterization in terms of *invariances*, i.e., specific quantities that are left unchanged by the transformations.

4.1.1 Degrees of freedom

We will also characterize the transformation groups in terms of their degrees of freedom². As we will see, the degrees of freedom of a particular transformation group grows with the complexity of the transformation.

A related issue is how many points that need to be specified, before and after a transformation, to completely determine which transformation it is. Let $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$ be the Cartesian coordinates of a point before and after a transformation \mathbf{T} is applied:

$$\bar{\mathbf{y}}_2 = \mathbf{T}(\bar{\mathbf{y}}_1). \quad (4.1)$$

¹Groups are defined in Toolbox Section 2.2.

²Degrees of freedom is discussed in Toolbox Section 4.5.

In this expression, $\mathbf{T} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is a general and non-linear function that is applied to the Cartesian coordinates $\bar{\mathbf{y}}_1$ of a point in \mathbb{E}^2 , to produce the transformed coordinates $\bar{\mathbf{y}}_2$. If \mathbf{T} has r degrees of freedom, this means that \mathbf{T} is specified by a minimum of r parameters, forming the elements of a vector $\mathbf{z} \in \mathbb{R}^r$. To make this *parameterization* of \mathbf{T} explicit, we can write

$$\mathbf{T} = \mathbf{T}_{\mathbf{z}}, \quad \Rightarrow \quad \bar{\mathbf{y}}_2 = \mathbf{T}_{\mathbf{z}}(\bar{\mathbf{y}}_1). \quad (4.2)$$

This last expression can be interpreted in two ways. If $\bar{\mathbf{y}}_1$ is known, and we also know the transformation parameters \mathbf{z} , then $\mathbf{T}_{\mathbf{z}}$ can be determined and applied onto $\bar{\mathbf{y}}_1$ to give $\bar{\mathbf{y}}_2$. Alternatively, if $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$ are known, but not \mathbf{z} , Equation (4.2) provides two constraints in \mathbf{z} . In general, these two constraints come in the form of 2 non-linear equations in the elements of \mathbf{z} . As a general rule of thumb, this observation means that we need at least $r/2$ distinct points, with known coordinates before and after the transformation, in order to completely determine \mathbf{z} . In some cases, there may be circumstances that require us to use more than $r/2$ points, and even with a sufficient number of points we may have to deal with awkward non-linear equations to determine the transformation parameters. This chapter presents some examples of how to parameterize different types of transformations, but we do not go into details about how to solve for \mathbf{z} given that a minimum number of points have been specified. For certain transformation groups, this problem is instead discussed in Part II.

General configuration

It is worth noting that when we specify that a specific minimum number of distinct points are needed to determine an instance of a specific transformation group, it is implicit that these points should be in *general configuration*, i.e., they do not satisfy any particular constraint other than Equation (4.1) for a fixed \mathbf{T} . Depending on the type of transformation, it may be impossible to determine \mathbf{T} based on the constraints from the points if they happen to be in a *degenerate configuration*. Typical examples of degenerate point configurations are when all points coincide or are co-linear. In these degenerate cases the constraints in \mathbf{z} that are formed by Equation (4.2) become dependent, and do not provide sufficient information to completely determine \mathbf{z} .

4.2 Rigid transformations

In a Euclidean space, like \mathbb{E}^2 , there are two fundamental transformations: rotations and translations. Together they form what is referred to as rigid transformations. We will shortly give this class of transformations a precise geometric characterization, but we will first have a look at the two basic transformations which it consists of.

Rotations

Beginning with the rotations, a rotation is geometrically specified by a point $\bar{\mathbf{y}}$, the rotation center, and a rotation angle α . As a matter of convention, a positive rotation angle in \mathbb{E}^2 implies rotating the point counter-clockwise relative to the coordinate system.

The algebraic representation of a rotation, however, always assumes the rotation center to be at the origin. In this case, a rotation is represented in terms of a 2×2 rotation matrix \mathbf{R} . It is applied to the Cartesian coordinates of a point $\bar{\mathbf{y}}_1 = (u_1, v_1)$ to get the coordinates of the rotated point $\bar{\mathbf{y}}_2 = (u_2, v_2)$:

$$\bar{\mathbf{y}}_2 = \begin{pmatrix} u_2 \\ v_2 \end{pmatrix} = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \begin{pmatrix} u_1 \\ v_1 \end{pmatrix} = \mathbf{R} \bar{\mathbf{y}}_1, \quad \text{where } \mathbf{R} = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix}. \quad (4.3)$$

The previous chapter introduced homogeneous representations of points and lines, and we now want to consider how a rotation is represented in homogeneous coordinates. Starting with the P-normalized homogeneous coordinates of the two points before and after the rotation, it follows that

$$\begin{pmatrix} \bar{\mathbf{y}}_2 \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \bar{\mathbf{y}}_1 \\ 1 \end{pmatrix}. \quad (4.4)$$

With \mathbf{y}_1 and \mathbf{y}_2 as the homogeneous coordinates of point $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$, respectively, we get

$$\mathbf{y}_2 \sim \mathbf{T} \mathbf{y}_1, \quad \text{where } \mathbf{T} \sim \begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (4.5)$$

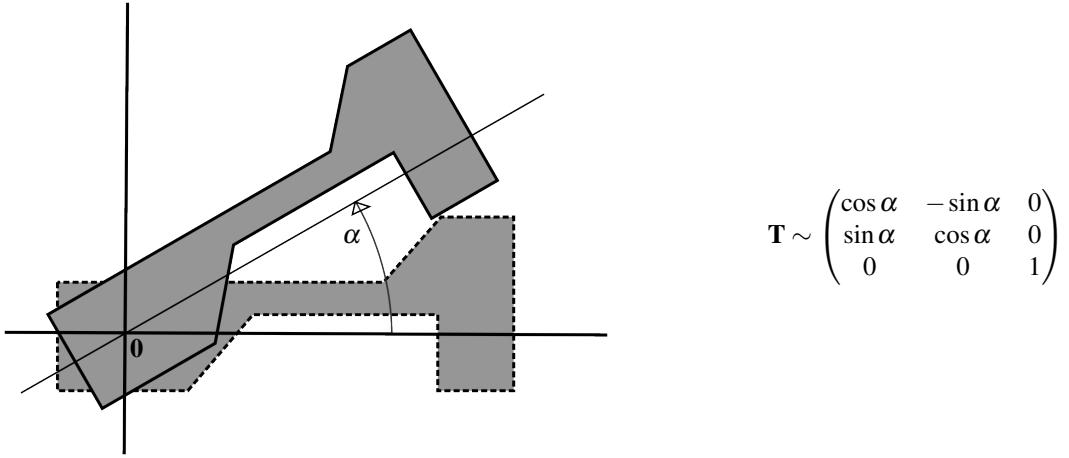


Figure 4.1: Left: an object transformed by a rotation about the origin. Dashed lines: before the transformation is applied. Solid lines: after the transformation is applied. Right: the corresponding transformation matrix \mathbf{T} .

Consequently, \mathbf{T} is a 3×3 matrix, and when multiplied onto the homogeneous coordinates of some point in \mathbb{E}^2 the result is the homogeneous coordinates of the rotated point. The rotation is about the origin of the coordinate system, and counter-clockwise relative to the angle α . A transformation of this type is illustrated in Figure 4.1, together with the corresponding transformation matrix \mathbf{T} .

Before we continue, let us make some important observations. As a start, since the rotation we are discussing here is about the origin of the coordinate system, it can be implemented as a special type of linear transformation on the Cartesian coordinates of the point to be rotated, a special orthogonal transformation³ $\mathbf{R} \in SO(2)$. As we have seen, this transformation corresponds to a linear transformation \mathbf{T} also on the homogeneous coordinates, defined in Equation (4.5).

Linear transformations of projective elements themselves form a projective space⁴. Consequently, any multiplication by a non-zero scalar onto \mathbf{T} in Equation (4.5) is a representative of the same linear transformation on the homogeneous coordinates of points in \mathbb{E}^2 . We also see that rotations in \mathbb{E}^2 about the origin are specified by a single parameter, the angle α , i.e., these transformations have one degree of freedom. Finally, the set of special orthogonal transformations, $SO(3)$, forms a group.

In the end we want to describe general rotations, which can have an arbitrary rotation center, but we need to look at translations first and return to general rotations shortly.

Translations

A translation from $\bar{\mathbf{y}}_1$ to $\bar{\mathbf{y}}_2$ is described as adding a constant (t_1, t_2) to the coordinates:

$$\bar{\mathbf{y}}_2 = \begin{pmatrix} u_2 \\ v_2 \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} + \begin{pmatrix} u_1 \\ v_1 \end{pmatrix} = \bar{\mathbf{t}} + \bar{\mathbf{y}}_1, \quad \text{where } \bar{\mathbf{t}} = \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}. \quad (4.6)$$

Alternatively, we express the transformation in terms of the canonical forms of the homogeneous coordinates for the two points:

$$\begin{pmatrix} \bar{\mathbf{y}}_2 \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \bar{\mathbf{y}}_1 \\ 1 \end{pmatrix}. \quad (4.7)$$

Returning to general homogeneous coordinates, the last relation can be expressed as

$$\mathbf{y}_2 \sim \mathbf{T} \mathbf{y}_1, \quad \text{where } \mathbf{T} \sim \begin{pmatrix} \mathbf{I} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{pmatrix}. \quad (4.8)$$

³The special rotation group $SO(2)$ is presented in Toolbox Section 3.3.4.

⁴Linear transformations on projective spaces are defined in Toolbox Section 7.1.3.

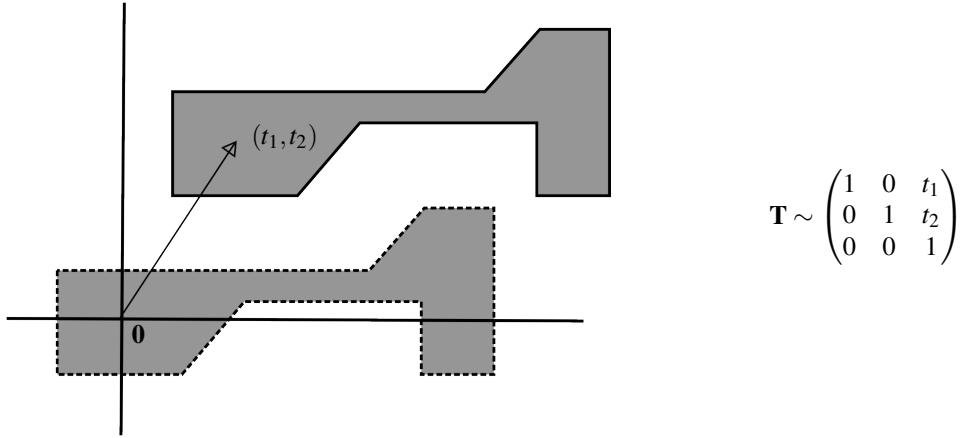


Figure 4.2: Left: an object transformed by a translation. Dashed lines: before the transformation is applied. Solid lines: after the transformation is applied. Right: the corresponding transformation matrix \mathbf{T} .

Similar to what was observed previously for rotation transformations, this \mathbf{T} is a representative of a projective element which transforms homogeneous coordinates according to a translation in \mathbb{E}^2 . An important observation is that although a translation is *not* a linear transformation on the Cartesian coordinates of points in \mathbb{E}^2 , Equation (4.6), it is represented as a linear transformation on the corresponding homogeneous coordinates, Equation (4.8). Furthermore, the set of all the translation matrices \mathbf{T} in Equation (4.8) forms a group, the *translation group* in \mathbb{E}^2 . Translations in this group are specified by two parameters, (t_1, t_2) , i.e., the translation group has two degrees of freedom. Translation is illustrated in Figure 4.2, together with the corresponding transformation matrix \mathbf{T} .

General rigid transformations

We now combine a rotation about the origin with a translation. Both of them can be represented as linear transformations in homogeneous coordinates, and their combined action on the same homogeneous coordinates is the matrix product given by

$$\begin{pmatrix} \mathbf{I} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix}. \quad (4.9)$$

This transformation reads: *first* rotation by \mathbf{R} and *then* translation by $\bar{\mathbf{t}}$. The opposite order is instead given by

$$\begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{I} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{R}\bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix}. \quad (4.10)$$

In general, the two transformations in Equations (4.9) and (4.10) are distinct, and this result illustrates the fact that transformations are not *commutative* in the general case, it matters in which order they are applied. It also illustrates that a transformation can be the result of a sequence of, perhaps, simpler transformations, and that one and the same transformation can be obtained by several different sequences of transformations. For example, the transformation in Equation (4.10) can be obtained by first translating by $\bar{\mathbf{t}}$ and then rotating by \mathbf{R} , but it is also the result of first rotating by \mathbf{R} and then translating by $\mathbf{R}\bar{\mathbf{t}}$.

We now return to the issue of rotation about an arbitrary point in \mathbb{E}^2 , denoted $\bar{\mathbf{y}}_0$. Such a transformation can be implemented as a sequence of the rotations and translations already described. First, we translate by $-\bar{\mathbf{y}}_0$, moving $\bar{\mathbf{y}}_0$ to the origin, then we do the rotation about the origin and, finally, translate the origin back to $\bar{\mathbf{y}}_0$. This means that the total transformation can be expressed as

$$\underbrace{\begin{pmatrix} \mathbf{I} & \bar{\mathbf{y}}_0 \\ \mathbf{0} & 1 \end{pmatrix}}_{\text{translation by } \bar{\mathbf{y}}_0} \underbrace{\begin{pmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}}_{\text{rotation by } \mathbf{R}} \underbrace{\begin{pmatrix} \mathbf{I} & -\bar{\mathbf{y}}_0 \\ \mathbf{0} & 1 \end{pmatrix}}_{\text{translation by } -\bar{\mathbf{y}}_0} = \begin{pmatrix} \mathbf{R} & \bar{\mathbf{y}}_0 - \mathbf{R}\bar{\mathbf{y}}_0 \\ \mathbf{0} & 1 \end{pmatrix}. \quad (4.11)$$

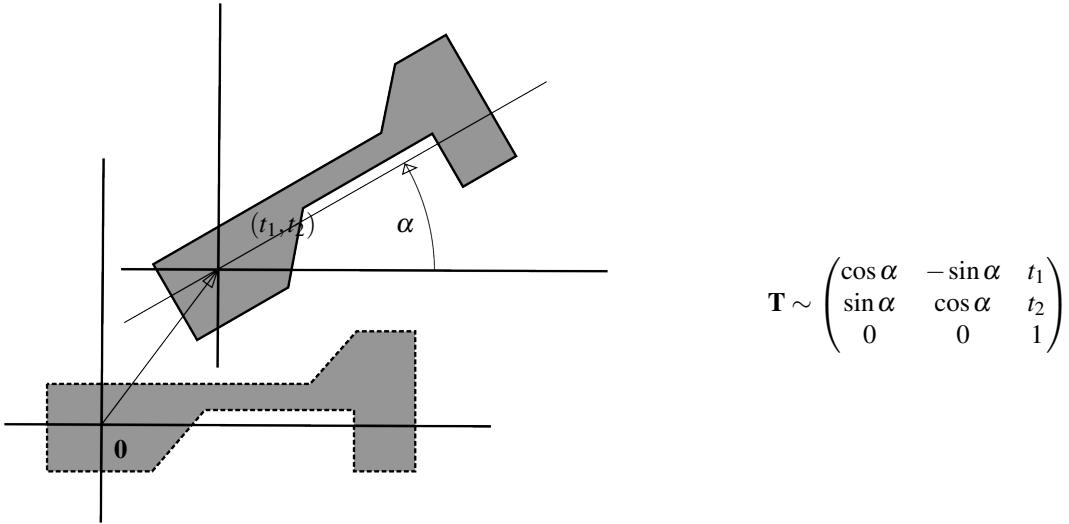


Figure 4.3: Left: an object subject to a rigid transformation. Dashed lines: before the transformation is applied. Solid lines: after the transformation is applied. Notice that the figure is *first* rotated about the origin by the angle α , and *then* translated by (t_1, t_2) . Right: the corresponding transformation matrix \mathbf{T} .

In the light of the previous discussion, we note that this transformation can also be described as: first a rotation about the origin by \mathbf{R} , followed by a translation by $\bar{\mathbf{y}}_0 - \mathbf{R}\bar{\mathbf{y}}_0$.

In fact, any combination of rotations and translations result in a transformation matrix of the form Equation (4.9), for some \mathbf{R} and $\bar{\mathbf{t}}$. This is the set of *Euclidean transformations*. In the literature an Euclidean transformation is also commonly referred to as a *rigid transformation*, which is what they will be called in this compendium⁵. The set of rigid transformations forms a group, the *rigid transformation group* in \mathbb{E}^2 , denoted $SE(2)$. We conclude that the general representation of a rigid transformation when applied to homogeneous coordinates is a 3×3 matrix of the form

$$\mathbf{T} \sim \begin{pmatrix} \mathbf{R} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \cos \alpha & -\sin \alpha & t_1 \\ \sin \alpha & \cos \alpha & t_2 \\ 0 & 0 & 1 \end{pmatrix}, \quad (4.12)$$

where $\mathbf{R} \in SO(2)$ and $\bar{\mathbf{t}} \in \mathbb{R}^2$. Furthermore, any multiplication by a non-zero scalar onto such a matrix is also a representation of the same rigid transformation. An illustration of a rigid transformation is shown in Figure 4.3.

Invariances

The algebraic characterization of a rigid transformation that is made in Equation (4.12) implies that the transformation preserves Euclidean distances:

$$d_{PP}(\mathbf{T}\mathbf{y}_1, \mathbf{T}\mathbf{y}_2) = d_{PP}(\mathbf{y}_1, \mathbf{y}_2), \quad (4.13)$$

for all points $\mathbf{y}_1, \mathbf{y}_2$ when $\mathbf{T} \in SE(2)$. In fact, preservation of distances and of handedness completely characterizes $SE(2)$. From this follows that also areas and angles are preserved by any rigid transformation.

- Handedness of coordinate systems is described in Toolbox Section 3.7.1.

Degrees of freedom

To determine a rigid transformation in \mathbb{E}^2 , we need to specify the rotation angle in \mathbf{R} , this is one single parameter, and the translation vector $\bar{\mathbf{t}}$, which adds two more parameters. In total we need three parameters to determine a rigid transformation in \mathbb{E}^2 , i.e., $SE(2)$ has three degrees of freedom. In accordance with Section 4.1.1, this means

⁵In the literature it is sometimes the case that Euclidean transformations and rigid transformations also include reflections, which can change the handedness of a figure. In this compendium, we use the term rigid transformation to refer to rotations, translations and their combinations, but excluding reflections. This set of transformations is referred to as the *special Euclidean transformation group*, denoted as $SE(2)$.

that we need at least 2 points before and after the transformation to determine what rigid transformation it is. Notice that they must satisfy the rigidity constraint, i.e., the distance between any pair of points is the same before and after the transformation.

4.3 Similarity transformations

Uniform scaling of \mathbb{E}^2 can be done geometrically relative to any specified point. To obtain a simple algebraic representation, we choose this point as the origin. In this case, uniform scaling by a factor $s \neq 0$ is defined as

$$\bar{\mathbf{y}}_2 = s \bar{\mathbf{y}}_1, \quad (4.14)$$

where $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$ are the Cartesian coordinates of a point before and after the scaling, respectively. To obtain a transformation of the corresponding homogeneous coordinates, we write

$$\mathbf{y}_2 \sim \begin{pmatrix} \bar{\mathbf{y}}_2 \\ 1 \end{pmatrix} = \begin{pmatrix} s\mathbf{I} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \bar{\mathbf{y}}_1 \\ 1 \end{pmatrix} \sim \begin{pmatrix} s\mathbf{I} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \mathbf{y}_1. \quad (4.15)$$

This means that the canonical representation of this scaling transformation is given as

$$\mathbf{T} \sim \begin{pmatrix} s\mathbf{I} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (4.16)$$

Combining uniform scaling with rigid transformations gives us the set of *similarity transformations*. If we first do a scaling according to Equation (4.16) and then a rigid transformation according to Equation (4.12), the result is a transformation that is represented by

$$\mathbf{T} \sim \begin{pmatrix} \mathbf{R} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} s\mathbf{I} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} s\mathbf{R} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} s \cos \alpha & -s \sin \alpha & t_1 \\ s \sin \alpha & s \cos \alpha & t_2 \\ 0 & 0 & 1 \end{pmatrix}, \quad (4.17)$$

or, if we do it the other way around:

$$\mathbf{T} \sim \begin{pmatrix} s\mathbf{I} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} s\mathbf{R} & s\bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} s \cos \alpha & -s \sin \alpha & st_1 \\ s \sin \alpha & s \cos \alpha & st_2 \\ 0 & 0 & 1 \end{pmatrix}. \quad (4.18)$$

In both cases, the upper left 2×2 submatrix is a rotation matrix times a scalar. This is the defining character of the transformation matrix of a similarity transformation. To see this, consider two similarity transformations, with scaling parameters s_1, s_2 , rotations $\mathbf{R}_1, \mathbf{R}_2$, and translations $\bar{\mathbf{t}}_1, \bar{\mathbf{t}}_2$, and the result of applying them in sequence:

$$\begin{pmatrix} s_1\mathbf{R}_1 & \bar{\mathbf{t}}_1 \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} s_2\mathbf{R}_2 & \bar{\mathbf{t}}_2 \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} s_1s_2\mathbf{R}_1\mathbf{R}_2 & s_1\mathbf{R}_1\bar{\mathbf{t}}_2 + \bar{\mathbf{t}}_1 \\ \mathbf{0} & 1 \end{pmatrix} \quad (4.19)$$

This is, again, a similarity transformation. It has a scaling parameter given as s_1s_2 , a rotation given by $\mathbf{R}_1\mathbf{R}_2$, and a translation given by $s_1\mathbf{R}_1\bar{\mathbf{t}}_2 + \bar{\mathbf{t}}_1$. Consequently, sequences of similarity transformations always produce similarity transformations, and together they form the group of similarity transformations in \mathbb{E}^2 . An example of a similarity transformation is shown in Figure 4.4.

In order for the set of similarity transformations to form a group, we must exclude the case $s = 0$. In this case all points are mapped to a single point, and this is not an invertible transformation. Furthermore, in \mathbb{E}^2 , a scaling by $s = -1$ is equivalent to a rotation by 180° about the origin. This means that in the case of \mathbb{E}^2 we can assume $s > 0$, since otherwise the rotation parameters become ambiguous. Incidentally, this result does not hold in \mathbb{E}^3 .

Geometric characterization

In contrast to rigid transformations, general similarity transformations do not preserve distances. Instead, a similarity transformation scales all distances uniformly, which means that fractions of two distances, or areas, in \mathbb{E}^2 , are preserved by a similarity transformation. Furthermore, also angles are preserved by a similarity transformation, and we can use either of these conservation properties as a defining character of what a similarity transformation is from a geometric point of view.

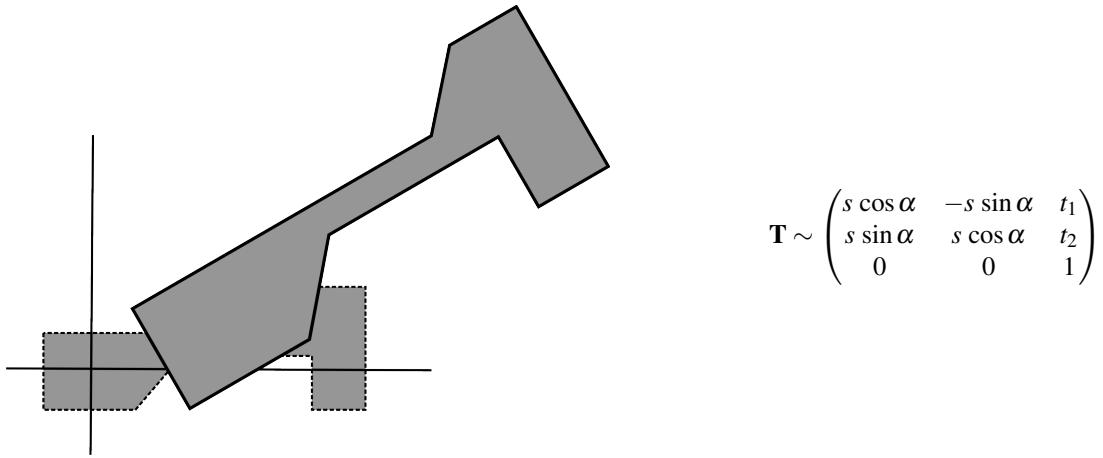


Figure 4.4: An object subject to a similarity transformation. Dashed lines: before the transformation is applied. Solid lines: after the transformation is applied. Right: the corresponding transformation matrix \mathbf{T} .

Degrees of freedom

We need four parameters to specify a particular similarity transformation: first three parameters to specify a rigid transformation and an additional parameter for the scaling. This means that a similarity transformation in \mathbb{E}^2 has four degrees of freedom. Consequently, we need a minimum of 2 points before and after the transformation in order to determine the parameters of a similarity transformation in \mathbb{E}^2 .

4.4 Affine transformations

If we want to define transformations that are more general than the similarity transformations, there are several options. We could consider non-uniform scaling, for example by scaling with distinct parameters along the axes of the coordinate system, or reflections that “mirror” the points of \mathbb{E}^2 relative to a specific line. We will get there, but first we define the more general class of *affine transformations* and look at how specific types of transformations are defined as special cases of affine transformations.

An affine transformation modifies the Cartesian coordinates according to:

$$\bar{\mathbf{y}}_2 = \mathbf{A} \bar{\mathbf{y}}_1 + \bar{\mathbf{t}}, \quad (4.20)$$

where $\mathbf{A} \in GL(2)$, and $\bar{\mathbf{t}} \in \mathbb{R}^2$ is a translation vector. Affine transformations can also be defined in terms of a transformation on homogeneous coordinates:

$$\mathbf{y}_2 \sim \mathbf{T} \mathbf{y}_1, \quad \text{where} \quad \mathbf{T} \sim \begin{pmatrix} \mathbf{A} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix}. \quad (4.21)$$

• The general linear transformation group $GL(2)$ is defined in Toolbox Section 3.3.4.

It is a straightforward exercise to show both that the composition of two affine transformations is, again, an affine transformation, and that the inverse of an affine transformation is affine. This means that they form a group, the *affine transformation group*. The formulation of affine transformations in Equation (4.21) means that they include rigid transformations ($\mathbf{A} = \mathbf{R}$) as well as similarity transformations ($\mathbf{A} = s \mathbf{R}$), but these are merely two special cases of affine transformations. As we will see shortly, they include also non-uniform scalings and reflections, but also more general types of transformations.

Degrees of freedom

The matrix $\mathbf{A} \in GL(2)$ has 4 degrees of freedom⁶, and the translation $\bar{\mathbf{t}}$ has 2 additional degrees of freedom, so an affine transformation has in total 6 degrees of freedom. This means that we need at minimum 3 points, before and

⁶Notice that the elements of \mathbf{A} cannot be chosen completely freely, it must be the case that $\det(\mathbf{A}) \neq 0$.

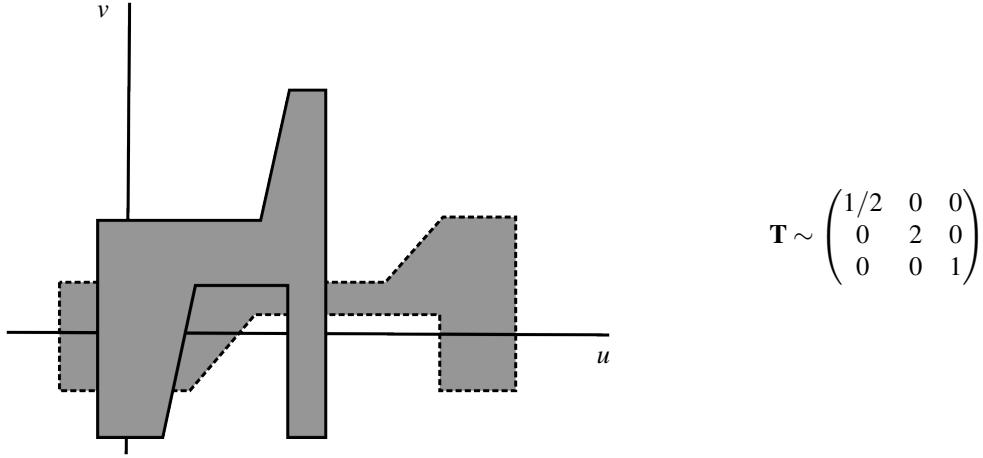


Figure 4.5: Left: an object subject to a non-uniform scaling. Dashed lines: before the transformation is applied. Solid lines: after the transformation is applied. Right: the corresponding transformation matrix \mathbf{T} .

after the transformation, to be able to determine \mathbf{A} and $\bar{\mathbf{t}}$. The issue of what geometric properties are preserved by an affine transformation is discussed in Section 4.5.1.

4.4.1 Non-uniform scaling

A basic type of non-uniform scaling can be implemented by scaling the two Cartesian coordinates of a point in \mathbb{E}^2 by non-zero and distinct parameters s_1 and s_2 , respectively. The corresponding 3×3 transformation matrix on homogeneous coordinates is

$$\mathbf{T} \sim \begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (4.22)$$

In this setting, we can even allow one of the two scaling parameters to be negative, which implies a reflection of the corresponding coordinate axis before scaling by the absolute value of the parameter. As already mentioned in the discussion about similarity transformations, Section 4.3, if both s_1 and s_2 are negative, this corresponds to a rotation by 180° before a non-uniform scaling by the parameters $|s_1|$ and $|s_2|$.

If we want the non-uniform scaling to refer to two arbitrary but perpendicular axes, we first rotate such that the axes of the coordinate system are aligned with the requested ones, do a non-uniform scaling, and then rotate back to the original axes. If, in addition, we want the non-uniform scaling to take place relative to two axes that do not intersect at the origin, but at some other point $\bar{\mathbf{y}}_0$, we first translate such that $\bar{\mathbf{y}}_0$ is at the origin, do whatever non-uniform scaling we are interested in, and then translate the origin back again. All these non-uniform scalings can be implemented by suitable choices of \mathbf{A} and \mathbf{t} in Equation (4.21). An example of a non-uniform scaling is shown in Figure 4.5.

4.4.2 Reflections

We have already seen that reflection can be implemented in terms of non-uniform scaling where one but not both of the scaling parameters are negative. For certain problems, we may want to describe a reflection relative to a specific line in \mathbb{E}^2 , represented by the dual homogeneous coordinates

$$\mathbf{l} = \begin{pmatrix} l_1 \\ l_2 \\ -\Delta \end{pmatrix}. \quad (4.23)$$

Here we assume that \mathbf{l} is not the line at infinity, i.e., it is properly P-normalized. As a consequence, the two vectors $\hat{\mathbf{l}}_1 = (l_1, l_2)$ and $\hat{\mathbf{l}}_2 = (l_2, -l_1)$ form an ON-basis of \mathbb{R}^2 . From Section 3.4.2 we know that a point $\bar{\mathbf{y}}_1$ in \mathbb{E}^2 , with

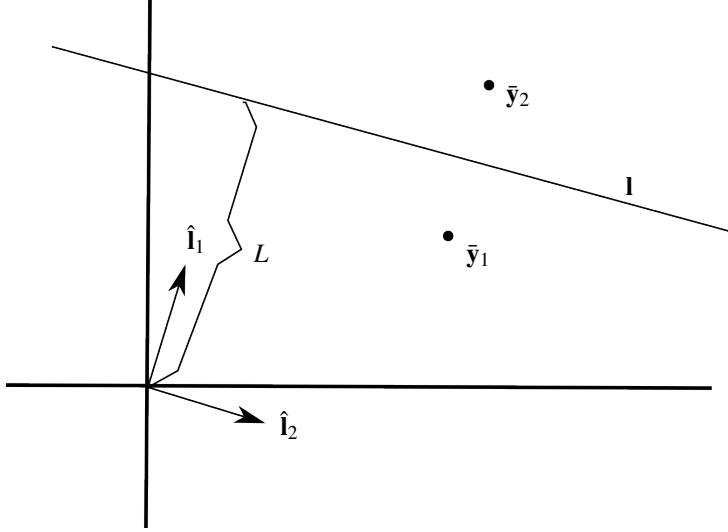


Figure 4.6: The reflection of a point \bar{y}_1 relative to the line \mathbf{I} implies keeping its projection onto $\hat{\mathbf{i}}_2$ fixed while the signed distance to the line changes sign.

P-normalized homogeneous coordinates \mathbf{y}_1 , has a signed distance to the line given as $\mathbf{l} \cdot \mathbf{y}_1 = \hat{\mathbf{i}}_1 \cdot \bar{y}_1 - \Delta$, when measured in the direction of the normal vector $\hat{\mathbf{i}}_1$. Reflection means to put the resulting point at the same distance to the line, but on the other side, i.e., change the sign of this distance. Thus, the resulting point $\bar{y}_2 \in \mathbb{E}^2$, with P-normalized homogeneous coordinates \mathbf{y}_2 , must satisfy $\mathbf{l} \cdot \mathbf{y}_2 = \Delta - \hat{\mathbf{i}}_1 \cdot \bar{y}_1$, and also $\bar{y}_2 \cdot \hat{\mathbf{i}}_2 = \bar{y}_1 \cdot \hat{\mathbf{i}}_2$. See Figure 4.6 for an illustration of variables that appear in this discussion. The Cartesian coordinates of the reflected point are therefore given as

$$\bar{y}_2 = \hat{\mathbf{i}}_1(2\Delta - \hat{\mathbf{i}}_1 \cdot \bar{y}_1) + \hat{\mathbf{i}}_2(\hat{\mathbf{i}}_2 \cdot \bar{y}_1) = (-\hat{\mathbf{i}}_1 \hat{\mathbf{i}}_1^\top + \hat{\mathbf{i}}_2 \hat{\mathbf{i}}_2^\top) \bar{y}_1 + 2\Delta \hat{\mathbf{i}}_1. \quad (4.24)$$

In terms of homogeneous coordinates, the corresponding transformation matrix is

$$\mathbf{T} \sim \begin{pmatrix} -\hat{\mathbf{i}}_1 \hat{\mathbf{i}}_1^\top + \hat{\mathbf{i}}_2 \hat{\mathbf{i}}_2^\top & 2\Delta \hat{\mathbf{i}}_1 \\ \mathbf{0} & 1 \end{pmatrix}. \quad (4.25)$$

This is an affine transformation, with $\mathbf{A} = -\hat{\mathbf{i}}_1 \hat{\mathbf{i}}_1^\top + \hat{\mathbf{i}}_2 \hat{\mathbf{i}}_2^\top$ and $\mathbf{t} = 2\Delta \hat{\mathbf{i}}_1$. Using the fact that $l_1^2 + l_2^2 = 1$ and denoting the third element of \mathbf{l} as $l_3 = -\Delta$, we can rewrite \mathbf{T} in terms of a quadratic function in the elements of \mathbf{l} :

$$\mathbf{T} \sim \begin{pmatrix} -l_1^2 + l_2^2 & -2l_1l_2 & -2l_1l_3 \\ -2l_1l_2 & l_1^2 - l_2^2 & -2l_2l_3 \\ 0 & 0 & l_1^2 + l_2^2 \end{pmatrix}. \quad (4.26)$$

In the case of a reflection, the matrix \mathbf{A} in Equation (4.20) and Equation (4.21) is described by the upper left 2×2 submatrix of \mathbf{T} in Equation (4.26). Consequently,

$$\det(\mathbf{A}) = -(l_1^2 - l_2^2)^2 - 4l_1^2l_2^2 = -(l_1^2 + l_2^2)^2 = -1. \quad (4.27)$$

This result provides an algebraic characterization of reflections in \mathbb{E}^2 : any affine transformation for which $\det(\mathbf{A})$ is negative involves a reflection. The combination of two reflections is not a reflection, which means that reflections do not form a transformation group. The application of one and the same reflection twice is the same as the identity operation, and it is a straightforward exercise to use Equation (4.26) and show that

$$\mathbf{T}^2 = \mathbf{T}\mathbf{T} \sim \mathbf{I}. \quad (4.28)$$

Such a transformation is its own inverse, a so-called *involution*, although this character does not only refer to reflections. Rotations about some point by the angle 180° are also involutions.

An example of a reflection relative to a line \mathbf{I} is illustrated in Figure 4.7.

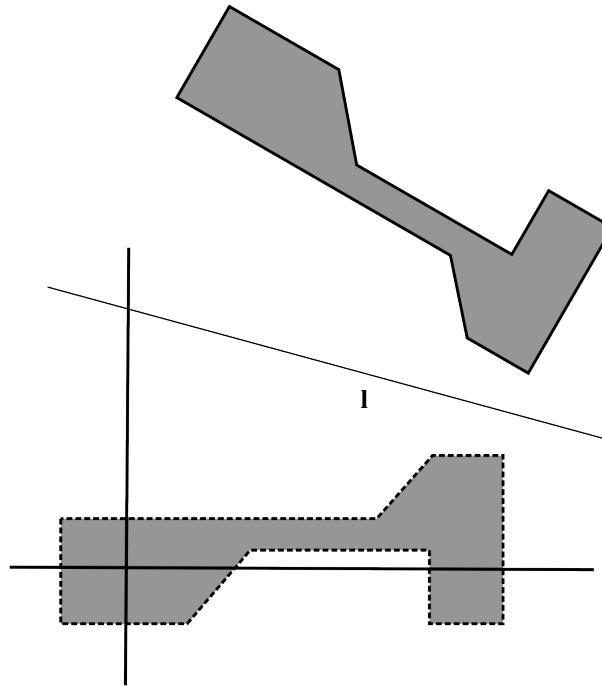


Figure 4.7: An object subject to a reflection relative to a line \mathbf{I} . Dashed lines: before the transformation is applied. Solid lines: after the transformation is applied.

4.4.3 Shearing

A special class of transformations among the affine transformations is *shearing*, sometimes also known as *skewing*. A shearing transformation preserves the area of any figure in \mathbb{E}^2 . A square in \mathbb{E}^2 transformed by the affine transformation in Equation (4.21), will have its area scaled by $\det(\mathbf{A})$. A shearing transformation is then the same as a general affine transformation where $\det(\mathbf{A}) = 1$. Furthermore, a general affine transformation for which $\det(\mathbf{A}) = -1$ is a combination of a shearing and a reflection.

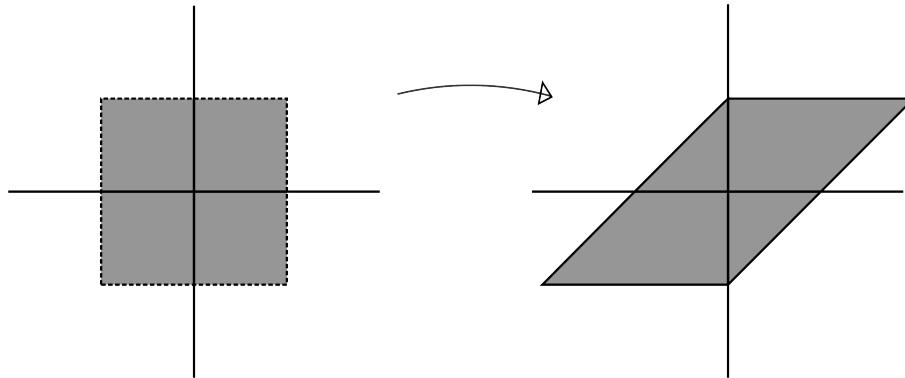


Figure 4.8: An object subject to the shearing transformation represented by \mathbf{T} in Equation (4.29). Dashed lines: before the transformation is applied. Solid lines: after the transformation is applied.

As an example, consider the transformation represented by the matrix

$$\mathbf{T} \sim \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (4.29)$$

The action of this transformation onto a square is illustrated in Figure 4.8.

A general shearing transformation can also have a translation part, specified by $\bar{\mathbf{t}}$ in Equation (4.21). The set of all shearing transformations forms a subgroup of the affine transformations.

4.4.4 Decomposition of an affine transformation

Given the definition of affine transformations, Equation (4.21), and the different subclasses of affine transformations presented above, we can make two observations. First, an affine transformation does not have to fall into any of these subclasses, it can be a general affine transformation. Second, we do not need all these subclasses to explain what an affine transformation is. For example, we can decompose a general affine transformation as a sequence of a rotation, a non-uniform scaling, a second rotation, and a translation. And this is only one of many ways of decomposing an affine transformation into simpler ones. Since it is simple to describe and also useful in certain applications, we stick to this particular decomposition.

Let \mathbf{T} be a general affine transformation, Equation (4.21). A special⁷ SVD of the linear transformation \mathbf{A} gives $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$, where $\mathbf{U}, \mathbf{V} \in SO(2)$, and \mathbf{S} is diagonal and non-singular. This leads to

$$\mathbf{T} \sim \begin{pmatrix} \mathbf{A} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{U}\mathbf{S}\mathbf{V}^\top & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{I} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix}}_{\text{translation by } \bar{\mathbf{t}}} \underbrace{\begin{pmatrix} \mathbf{U} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}}_{\text{non-uniform scaling by } \mathbf{S}} \underbrace{\begin{pmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}}_{\text{rotation by } \mathbf{U}} \underbrace{\begin{pmatrix} \mathbf{V}^\top & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}}_{\text{rotation by } \mathbf{V}^\top}. \quad (4.30)$$

We interpret this as: first apply a rotation about the origin, defined by \mathbf{V}^\top , then a non-uniform scaling given by the diagonal elements of \mathbf{S} , then another rotation, defined by \mathbf{U} . Finally, we need a translation, specified by $\bar{\mathbf{t}}$. This decomposition is applicable on any affine transformation, including reflections and shearing transformations.

4.5 Dual transformations

So far, we have discussed points in \mathbb{E}^2 , and transformations on these points. This lead to a representation of transformations on points: as 3×3 matrices applied to homogeneous coordinates. Other geometric objects consist of point-sets, for example lines. These objects, too, can be subject to transformations. In Section 3.2 we have seen how dual homogeneous coordinates form a representation of lines in \mathbb{E}^2 . If a specific matrix transforms the homogeneous coordinates of a point, how do the dual homogeneous coordinates of a line change?

Let \mathbf{T} be a 3×3 transformation matrix. The homogeneous coordinates of a point before the transformation, \mathbf{y}_1 , and after the transformation, \mathbf{y}_2 , are then related as

$$\mathbf{y}_2 \sim \mathbf{T}\mathbf{y}_1. \quad (4.31)$$

Let \mathbf{l}_1 and \mathbf{l}_2 be the dual homogeneous coordinates of a line, before and after the same transformation. The above question boils down to: how are \mathbf{l}_1 and \mathbf{l}_2 related? A first guess may be that we can write $\mathbf{l}_2 \sim \mathbf{T}\mathbf{l}_1$, but *this is not correct in the general case*.

To see why this is so, let \mathbf{y}_1 be some point on the line \mathbf{l}_1 , i.e., $\mathbf{y}_1 \cdot \mathbf{l}_1 = 0$. If we transform both the point and the line with the same transformation, the transformed point should remain on the transformed line, i.e., $\mathbf{y}_2 \cdot \mathbf{l}_2 = 0$. We use $\tilde{\mathbf{T}}$ to denote the transformation on the dual homogeneous coordinates of the line:

$$\mathbf{l}_2 \sim \tilde{\mathbf{T}}\mathbf{l}_1, \quad (4.32)$$

and we require that $\mathbf{y}_1^\top \mathbf{l}_1 = 0 \Rightarrow \mathbf{y}_2^\top \mathbf{l}_2 = 0$, i.e.,

$$\mathbf{y}_1^\top \mathbf{l}_1 = 0 \Rightarrow 0 = (\mathbf{T}\mathbf{y}_1)^\top \tilde{\mathbf{T}}\mathbf{l}_1 = \mathbf{y}_1^\top \mathbf{T}^\top \tilde{\mathbf{T}}\mathbf{l}_1 = 0. \quad (4.33)$$

⁷The special form of SVD is described in Toolbox Section 8.2.7.

This last relation must hold not only for a particular line, it should be true for every line \mathbf{l}_1 and any point \mathbf{y}_1 on the line. This happens if, and only if, $\mathbf{T}^\top \tilde{\mathbf{T}} = \mathbf{I}$ or

$$\tilde{\mathbf{T}} = \mathbf{T}^{-\top}. \quad (4.34)$$

In this context, we call $\tilde{\mathbf{T}}$ the *dual transformation* of \mathbf{T} .

The introduction of dual transformations means that we must treat a vector in \mathbb{R}^3 in two distinct ways. It can represent either the homogeneous coordinates of a point, or the dual homogeneous coordinates of a line. In the former case the transformation matrix is \mathbf{T} , in the latter case it is $\tilde{\mathbf{T}}$, the dual of \mathbf{T} . This explains why we insist on referring to \mathbf{l} in Equation (3.15) as *dual* homogeneous coordinates. \mathbf{l} change according to the dual transformation $\tilde{\mathbf{T}}$, when \mathbf{T} transform homogeneous coordinates.

Duality works both ways: the dual transformation of $\tilde{\mathbf{T}}$ is $\tilde{\mathbf{T}}^{-\top} = \mathbf{T}$. This means that transformations on lines are dual, not by nature, but relative to transformations applied to points. And transformations of points are dual relative to the corresponding transformations of lines.

In general it is the case that $\mathbf{T} \neq \tilde{\mathbf{T}}$, but it should be clear that $\mathbf{T} = \tilde{\mathbf{T}}$ in the case that $\mathbf{T} \in O(3)$. This case includes rotations about the origin, but also other more general transformations. These general transformations, called homographies, will be the topic of Section 7.1.

4.5.1 Geometric characterization of affine transformations

Consider an affine transformation that is represented by the 3×3 matrix

$$\mathbf{T} = \begin{pmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}, \quad \text{with a corresponding dual transformation } \tilde{\mathbf{T}} = \mathbf{T}^{-\top} = \begin{pmatrix} \mathbf{A}^{-\top} & \mathbf{0} \\ -\mathbf{t}^\top \mathbf{A}^{-\top} & 1 \end{pmatrix}. \quad (4.35)$$

This allows us to present a geometric characterization of affine transformations. Consider two parallel lines,

$$\mathbf{l}_1 = \begin{pmatrix} \hat{\mathbf{l}} \\ \Delta_1 \end{pmatrix}, \quad \mathbf{l}_2 = \begin{pmatrix} \hat{\mathbf{l}} \\ \Delta_2 \end{pmatrix}, \quad (4.36)$$

and map them by the affine transformation

$$\mathbf{l}'_1 = \tilde{\mathbf{T}} \mathbf{l}_1 = \begin{pmatrix} \mathbf{A}^{-\top} & \mathbf{0} \\ -\mathbf{t}^\top \mathbf{A}^{-\top} & 1 \end{pmatrix} \begin{pmatrix} \hat{\mathbf{l}} \\ \Delta_1 \end{pmatrix} = \begin{pmatrix} \mathbf{A}^{-\top} \hat{\mathbf{l}} \\ \Delta_1 - \mathbf{t}^\top \mathbf{A}^{-\top} \hat{\mathbf{l}} \end{pmatrix}, \quad (4.37)$$

$$\mathbf{l}'_2 = \tilde{\mathbf{T}} \mathbf{l}_2 = \begin{pmatrix} \mathbf{A}^{-\top} & \mathbf{0} \\ -\mathbf{t}^\top \mathbf{A}^{-\top} & 1 \end{pmatrix} \begin{pmatrix} \hat{\mathbf{l}} \\ \Delta_2 \end{pmatrix} = \begin{pmatrix} \mathbf{A}^{-\top} \hat{\mathbf{l}} \\ \Delta_2 - \mathbf{t}^\top \mathbf{A}^{-\top} \hat{\mathbf{l}} \end{pmatrix}. \quad (4.38)$$

We see that both of the resulting lines have $\mathbf{A}^{-\top} \hat{\mathbf{l}}$ as a normal vector, which means that they are parallel also after the transformation.

We conclude that parallel lines remain parallel after an affine transformation. Vice versa, if the two lines are not parallel, an affine transformation cannot make them parallel. We interpret a point at infinity as the intersection of parallel lines, and our observations mean that a point at infinity remains there after an affine transformation. Also, proper points remain proper points after an affine transformation. The same applies to lines, the line at infinity stays at infinity after an affine transformation, and a proper line stays proper. In summary, the following characterization applies to affine transformations:

4.1 Affine transformations preserve parallel lines as parallel, and preserve non-parallel lines as non-parallel.

And, as a consequence:

4.2 Affine transformations preserve proper points as proper, and points at infinity stay at infinity.

These observations do not apply to a more general transformations, e.g., the homographies discussed in Chapter 7.

4.6 Projections

Projections form a special class of operations in \mathbb{E}^2 . Here, we focus on orthogonal projections onto a line \mathbf{l} , see Figure 4.9 for an illustration. This operation is special in the sense that it flattens any 2D shape to a line segment. As a result, the resulting area is always zero, and it follows that this operation cannot have an inverse. It is special also in the sense that repeating the projection does not change the result. This aside, the orthogonal projection can still be realized as a 3×3 matrix applied to homogeneous coordinates.

We can use a similar procedure to derive \mathbf{T} , as we did in Section 4.4.2 for the reflection. The Cartesian coordinates of the original point, $\bar{\mathbf{y}}_1$, and of the projected point, $\bar{\mathbf{y}}_2$, are related as

$$\bar{\mathbf{y}}_2 = \hat{\mathbf{l}}_2 (\hat{\mathbf{l}}_2 \cdot \bar{\mathbf{y}}_1) + \Delta \hat{\mathbf{l}}_l = (\hat{\mathbf{l}}_2 \hat{\mathbf{l}}_2^\top - \Delta \hat{\mathbf{l}}_l) \begin{pmatrix} \bar{\mathbf{y}} \\ 1 \end{pmatrix}, \quad \text{which leads to } \mathbf{T} \sim \begin{pmatrix} \hat{\mathbf{l}}_2 \hat{\mathbf{l}}_2^\top & \Delta \hat{\mathbf{l}}_l \\ \mathbf{0} & 1 \end{pmatrix}. \quad (4.39)$$

Based on $l_1^2 + l_2^2 = 1$, we rewrite \mathbf{T} as a quadratic function in the elements of $\mathbf{l} = (l_1, l_2, l_3)$:

$$\mathbf{T} \sim \begin{pmatrix} l_2^2 & -l_1 l_2 & -l_1 l_3 \\ -l_1 l_2 & l_1^2 & -l_2 l_3 \\ 0 & 0 & l_1^2 + l_2^2 \end{pmatrix}. \quad (4.40)$$

The matrix \mathbf{T} in Equation (4.40) does not correspond to an affine transformation. In the case of a projection, we get $\det(\mathbf{T}) = 0$ which means that \mathbf{T} is not invertible. An examination of \mathbf{T} shows that $\mathbf{T}\mathbf{y}_0 = \mathbf{0}$ when $\mathbf{y}_0 \sim (l_1, l_2, 0)$. This is the homogeneous representation of the point at infinity in the direction $\pm \hat{\mathbf{l}}_l$, i.e., the directions perpendicular to the line. This result may seem peculiar. We can project all other points at infinity onto the line without problems. Why not this particular point at infinity as well? The reason is that it does not have a well-defined position *along* the line *after* the projection. The result $\mathbf{T}\mathbf{y}_0 = \mathbf{0}$ flags an undetermined result in this case. \mathbf{T} has a left null space spanned by \mathbf{l} , which follows trivially from the fact that $\mathbf{T}\mathbf{y}$ is a point on the line \mathbf{l} for any proper point \mathbf{y} .

4.7 Concluding remarks

We have presented common types of transformations in \mathbb{E}^2 , implemented as linear mappings on homogeneous coordinates. The most general type, where the corresponding 3×3 matrix is any element in $GL(3)$, is not yet discussed. To properly introduce these transformations, more results are needed and return to this topic in Chapter 7.

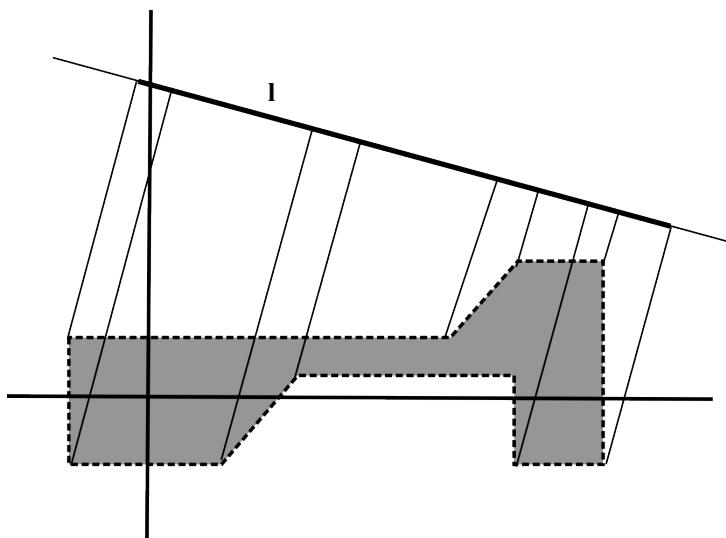


Figure 4.9: Projection onto the line \mathbf{l} .

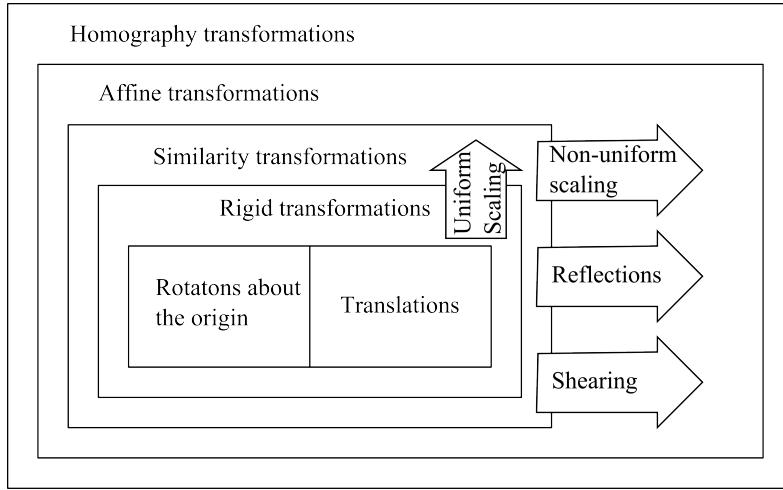


Figure 4.10: A hierarchy of transformations. Homographies are described in Chapter 7.

The following list is a set of observations that we can make from the presentation in this chapter.

- The different sets of transformations discussed here, with the exception of projections, form groups and these can be arranged in a hierarchy, as illustrated in Figure 4.10. Translations and rotations by themselves form two distinct groups, and together they form the group of rigid transformations. Add to that uniform scaling, and we get the group of similarity transformations. If we add to that only non-uniform scaling, including the case where one scaling parameter can be negative, we can generate the entire affine group, including reflection and shearing.
- The 3×3 matrix \mathbf{T} that has been described for each transformation group is in fact a canonical form, seen as a representative of a projective element. We can multiply such a matrix by a non-zero scalar and the result is another representative of the same transformation.
- We need to distinguish between transformations of points and of lines when applying their algebraic representation, in terms of matrices, to homogeneous coordinates. A transformation of one type, e.g., on points, has a dual transformation on lines, and vice versa.

Chapter 5

Homogeneous Representations in 3D

Before you read this chapter, you should have a thorough understanding of the homogeneous representations of points and lines that are presented in Chapter 3. You may also want to have a look at the cross product operator, defined in Toolbox Section 3.7.3, at the singular value decomposition, defined in Toolbox Section 8.2, and at the characterization of the eigensystem of an anti-symmetric matrix that is made in Toolbox Section 8.6.1.

The homogeneous representation of geometric objects in \mathbb{E}^3 is largely a straightforward extension of the results presented in Chapter 3 for \mathbb{E}^2 . The major difference is that dual homogeneous representation described for a line in \mathbb{E}^2 extends to a plane in \mathbb{E}^3 , rather than to a line, and that we need an additional homogeneous representation for a 3D line in the form of Plücker coordinates. This means that the presentation of the early parts of this chapter, extending the homogeneous representation of points and lines in \mathbb{E}^2 to points and planes in \mathbb{E}^3 , is relatively brief.

5.1 Homogeneous coordinates of 3D points

Let $\bar{\mathbf{x}}$ be a point in \mathbb{R}^3 :

$$\bar{\mathbf{x}} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}. \quad (5.1)$$

The elements of $\bar{\mathbf{x}}$ are the Cartesian coordinates of a point relative to a coordinate system in \mathbb{E}^3 , and we define the *canonical form* of the homogeneous coordinates of this point as the vector

$$\begin{pmatrix} \bar{\mathbf{x}} \\ 1 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix}. \quad (5.2)$$

The *homogeneous coordinates* of $\bar{\mathbf{x}}$ is the corresponding projective element:

$$\mathbf{x} \sim \begin{pmatrix} \bar{\mathbf{x}} \\ 1 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix}. \quad (5.3)$$

This expression defines a mapping $\mathbb{R}^3 \rightarrow P(\mathbb{R}^4)$, that takes the 3-dimensional vector $\bar{\mathbf{x}}$ and concatenates it with an extra fourth dimension, where the new coordinate assumes a constant value = 1. The result is a vector in \mathbb{R}^4 , the canonical form of the homogeneous coordinates, Equation (5.2). This vector is a representative of a projective element in $P(\mathbb{R}^4)$, here denoted \mathbf{x} . Formally, this projective element is what we mean by the homogeneous coordinates of $\bar{\mathbf{x}}$.

In the same way as for the 2D case, we use \mathbf{x} to denote a projective element in $P(\mathbb{R}^4)$, as well as a specific vector in \mathbb{R}^4 that is a representative of that projective element. In addition to these two interpretations, we sometimes also use \mathbf{x} to refer to the point in \mathbb{E}^3 that has Cartesian coordinates $\bar{\mathbf{x}}$. We rely on the context to indicate if \mathbf{x} refers to a projective element in $P(\mathbb{R}^4)$, a vector in \mathbb{R}^4 , or to point in \mathbb{E}^3 .

P-normalization

For a given vector $\mathbf{x} \in \mathbb{R}^4$, representing the homogeneous coordinates of a point $\bar{\mathbf{x}}$, we are sometimes interested in knowing the Cartesian coordinates of $\bar{\mathbf{x}}$. This can be done by applying the P-normalization described in Section 3.1.1. It produces the Cartesian coordinates $\bar{\mathbf{x}}$ from any vector \mathbf{x} that contains the corresponding homogeneous coordinates, as long as the fourth element of \mathbf{x} is not zero.

Points at infinity

A 3D point at infinity has homogeneous coordinates given as

$$\mathbf{x} \sim \begin{pmatrix} \bar{\mathbf{t}} \\ 0 \end{pmatrix}. \quad (5.4)$$

This homogeneous representation cannot be P-normalized, and it describes a point at infinite distance from the origin in \mathbb{E}^3 , in fact, at infinite distance from any point in \mathbb{E}^3 . Intuitively, we can arrive at this point by taking a line that has $\bar{\mathbf{t}}$ as a tangent vector and follow this line “all the way” to infinity. It does not matter if you follow the line in one direction or in the opposite direction, or if you take another line that is parallel to the first line, you will end up at the same point at infinity. An alternative way of thinking about points at infinity in the 3D case is a the intersection of two or more parallel lines. Points that are not at infinity are proper points. We return to points at infinity for the 3D case in Section 5.4.1.

5.2 Dual homogeneous coordinates of 3D planes

Section 2.3 gives an algebraic definition of a plane in \mathbb{E}^3 in accordance with Equation (2.18), which we here rewrite as

$$\bar{\mathbf{x}} \cdot \hat{\mathbf{p}} - \Delta = x_1 p_1 + x_2 p_2 + x_3 p_3 - \Delta = 0, \quad (5.5)$$

where $\hat{\mathbf{p}} = (p_1, p_2, p_3)$ is a normal vector of the plane, and Δ is the distance from the origin to the plane. Recall that any point in \mathbb{E}^3 with Cartesian coordinates $\bar{\mathbf{x}} = (x_1, x_2, x_3)$ lies in the plane if and only if Equation (5.5) is satisfied. Furthermore, the parameters for a specific plane are ambiguous unless they are normalized, e.g., such that $\|\hat{\mathbf{p}}\| = 1$ and $\Delta \geq 0$. In this case, $\hat{\mathbf{p}}$ is a normal vector of the plane and Δ is the distance from the origin to the plane.

We rewrite Equation (5.5) in the following form

$$\begin{pmatrix} \bar{\mathbf{x}} \\ 1 \end{pmatrix} \cdot \begin{pmatrix} \hat{\mathbf{p}} \\ -\Delta \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ -\Delta \end{pmatrix} = 0. \quad (5.6)$$

This last expression is a scalar product between two vectors, where the first represents the homogeneous coordinates of the point $\bar{\mathbf{x}}$. The second vector depends entirely on the parameters of the plane, and defines a *canonical form* for the dual homogeneous coordinates of the plane as the vector

$$\begin{pmatrix} \hat{\mathbf{p}} \\ -\Delta \end{pmatrix} = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ -\Delta \end{pmatrix}. \quad (5.7)$$

A homogeneous representation of the plane is defined as the corresponding projective element

$$\mathbf{p} \sim \begin{pmatrix} \hat{\mathbf{p}} \\ -\Delta \end{pmatrix} = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ -\Delta \end{pmatrix}. \quad (5.8)$$

Consequently, Equation (5.6) can be written in the more compact form as

$$\mathbf{x} \cdot \mathbf{p} = \mathbf{x}^\top \mathbf{p} = 0. \quad (5.9)$$

The projective element represented by \mathbf{p} in Equation (5.8) is here referred to as the *dual homogeneous coordinates* of the plane with parameters $(\hat{\mathbf{p}}, \Delta)$. We use \mathbf{p} to refer both to the projective element defined in Equation (5.8), and to a vector that is a representative of this projective element. We also use \mathbf{p} to refer to the corresponding plane in \mathbb{E}^3 , and rely on the context to indicate which of the three options that is relevant.

We summarize this section by concluding that we have defined a new type of projective element, a homogeneous representation of a plane in \mathbb{E}^3 in terms of its dual homogeneous coordinates. Furthermore, the geometric statement that a point $\bar{\mathbf{x}}$ lies on a plane is equivalent to the algebraic statement that the homogeneous coordinates of the point, \mathbf{x} , and the dual homogeneous coordinates of the plane, \mathbf{p} , are orthogonal: $\mathbf{x} \cdot \mathbf{p} = 0$.

D-normalization

Given a vector $\mathbf{p} \in \mathbb{R}^4$, representing the dual homogeneous coordinates of a plane in \mathbb{E}^3 , we could be interested in knowing exactly which plane it is, i.e., we want to determine the parameters of the plane in terms of the normal vector $\hat{\mathbf{p}}$ and distance to the origin Δ . We know that \mathbf{p} is equal to the canonical form of the parameters given in Equation (5.7) times some non-zero scalar. To extract the plane parameters we can apply the D-normalization described in Section 3.2.1 to the vector \mathbf{p} . This operation requires that the first three elements of \mathbf{p} not all are zero.

The plane at infinity

There is a single plane at infinity, with a homogeneous representation given by

$$\mathbf{p}_\infty \sim \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}. \quad (5.10)$$

All other planes are proper planes. A proper plane intersects with a set of proper points, but also with a subset of the points at infinity. The plane at infinity, however, includes all points at infinity, but no proper points.

5.2.1 Parametric representation of a plane

Let \mathbf{p} be a plane in \mathbb{E}^3 . This plane can be identified with \mathbb{E}^2 and, consequently, we can introduce a coordinate system in the plane and represent any point in the plane by its Cartesian 2D coordinates. This can be done as follows: let $\bar{\mathbf{x}}_0$ be a point in the plane that acts as the origin of the coordinate system, and let $\hat{\mathbf{t}}_1$ and $\hat{\mathbf{t}}_2$ be two normalized and perpendicular tangent vectors of the plane. This implies that $\hat{\mathbf{t}}_1$ and $\hat{\mathbf{t}}_2$ are perpendicular also to any normal vector of the plane, and they serve as the axes of the coordinate system. Hence, any point $\bar{\mathbf{x}}$ in the plane can be written as¹

$$\bar{\mathbf{x}} = \bar{\mathbf{x}}_0 + \hat{\mathbf{t}}_1 u + \hat{\mathbf{t}}_2 v. \quad (5.11)$$

The last relation expresses how the Cartesian 3D coordinates of the point $\bar{\mathbf{x}}$ is related to its Cartesian 2D coordinates (u, v) relative to the coordinate system in the plane. We can also express this relation in terms of homogeneous coordinates:

$$\mathbf{x} \sim \begin{pmatrix} \bar{\mathbf{x}}_0 \\ 1 \end{pmatrix} + \begin{pmatrix} \hat{\mathbf{t}}_1 \\ 0 \end{pmatrix} u + \begin{pmatrix} \hat{\mathbf{t}}_2 \\ 0 \end{pmatrix} v = \begin{pmatrix} \hat{\mathbf{t}}_1 & \hat{\mathbf{t}}_2 & \bar{\mathbf{x}}_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}. \quad (5.12)$$

It then makes sense to define a 4×3 matrix \mathbf{P} as

$$\mathbf{P} \sim \begin{pmatrix} \hat{\mathbf{t}}_1 & \hat{\mathbf{t}}_2 & \bar{\mathbf{x}}_0 \\ 0 & 0 & 1 \end{pmatrix}, \quad (5.13)$$

and rewrite Equation (5.12) in the compact form

$$\mathbf{x} \sim \mathbf{P} \mathbf{y}. \quad (5.14)$$

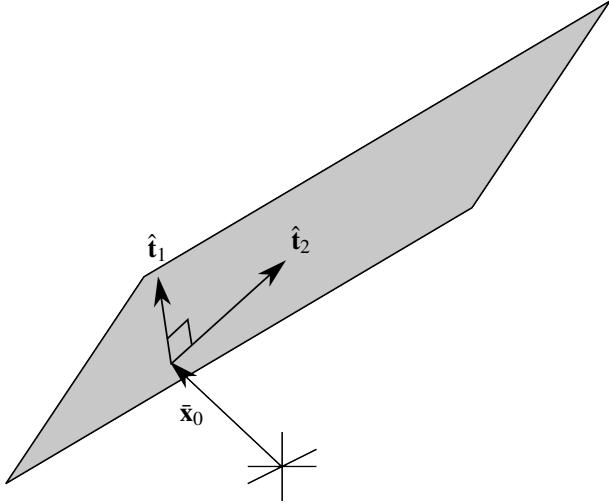


Figure 5.1: A point in a plane can be parameterized using its Cartesian coordinates relative to a coordinate system in the plane. The origin is located at a fixed point $\bar{\mathbf{x}}_0$ in the plane, and has two axes defined by the orthogonal vectors $\hat{\mathbf{t}}_1$ and $\hat{\mathbf{t}}_2$, both tangent vectors of the plane.

Here, \mathbf{y} are the homogeneous coordinates of $\bar{\mathbf{x}}$ relative to the coordinate system defined by \mathbf{P} in the plane \mathbf{p} . \mathbf{P} is a full rank matrix, from which follows that it has a left pseudo-inverse, \mathbf{P}^+ , such that $\mathbf{P}^+ \mathbf{P} = \mathbf{I}$. This observation will be useful in later on.

- Pseudo-inverses are defined in Toolbox Section 3.4.2.

5.3 Homogeneous representations of 3D lines

The algebraic description of a line in \mathbb{E}^2 , the equation of the line, is extended in Section 5.2 to \mathbb{E}^3 , but the result is a plane rather than a line. If we are interested in a line in \mathbb{E}^3 , one option is to use a parametric representation, described below in Section 5.3.1. Although this approach is useful in some applications, it has the usual disadvantage of such representations: it does not provide a unique of the line. An alternative is to use the fact that the homogeneous coordinates of a point on a line must be restricted to a particular 2-dimensional subspace of \mathbb{R}^4 . For a given line, this subspace is unique and can be given a homogeneous representation in terms of Plücker coordinates, using an approach that is very similar to how we treated the 2D case in Section 3.7. This representation is presented in Section 5.3.2, and forms the basis of the results derived in this section.

5.3.1 Parametric representation of a 3D line

A line in \mathbb{E}^3 can be expressed in parametric form, in accordance with either of the two alternatives described in Section 2.3. Based on the parametric forms for the 2D case in Section 3.6, we can formulate the parametric representation of a line in \mathbb{E}^3 as

$$\mathbf{x} \sim \lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2. \quad (5.15)$$

Here, \mathbf{x} is the homogeneous representation of an arbitrary point on the line, and $\mathbf{x}_1, \mathbf{x}_2$ are the homogeneous coordinates of two distinct and fixed points on the line. Which point \mathbf{x} refers to depends on how we choose the parameters $\lambda_1, \lambda_2 \in \mathbb{R}$, but since \mathbf{x} is a projective element, the point is invariant to a common scaling of λ_1, λ_2 . As a consequence, we can parameterize \mathbf{x} using a single parameter, for example as

$$\mathbf{x}(\lambda) \sim \lambda \mathbf{x}_1 + (1 - \lambda) \mathbf{x}_2. \quad (5.16)$$

When we use this parameterization based on homogeneous coordinates, we can allow one of the two points to be a point at infinity. In this case, the point at infinity represents the tangent orientation of the line, and we can

¹Compare this to the discussion that leads to Equation (2.19) in Section 2.3. The difference is that, here, we construct a Cartesian basis. Consequently, the basis vectors in the plane must be orthogonal.

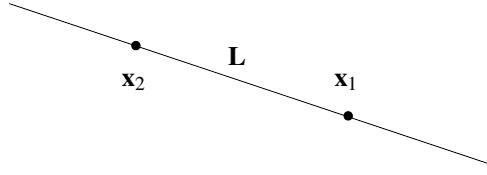


Figure 5.2: The Plücker coordinates of a 3D line \mathbf{L} is formed from the homogeneous coordinates of two points on the line, \mathbf{x}_1 and \mathbf{x}_2 , using Equation (5.18).

write

$$\mathbf{x}(s) \sim \mathbf{x}_0 + s \mathbf{t} = \begin{pmatrix} \bar{\mathbf{x}}_0 \\ 1 \end{pmatrix} + s \begin{pmatrix} \hat{\mathbf{t}} \\ 0 \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{x}}_0 + s \hat{\mathbf{t}} \\ 1 \end{pmatrix}, \quad (5.17)$$

where \mathbf{x}_0 is a fixed point on the line, and \mathbf{t} is the homogeneous coordinates of the point at infinity.

In the same way as for the case of lines in \mathbb{E}^2 , the homogeneous coordinates of points on a line span a 2-dimensional subspace, now in \mathbb{R}^4 . Next, we define a homogeneous representation for such a subspace in terms of its Plücker coordinates.

5.3.2 Plücker coordinates of 3D lines

As an alternative to the parametric representation of a 3D line, described in the previous section, a more unique and compact representation is offered by the Plücker coordinates of the 2-dimensional subspace in \mathbb{R}^4 spanned by the homogeneous coordinates of all points that lie on the line. This is done based on the discussion in Section 3.7, and results in a projective element that is represented by 4×4 anti-symmetric matrix $\mathbf{L} \in so(4)$, constituting the *Plücker coordinates* of the line, defined as:

$$\mathbf{L} \sim \mathbf{x}_1 \mathbf{x}_2^\top - \mathbf{x}_2 \mathbf{x}_1^\top, \quad (5.18)$$

\bullet $so(3)$ is defined in Toolbox Section 3.3.4.1.

where \mathbf{x}_1 and \mathbf{x}_2 are the homogeneous coordinates of two points on the line, see Figure 5.2. In the same way as for the Plücker coordinates of a 2D line, as a projective element \mathbf{L} is independent to the choice of \mathbf{x}_1 and \mathbf{x}_2 as long as they are distinct and lie on the line.

The 4×4 matrix \mathbf{L} is anti-symmetric and of rank 2. It has a range spanned by the vectors \mathbf{x}_1 and \mathbf{x}_2 , and the range is orthogonal to its null space, i.e., any linearly independent set of vectors \mathbf{p}_1 and \mathbf{p}_2 that both are orthogonal to \mathbf{x}_1 and \mathbf{x}_2 spans $\text{Null}(\mathbf{L})$. Such vectors \mathbf{p}_1 and \mathbf{p}_2 correspond to planes that intersect both \mathbf{x}_1 and \mathbf{x}_2 , i.e., planes that intersect the line. We will return to these planes in Section 5.3.4, where dual Plücker coordinates are defined.

About notation

We use the same approach for the notation of geometric objects and for their homogeneous representation as before. Thus, \mathbf{L} denotes both a particular line in \mathbb{E}^3 and its homogeneous representation, its Plücker coordinates, which is a matrix in $so(4)$ or rather the projective element represented by this matrix.

Lines at infinity

A *line at infinity* can be seen as the intersecting line of two distinct points at infinity, similar to the 2D case. The difference is that in the 2D case there is only one line at infinity, while in the 3D case there are infinitely many lines at infinity. The general form for the Plücker coordinates of such a line is

$$\mathbf{L} = \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & 0 \end{pmatrix}, \quad (5.19)$$

where $\mathbf{A} \in so(3)$. A line that is not at infinity is a *proper line*. A proper line always intersects with exactly one point at infinity, but also includes proper points, while a line at infinity intersects only points at infinity.

5.3.3 L-normalization

Let \mathbf{x}_1 and \mathbf{x}_2 be two points lying on the proper line \mathbf{L} in \mathbb{E}^3 . In addition to this, we assume that \mathbf{x}_1 and \mathbf{x}_2 are P-normalized as vectors in \mathbb{R}^4 , and also that the points are at unit distance²: $d_{\text{PP}}(\mathbf{x}_1, \mathbf{x}_2) = 1$. These two vectors are then given as

$$\mathbf{x}_1 = \begin{pmatrix} \bar{\mathbf{x}}_1 \\ 1 \end{pmatrix}, \quad \mathbf{x}_2 = \begin{pmatrix} \bar{\mathbf{x}}_1 + \hat{\mathbf{t}} \\ 1 \end{pmatrix}, \quad (5.20)$$

where $\bar{\mathbf{x}}_1$ is the Cartesian coordinates of the first point, and $\hat{\mathbf{t}}$ is a normalized tangent vector of the line. The Plücker coordinates of the line that intersects \mathbf{x}_1 and \mathbf{x}_2 is given as the following 4×4 matrix:

$$\mathbf{L} \sim \mathbf{x}_1 \mathbf{x}_2^\top - \mathbf{x}_2 \mathbf{x}_1^\top = \begin{pmatrix} \bar{\mathbf{x}}_1 \hat{\mathbf{t}}^\top - \hat{\mathbf{t}} \bar{\mathbf{x}}_1^\top & -\hat{\mathbf{t}} \\ \hat{\mathbf{t}}^\top & 0 \end{pmatrix}. \quad (5.21)$$

Consequently, given an arbitrary representative of the Plücker coordinates of a proper line \mathbf{L} , we can normalize it such that the three first elements in the fourth column has unit norm. The resulting matrix equals the right-hand side of Equation (5.21) for P-normalized vectors \mathbf{x}_1 and \mathbf{x}_2 , representing two points on the line at unit distance from each other.

We refer to this normalization of \mathbf{L} as *line normalization*, or *L-normalization* for short. It is described in terms of the function $\text{norm}_{\mathbf{L}}$, defined on $so(4)$ as

$$\text{norm}_{\mathbf{L}}(\mathbf{L}) = \frac{\mathbf{L}}{\|\mathbf{L}_{1:3,4}\|} = \begin{pmatrix} \mathbf{A} & -\hat{\mathbf{t}} \\ \hat{\mathbf{t}} & 0 \end{pmatrix} = \mathbf{x}_1 \mathbf{x}_2^\top - \mathbf{x}_2 \mathbf{x}_1^\top. \quad (5.22)$$

Here, $\|\mathbf{L}_{1:3,4}\|$ is the norm of the first three elements in the fourth column of \mathbf{L} . This norm vanishes in the case that \mathbf{L} represents a line at infinity, which means that L-normalization is only applicable for proper lines.

In the case of a general, but proper, line \mathbf{L} , $\text{norm}_{\mathbf{L}}(\mathbf{L})$ allows us to determine some relevant parameters of the line. For example, the first three elements of the fourth column in $\text{norm}_{\mathbf{L}}(\mathbf{L})$ contains $\hat{\mathbf{t}}$, a tangent vector of the line. This vector has no well-defined sign, but it represents the orientation of the line. Furthermore, the point $\bar{\mathbf{x}}_1$ is not well-defined other than it must lie on the line. We can, for example, choose it as $\bar{\mathbf{x}}_0$, the closest point on the line to the origin. In this case: $\bar{\mathbf{x}}_0 \cdot \hat{\mathbf{t}} = 0$, and we can use this to obtain

$$-\frac{(\mathbf{L}_{1:3,1:3})(\mathbf{L}_{1:3,4})}{\|\mathbf{L}_{1:3,4}\|^2} = \mathbf{A} \hat{\mathbf{t}} = (\bar{\mathbf{x}}_0 \hat{\mathbf{t}}^\top - \hat{\mathbf{t}} \bar{\mathbf{x}}_0^\top) \hat{\mathbf{t}} = \bar{\mathbf{x}}_0, \quad (5.23)$$

where \mathbf{A} is the upper left 3×3 submatrix of $\text{norm}_{\mathbf{L}}(\mathbf{L})$ in Equation (5.22).

In summary: $\text{norm}_{\mathbf{L}}(\mathbf{L})$ can be seen as a *canonical form* of the Plücker coordinates of a line, which provides us both with a tangent vector of the line, $\hat{\mathbf{t}}$, and a point on the line. More specifically, Equation (5.23) provides $\bar{\mathbf{x}}_0$, the

²Here we use the distance function that is defined in Section 5.5.1.

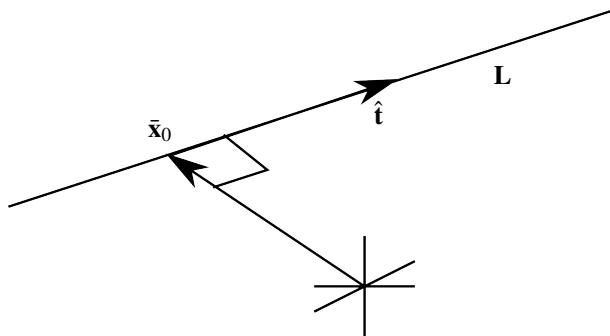


Figure 5.3: A line in \mathbb{E}^3 , here denoted \mathbf{L} , can be characterized by a point $\bar{\mathbf{x}}_0$ on the line, which lies closest to the origin, and a tangent vector $\hat{\mathbf{t}}$.

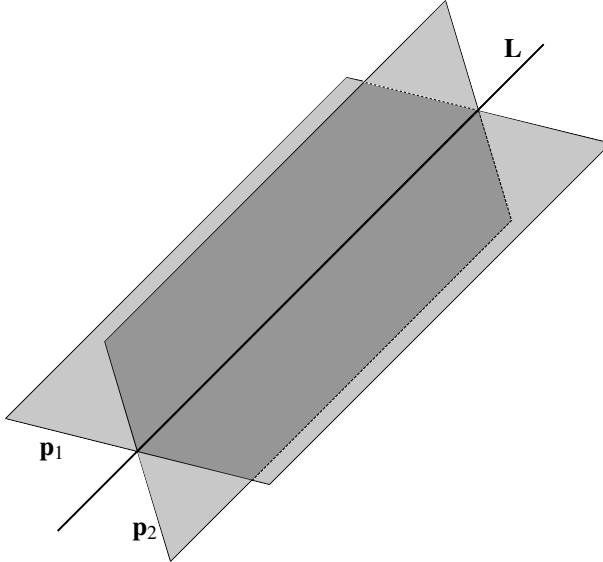


Figure 5.4: The dual Plücker coordinates of a 3D line are constructed from two distinct planes that intersect the line.

point on the line closest to the origin. See Figure 5.3 for an illustration of these parameters. The canonical form of a line is ambiguous with respect to the sign of \hat{t} , since both directions are consistent with the same line. Notice that $\|\bar{x}_0\|$ is the perpendicular distance to the line from the origin.

5.3.4 Dual Plücker coordinates

A line in \mathbb{E}^3 can be defined in terms of the set of points that lie on the line, but it can also be defined in terms of intersecting planes. We have already concluded that the Plücker coordinates \mathbf{L} of a line in \mathbb{E}^3 is a representation of the 2-dimensional subspace in \mathbb{R}^4 that contains the homogeneous coordinates \mathbf{x} of all points on the line. This 2-dimensional subspace is uniquely defined by its orthogonal complement in \mathbb{R}^4 and it, too, is a 2-dimensional subspace. It consists of all vectors $\mathbf{p} \in \mathbb{R}^4$ that are orthogonal to all these \mathbf{x} . Geometrically, such a \mathbf{p} represents the dual homogeneous coordinates of a plane that intersect the line.

Let \mathbf{p}_1 and \mathbf{p}_2 be the dual homogeneous coordinates of two distinct planes that intersect a particular line in \mathbb{E}^3 . We defined the *dual Plücker coordinates* of the line as

$$\tilde{\mathbf{L}} \sim \mathbf{p}_1 \mathbf{p}_2^\top - \mathbf{p}_2 \mathbf{p}_1^\top. \quad (5.24)$$

As a projective element, $\tilde{\mathbf{L}}$ is independent of which two plane we choose as long as they are distinct and intersect the line, see Figure 5.4. Furthermore, $\tilde{\mathbf{L}} \in so(4)$ and it is of rank 2. When it is necessary to distinguish the two types of Plücker coordinates, the one defined in Equation (5.18) is here referred to as *standard Plücker coordinates*. The dual Plücker representation works also when the two planes are parallel, in which case the resulting line of intersection is a line at infinity.

We now have two possible descriptions of a line in \mathbb{E}^3 , its Plücker coordinates \mathbf{L} and its dual Plücker coordinates $\tilde{\mathbf{L}}$. One is related to the points on the line, and the other to the planes that intersect the line. This means that the two descriptions are not identical but they both provide a compact and complete representation of a line. In fact, the two representations are very much related. With \mathbf{L} and $\tilde{\mathbf{L}}$ representing one and the same line, this relation implies that

$$\mathbf{L} \cdot \tilde{\mathbf{L}} = \text{trace}(\mathbf{L}^\top \tilde{\mathbf{L}}) = \text{trace}\left((\mathbf{x}_1 \mathbf{x}_2^\top - \mathbf{x}_2 \mathbf{x}_1^\top)^\top (\mathbf{p}_1 \mathbf{p}_2^\top - \mathbf{p}_2 \mathbf{p}_1^\top)\right) = 0. \quad (5.25)$$

Here we have used the Frobenius scalar product³ between matrices, together with the fact that $\mathbf{x}_1, \mathbf{x}_2$ both are orthogonal to both of $\mathbf{p}_1, \mathbf{p}_2$. This result shows that the two projective elements \mathbf{L} and $\tilde{\mathbf{L}}$ are orthogonal in $P(\mathbb{R}^{4 \times 4})$.

³The Frobenius scalar product is defined in Toolbox Section 3.4.1.

About notation

We will use $\tilde{\mathbf{L}}$ to denote: (1) the projective element in $P(so(4))$ which contains the Plücker coordinates of a specific line in \mathbb{E}^3 , (2) a matrix in $so(4)$ which represent such a projective element, and (3) the line in \mathbb{E}^3 . The context should make clear which case $\tilde{\mathbf{L}}$ refers to. Hence, a line in \mathbb{E}^3 can be denoted either as \mathbf{L} or $\tilde{\mathbf{L}}$.

5.3.5 The duality mapping

In some cases it may be useful to have a mapping on $P(so(4))$ that produces $\tilde{\mathbf{L}}$ from \mathbf{L} , the *duality mapping*. In the light of the previous discussion we have the following characterization of this mapping: (1) $\tilde{\mathbf{L}}$ must be orthogonal to \mathbf{L} , (2) $\tilde{\mathbf{L}}$ has rank two. Another piece of information that is relevant here comes from Toolbox Section 8.6.1: an anti-symmetric matrix has always an even rank, i.e., in the case of $so(4)$ the rank is either 0, 2, or 4. Rank 0 means that all elements are = 0, and rank 4 occurs if and only if the determinant is not zero. We can summarize these observations as: both \mathbf{L} and its dual representation $\tilde{\mathbf{L}}$ should have a vanishing determinant.

Let \mathbf{L} be given as

$$\mathbf{L} \sim \begin{pmatrix} 0 & a & b & c \\ -a & 0 & d & e \\ -b & -d & 0 & f \\ -c & -e & -f & 0 \end{pmatrix}. \quad (5.26)$$

Instead of formally deriving the duality mapping based on the given properties, we define it directly as

$$\tilde{\mathbf{L}} \sim \begin{pmatrix} 0 & f & -e & d \\ -f & 0 & c & -b \\ e & -c & 0 & a \\ -d & b & -a & 0 \end{pmatrix}. \quad (5.27)$$

From this follows that

$$\det \mathbf{L} = \det \tilde{\mathbf{L}} = (af - be + cd)^2, \quad (5.28)$$

which means that $\text{rank}(\tilde{\mathbf{L}}) = \text{rank}(\mathbf{L})$, i.e., $\tilde{\mathbf{L}}$ has rank 2 whenever \mathbf{L} has rank 2. Furthermore, it is a straightforward exercise to show that $\mathbf{L} \cdot \tilde{\mathbf{L}} = 0$ whenever $\det(\mathbf{L}) = \det(\tilde{\mathbf{L}}) = 0$. This means that $\tilde{\mathbf{L}}$ in Equation (5.27) defines a duality mapping of \mathbf{L} , it produces the dual Plücker coordinates $\tilde{\mathbf{L}}$ of the line for which \mathbf{L} are the standard Plücker coordinates. Both standard and dual Plücker coordinates are unique for a particular line, if we also take into account that both \mathbf{L} and $\tilde{\mathbf{L}}$ are projective elements, which means that the formulation of the duality mapping in Equation (5.27) is unique. Notice that the duality mapping in Equation (5.27) is a linear transformation that can be applied to any $\mathbf{L} \in P(so(4))$, but it makes sense only when \mathbf{L} has rank 2.

Finally, we observe that if we apply the duality mapping to $\tilde{\mathbf{L}}$, the result is \mathbf{L} , i.e., $\tilde{\tilde{\mathbf{L}}} = \mathbf{L}$. Consequently, we can use the same mapping to transform \mathbf{L} to $\tilde{\mathbf{L}}$, or transform $\tilde{\mathbf{L}}$ to \mathbf{L} .

5.3.6 Internal constraint and degrees of freedom

For points and lines in \mathbb{E}^2 , we can pick any non-zero vector in \mathbb{R}^3 and claim that it represents a well-defined point or line, possibly at infinity. Similarly, we can pick any non-zero vector in \mathbb{R}^4 and use it to represent a point or a plane in \mathbb{E}^3 . We have seen that a line in \mathbb{E}^3 has a homogeneous representation in terms of its Plücker coordinates \mathbf{L} , an element of $P(so(4))$ where $\text{rank}(\mathbf{L}) = 2$. The last property implies that $\det(\mathbf{L}) = 0$. In fact, due to Equation (5.28), we get $\det(\mathbf{L}) = \det(\tilde{\mathbf{L}}) = 0$ for any proper set of Plücker coordinates. This means that we cannot pick any matrix in $so(4)$ and use it as Plücker coordinates. Only matrices in $so(4)$ that satisfy an *internal constraint*, in the form of a vanishing determinant, are valid as Plücker coordinates. In this context, we refer to a matrix that satisfy the internal constraint as *consistent Plücker coordinates*. Based on Equation (5.28), we summarize this result as

$$\mathbf{L} \text{ and } \tilde{\mathbf{L}} \text{ are valid (dual) Plücker coordinates} \iff af - be + cd = 0. \quad (5.29)$$

5.1 Algebraic representations of geometric objects can sometimes have internal constraints.

Equation (5.29) is the first of many internal constraints that can be derived for homogeneous representations of geometric objects. It is a quadratic form in the elements of \mathbf{L} or $\tilde{\mathbf{L}}$ that vanishes for Plücker coordinates. This

means that the set of Plücker coordinates in the 6-dimensional vector space $so(4)$ forms a second order surface defined by Equation (5.29). The projective space $P(so(4))$ has 5 degrees of freedom, because of the \sim equivalence, and due to Equation (5.29) the set of Plücker coordinates has 4 degrees of freedom.

5.2 A line in \mathbb{E}^3 has 4 degrees of freedom.

5.3.7 DL-normalization

To derive distances related to lines in \mathbb{E}^3 , we need to extend the idea of L-normalization also to dual Plücker coordinates of 3D lines. Let $\tilde{\mathbf{L}}$ be a line in \mathbb{E}^3 . We assume that we are dealing with a proper line for which we can find two distinct and proper planes \mathbf{p}_1 and \mathbf{p}_2 that intersect the line, see Figure 5.4. We can always select the planes to be perpendicular in \mathbb{E}^3 , and such that one of them, e.g., \mathbf{p}_2 , intersects the origin. The corresponding dual homogeneous coordinates of the planes are:

$$\mathbf{p}_1 = \begin{pmatrix} \hat{\mathbf{p}}_1 \\ -\Delta_1 \end{pmatrix}, \quad \mathbf{p}_2 = \begin{pmatrix} \hat{\mathbf{p}}_2 \\ 0 \end{pmatrix}, \quad \text{where } \hat{\mathbf{p}}_1 \cdot \hat{\mathbf{p}}_2 = 0. \quad (5.30)$$

The corresponding matrix $\tilde{\mathbf{L}} \in \mathbb{R}^{4 \times 4}$, is then given as

$$\tilde{\mathbf{L}} \sim \mathbf{p}_1 \mathbf{p}_2^\top - \mathbf{p}_2 \mathbf{p}_1^\top = \begin{pmatrix} \hat{\mathbf{p}}_1 \hat{\mathbf{p}}_2^\top - \hat{\mathbf{p}}_2 \hat{\mathbf{p}}_1^\top & \Delta_1 \hat{\mathbf{p}}_2 \\ -\Delta_1 \hat{\mathbf{p}}_2^\top & 0 \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{b} \\ -\mathbf{b}^\top & 0 \end{pmatrix}, \quad (5.31)$$

where $\mathbf{A} \in so(3)$. From this follows that

$$\|\mathbf{A}\|_F^2 = \text{trace}(\mathbf{A}^\top \mathbf{A}) = 2 (\|\hat{\mathbf{p}}_1\|^2 \|\hat{\mathbf{p}}_2\|^2 - (\hat{\mathbf{p}}_1 \cdot \hat{\mathbf{p}}_2)^2) = 2. \quad (5.32)$$

Hence, given $\tilde{\mathbf{L}} \in so(4)$, a representative of the dual Plücker coordinates of a proper line, it can be normalized such that the Frobenius norm of the upper left 3×3 submatrix equals $\sqrt{2}$. The resulting matrix equals Equation (5.31) for D-normalized vectors \mathbf{p}_1 and \mathbf{p}_2 that represent two perpendicular planes intersecting the line.

We refer to this normalization as *dual line normalization*, or *DL-normalization* for short. It is described in terms of the function norm_{DL} on $so(4)$, defined as

$$\text{norm}_{\text{DL}}(\tilde{\mathbf{L}}) = \frac{\sqrt{2}}{\|\tilde{\mathbf{L}}_{1:3,1:3}\|_F} \tilde{\mathbf{L}} = \begin{pmatrix} \mathbf{A} & \mathbf{b} \\ -\mathbf{b}^\top & 0 \end{pmatrix}, \quad (5.33)$$

where $\tilde{\mathbf{L}}_{1:3,1:3}$ is the upper left 3×3 submatrix of $\tilde{\mathbf{L}}$. The result can be seen as a *canonical form* of the dual Plücker coordinates of a 3D line. This normalization cannot be done for a line at infinity, since $\tilde{\mathbf{L}}_{1:3,1:3} = \mathbf{0}$ in this case.

From the DL-normalized version of $\tilde{\mathbf{L}}$ we can immediately recover the parameters $\bar{\mathbf{x}}_0$ and $\hat{\mathbf{t}}$, discussed in Section 5.3.3, where $\bar{\mathbf{x}}_0$ is the point on the line that is closest to the origin, and $\hat{\mathbf{t}}$ is a tangent vector of the line, see Figure 5.3. From the construction of the two planes, it follows that $\bar{\mathbf{x}}_0 = \Delta_1 \hat{\mathbf{p}}_1$, and we get

$$2 \frac{(\tilde{\mathbf{L}}_{1:3,1:3})(\tilde{\mathbf{L}}_{1:3,4})}{\|\tilde{\mathbf{L}}_{1:3,1:3}\|_F^2} = \mathbf{A} \mathbf{b} = (\hat{\mathbf{p}}_1 \hat{\mathbf{p}}_2^\top - \hat{\mathbf{p}}_2 \hat{\mathbf{p}}_1^\top) \Delta_1 \hat{\mathbf{p}}_2 = \Delta_1 \hat{\mathbf{p}}_1 = \bar{\mathbf{x}}_0. \quad (5.34)$$

Furthermore, using Equation (3.142), a tangent vector $\hat{\mathbf{t}}$ can be derived as

$$\mathbf{A} = [\hat{\mathbf{t}}]_\times, \quad \Rightarrow \quad \hat{\mathbf{t}} = [\mathbf{A}]^\times. \quad (5.35)$$

Since \mathbf{L} is a projective element and does not have a well-defined sign, the sign of $\hat{\mathbf{t}}$ is ambiguous. Hence, the perpendicular distance to the line from the origin is given as $\|\bar{\mathbf{x}}_0\|$.

• The Frobenius norm on matrices is defined in Toolbox Section 3.4.1.

5.4 Incidence relations between points, planes, and lines

In this section, the results derived in Section 3.3 for \mathbb{E}^2 are extended to \mathbb{E}^3 . Similar concepts reappear here, but the algebraic implementation looks slightly different. As before, the results derived here are based on an orthogonality relation: $\mathbf{x} \cdot \mathbf{p} = 0$, where \mathbf{x} and \mathbf{p} are the homogeneous coordinates and dual homogeneous coordinates of a point and a plane that intersect.

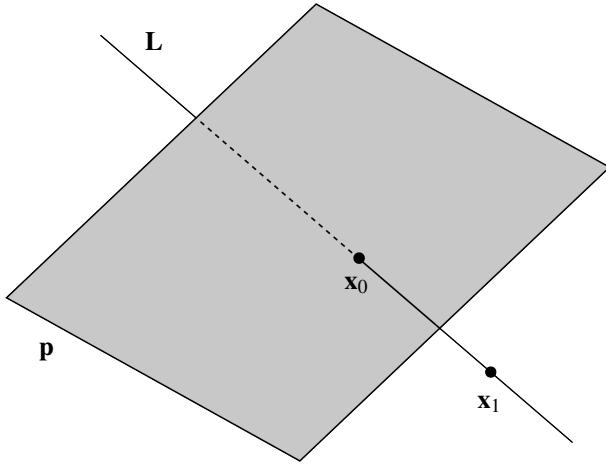


Figure 5.5: The point x_0 is the intersection of the line \mathbf{L} with the plane \mathbf{p} .

5.4.1 The intersection of a line with a plane

Let \mathbf{L} be the Plücker coordinates of a line and let \mathbf{p} be the dual homogeneous coordinates of a plane, both in \mathbb{E}^3 . What is the point of intersection, x_0 , between the line and the plane? Since x_0 must lie on the line \mathbf{L} , we can write

$$\mathbf{L} \sim x_0 x_1^\top - x_1 x_0^\top, \quad (5.36)$$

where x_1 is another point on the line, distinct from x_0 , see Figure 5.5. Since the point x_0 lies in the plane \mathbf{p} , it follows that $x_0 \cdot \mathbf{p} = 0$. Consequently:

$$\mathbf{L} \mathbf{p} \sim (x_0 x_1^\top - x_1 x_0^\top) \mathbf{p} = x_0(x_1 \cdot \mathbf{p}) - x_1 \underbrace{(x_0 \cdot \mathbf{p})}_{=0} = x_0(x_1 \cdot \mathbf{p}) \sim x_0. \quad (5.37)$$

This result assumes that $x_1 \cdot \mathbf{p} \neq 0$, i.e., x_1 does not lie in the plane \mathbf{p} . In the general case, and since x_1 and x_0 are distinct, we can always find such a point, and then $\mathbf{L} \mathbf{p}$ gives the homogeneous coordinates of the requested point.

In the special case that the line \mathbf{L} lies in the plane \mathbf{p} , any x_1 on the line gives $x_1 \cdot \mathbf{p} = 0$, leading to $\mathbf{L} \mathbf{p} = \mathbf{0}$. The zero vector flags that the result is ambiguous: there is no distinct point of intersection that can be determined since the line \mathbf{L} lies in the plane \mathbf{p} . An alternative formulation of this observation is:

$$\text{The line } \mathbf{L} \text{ lies in the plane } \mathbf{p} \Leftrightarrow \mathbf{L} \mathbf{p} = \mathbf{0}. \quad (5.38)$$

Points at infinity revisited

In Section 3.5.1, points at infinity are introduced and defined as the intersection of two distinct but parallel lines in \mathbb{E}^2 . For the 3D case, in Section 5.1 we have said that a point at infinity has a specific form of homogeneous representation, which intuitively can be seen at points at infinite distance from the origin and in a certain orientation.

Given the results derived here, an alternative interpretation of points at infinity can be made: as the intersection of a line and a plane where the line is parallel to the plane, i.e., a tangent of the line is orthogonal to the plane. This result can be obtained in a similar way as in Section 3.5.1, by starting with a plane and a line, where the line is almost parallel to the plane. The intersection between the two is then a point somewhere in the plane. By making the line more and more parallel to the plane, the point moves further and further away from the origin. In the limit case, it will have moved to an infinite distance but still in a orientation specified by the line.

5.4.2 The plane that intersects a line and a point

Let \mathbf{L} be the Plücker coordinates of a line and let \mathbf{x} be the homogeneous coordinates of a point, both in \mathbb{E}^3 . What plane \mathbf{p}_0 intersects both the line and the point? Consider the dual Plücker coordinates of the line, $\tilde{\mathbf{L}}$. Since \mathbf{p}_0

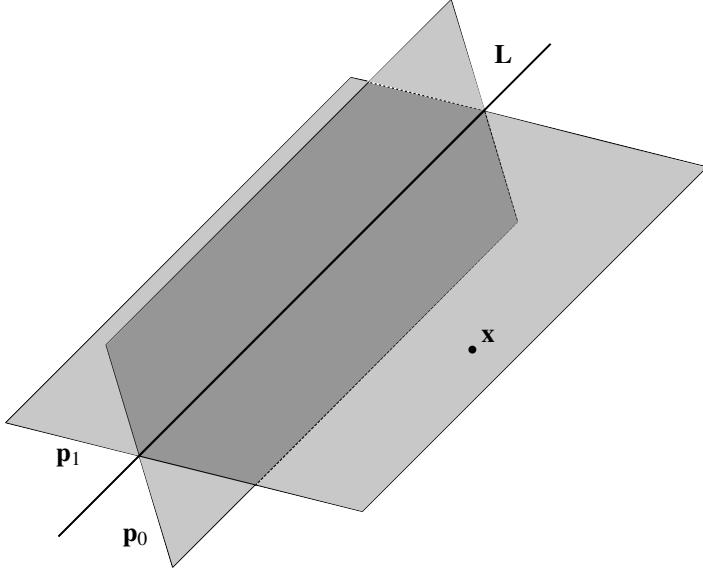


Figure 5.6: The plane \mathbf{p}_1 is constructed as including both the line \mathbf{L} and the point \mathbf{x} . \mathbf{p}_0 is a second plane, used here to define \mathbf{L} as the intersection of \mathbf{p}_1 and \mathbf{p}_0 .

intersects the line, we can write

$$\tilde{\mathbf{L}} \sim \mathbf{p}_0 \mathbf{p}_1^\top - \mathbf{p}_1 \mathbf{p}_0^\top, \quad (5.39)$$

where \mathbf{p}_1 is another plane that intersects the line, too, see Figure 5.6. Since the plane \mathbf{p}_0 includes the point \mathbf{x} , it follows that $\mathbf{p}_0 \cdot \mathbf{x} = 0$. Consequently:

$$\tilde{\mathbf{L}} \mathbf{x} \sim (\mathbf{p}_0 \mathbf{p}_1^\top - \mathbf{p}_1 \mathbf{p}_0^\top) \mathbf{x} = \mathbf{p}_0(\mathbf{p}_1 \cdot \mathbf{x}) - \mathbf{p}_1 \underbrace{(\mathbf{p}_0 \cdot \mathbf{x})}_{=0} = \mathbf{p}_0(\mathbf{p}_1 \cdot \mathbf{x}) \sim \mathbf{p}_0. \quad (5.40)$$

This result assumes that $\mathbf{p}_1 \cdot \mathbf{x} \neq 0$, i.e., \mathbf{p}_1 does not include the point \mathbf{x} . In the general case, and since \mathbf{p}_1 and \mathbf{p}_0 are distinct, we can always find such a plane, and then $\tilde{\mathbf{L}} \mathbf{x}$ gives the dual homogeneous coordinates of the requested plane.

In the special case that the line $\tilde{\mathbf{L}}$ intersects the point \mathbf{x} , any \mathbf{p}_1 that intersects the line gives $\mathbf{p}_1 \cdot \mathbf{x} = 0$, leading to $\tilde{\mathbf{L}} \mathbf{x} = \mathbf{0}$. The zero vector flags that the result is ambiguous: there is no distinct plane that can be determined since the point \mathbf{x} lies on the line \mathbf{L} . An alternative formulation of this observation is:

$$\text{The line } \tilde{\mathbf{L}} \text{ includes the point } \mathbf{x} \iff \tilde{\mathbf{L}} \mathbf{x} = \mathbf{0}. \quad (5.41)$$

5.4.3 The point that intersects three planes

Let $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ be three planes in \mathbb{E}^3 . In general, the planes intersect at a point \mathbf{x} , and we want to determine this point, see Figure 5.7. The algebraic formulation of this problem is that we seek the homogeneous coordinates of a point \mathbf{x} such that

$$\mathbf{p}_1 \cdot \mathbf{x} = \mathbf{p}_2 \cdot \mathbf{x} = \mathbf{p}_3 \cdot \mathbf{x} = 0. \quad (5.42)$$

In the 2D case, this corresponds to finding the point \mathbf{y} that is the intersection of two lines \mathbf{l}_1 and \mathbf{l}_2 , given as $\mathbf{y} \sim \mathbf{l}_1 \times \mathbf{l}_2$. We need a generalization of the cross product to \mathbb{R}^4 to repeat this result for this current problem.

Although it is possible to define such an operation, the theoretical framework required for doing that is outside the scope of this presentation⁴. Instead we make use of the result derived in Section 5.4.1 and form $\tilde{\mathbf{L}}$, the dual Plücker coordinates of the line that lies in the intersection of \mathbf{p}_1 and \mathbf{p}_2 . Then, we apply the duality operator to get

⁴The interested reader should consult texts on geometric algebra or Clifford algebra.

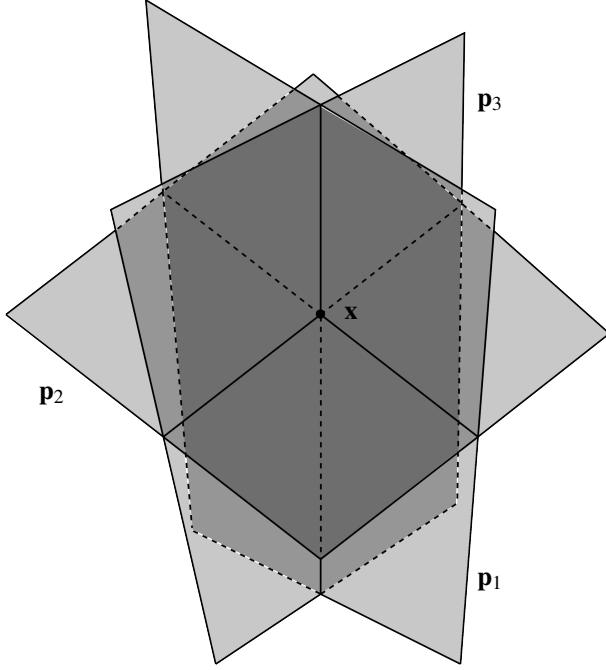


Figure 5.7: The planes, $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ intersect at the point \mathbf{x} .

the standard Plücker coordinates of the same line as $\mathbf{L} = \tilde{\mathbf{L}}$. Finally, the point \mathbf{x} that we seek is the intersection of \mathbf{L} with the third plane \mathbf{p}_3 , given as $\mathbf{x} = \mathbf{L}\mathbf{p}_3$. In summary:

$$\mathbf{x} \sim \underbrace{\left(\mathbf{p}_1 \mathbf{p}_2^\top - \mathbf{p}_2 \mathbf{p}_1^\top \right)}_{\mathbf{L}} \mathbf{p}_3. \quad (5.43)$$

Although the expression in the right-hand side appears to depend on how the three planes are labeled, it is invariant to permutations of the planes. Finally, the right-hand side equals $\mathbf{0} \in \mathbb{R}^4$ if, and only if, the three planes intersect along a line.

5.4.4 The plane that intersects three points

Let $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ be three points in \mathbb{E}^3 . In general, the three points intersect a unique plane \mathbf{p} , and we want to determine this plane, see Figure 5.8. The algebraic formulation of this problem is that we seek the dual homogeneous coordinates of the plane \mathbf{p} such that

$$\mathbf{x}_1 \cdot \mathbf{p} = \mathbf{x}_2 \cdot \mathbf{p} = \mathbf{x}_3 \cdot \mathbf{p} = 0. \quad (5.44)$$

We can analyze this problem in a similar way to the problem in the last section. Hence, we form the Plücker coordinates \mathbf{L} of the line that passes through \mathbf{x}_1 and \mathbf{x}_2 , then apply the duality mapping to obtain the corresponding dual Plücker coordinates $\tilde{\mathbf{L}}$. Finally, in accordance with Section 5.4.5 we get the plane as $\mathbf{p} = \tilde{\mathbf{L}}\mathbf{x}_3$. In summary:

$$\mathbf{p} \sim \underbrace{\left(\mathbf{x}_1 \mathbf{x}_2^\top - \mathbf{x}_2 \mathbf{x}_1^\top \right)}_{\tilde{\mathbf{L}}} \mathbf{x}_3. \quad (5.45)$$

The right-hand side equals $\mathbf{0} \in \mathbb{R}^4$ if, and only if, the three points are co-linear.

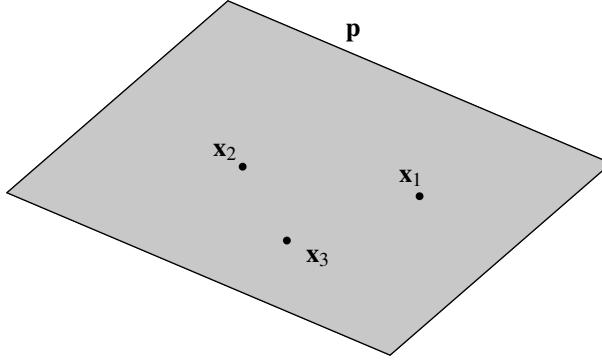


Figure 5.8: The plane \mathbf{p} intersects three points, $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$.

5.4.5 Points on a plane, or not

Three distinct points always intersect with a plane in \mathbb{E}^3 , but what if we have four or more points? How can we test if they are all on a plane? And if they are, what plane is it? Let $\mathbf{x}_1, \dots, \mathbf{x}_m$ be the homogeneous coordinates of m points in \mathbb{E}^3 . If they all are lying on one and the same plane \mathbf{p} , it must be the case that $\mathbf{x}_k \cdot \mathbf{p} = 0$ for $k = 1, \dots, m$. A more compact way of describing this statement is to first form the $4 \times m$ matrix $\mathbf{X} = (\mathbf{x}_1 \dots \mathbf{x}_m)$, holding the homogeneous coordinates in its columns. If all the points lie on the plane \mathbf{p} , this is equivalent to

$$\mathbf{X}^\top \mathbf{p} = \mathbf{0}. \quad (5.46)$$

This means that \mathbf{p} lies in the null space of \mathbf{X}^\top , but it also means that a necessary and sufficient condition for the existence of a plane that intersects all m points is that \mathbf{X} is rank deficient. Assuming that the points are distinct and co-planar, but not co-linear, the null space of \mathbf{X}^\top is 1-dimensional and corresponds to the homogeneous coordinates of \mathbf{p} . In the special case that $m = 4$, we can formulate the necessary and sufficient condition the existence of an intersecting plane as: $\det(\mathbf{X}) = 0$. In the general case, the necessary and sufficient condition can instead be expressed as $\det(\mathbf{X} \mathbf{X}^\top) = 0$.

A computational approach for solving the problem discussed here is to do an SVD of \mathbf{X}^\top . The rank of \mathbf{X}^\top must be 1, 2, 3, or 4, and the solution to the posed problem can be described by the following four cases:

- **Rank 4:** $\text{Null}(\mathbf{X}^\top)$ is trivial and does not contain the dual homogeneous coordinates of any plane. This indicates that the points are not co-planar.
- **Rank 3:** $\text{Null}(\mathbf{X}^\top)$ is 1-dimensional and corresponds to a unique projective element \mathbf{p} that solves Equation (5.46). This implies that the points are co-planar but not co-linear, and the projective element \mathbf{p} is unique and is represented by any right singular vector of \mathbf{X}^\top corresponding to the singular value = 0.
- **Rank 2:** $\text{Null}(\mathbf{X}^\top)$ is 2-dimensional. This indicates that the points are co-linear, they lie on a well-defined line. Consequently, there are infinitely many planes that intersect that line, and their dual homogeneous coordinates span $\text{Null}(\mathbf{X}^\top)$: any linear combination of two linearly independent null vectors of \mathbf{X}^\top represents a valid plane.
- **Rank 1:** $\text{Null}(\mathbf{X}^\top)$ is 3-dimensional. This indicates that the points are not distinct, they refer to one and the same point. Consequently, there are infinitely many planes that intersect that single point, with all possible orientations, and their dual homogeneous coordinates span $\text{Null}(\mathbf{Y}^\top)$: any linear combination of three linearly independent null vectors of \mathbf{Y}^\top represents a valid plane.

Before looking at the dual problem, of planes intersecting at a point, we remind about the discussion in Section 3.3.3, where it was said that the issue of whether singular values are equal to zero or not in practice is a delicate matter. Especially, this is the case when the positions of points or the parameters of planes cannot be determined with arbitrary accuracy.

5.4.6 Planes through a point, or not

The results of the last section have a similar algebraic form if we instead consider a set of m planes, $\mathbf{p}_1, \dots, \mathbf{p}_m$, and ask the question: do they all intersect at a common point? And if they do, what point is it? The intersection point \mathbf{x} must satisfy

$$\tilde{\mathbf{X}}^\top \mathbf{x} = \mathbf{0}, \quad (5.47)$$

where $\tilde{\mathbf{X}} = (\mathbf{p}_1 \dots \mathbf{p}_m)$ holds the dual homogeneous coordinates of the planes in its columns. A necessary and sufficient condition for the existence of a common point of intersection is that $\tilde{\mathbf{X}}$ is rank deficient. In the case that $m = 4$, this condition is equivalent to $\det(\tilde{\mathbf{X}}) = 0$, and in the general case to $\det(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top) = 0$.

In the case that \mathbf{x} exists, it is found in $\text{Null}(\tilde{\mathbf{X}}^\top)$ which, in turn, can be determined from an SVD of $\tilde{\mathbf{X}}^\top$. There are 4 possible cases, depending on the rank of $\tilde{\mathbf{X}}^\top$:

- **Rank 4:** $\text{Null}(\tilde{\mathbf{X}}^\top)$ is trivial and does not contain the homogeneous coordinates of any point. This indicates that the planes do not intersect at a common point.
- **Rank 3:** $\text{Null}(\tilde{\mathbf{X}}^\top)$ is 1-dimensional and corresponds to a unique projective element \mathbf{x} that solves Equation (5.47). This implies that the planes intersect at a common point, and the projective element \mathbf{x} is unique and represented by any right singular vector of $\tilde{\mathbf{X}}^\top$ corresponding to the singular value = 0.
- **Rank 2:** $\text{Null}(\tilde{\mathbf{X}}^\top)$ is 2-dimensional. This indicates that the planes are co-linear, they intersect at a well-defined line. Consequently, there are infinitely many points that lie on all planes, and their homogeneous coordinates span $\text{Null}(\tilde{\mathbf{X}}^\top)$, any linear combination of two linearly independent null vectors of $\tilde{\mathbf{X}}^\top$ represents a valid point.
- **Rank 1:** $\text{Null}(\tilde{\mathbf{X}}^\top)$ is 3-dimensional. This indicates that the planes are not distinct, they refer to one and the same plane. Consequently, there are infinitely many points that lie on that single plane, and their homogeneous coordinates span $\text{Null}(\tilde{\mathbf{X}}^\top)$, any linear combination of three linearly independent null vectors of $\tilde{\mathbf{X}}^\top$ represents a valid point.

5.4.7 The intersection of two lines

Let \mathbf{L}_1 and \mathbf{L}_2 be two lines in \mathbb{E}^3 . In general, the two lines do not have to intersect, but how can we know if this happens or not? Notice that the case where the two lines intersect is equivalent to the case that they are co-planar, i.e., lie in a common plane. This means that the current problem can be rephrased as: how can we determine if \mathbf{L}_1 and \mathbf{L}_2 lie in a common plane or not? We show first that there is a necessary condition for the lines to intersect, or lie in a common plane.

We assume that the two lines intersect. Consider the dual Plücker coordinate $\tilde{\mathbf{L}}_1$ of the first line, which can be written

$$\tilde{\mathbf{L}}_1 \sim \mathbf{p}_1 \mathbf{p}_2^\top - \mathbf{p}_2 \mathbf{p}_1^\top. \quad (5.48)$$

Here, \mathbf{p}_1 and \mathbf{p}_2 are two arbitrary but distinct planes that intersect the line $\tilde{\mathbf{L}}_1$. Since the two lines intersect, we can choose \mathbf{p}_1 as the plane that includes both lines, and \mathbf{p}_2 as another distinct plane that only needs to include the first line. The Plücker coordinates of the second line, \mathbf{L}_2 , is given as

$$\mathbf{L}_2 \sim \mathbf{x}_1 \mathbf{x}_2^\top - \mathbf{x}_2 \mathbf{x}_1^\top, \quad (5.49)$$

where \mathbf{x}_1 and \mathbf{x}_2 are two distinct points on the line. Since \mathbf{p}_1 includes the second line it follows that $\mathbf{p}_1 \cdot \mathbf{x}_1 = \mathbf{p}_1 \cdot \mathbf{x}_2 = 0$. The Frobenius scalar product between $\tilde{\mathbf{L}}_1$ and \mathbf{L}_2 can then be expanded as

$$\tilde{\mathbf{L}}_1 \cdot \mathbf{L}_2 = \text{trace}(\tilde{\mathbf{L}}_1^\top \mathbf{L}_2) = 2 \left(\underbrace{(\mathbf{p}_1 \cdot \mathbf{x}_1)(\mathbf{p}_2 \cdot \mathbf{x}_2)}_{=0} - \underbrace{(\mathbf{p}_1 \cdot \mathbf{x}_2)(\mathbf{p}_2 \cdot \mathbf{x}_1)}_{=0} \right) = 0. \quad (5.50)$$

Consequently, Equation (5.50) is a necessary condition for the two lines to intersect.

In fact, the condition in Equation (5.50) is also sufficient. To prove this, we now assume that Equation (5.50) is satisfied:

$$0 = \tilde{\mathbf{L}}_1 \cdot \mathbf{L}_2 = \text{trace}(\tilde{\mathbf{L}}_1^\top \mathbf{L}_2) = \text{trace}((\mathbf{p}_1 \mathbf{p}_2^\top - \mathbf{p}_2 \mathbf{p}_1^\top)^\top \mathbf{L}_2) = \mathbf{p}_1^\top \mathbf{L}_2 \mathbf{p}_2 - \mathbf{p}_2^\top \mathbf{L}_2 \mathbf{p}_1 = 2 \mathbf{p}_1^\top \mathbf{L}_2 \mathbf{p}_2. \quad (5.51)$$

• The Frobenius scalar product is defined in Toolbox Section 3.4.1.

One way to satisfy this equation is to assume that $\mathbf{L}_2 \mathbf{p}_2 = 0$, i.e., the line \mathbf{L}_2 lies in \mathbf{p}_2 , Equation (5.38). But \mathbf{p}_2 already includes the line \mathbf{L}_1 , which means that both \mathbf{L}_1 and \mathbf{L}_2 lie in the plane \mathbf{p}_2 , i.e., the two lines intersect. The only other alternative is that $\mathbf{x} = \mathbf{L}_2 \mathbf{p}_2$ is the unique point of intersection between the line \mathbf{L}_2 and the plane \mathbf{p}_2 , and \mathbf{x} lies in the plane \mathbf{p}_1 , i.e., $\mathbf{p}_1 \cdot \mathbf{x} = 0$. This means that \mathbf{x} lies on the line \mathbf{L}_2 , and in both the planes \mathbf{p}_1 and \mathbf{p}_2 . Consequently, \mathbf{x} lies also on the line \mathbf{L}_1 , i.e., it is the point of intersection between \mathbf{L}_1 and \mathbf{L}_2 .

The results derived here are symmetric in the two lines, and they can be summarized as

5.3 Two 3D lines \mathbf{L}_1 and \mathbf{L}_2 intersect if and only if

$$\mathbf{L}_1 \text{ and } \mathbf{L}_2 \text{ lie in a common plane} \iff \tilde{\mathbf{L}}_1 \cdot \mathbf{L}_2 = 0 \iff \mathbf{L}_1 \cdot \tilde{\mathbf{L}}_2 = 0. \quad (5.52)$$

5.5 Distances

In this section we extend the discussion Section 3.4 to distances between various types of geometric objects in \mathbb{E}^3 . In the same way as for the 2D case, we assume here that all points, planes, and lines are proper ones. The concept of distance does not make sense when we are dealing with points, lines, or the plane at infinity.

5.5.1 Distance between two points

Using the same arguments that were presented for the 2D case in Section 3.4.1, we formulate the point-to-point distance in \mathbb{E}^3 as

$$d_{PP}(\mathbf{x}_1, \mathbf{x}_2) = \|\text{cart}(\mathbf{x}_1) - \text{cart}(\mathbf{x}_2)\| = \|\text{norm}_{\mathbf{P}}(\mathbf{x}_1) - \text{norm}_{\mathbf{P}}(\mathbf{x}_2)\|. \quad (5.53)$$

Here, \mathbf{x}_1 and \mathbf{x}_2 are the homogeneous coordinates of two proper points in \mathbb{E}^3 . Formally, there is a distinction between the distance functions in Equation (3.31) and in Equation (5.53), both with the same name but applied to different types of variables. In the following presentation, we rely on the context to determine which of the two functions that is used in a particular expression.

5.5.2 Signed distance between a point and a plane

Section 3.4.2 describes a signed distance between a point and a line in \mathbb{E}^2 . Let \mathbf{x} and \mathbf{p} be the homogeneous representations of a point and a plane in \mathbb{E}^3 . Using the same arguments as in the 2D case, we define a signed distance between the point and the plane as

$$d_{PD}(\mathbf{x}, \mathbf{p}) = \text{norm}_{\mathbf{P}}(\mathbf{x}) \cdot \text{norm}_{\mathbf{D}}(\mathbf{p}). \quad (5.54)$$

It is a straightforward exercise to show that $|d_{PD}(\mathbf{x}, \mathbf{p})|$ equals the Euclidean distance from the point to the plane, measured in the direction perpendicular to the plane.

We can also show that

$$d_{PD}(\mathbf{x}, \mathbf{p}) \begin{cases} > 0 & \text{when the point and the origin lie on opposite sides of the plane,} \\ = 0 & \text{when the point lies in the plane,} \\ < 0 & \text{when the point and the origin lie on the same side of the plane,} \end{cases} \quad (5.55)$$

when the plane does not intersect the origin. Furthermore, by checking the signs of the signed distances from two points to the same plane, it is possible to determine if they lie on the same side or on opposite sides of the plane.

5.5.3 Distance between a point and a line

Let \mathbf{L} be a proper line and \mathbf{x} a proper point in \mathbb{E}^3 . What is the distance from the point to the line, measured perpendicular to the line? See Figure 5.9 for an illustration. Let \mathbf{p}_1 and \mathbf{p}_2 be two planes that define the corresponding dual Plücker coordinates $\tilde{\mathbf{L}}$, where \mathbf{p}_1 is a plane that intersects with both the line $\tilde{\mathbf{L}}$ and the point \mathbf{x} . We get

$$\tilde{\mathbf{L}} \mathbf{x} = (\mathbf{p}_1 \mathbf{p}_2^\top - \mathbf{p}_2 \mathbf{p}_1^\top) \mathbf{x} = \mathbf{p}_1 (\mathbf{p}_2 \cdot \mathbf{x}) - \underbrace{\mathbf{p}_2 (\mathbf{p}_1 \cdot \mathbf{x})}_{=0} = \mathbf{p}_1 (\mathbf{p}_2 \cdot \mathbf{x}). \quad (5.56)$$

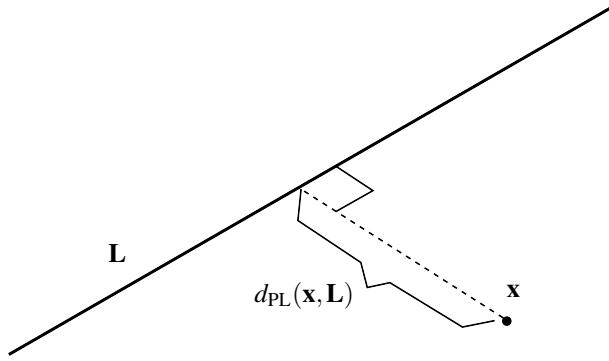


Figure 5.9: The distance $d_{PL}(x, L)$ is measured from the point x perpendicular to the line L .

Next, we apply a DL-normalization on $\tilde{\mathbf{L}}$, in accordance with Section 5.3.7, and a P-normalization on \mathbf{x} . This means that $\text{norm}_{DL}(\tilde{\mathbf{L}})$ is given by Equation (5.31) for any two D-normalized vectors \mathbf{p}_1 and \mathbf{p}_2 representing two perpendicular planes that intersect along the line. For example, we can choose the plane \mathbf{p}_1 as before, intersecting both the line L and the point x , and the second plane \mathbf{p}_2 as perpendicular to the plane \mathbf{p}_1 . Using these properties in Equation (5.56) gives:

$$\text{norm}_{DL}(\tilde{\mathbf{L}}) \text{ norm}_P(\mathbf{x}) = \text{norm}_D(\mathbf{p}_1) \underbrace{(\text{norm}_D(\mathbf{p}_2) \cdot \text{norm}_P(\mathbf{x}))}_{=d} = d \cdot \text{norm}_D(\mathbf{p}_1). \quad (5.57)$$

Here, d is the signed distance between point x and plane \mathbf{p}_2 . Since the plane \mathbf{p}_2 is perpendicular to the plane \mathbf{p}_1 , and the latter includes the point x , it follows that $|d|$ is the requested distance between the point and the line. The vector \mathbf{p}_1 is D-normalized, i.e., the first three elements have unit norm, leading to the following expression for the point-to-line distance in \mathbb{E}^3 :

$$d_{PL}(x, \tilde{\mathbf{L}}) = |d| = \|(\text{norm}_{DL}(\tilde{\mathbf{L}}) \text{ norm}_P(\mathbf{x}))_{1:3}\|. \quad (5.58)$$

In this expression, $\text{norm}_{DL}(\tilde{\mathbf{L}}) \text{ norm}_P(\mathbf{x})$ is a vector in \mathbb{R}^4 , and the right-hand side of Equation (5.58) is the norm of the first three elements of this vector.

Intersection of a point and a line in \mathbb{E}^3

As a straightforward side-effect of the above results, we can formulate a sufficient and necessary condition for the intersection of a point x and a line $\tilde{\mathbf{L}}$ in \mathbb{E}^3 as $\tilde{\mathbf{L}}\mathbf{x} = \mathbf{0}$. This corresponds to the distance $d = 0$ in Equation (5.57), but here we can skip the normalizations that only scale an expression that is equal to zero. This is consistent with the result already derived in Section 5.4.2.

5.6 The duality principle in \mathbb{E}^3

Section 3.8 formulates a duality principle for homogeneous representation of geometric objects in \mathbb{E}^2 : in every correct geometric statement we can swap “point” and “line” and get a new correct statement. Given the presentation of homogeneous representations for the geometry of \mathbb{E}^3 made in this chapter, it should be clear that there is a similar duality principle also for this case: in every correct geometric statement we can swap “point” and “plane” and get a new correct statement. For an example, notice the similarity of the text in Sections 5.4.1 and 5.4.2, in Sections 5.4.3 and 5.4.4, and in Sections 5.4.5 and 5.4.6. In the corresponding algebraic relations, we also swap the homogeneous representation of a point, \mathbf{x} , with the homogeneous representation of a plane, \mathbf{p} . To become complete, we must also deal with the homogeneous representations of a 3D line, in terms of Plücker coordinates: we swap the standard Plücker coordinates \mathbf{L} with the dual Plücker coordinates $\tilde{\mathbf{L}}$.

Chapter 6

Transformations in 3D

Before you read this chapter, you should have a thorough understanding of the representations of transformations in \mathbb{E}^2 , of both points and lines, which are presented in Chapter 4. You may also want to have a look at Toolbox Section 8.6.2 and Toolbox Section 8.6.4 that show how $SO(3)$ can be parameterized by the 3 free parameters of an anti-symmetric matrix in $so(3)$.

The presentation on various types of 2D transformations in Chapter 4 is here extended in a more or less straightforward way to 3D transformations. There are only minor differences, mainly in the degrees of freedom for the different types of transformations. This means that the presentation in this chapter is rather brief, and refers to the more detailed discussions in Chapter 4.

6.1 Degrees of freedom

Using the same argumentation as in Section 4.1.1 we define the degree of freedom of a 3D transformation as the minimum number of parameters, r , which are needed to determine a specific transformation. In the 3D case each point that is specified before and after the transformation provides 3 independent equations in these parameters. Consequently, we need at least $r/3$ such points in general configuration to fully determine all r parameters of a transformation. In some cases, however, there may be circumstances that require us to use more than this minimum number of points.

6.2 Rigid transformations

A rigid transformation in \mathbb{E}^3 is represented by a 4×4 matrix of the form

$$\mathbf{T} \sim \begin{pmatrix} \mathbf{R} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix}. \quad (6.1)$$

Here, $\mathbf{R} \in SO(3)$ describes the rotation part of the transformation and $\bar{\mathbf{t}} \in \mathbb{R}^3$ describes the translation part of the transformation. The decomposition of Equation (6.1) into a rotation and a translation implies that the rotation defined by \mathbf{R} is applied first, followed by the translation defined by $\bar{\mathbf{t}}$. This set of linear transformations form the rigid transformation group, which preserves distances, angles, areas, and volumes in \mathbb{E}^3 . The group is also referred to as the *special Euclidean transformation group* in \mathbb{E}^3 , or $SE(3)$ for short. It has 6 degrees of freedom: 3 from the rotation part¹ and 3 from the translation part. The above rule for the minimum number of points implies that we need at least two points correspondences, before and after the rigid transformation, to determine the transformation. This is correct, but 2 is not the minimum number in this case. The reason is that the two point correspondences

¹This follows from the results in Toolbox Section 8.6.2 and Toolbox Section 8.6.4. A more general overview on how $SO(3)$ can be parameterized is presented in Chapter 11.

cannot be chosen independently. They have to satisfy the rigidity constraint: the distance between the two points must be preserved after the transformation. Therefore, we need a minimum of three point correspondences to determine a 3D rigid transformation.

6.3 Similarity transformations

A similarity transformation in \mathbb{E}^3 is represented by a 4×4 matrix of the form

$$\mathbf{T} \sim \begin{pmatrix} s\mathbf{R} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix}. \quad (6.2)$$

Here, $\mathbf{R} \in SO(3)$ describes the rotation part of the transformation, $\bar{\mathbf{t}} \in \mathbb{R}^3$ describes the translation part of the transformation, and $s \in \mathbb{R}$ describes a uniform scaling. Furthermore, we assume that $s > 0$ to assure that the transformation preserves the handedness of an object in \mathbb{E}^3 . In other presentations, it may be the case that s is a non-zero scalar, which allows for more general transformations that also can change the handedness, e.g., of a basis. In the 2D case, scaling by a negative s corresponds to a rotation by 180° followed by a scaling by $|s|$. In the 3D case, however, a negative s cannot be interpreted as a 3D rotation and a scaling by $|s|$. Instead, we must also include reflections into the transformations if s is allowed to be negative.

A similarity transformation in \mathbb{E}^3 has 7 degrees of freedom, 6 from the rigid transformation described by \mathbf{R} and $\bar{\mathbf{t}}$, and one additional degree from the scaling parameters s . This means that at least 3 points are required to determine a specific similarity transformation in \mathbb{E}^3 . A similarity transformation preserves angles and as well as fractions between two distances, areas, or volumes.

6.4 Affine transformations

An affine transformation in \mathbb{E}^3 is represented by a 4×4 matrix of the form

$$\mathbf{T} \sim \begin{pmatrix} \mathbf{A} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix}. \quad (6.3)$$

Here, $\mathbf{A} \in GL(3)$ describes the linear part of the transformation, and $\bar{\mathbf{t}} \in \mathbb{R}^3$ describes the translation part of the transformation.

An affine transformation in \mathbb{E}^3 preserves parallel lines or planes in \mathbb{E}^3 as parallel, and proper points, lines, or planes, as proper. Similarly, these transformations make points, lines, and the plane at infinity to all stay at infinity. An affine transformation in \mathbb{E}^3 has 12 degrees of freedom: 9 from the 3×3 matrix \mathbf{A} and 3 more from the vector $\bar{\mathbf{t}}$. This means that 4 points are sufficient for determining an affine transformation in \mathbb{E}^3 .

Specific classes of affine transformations are sometimes of interest, for example, non-uniform scaling, reflections, and shearing transformations.

Non-uniform scaling

Affine transformations include non-uniform scaling that is characterized by $\bar{\mathbf{t}} = \mathbf{0}$ and

$$\mathbf{A} = \begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{pmatrix}, \quad \Rightarrow \quad \mathbf{T} \sim \begin{pmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \quad (6.4)$$

Here, s_1, s_2, s_3 are the independent non-zero scaling parameters for each of the three dimensions of the coordinate system.

Non-uniform scaling can also be performed along three arbitrary but orthogonal axes, possibly intersecting at an arbitrary point. This is done by first applying a rigid transformation of the coordinate system to align it with the chosen axes, then apply the non-uniform scaling \mathbf{T} in Equation (6.4) and, finally, apply the inverse rigid transformation to return to the original coordinate system.

Reflection in a plane

In \mathbb{E}^3 a reflection can be defined relative to a plane, such that the reflection transformation leaves the orthogonal projection of a point onto the plane unchanged, but the signed distance to the plane changes sign. Let

$$\mathbf{p} = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{p}} \\ -\Delta \end{pmatrix}, \quad (6.5)$$

be the D-normalized dual homogeneous coordinates of a plane in \mathbb{E}^3 , i.e., $\hat{\mathbf{p}}$ is a normal vector to the plane of unit norm, and $\Delta \geq 0$ is the distance between the plane and the origin. The corresponding reflection transformation is then represented by the matrix

$$\mathbf{T} \sim \begin{pmatrix} \mathbf{I} - 2\hat{\mathbf{p}}\hat{\mathbf{p}}^\top & 2\Delta\hat{\mathbf{p}} \\ \mathbf{0} & 1 \end{pmatrix}. \quad (6.6)$$

The transformation matrix \mathbf{T} can be expressed as a quadratic function in the elements of \mathbf{p} :

$$\mathbf{T} \sim \begin{pmatrix} -p_1^2 + p_2^2 + p_3^2 & -2p_1p_2 & -2p_1p_3 & -2p_1p_4 \\ -2p_1p_2 & p_1^2 - p_2^2 + p_3^2 & -2p_2p_3 & -2p_2p_4 \\ -2p_1p_3 & -2p_2p_3 & p_1^2 + p_2^2 - p_3^2 & -2p_3p_4 \\ 0 & 0 & 0 & p_1^2 + p_2^2 + p_3^2 \end{pmatrix}. \quad (6.7)$$

We leave it as an exercise for the reader to prove the last two expressions.

Reflection in a line

As an alternative, in 3D we can instead reflect a point relative to a line. With $\hat{\mathbf{t}}$ a tangent vector of the line, and $\tilde{\mathbf{p}}_1$ and $\tilde{\mathbf{p}}_2$ two perpendicular normal vectors of the line, all three vectors normalized, the corresponding transformation matrix is

$$\mathbf{T} \sim \begin{pmatrix} \hat{\mathbf{t}}\hat{\mathbf{t}}^\top - \tilde{\mathbf{p}}_1\tilde{\mathbf{p}}_1^\top - \tilde{\mathbf{p}}_2\tilde{\mathbf{p}}_2^\top & 2\bar{\mathbf{x}}_0 \\ \mathbf{0} & 1 \end{pmatrix}, \quad (6.8)$$

where $\bar{\mathbf{x}}_0$ is the point on the line closest to the origin. Incidentally, this type of reflection is nothing but a 180° rotation about the line.

Shearing

Shearing in \mathbb{E}^3 is defined as an affine transformation, Equation (6.3), where $\det(\mathbf{A}) = 1$. This means that the volume of a body does not change after a shearing transformation.

Decomposition of an affine transformation

Similar to a 2D affine transformation, a 3D affine transformation can be decomposed into a sequence of a rotation, a non-uniform scaling, a second rotation, and a translation. Consequently, we can write

$$\mathbf{T} \sim \begin{pmatrix} \mathbf{A} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{I} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix}}_{\text{translation by } \bar{\mathbf{t}}} \begin{pmatrix} \mathbf{U} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \underbrace{\begin{pmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}}_{\text{non-uniform scaling by } \mathbf{S}} \underbrace{\begin{pmatrix} \mathbf{V}^\top & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}}_{\text{rotation by } \mathbf{V}^\top}. \quad (6.9)$$

Here, $\mathbf{U}, \mathbf{S}, \mathbf{V}$ are given by a special SVD of $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$, which guarantees $\mathbf{U}, \mathbf{V} \in SO(3)$. This type of decomposition is not unique.

6.5 Dual transformations

Dual transformations in 3D work in the same ways as for the 2D case. Let \mathbf{x} and \mathbf{p} be homogeneous representations of a point and a plane, respectively, relative to the same coordinate system in \mathbb{E}^3 . \mathbf{T} is a transformation of the homogeneous coordinates of the point to new coordinates $\mathbf{x}' = \mathbf{T}\mathbf{x}$, either by moving the point relative to the coordinate system, or by moving the coordinate system in \mathbb{E}^3 . The dual transformation $\tilde{\mathbf{T}}$ is then given as $\tilde{\mathbf{T}} = \mathbf{T}^{-\top}$, it gives the corresponding transformation of the plane's dual homogeneous coordinates: $\mathbf{p}' = \tilde{\mathbf{T}}\mathbf{p}$. With these transformations made, \mathbf{x}' and \mathbf{p}' refer to a point and a plane that both are transformed by the *same* transformation and relative to the *same* coordinate system.

6.6 Line transformations

Let \mathbf{T} be a 4×4 matrix that represents a transformation on the homogeneous coordinates of a point in \mathbb{E}^3 . Then $\tilde{\mathbf{T}} = \mathbf{T}^{-\top}$ represents the corresponding dual transformation that transforms the dual homogeneous coordinates of a plane. What about the Plücker coordinates of a line, how are they transformed in this case? And how are the dual Plücker coordinates of the same line transformed?

Let \mathbf{x}_1 and \mathbf{x}_2 be two points on the line \mathbf{L} :

$$\mathbf{L} \sim \mathbf{x}_1\mathbf{x}_2^\top - \mathbf{x}_2\mathbf{x}_1^\top. \quad (6.10)$$

We apply the transformation \mathbf{T} onto the two points to get

$$\mathbf{x}'_1 \sim \mathbf{T}\mathbf{x}_1, \quad \mathbf{x}'_2 \sim \mathbf{T}\mathbf{x}_2. \quad (6.11)$$

The Plücker coordinates \mathbf{L}' of the transformed line is then given as

$$\mathbf{L}' \sim \mathbf{x}'_1\mathbf{x}'_2^\top - \mathbf{x}'_2\mathbf{x}'_1^\top = \mathbf{T}(\mathbf{x}_1\mathbf{x}_2^\top - \mathbf{x}_2\mathbf{x}_1^\top)\mathbf{T}^\top = \mathbf{T}\mathbf{L}\mathbf{T}^\top. \quad (6.12)$$

This means that the transformation from \mathbf{L} to \mathbf{L}' is well-defined, although here it is not formulated in a conventional way as a single transformation applied from left onto \mathbf{L} , but rather as a *sandwich product* where \mathbf{T} and \mathbf{T}^\top are multiplied from left and right, respectively, onto \mathbf{L} in the middle.

The same arguments lead to

$$\tilde{\mathbf{L}}' \sim \mathbf{T}^{-\top}\tilde{\mathbf{L}}\mathbf{T}^{-1} = \tilde{\mathbf{T}}\tilde{\mathbf{L}}\tilde{\mathbf{T}}^\top, \quad (6.13)$$

where $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{L}}'$ are the dual Plücker coordinates of a line, before and after the transformation.

Chapter 7

Homographies

Before you read this chapter, you should have a thorough understanding of the homogeneous representations for \mathbb{E}^2 and \mathbb{E}^3 that are presented in Chapters 3 and 5.

This chapter wraps up the presentations on transformations in 2D and 3D made in Chapter 4 and Chapter 6 by describing the most general type of linear transformation on homogeneous coordinates: homographies. This very general type of transformation has a wide range of applications, as will be demonstrated in subsequent chapters.

7.1 Homographies in 2D

Consider two distinct planes in \mathbb{E}^3 , represented by their dual homogeneous coordinates \mathbf{p}_1 and \mathbf{p}_2 , and a point in \mathbb{E}^3 with homogeneous coordinates \mathbf{n} , the *projection point*, not lying in any of the two planes. Let \mathbf{x}_1 be an arbitrary point in \mathbf{p}_1 . It defines a 3D line \mathbf{L} that intersects both \mathbf{n} and \mathbf{x}_1 , and we refer to this line as a *projection line*. The projection line, in turn, intersects with \mathbf{p}_2 at some point \mathbf{x}_2 . We refer to \mathbf{x}_2 as the *projection* of \mathbf{x}_1 through the projection point \mathbf{n} onto \mathbf{p}_2 . Assuming that $\mathbf{p}_1, \mathbf{p}_2$, and \mathbf{n} are fixed, it follows that \mathbf{x}_2 is a function of \mathbf{x}_1 , as is illustrated in Figure 7.1. This function is referred to as a *homography transformation* or *homography* for short. In the literature this function is also referred to as a *projectivity* or as a *collineation*.

7.1 Geometrically, we define a homography as a mapping between two planes in \mathbb{E}^3 , by projecting through a fixed point \mathbf{n} .

A more algebraic representation of the mapping from \mathbf{x}_1 to \mathbf{x}_2 can be formulated by first defining the Plücker coordinates of the projection line:

$$\mathbf{L} \sim \mathbf{x}_1 \mathbf{n}^\top - \mathbf{n} \mathbf{x}_1^\top. \quad (7.1)$$

The point \mathbf{x}_2 is the intersection of the line \mathbf{L} with the plane \mathbf{p}_2 , and in accordance with the results derived in Section 5.4.2 it can be expressed as:

$$\mathbf{x}_2 \sim \mathbf{L} \mathbf{p}_2 = (\mathbf{x}_1 \mathbf{n}^\top - \mathbf{n} \mathbf{x}_1^\top) \mathbf{p}_2 = (\mathbf{n} \cdot \mathbf{p}_2) \mathbf{x}_1 - \mathbf{n} \mathbf{p}_2^\top \mathbf{x}_1 = ((\mathbf{n} \cdot \mathbf{p}_2) \mathbf{I} - \mathbf{n} \mathbf{p}_2^\top) \mathbf{x}_1. \quad (7.2)$$

Consequently, we can define a linear transformation, which depends on a plane \mathbf{p} and the projection point \mathbf{n} , as

$$\mathbf{M}(\mathbf{p}, \mathbf{n}) = (\mathbf{n} \cdot \mathbf{p}) \mathbf{I} - \mathbf{n} \mathbf{p}^\top, \quad (7.3)$$

such that

$$\mathbf{x}_2 \sim \mathbf{M}(\mathbf{p}_2, \mathbf{n}) \mathbf{x}_1. \quad (7.4)$$

The mapping \mathbf{M} , however, does not really capture the true nature of the homography. A homography it is not a mapping from a general 3D point \mathbf{x}_1 to a 3D point \mathbf{x}_2 . This algebraic construction of the homography does not take into account that \mathbf{x}_1 must be a point in \mathbf{p}_1 and \mathbf{x}_2 a point in \mathbf{p}_2 .

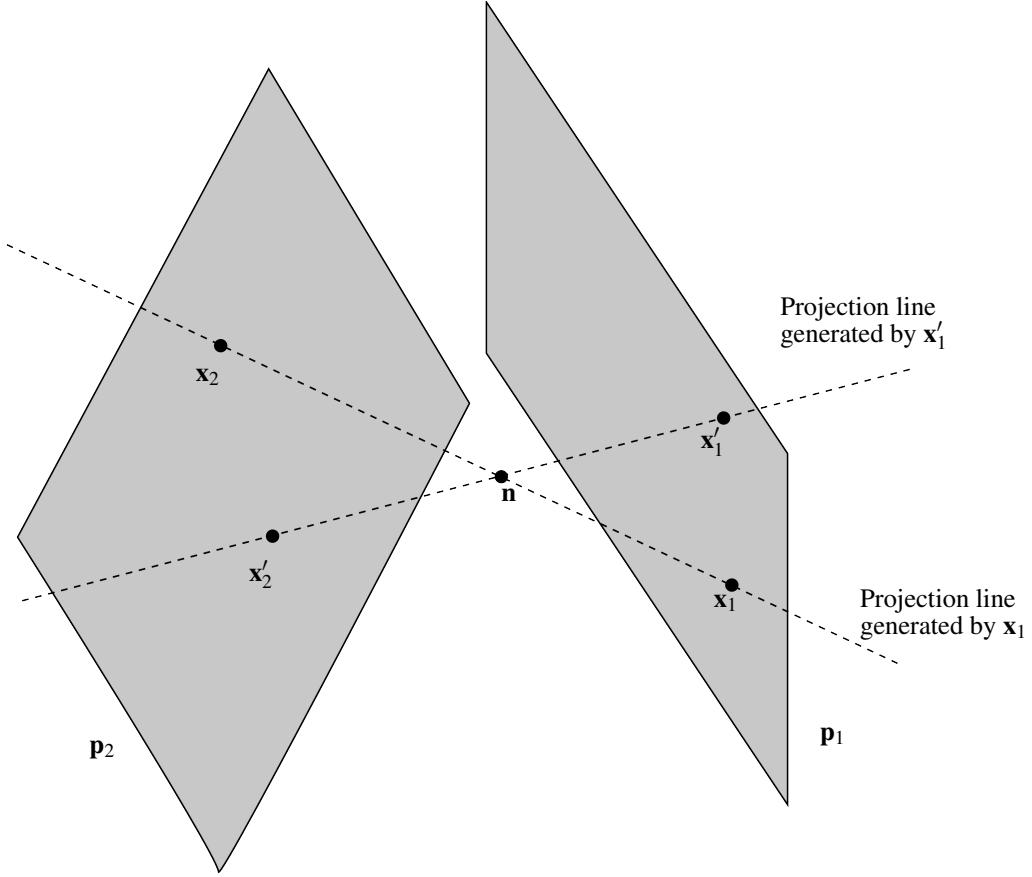


Figure 7.1: A homography maps the points \mathbf{x}_1 and \mathbf{x}'_1 in the plane \mathbf{p}_1 to points \mathbf{x}_2 and \mathbf{x}'_2 by projecting each point through \mathbf{n} and finding the intersection of the projection line with the plane \mathbf{p}_2 .

To make the fact that \mathbf{x}_1 lies in \mathbf{p}_1 and \mathbf{x}_2 lies in \mathbf{p}_2 explicit we will express each of the two points in terms of their Cartesian 2D coordinates relative to a coordinate system in each of the plane. Based on the discussion in Section 5.2.1, we introduce coordinate systems \mathbf{P}_1 and \mathbf{P}_2 in the planes \mathbf{p}_1 and \mathbf{p}_2 , respectively. This means that $\mathbf{x}_1 = \mathbf{P}_1 \mathbf{y}_1$, where \mathbf{y}_1 is the coordinates of \mathbf{x}_1 relative to coordinate system \mathbf{P}_1 , and similarly we write $\mathbf{x}_2 = \mathbf{P}_2 \mathbf{y}_2$. This allows us to formulate the homography transformation in terms of how \mathbf{y}_1 is mapped to \mathbf{y}_2 :

$$\mathbf{P}_2 \mathbf{y}_2 \sim \mathbf{M}(\mathbf{p}_2, \mathbf{n}) \mathbf{P}_1 \mathbf{y}_1, \quad \Rightarrow \quad \mathbf{y}_2 \sim \mathbf{P}_2^+ \mathbf{M}(\mathbf{p}_2, \mathbf{n}) \mathbf{P}_1 \mathbf{y}_1. \quad (7.5)$$

• Pseudo-inverses are defined in Toolbox Section 3.4.2.

Since \mathbf{P}_2 has full column rank its pseudo-inverse is well-defined. Consequently, we see that the mapping from \mathbf{y}_1 to \mathbf{y}_2 is represented by a 3×3 matrix \mathbf{H} given as

$$\mathbf{H} = \mathbf{P}_2^+ \mathbf{M}(\mathbf{p}_2, \mathbf{n}) \mathbf{P}_1. \quad (7.6)$$

\mathbf{H} is well-defined as soon as we have specified the two planes $\mathbf{p}_1, \mathbf{p}_2$, the projection point \mathbf{n} , and the coordinate systems in each of the two planes, \mathbf{P}_1 and \mathbf{P}_2 . In the following, we refer also to \mathbf{H} as a homography or a homography transformation, but this definition is not exactly equivalent to the first one. While the previous geometric definition only depends on the two planes and the projection point, this last definition also requires us to introduce a coordinate system for each plane. In practice this is not a big issue since a coordinate system always can be

defined when needed.

7.2 Algebraically, we can define a homography as the consequence of the geometric definition when coordinate systems has been defined in each of the two planes.

The geometric definition of a homography is symmetric relative to the two planes: we can equally well consider a point \mathbf{x}_2 in \mathbf{p}_2 which defines a projection line through \mathbf{n} that intersects \mathbf{p}_1 at the point \mathbf{x}_1 . If the mapping goes from \mathbf{p}_1 to \mathbf{p}_2 or the other way is just a matter of deciding which plane is labeled as “1” and which as “2”. Once we have made this choice and also introduced coordinate systems for each of the two planes, the matrix \mathbf{H} defines a mapping from plane \mathbf{p}_1 to plane \mathbf{p}_2 . The inverse mapping, from plane \mathbf{p}_2 to plane \mathbf{p}_1 , must then be given by \mathbf{H}^{-1} for the same coordinate systems \mathbf{P}_1 and \mathbf{P}_2 as before.

Since we assume that \mathbf{n} is not included neither in \mathbf{p}_1 nor in \mathbf{p}_2 , it follows that \mathbf{H} defined in Equation (7.6) is a general 3×3 and non-singular matrix: $\det(\mathbf{H}) \neq 0$. As usual in the context of homogeneous representations, the matrix is actually a representative of a projective element in $P(\mathbb{R}^{3 \times 3})$. Any non-zero scalar multiplied onto \mathbf{H} is also a representative of the same homography, since all equivalent matrices transform homogeneous coordinates in the same way. This observation leaves us with a third way of defining what a homography is: a general non-singular 3×3 matrix \mathbf{H} applied to homogeneous coordinates. In this case, \mathbf{H} does not have to be related to planes or points in \mathbb{E}^3 . In the case of proper points, the mapping from \mathbf{y}_1 to \mathbf{y}_2 in Equation (7.5) can be reformulated as

$$\mathbf{y}_2 \sim \mathbf{H}\mathbf{y}_1, \quad \Rightarrow \quad \underbrace{\begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix}}_{=\mathbf{y}_2} \sim \underbrace{\begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}}_{=\mathbf{H}} \underbrace{\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix}}_{=\mathbf{y}_1}. \quad (7.7)$$

After P-normalization of the homogeneous coordinates, the resulting Cartesian coordinates are

$$u_2 = \frac{h_{11}u_1 + h_{12}v_1 + h_{13}}{h_{31}u_1 + h_{32}v_1 + h_{33}} = \frac{\mathbf{h}_1 \cdot \mathbf{y}_1}{\mathbf{h}_3 \cdot \mathbf{y}_1}, \quad v_2 = \frac{h_{21}u_1 + h_{22}v_1 + h_{23}}{h_{31}u_1 + h_{32}v_1 + h_{33}} = \frac{\mathbf{h}_2 \cdot \mathbf{y}_1}{\mathbf{h}_3 \cdot \mathbf{y}_1}. \quad (7.8)$$

Here, $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3$ are the three rows of the matrix \mathbf{H} . In general, this is a non-linear mapping of the 2D Cartesian coordinates (u_1, v_1) to (u_2, v_2) , but it is nevertheless represented as the *linear* transformation \mathbf{H} on the homogeneous coordinates.

The set of 2D homographies forms a group, corresponding to $P(GL(3))$, i.e., 3×3 matrices with non-zero determinants but disregarding any non-zero scalar multiplication. We refer to this group as the *homography group*, but the term *projective group* is also used in the literature. This group includes the 2D affine transformation, described in Section 4.4, as a subgroup. More precisely, affine transformations appear as the subgroup of homographies where $h_{31} = h_{32} = 0$.

7.3 Algebraically, we can also define homographies as the projective counterpart of the matrix group $GL(3)$.

Dual transformations

The idea of dual transformations, introduced in Section 4.5, applies also to homographies. When \mathbf{H} is a homography that transforms the homogeneous coordinates of points, then $\mathbf{H}^{-\top}$ is the corresponding dual transformation that implements the same geometric transformation on the dual homogeneous coordinates of lines. A homography always maps a line to a line, but in contrast to affine transformations, a homography can transport a proper point all the way to infinity, and vice versa. This means that lines that are parallel may not be parallel after a homography is applied, an issue we will return to in Section 7.1.2.

7.4 A 2D homography always maps a line to a line.

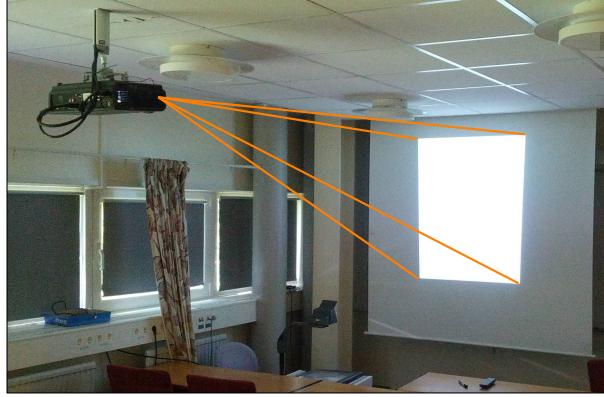


Figure 7.2: Illustration of the keystone effect related to projectors.

Degrees of freedom

Since the matrix \mathbf{H} has 9 elements but scalar multiplication does not matter, a homography has 8 degrees of freedom. Consequently, in the case of a 2D homography transformation we need at least 4 points, before and after the transformation, to determine which homography it is.

7.1.1 Applications of homographies

Homographies appear in real life situations, but are perhaps not always explicitly described as such. For example, modern meetings rooms often include a projector that can project the image from a computer or similar digital device onto a screen. A usual arrangement is that the projector is mounted onto the ceiling of the room and projects to a screen parallel to one of the walls of the room. As a result of this arrangement, the optical axis of the projector is not perpendicular to the screen, and if no correction is made the projected rectangular image would no longer appear as a rectangle: its upper side is shorter than its lower side. The reason is that the mapping from the image plane of the projector to the screen in this case forms a homography according to the geometric definition in Section 7.1. The first plane is the image plane of the projector, the second plane is the screen, and the projection point is the optical center of the projector. This mapping is often referred to as the *keystone effect* and can be compensated for by applying the inverse homography transformation to the image before it is projected onto the screen. As a result, the projected image appears rectangular, which is illustrated in Figure 7.2.

Another example is illustrated by street paintings. The idea here is to produce an image on the flat ground that, when viewed from a certain point, appears like a full 3D object or scene. Looking at it from other points makes the painting appear more or less deformed depending on how far from the optimal point you are. To produce a street painting, you need a 2D image of the object or scene, apply a homography transformation to it, and finally paint the transformed image onto the ground. The homography is defined geometrically in terms of two planes, where the first is the undeformed image placed at the intended position relative to the viewing point, and the second plane is the flat ground. The projection point \mathbf{n} is the viewing point.

Additional applications of 2D homographies are described in Sections 9.3 and 9.2.

7.1.2 Points at infinity revisited

At this point, we return to the idea of points at infinity. Recall that in the beginning of Chapter 4 it is said that the transformations that change a set of coordinates to another set of coordinates can be interpreted either as: the point is moved in 3D space to a new position relative to a fixed coordinate system, or as the transformation of the coordinates of a fixed point between two different coordinate systems.

We begin with the first description: a transformation that moves points from one place to another. We can think of the two planes \mathbf{p}_1 and \mathbf{p}_2 as two copies of \mathbb{E}^2 , each equipped with its own coordinate system, \mathbf{P}_1 and \mathbf{P}_2 , respectively. The homography \mathbf{H} maps the point \mathbf{y}_1 in the first copy of \mathbb{E}^2 to the point \mathbf{y}_2 , in the second copy of \mathbb{E}^2 . Assuming that the two planes \mathbf{p}_1 and \mathbf{p}_2 are not parallel, we can always find *proper points* \mathbf{y}_1 that

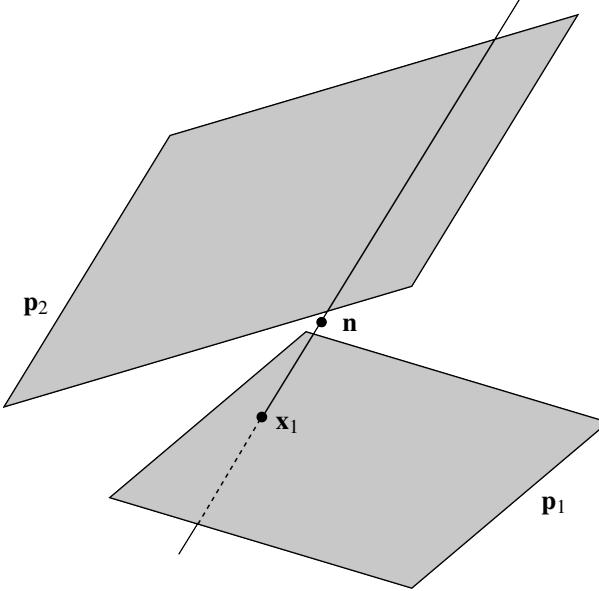


Figure 7.3: A homography is defined as the projection from the plane \mathbf{p}_1 through the point \mathbf{n} onto the plane \mathbf{p}_2 . The projection line generated by \mathbf{x}_1 in the plane \mathbf{p}_1 is parallel to \mathbf{p}_2 , and intersects \mathbf{p}_2 at infinity. This means that the proper point \mathbf{x}_1 is mapped by the homography to a point at infinity

generate projection lines that are parallel to \mathbf{p}_2 . In accordance with Section 5.4.1, this implies that \mathbf{y}_2 is a point at infinity. Consequently, for the general case of non-parallel planes, the geometric definition of a homography always includes the case that a proper point is mapped to *a point at infinity*. If the projection line instead is chosen as parallel to \mathbf{p}_1 , there exists a point at infinity, \mathbf{y}_1 , which is mapped to a proper point \mathbf{y}_2 . See Figure 7.3 for an illustration.

We can also take the second position and consider fixed points and a homography that transforms their coordinates from one coordinate system to the other. The homography transforms lines to lines, but may not necessarily preserve other properties that we normally associate with a coordinate system, e.g., perpendicular axes. Let $\bar{\mathbf{y}}_0 = (0, 0)$ be the origin of a 2D Cartesian coordinate system, its axes are perpendicular and the origin is a well-defined point in \mathbb{E}^2 . Now, let \mathbf{H} have the form:

$$\mathbf{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 0 \end{pmatrix}. \quad (7.9)$$

This means that the point \mathbf{y}_0 has transformed coordinates $\mathbf{y}'_0 \sim \mathbf{H}\mathbf{y}$ given as

$$\mathbf{y}' \sim \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} h_{13} \\ h_{23} \\ 0 \end{pmatrix}. \quad (7.10)$$

This is a point at infinity. It is a point at infinity relative to the transformed coordinate system, but not relative to the original coordinate system. Consequently, the idea of a point at infinity does not necessarily have to refer to something that lies infinitely far away from everything else. A point may end up at infinity simply because we have chosen a coordinate system that places it there. Conversely, if we happen to come across a point at infinity, it can become a proper point by choosing a suitable coordinate system and transforming the coordinates of that point to the new coordinate system.

7.2 Homographies in 3D

In principle, we can introduce homographies as transformations on 3D space in a similar way as was done for the 2D case in Section 7.1. Although the construction of a 2D homography in terms of a projection through a fixed point in \mathbb{E}^3 can be extended to a 4-dimensional Euclidean space, the practical application of this is limited. Instead we use the following definition: in \mathbb{E}^3 , a homography transformation is any element of the group $P(GL(4))$ applied on the homogeneous coordinates of points.

This means that the group of 3D homography transformations includes the 3D affine transformation group, but also other more general transformations. It has 15 degrees of freedom and requires a minimum of 5 points to be determined. A 3D homography transformation maps lines in \mathbb{E}^3 to lines, and maps planes to planes, and a general homography transformation may move points, lines, and planes to and from infinity. Homography transformations in 3D space have a much more limited use compared to the 2D case, but they are handy in certain applications.

7.3 Canonical set of homogeneous coordinates

A common trick in algebra is to change basis to solve a problem more easily. For example, by changing basis to a one given by the eigenvectors or singular vectors of some linear transformation it is often easier to solve a problem where the linear transformation is involved. The same idea applies to solving problems that involve homogeneous coordinates, e.g., of points in \mathbb{E}^2 or \mathbb{E}^3 . Given a set of such points, their homogeneous coordinates can be transformed to make the analysis simpler. This is what a set of canonical homogeneous coordinates are all about, they define a particular coordinate transformation that leads to simpler coordinates.

7.3.1 Canonical set homogeneous coordinates in 2D

Consider a set of four 2D points with homogeneous coordinates $\mathbf{y}_1, \dots, \mathbf{y}_4$. Based on these points, we define a homography transformation \mathbf{H}_a with columns given by $(\mathbf{y}_1 \mathbf{y}_2 \mathbf{y}_3)$, the homogeneous coordinates of the first three points. To be a proper homography we require that \mathbf{H}_a is non-singular which in this case means that the first three points are not co-linear. We then map the four points with \mathbf{H}_a^{-1} and the result is the transformed homogeneous coordinates $\mathbf{y}'_k = \mathbf{H}_a^{-1} \mathbf{y}_k$, which must look like:

$$\mathbf{y}'_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{y}'_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{y}'_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \mathbf{y}'_4 = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix}. \quad (7.11)$$

Already here, it should be clear that the transformed homogeneous coordinates \mathbf{y}'_k have a simple appearance, at least for the first three points. We can simplify the fourth point as follows. Define a diagonal matrix

$$\mathbf{H}_b = \begin{pmatrix} \alpha_1 & 0 & 0 \\ 0 & \alpha_2 & 0 \\ 0 & 0 & \alpha_3 \end{pmatrix}. \quad (7.12)$$

If the fourth point is not co-linear with two of the first three points, all three α_k are non-zero and \mathbf{H}_b has a well-defined matrix inverse. We apply this inverse of \mathbf{H}_b on the transformed points, to get $\mathbf{y}''_k = \mathbf{H}_b^{-1} \mathbf{H}_a^{-1} \mathbf{y}_k$. They look like:

$$\mathbf{y}''_1 \sim \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{y}''_2 \sim \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{y}''_3 \sim \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \mathbf{y}''_4 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}. \quad (7.13)$$

\mathbf{H}_b^{-1} , too, is diagonal and the application of this transformation onto the first three points \mathbf{y}'_k only changes their homogeneous coordinates in terms of a multiplication by a scalar, i.e., as projective elements they are unchanged. The final result is that now also the fourth point has a simple form, although in a different way than the first three.

The homogeneous coordinates presented in Equation (7.13) are referred to as a *canonical set of homogeneous coordinates*. We can summarize the discussion above as: given a set of four points in \mathbb{E}^2 , where not three of the four points are co-linear, it is always possible to determine a homography $\mathbf{H}_c = \mathbf{H}_b^{-1} \mathbf{H}_a^{-1}$ that transforms them to the canonical set of homogeneous coordinates in Equation (7.13). If we have fewer than four points, we can

still find a homography that transforms their homogeneous coordinates to the canonical set, but in this case, the transformation is not unique. If we have more than four points, a unique homography can be determined, but it will only include four of the points to the canonical set, the homogeneous coordinates of the other points will remain general.

7.3.2 Canonical set of homogeneous coordinates in 3D

Let $\mathbf{x}_1, \dots, \mathbf{x}_5$ be the homogeneous coordinates of five points in \mathbb{E}^3 , where not four of them are co-planar. In light of the results derived in the last section, we define a 3D homography $\mathbf{H}_a = (\mathbf{x}_1 \mathbf{x}_2 \mathbf{x}_3 \mathbf{x}_4)$, with the columns as the homogeneous coordinates of the first four point. Since the four points are not co-planar, \mathbf{H}_a is non-singular and \mathbf{H}_a^{-1} is well-defined. By applying \mathbf{H}_a^{-1} to the homogeneous coordinates we get $\mathbf{x}'_k = \mathbf{H}_a^{-1}\mathbf{x}_k$, where now the homogeneous coordinates of the first four points are in the canonical set, but the last one is given as $\mathbf{x}'_5 = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$. Since no four of the five points are co-planar, it must be case that all α_k are non-zero, and we form a diagonal matrix \mathbf{H}_b that holds α_k in the diagonal. Finally, by applying \mathbf{H}_b^{-1} to \mathbf{x}'_k we arrive at $\mathbf{x}''_k = \mathbf{H}_b^{-1}\mathbf{H}_a^{-1}\mathbf{x}_k$, where

$$\mathbf{x}''_1 \sim \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}''_2 \sim \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{x}''_3 \sim \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad \mathbf{x}''_4 \sim \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad \mathbf{x}''_5 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}. \quad (7.14)$$

The main difference to the 2D case of the canonical set of homogeneous coordinates, which can include up to 4 points, is that the canonical set includes up to 5 points for the 3D case.

7.3.3 Concluding remarks

An observation that can be made here is that the first few point in the canonical set lie at infinity. Do not let this distract you, points at infinity work as any proper point relative to the algebraic framework for homogeneous representations that is presented here. Just carry on, and do not think much about it. Also notice that the transformation that maps the homogeneous coordinates of some points into the canonical set is *data dependent*. For two different sets of points, we need to apply two different transformations to map the homogeneous coordinates of the points to the canonical set: both \mathbf{H}_a and \mathbf{H}_b depend on the points under consideration.

The main application for these results is that they allow you to transform a set of points to the canonical set, where you then apply some analysis in the transformed space and, if necessary, transform the solution back to the original space. The analysis is sometimes significantly simplified when done on the canonical set, rather than on general homogeneous coordinates.

Chapter 8

The Pinhole Camera

Before you read this chapter, you should have a thorough understanding of the homogeneous representations for \mathbb{E}^2 and \mathbb{E}^2 that are presented in Chapters 3 and 5. 2D homographies, described in Section 7.1 will be used throughout this chapter. You may also want to have a look at the QR-decomposition, described in Toolbox Section 8.7, in particular the special variant of the QR-decomposition described in Toolbox Section 8.7.1.

The Euclidean space \mathbb{E}^3 that we live in is not accessible to us directly in terms of visual information. What our eyes see are rather projections of this space onto our retinas. The way these projections are formed serves as a model for how to make two-dimensional representations of the three-dimensional world. The simplest mathematical description of this projection is the pinhole camera model. Although this model is not entirely correct in all details, relative to how our eyes together with the visual cortex of the brain perceive visual stimuli, it is sufficiently correct to produce two-dimensional images that we can interpret as projections of the 3D world. The pinhole camera refers both to a mathematical model for how 3D points are projected onto a 2D image, as well as to a physical realization of this model, an approximation of the mathematical model.

8.1 Central projection

The geometric definition of a homography in Section 7.1 is based on mapping a point \mathbf{x}_1 in a plane by projecting it through a point \mathbf{n} and then determine the intersection of the projection line with a second plane \mathbf{p}_2 . The point of intersection, \mathbf{x}_2 , is determined by \mathbf{x}_1 , \mathbf{p}_2 , and \mathbf{n} as described in Equation (7.2). In that relation, we assume that \mathbf{x}_1 is restricted to a plane, and this is later taken care of by introducing coordinate systems in each of the two planes.

The mapping that we consider here is similar to a homography but with the difference that it does not require the first point, now called \mathbf{x} , to lie in a plane, in principle it can be any point in \mathbb{E}^3 . The second point \mathbf{x}_2 , however, is still restricted by Equation (7.2) to lie in a plane \mathbf{p} , as is illustrated in Figure 8.1. The mapping from an arbitrary 3D point \mathbf{x} to the point \mathbf{x}_2 that lies in the plane \mathbf{p} , as a projection through a fixed point \mathbf{n} , is referred to as a *central projection*. In the literature the same concept is also known as *central perspective* or *pinhole perspective*.

To obtain an algebraic representation for this mapping we introduce a coordinate system for the plane \mathbf{p} , such that \mathbf{x}_2 is given as

$$\mathbf{x}_2 = \mathbf{P} \mathbf{y}. \quad (8.1)$$

Here, \mathbf{P} is a 4×3 matrix that defines the coordinate system in \mathbf{p} , and \mathbf{y} holds the homogeneous coordinates of the point \mathbf{x}_2 in \mathbf{p} relative the coordinate system \mathbf{P} . We insert this into Equation (7.4) and get

$$\mathbf{P} \mathbf{y} \sim \mathbf{M}(\mathbf{p}, \mathbf{n}) \mathbf{x}, \quad \Rightarrow \quad \mathbf{y} \sim \mathbf{P}^+ \mathbf{M}(\mathbf{p}, \mathbf{n}) \mathbf{x}. \quad (8.2)$$

The function \mathbf{M} is the one defined in Equation (7.3). The last relation is well-defined when \mathbf{P} has full column rank, which happens when \mathbf{n} does not lie in \mathbf{p} . In this case, we can define a 3×4 matrix \mathbf{C} as

$$\mathbf{C} = \mathbf{P}^+ \mathbf{M}(\mathbf{p}, \mathbf{n}), \quad (8.3)$$

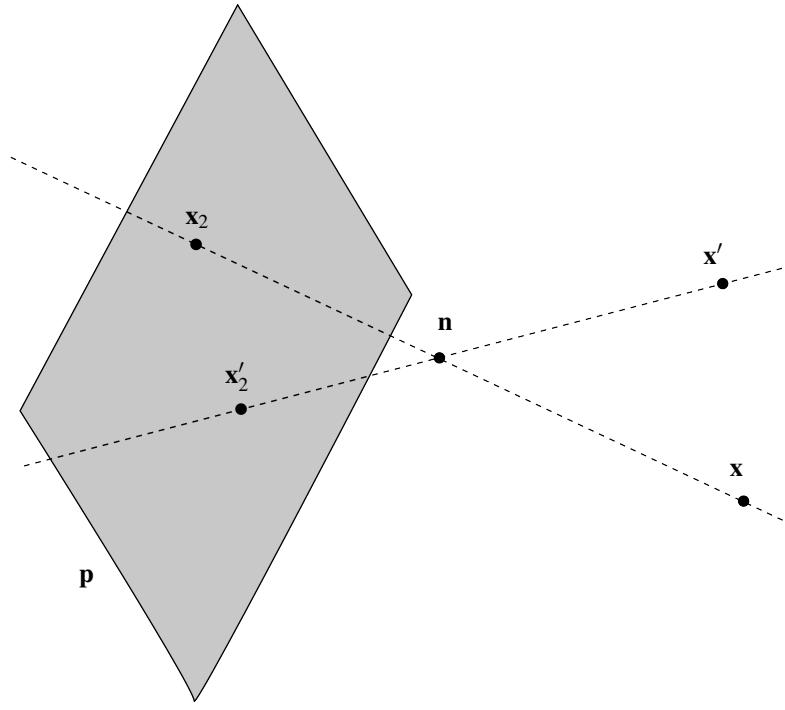


Figure 8.1: Central projection of the 3D points \mathbf{x} and \mathbf{x}' through the *camera center* \mathbf{n} onto the *image plane* \mathbf{p} where they intersect at \mathbf{x}_2 and \mathbf{x}'_2 . The latter two 3D points correspond to 2D image points \mathbf{y} and \mathbf{y}' .

and write

$$\mathbf{y} \sim \mathbf{C} \mathbf{x}. \quad (8.4)$$

The matrix \mathbf{C} is well-defined as soon as we have specified the projection point \mathbf{n} , the plane \mathbf{p} , and the coordinate system in the plane defined by \mathbf{P} . We refer to \mathbf{C} as a *camera projection matrix* or simply *camera matrix*. In the following presentation, the projection point \mathbf{n} is referred to as the *camera center*. Other textbooks refer to \mathbf{n} as the *focal point*, or the *optical center*. Furthermore, \mathbf{p} is referred to as the *image plane*, this is where the image of the 3D space is projected by the central projection. In this context, it is natural to refer to \mathbf{y} as an *image point*, or as *image coordinates*, depending on whether we see \mathbf{y} as a point in the image plane or as the homogeneous coordinates of this point relative to the coordinate system \mathbf{P} .

The projection lines that appear in Figure 8.1 can be constructed in two different ways. It can be done by selecting a 3D point \mathbf{x} , and finding the line that intersects \mathbf{x} and \mathbf{n} . This line intersects the image plane \mathbf{p} at some specific point \mathbf{x}_2 , with image coordinates \mathbf{y} . Alternatively, we start with a specific image point \mathbf{y} , corresponding to a 3D point \mathbf{x}_2 in the image plane. It has a projection line that intersects \mathbf{x}_2 and \mathbf{n} . This line consists of all 3D points that are projected onto the image point \mathbf{y} .

8.1 In the context of central projection, a projection line can be constructed either from selecting 3D points \mathbf{x} , or by selecting image points \mathbf{y} .

As a matrix \mathbf{C} has at most rank 3, so it must have a null space that is at least 1-dimensional. A close look at the function \mathbf{M} , Equation (7.3), reveals that $\mathbf{M}(\mathbf{p}, \mathbf{n})$ has rank 3 and that \mathbf{n} is a null vector. Furthermore, it has a range that is perpendicular to \mathbf{p} , and since \mathbf{p} is the only projective element that makes $\mathbf{P}^+ \mathbf{p} = \mathbf{0}$ it follows that $\text{rank}(\mathbf{C}) = 3$ and \mathbf{n} is a null vector of \mathbf{C} :

$$\mathbf{C} \mathbf{n} = \mathbf{0}. \quad (8.5)$$

This means that the camera center \mathbf{n} is the unique point in \mathbb{E}^3 that cannot be projected to a point in the plane \mathbf{p} by central projection. Finally, since \mathbf{C} is used to map homogeneous coordinates of 3D points to homogeneous

coordinates of 2D points, and both are projective elements, any scalar multiplication of the matrix \mathbf{C} produces a new matrix that represents the same central projection, relative to the same coordinate systems. Hence, we can see \mathbf{C} as an element of the projective space $P(\mathbb{R}^{3 \times 4})$.

8.2 The camera matrix \mathbf{C} is 3×4 and has full rank ($= 3$). The camera center \mathbf{n} is a null vector of \mathbf{C} .

The pinhole camera

In this presentation, the term *pinhole camera*, or simply *camera*, will be used in several related, but distinct, ways. For example, it can refer to the geometrical mapping from 3D space to an image point by central projection. A camera can also refer to the algebraic representation of this mapping in terms of the camera matrix \mathbf{C} . Compare this to the geometric and algebraic definition of a homography transformation, described in Section 7.1.

A key observation for the geometric interpretation of a pinhole camera is that it is defined primarily only by the position of its center, \mathbf{n} . As we will see in Section 9.3, all cameras that have a common center, but different image planes \mathbf{p} , are equivalent in the sense that they can be related by homography transformations of the image coordinates \mathbf{y} . For the algebraic interpretation, a key observation is that the numerical values of \mathbf{C} depend on which coordinate systems are being used, in 3D space and in the image plane. Keeping \mathbf{n} and \mathbf{p} fixed, but changing to another coordinate system in 3D space, or in \mathbf{p} , in general produces a new camera matrix. Hence, one and the same central projection has different numerical representations, depending on the choice of coordinate systems.

8.3 The term *camera* is used to denote a specific instance of central projection, specified by a camera center \mathbf{n} and an image plane \mathbf{p} .

There is also a third meaning for the concept of a pinhole camera: as any device that can implement central projection, at least to a sufficient degree of approximation. Such a device is also known as a *camera obscura*.

8.1.1 Camera obscura

For a physical implementation of a camera, the projection lines described above correspond to light rays that are reflected or emitted at various points in the scene and collected by the camera, allowing only those rays that intersect the camera center \mathbf{n} to pass into the camera and fall onto the image plane. For a film-based camera the light rays are converted by means of photo-chemical processes to molecular changes in the film that lies in the image plane, causing changes that make an image to appear in the film. For a digital camera, the light rays are converted, by means of photo-electrical processes in the *pixels* of the image sensor, either to voltage or electric charge that can be measured and stored as a digital image.

To construct an operational pinhole camera, you just take a box of a suitable size and make a small hole, an *aperture*, corresponding to the camera center, in one of the sides of the box. The light which enters the camera through its center is projected onto the opposite side where it produces an image, see Figure 8.2. In the simplest case the image can be captured using a photographic film. This device is called a *camera obscura* and has been known from antiquities, although it was originally not used to produce film based images but rather projections of a scene into a closed and dark chamber (in Latin: *camera obscura*). The first photographic image by Niépce in 1826 was taken by a camera obscura.

The main drawback of a camera obscura is that the aperture has to be small in order to make the projected image sharp, while a small aperture also prevents sufficient amount of light to enter the camera for most practical purposes. In practice, the camera obscura concept has been replaced by lens based cameras. In such a camera, the aperture can be of almost arbitrary size and it includes a lens or a system of lenses that focuses the light that enters the camera onto the image plane. Within rather general limits the typical lens based camera can still be seen as a reasonable approximation of a pinhole camera in terms of how light from points in the scene, depicted by the camera image, are projected onto the image plane. At this point, we will not go deeper into the issue of the image formation process of a camera. But we note that, although a mathematical pinhole camera has an infinitely small aperture point, a physical realization of a pinhole camera must allow a sufficient amount of light to enter the camera to produce an image. Therefore, a real camera requires an aperture of some finite size.

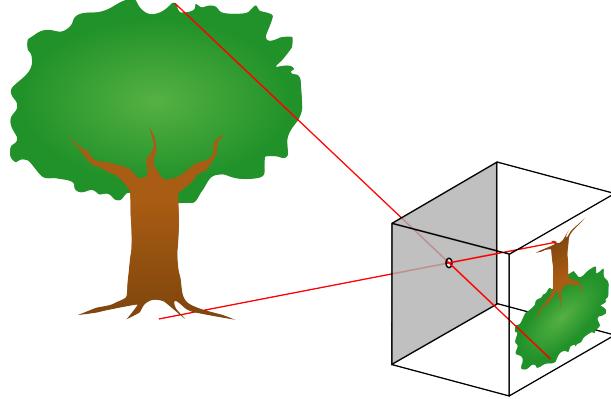


Figure 8.2: A pinhole camera maps points in 3D space to an image by means of central projection. The resulting image is rotated 180° relative to the 3D space. Image source: Wikimedia commons.

A characteristic property of the image produced by a camera obscura is that the projected image is rotated 180° relative how the scene appears to our eyes, as is illustrated in Figure 8.2. We will return to this issue shortly.

8.2 Camera centered coordinate systems

The discussion in the last section leads to a 3×4 camera matrix, \mathbf{C} in Equation (8.3), which represents the central projection of a pinhole camera as a mapping from \mathbb{E}^3 to \mathbb{E}^2 . In this section we investigate the camera matrix more in detail, to get both an algebraic and geometric interpretation of what it means.

We have an image plane \mathbf{p} and a camera center \mathbf{n} , and we will now choose two particularly suitable coordinate system for this investigation, one for \mathbb{E}^3 and another one for the image plane \mathbf{p} . See Figure 8.3 for an illustration. We begin by choosing the coordinate system of \mathbb{E}^3 such that the origin is at the camera center. The three axes of the coordinate system are chosen such that the third axis, $\hat{\mathbf{e}}_3$, is perpendicular to the image plane. This axis, the *principal axis* or *optical axis* of the camera, corresponds to a line in \mathbb{E}^3 , the *principal line*, which intersects with the image plane at the *principal point*. Another characterization of the principal plane is that, with the exception of the camera center, it contains exactly all 3D points that are projected to points at infinity in the image. The *principal plane* intersects the camera center and is parallel to the image plane, i.e., it includes the first two coordinate axes. The principal plane can be seen as the “front” of the camera, facing the scene that is depicted in the image plane. The distance between the camera center and the image plane is referred to as the *focal length* of the camera, denoted f . The other two axes, $\hat{\mathbf{e}}_1$ and $\hat{\mathbf{e}}_2$, can have arbitrary orientation as long as they form an ON-basis.

Next, we choose a coordinate system of the image plane that has its origin at the principal point and its axes, $\hat{\mathbf{b}}_1$ and $\hat{\mathbf{b}}_2$, aligned with the first two axes of the first coordinate system. The coordinate systems constructed here are referred to as *camera centered coordinate systems*. There are two camera centered coordinate systems, one for \mathbb{E}^3 ($\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \hat{\mathbf{e}}_3$), and one for the image plane \mathbb{E}^2 ($\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2$), and the two are related such that two of the axes of the first coordinate system are parallel to the axes of the second coordinate system.

In terms of a practical camera, we can choose the three axes of the 3D camera centered coordinates system such that the first axis $\hat{\mathbf{e}}_1$ points right, the second axis $\hat{\mathbf{e}}_2$ points down, and the third axis $\hat{\mathbf{e}}_3$ points in the viewing direction of the camera. This choice is arbitrary, but is practical since it implies a right-handed orthogonal coordinate system in 3D, and an *anti-clockwise* orthogonal system in the image plane. This assumes that the image is projected onto the image plane from a scene in front of the camera (in the direction of the principal axis) and it is the “front side” of the image plane that defines the image.

Other configurations of the coordinate systems appear in practical applications and in the literature. It is a common source of error in calculations not to have a thorough understanding of exactly how the coordinate axes are arranged for a particular set of coordinates that is given. This applies both to the image coordinates and to the

• Right-handed coordinate systems in \mathbb{R}^3 are defined in Toolbox Section 3.7.1.

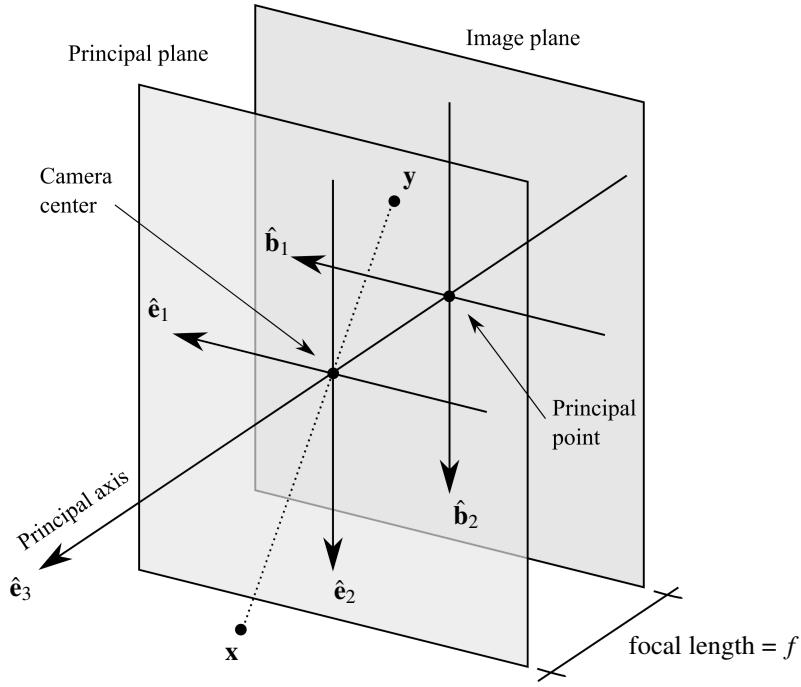


Figure 8.3: The camera centered coordinate systems of a pinhole camera, including the principal line, the principal point, the principal plane, and the focal length of the camera. The vectors $\hat{e}_1, \hat{e}_2, \hat{e}_3$ form an ON-basis of the 3D space, and \hat{b}_1, \hat{b}_2 form an ON-basis of the image plane. The illustration also shows a 3D point x that is projected to an image point y .

3D coordinates.

8.4 It is important to know the configuration of the coordinate systems that are used to define coordinates in 3D space and in the image. Otherwise, coordinates are meaningless!

Consider a 3D point that has Cartesian coordinates $\bar{x} = (x_1, x_2, x_3)$ relative to the camera centered coordinate system. We want to determine the camera centered image coordinates $\bar{y} = (u, v)$ of the projection of \bar{x} . To do this we can look at the whole scenario from “above”, along the \hat{e}_2 axis, as is illustrated in Figure 8.4. In this figure we see two similar triangles, the left one has sides x_3 and $-x_1$, and the right one with sides f and y_1 , leading to

$$\frac{u}{f} = -\frac{x_1}{x_3}. \quad (8.6)$$

The minus sign comes from the fact that axis \hat{e}_1 of the coordinate system in \mathbb{E}^3 points in the same direction as \hat{b}_1 in the image plane. It does not matter on which side of the principal axes the point x lies, there will always be a minus sign in the last equation. If we do the same exercise also along the \hat{e}_1 axis, the result can be summarized as

$$\bar{y} = \begin{pmatrix} u \\ v \end{pmatrix} = -\frac{f}{x_3} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}. \quad (8.7)$$

Let us discuss the last relation before we continue. We see that the image coordinates (u, v) are essentially proportional to the first two coordinates of the 3D point. They are also scaled such that the projection of an object shrinks when the object moves away from the camera center, i.e., when x_3 increases, and the projection grows when the object comes closer, i.e., when x_3 decreases. This is consistent with how we perceive a 3D scene: objects far away appear smaller than objects that are closer to the eye. Of course, our brain can manage this scale change and tell us that it is not the objects that are shrinking or growing in size, but rather their distances that vary.

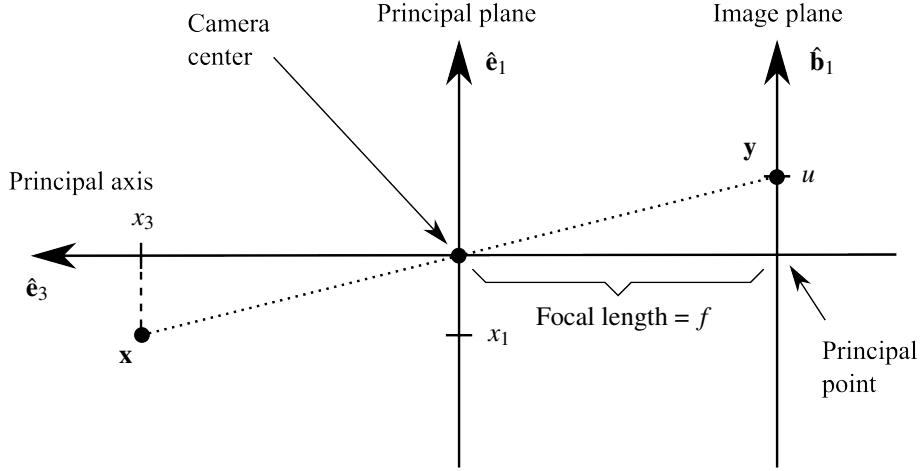


Figure 8.4: The camera centered coordinate systems in Figure 8.3 seen from “above”, along coordinate axis \hat{e}_2 .

A second observation is that the scale of the projection of an object is also determined by the focal length f . If we want a smaller or larger image of the same scene, we choose f as either small or large. Finally, the minus sign in Equation (8.7) implies that the projected image is rotated by 180° .

Next, we formulate the mapping from 3D coordinates $\bar{\mathbf{x}}$ to image coordinates $\bar{\mathbf{y}}$ in terms of their homogeneous coordinates. We get

$$\mathbf{y} \sim \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{x_3} \begin{pmatrix} -fx_1 \\ -fx_2 \\ x_3 \end{pmatrix} \sim \begin{pmatrix} -fx_1 \\ -fx_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} = \begin{pmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \mathbf{x}. \quad (8.8)$$

With reference to Equation (8.4) we can write the camera matrix as

$$\mathbf{C} \sim \begin{pmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (8.9)$$

In a camera centered coordinate system, the camera center has Cartesian coordinates $(0, 0, 0)$ and its homogeneous representation is

$$\mathbf{n} \sim \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \quad (8.10)$$

Trivially, this camera center \mathbf{n} satisfies Equation (8.5) when \mathbf{C} has the form in Equation (8.9).

In the initial expressions, e.g., Equation (8.7), we need to assume that $x_3 \neq 0$. But if $x_3 = 0$ we can express \mathbf{x} as

$$\mathbf{x} \sim \begin{pmatrix} x_1 \\ x_2 \\ 0 \\ 1 \end{pmatrix}, \quad \Rightarrow \quad \mathbf{y} \sim \mathbf{C} \mathbf{x} = \begin{pmatrix} fx_1 \\ fx_2 \\ 0 \end{pmatrix}. \quad (8.11)$$

This result shows that \mathbf{y} is a point at infinity when $x_3 = 0$, thus this case needs not be excluded from central projection. Instead, we can define the principal plane as the plane consisting of the 3D points that are projected onto points at infinity in the image. To be sure, there is one point in the principal plane that cannot be projected in this way: the camera center \mathbf{n} lies in the principal plane, but $\mathbf{C} \mathbf{n} = \mathbf{0}$, which means that there is no projection of \mathbf{n} . Furthermore, in the above expressions, there is no restriction on the sign of x_3 , it can be either positive or negative.

In practice, it must be the case that $x_3 > 0$, since we cannot project a 3D point that is “inside” or “behind” the camera, which is what $x_3 < 0$ means. Another practical restriction that relates to a real camera is that although the image plane is of infinite extension, only a finite area of this plane is exposed to the light in the scene in front of the camera, typically in the form of a rectangle. In this presentation, we will often ignore this fact and instead focus on the geometrical aspects of an ideal pinhole camera. We will keep these observations in mind and return to it later on, for example in Section 8.4.

Depth

Given a 3D point $\bar{\mathbf{x}} = (x_1, x_2, x_3)$, the third coordinate relative to the camera centered 3D coordinate system, x_3 is often referred to as the *depth* of $\bar{\mathbf{x}}$. As mentioned above, depth is in practice always positive. Notice that depth is not the same as distance to the camera, it is rather the distance from $\bar{\mathbf{x}}$ to the principal plane, measured in the direction orthogonal to this plane. Also notice, the depth refers specifically to the third coordinate relative to the camera centered coordinate system. As we will see shortly, 3D coordinates can often be defined relative to a world coordinate system which, in general, do not have to be related to the camera. In this case, we need to transform the coordinates of a point to the camera centered coordinate system before we can determine its depth.

8.5 In practice, a camera can only observe 3D points that lie in front of the camera, with positive depth.

As we will see later on, this *feasibility constraint* can often be used to check the validity of solutions to various geometric problems.

The virtual image plane

The image produced by pinhole camera is rotated 180° . In reality, this is not a problem: for a film based camera, it simply means that we need to rotate the image 180° before we look at it, and for a digital camera a de-rotated image is produced by reading out the pixels in the appropriate order from the sensor chip. In calculations, however, the minus sign in Equation (8.13) is a nuisance and they would be simpler without it. To remove the minus sign, there are two possible approaches: either we rotate one of the two camera centered coordinate systems 180° about the principal axis, or we move the image plane to the other side of the camera center.

The first option is possible but impractical. It would be necessary to always keep in mind that the two coordinate systems are rotated relative to each other. The second option cannot be realized in practice. We can, however, make a graphical representation of such a camera; it is not more difficult to draw the image plane “in front of” the camera center than “behind”. If the image plane is placed in front of the camera center, it is referred to as a *virtual image plane*. It produces a non-rotated version of the projected image, without a minus sign in the equations, and it can be drawn as easily as the “proper” image plane that lies behind the camera center. Consequently, in the following we will often use the concept of a virtual image plane instead of the standard approach described initially. See Figure 8.5 for an illustration of a central projection onto a virtual image plane.

When the idea of a virtual image plane is applied to the camera matrix in Equation (8.9), it becomes instead

$$\mathbf{C} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (8.12)$$

8.2.1 The normalized camera

In this section we consider the particular case when $f = 1$, corresponding to what is referred to as a *normalized camera*. In this case, and using the projection onto the virtual image plane, Equation (8.12), the *normalized camera matrix* is given as

$$\mathbf{C}_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (8.13)$$

A more compact, and also very useful way, of describing this matrix is

$$\mathbf{C}_0 = (\mathbf{I} | \mathbf{0}), \quad (8.14)$$

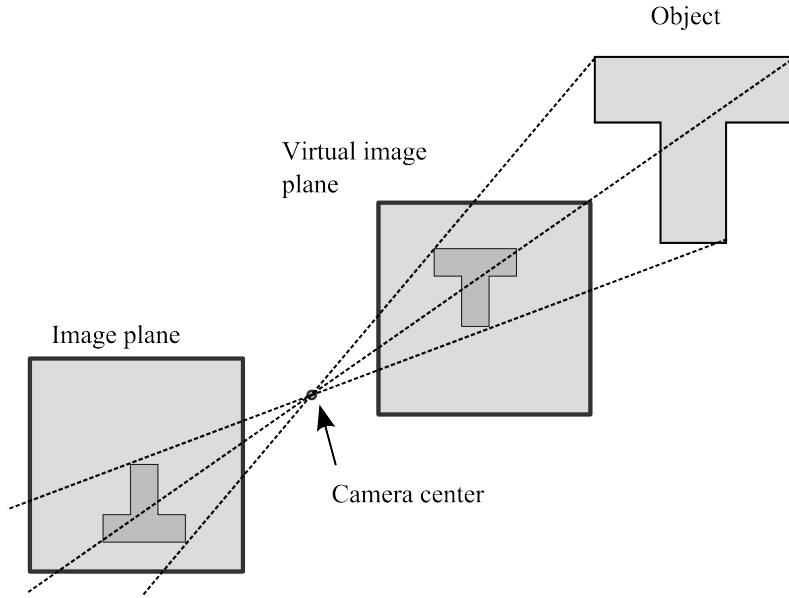


Figure 8.5: An object is projected onto the image plane through the camera center, producing an image that is rotated 180°. The figure also shows a virtual image plane, in front of the camera center, which produces an unrotated image.

where we see \mathbf{C}_0 as a concatenation of the 3×3 identity matrix \mathbf{I} and the zero vector $\mathbf{0} \in \mathbb{R}^3$. We refer to the image coordinates $\mathbf{y} \sim \mathbf{C}_0 \mathbf{x}$ that are produced by a normalized camera matrix as *camera normalized image coordinates*, or *C-normalized image coordinates* for short.

The matrix \mathbf{C}_0 describes the normalized camera relative to the camera centered coordinate system, where the origin is located at the camera center, \mathbf{n} , and the orientation of the image coordinate axes are aligned with the axes of the camera. Next, we consider the normalized camera represented in a general coordinate system of 3D space.

The world coordinate system

So far, we assume that both 3D points and image points refer to camera centered coordinate systems. In practice we sometimes have reasons to use a coordinate system for \mathbb{E}^3 than is not camera centered. A simple way to deal with that situation is to assume that we have a specific designated coordinate system for \mathbb{E}^3 , in this context often referred to as a *world coordinate system*, and that the camera centered coordinate system is translated and rotated relative to the world coordinate system. This means that the Cartesian coordinates of a specific point in \mathbb{E}^3 , relative to the world coordinate system and relative to the camera centered coordinate system, are related by a rigid transformation. This idea is based on both coordinate systems having the same handedness, and although this is not a strict requirement it tends to simplify the calculations involving the cameras. See Figure 8.6 for an illustration of how the two coordinate systems are related. Let $\bar{\mathbf{x}}_w$ denote the Cartesian coordinates of a point relative to the world coordinate system, and let $\bar{\mathbf{x}}_c$ denote its Cartesian coordinates relative to the camera centered coordinate system. The homogeneous coordinates of a 3D point relative the two coordinate systems are then given as

$$\mathbf{x}_c \sim \begin{pmatrix} \bar{\mathbf{x}}_c \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \bar{\mathbf{x}}_w \\ 1 \end{pmatrix} \sim \underbrace{\begin{pmatrix} \mathbf{R} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix}}_{:=\mathbf{T}_{wc}} \mathbf{x}_w = \mathbf{T}_{wc} \mathbf{x}_w. \quad (8.15)$$

Remember from Section 6.2 that the expression for the rigid transformation \mathbf{T}_{wc} from \mathbf{x}_w to \mathbf{x}_c should be interpreted as: first a rotation about the origin by \mathbf{R} and then a translation by $\bar{\mathbf{t}}$.

Equation (8.15) allows us to transform the coordinates of any 3D point, from world coordinates to camera centered coordinates, and in the latter coordinate system we can then project the 3D point onto the image plane of

• Handedness of coordinate systems is described in Toolbox Section 3.7.1.

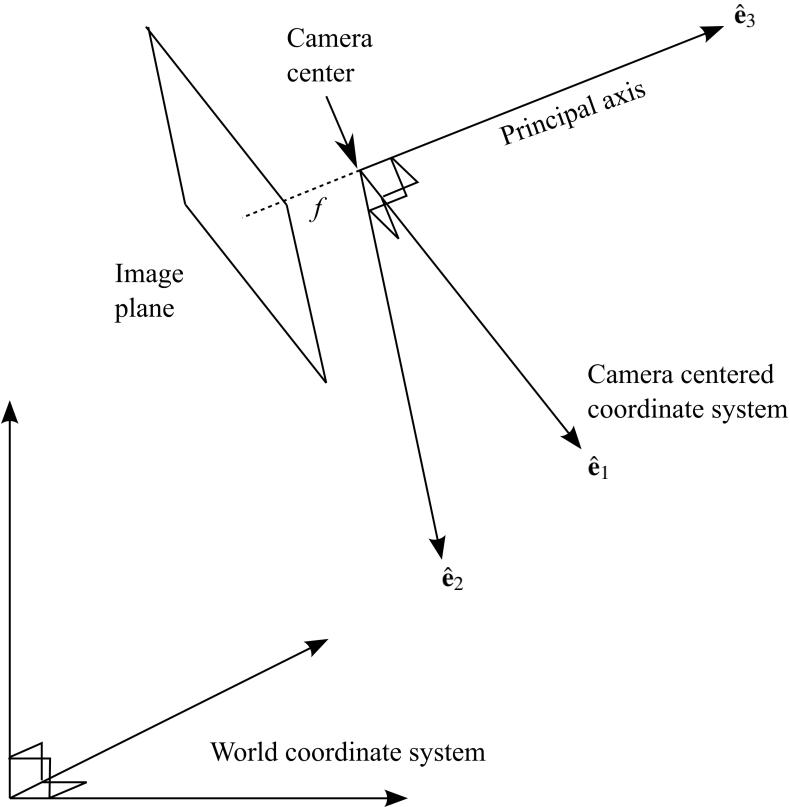


Figure 8.6: The camera centered coordinate system is rigidly transformed relative to the world coordinate system.

the normalized camera in Equation (8.14):

$$\mathbf{y} \sim \mathbf{C}_0 \mathbf{x}_c = (\mathbf{I} | \mathbf{0}) \begin{pmatrix} \mathbf{R} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} \mathbf{x}_w = (\mathbf{R} | \bar{\mathbf{t}}) \mathbf{x}_w. \quad (8.16)$$

This means that the image coordinates of the 3D point are produced by a camera matrix given as

$$\mathbf{C} = (\mathbf{R} | \bar{\mathbf{t}}), \quad \text{where} \quad \mathbf{R} \in SO(3), \quad \bar{\mathbf{t}} \in \mathbb{R}^3. \quad (8.17)$$

We refer also to this \mathbf{C} as a *normalized camera matrix* since the focal length remains as $f = 1$. We can see $\mathbf{R} \in SO(3)$ as an *internal constraint* on normalized camera matrices although, as a projective element, we can allow the leftmost 3×3 submatrix of a normalized camera to be a rotation matrix multiplied by an arbitrary non-zero real number. This camera projects 3D points defined relative to the world coordinate system to normalized image coordinates, where the transformation from the world coordinate system to the camera centered coordinate system is given by \mathbf{R} and $\bar{\mathbf{t}}$ in Equation (8.15).

8.6 The rigid transformation represented by \mathbf{R} and $\bar{\mathbf{t}}$ in Equation (8.17) describes the transformation *from* the world coordinate system *to* the camera centered coordinate system. World coordinates are *first* rotated by \mathbf{R} and *then* translated by $\bar{\mathbf{t}}$.

The rigid transformation \mathbf{T}_{wc} has an inverse transformation, denoted as \mathbf{T}_{cw} . This, too, is a rigid transformation and it transforms *from* the camera centered coordinate system *to* the world coordinate system. It is given as

$$\mathbf{T}_{cw} = \mathbf{T}_{wc}^{-1} = \begin{pmatrix} \mathbf{R}^\top & -\mathbf{R}^\top \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix}, \quad (8.18)$$

which can be easily verified since it leads to $\mathbf{T}_{cw} \mathbf{T}_{wc} = \mathbf{I}$. With explicit expressions for both \mathbf{T}_{wc} and \mathbf{T}_{cw} we can now derive several useful results.

The camera center

The camera center has coordinates $\mathbf{n}_c = (\mathbf{0} \ 1)$ relative to the camera coordinate system, and we use the rigid transformation \mathbf{T}_{cw} in Equation (8.18) to express the coordinates of the same point relative to the world coordinate system:

$$\mathbf{n}_w \sim \mathbf{T}_{cw}\mathbf{n}_c = \begin{pmatrix} \mathbf{R}^\top & -\mathbf{R}^\top \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} = \begin{pmatrix} -\mathbf{R}^\top \bar{\mathbf{t}} \\ 1 \end{pmatrix}. \quad (8.19)$$

Consequently, $\bar{\mathbf{t}}$ is *not* the Cartesian coordinates of the camera center relative to the world coordinate system. They are instead given by

$$\bar{\mathbf{n}} = \bar{\mathbf{n}}_w = -\mathbf{R}^\top \bar{\mathbf{t}}. \quad (8.20)$$

So, what does $\bar{\mathbf{t}}$ represent? Let \mathbf{x}_w be the origin world coordinate system, relative to the the same coordinate system, and transform these coordinates to the camera coordinate system:

$$\mathbf{x}_w = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}, \quad \Rightarrow \quad \mathbf{x}_c \sim \mathbf{T}_{wc}\mathbf{x}_w = \begin{pmatrix} \bar{\mathbf{t}} \\ 1 \end{pmatrix}. \quad (8.21)$$

Consequently, $\bar{\mathbf{t}}$ is the Cartesian coordinates of the origin in the world coordinate system but relative to the camera coordinate system. We summarize these findings as:

- 8.7** The normalized camera in Equation (8.17) has its center located at $\bar{\mathbf{n}}_w = -\mathbf{R}^\top \bar{\mathbf{t}}$ relative to the world coordinate system. The origin of the world coordinate system relative to the camera coordinate system is given as $\bar{\mathbf{t}}$.

Parameterization of the normalized camera

Solving for $\bar{\mathbf{t}}$ in Equation (8.20) gives $\bar{\mathbf{t}} = -\mathbf{R}\bar{\mathbf{n}}$, and we can insert that into the expression for the normalized camera matrix in Equation (8.17). The result is

$$\mathbf{C} = (\mathbf{R} | -\mathbf{R}\bar{\mathbf{n}}) = \mathbf{R}(\mathbf{I} | \bar{\mathbf{n}}). \quad (8.22)$$

Equations (8.17) and (8.22) represent two different ways of parameterizing a normalized camera, but give the same numerical values for the elements of \mathbf{C} . To fully specify the normalized camera, we need to know the rotation between the camera and world coordinate system, it has three degrees of freedom in accordance with Section 6.2, and we need the displacement between the two coordinate systems, which gives three more degrees of freedom. Consequently, a normalized camera has six degrees of freedom, same as a rigid transformation in 3D. The displacement can be represented either as $\bar{\mathbf{t}}$, the origin of the world coordinate system expressed in the camera centered coordinate system, or as $\bar{\mathbf{n}}$, the camera center relative to the world coordinate system.

The parameterization in Equation (8.22) is a bit unconventional since it uses a rotation from the world coordinate system to the camera coordinate system, \mathbf{R} , and a translation from the camera coordinate system to the world coordinate system, $\bar{\mathbf{n}}$. A more consistent parameterization is to use

$$\mathbf{C} = (\mathbf{R}_{cw}^\top | -\mathbf{R}_{cw}^\top \bar{\mathbf{n}}) = \mathbf{R}_{cw}^\top (\mathbf{I} | \bar{\mathbf{n}}). \quad (8.23)$$

where \mathbf{R}_{cw} is the rotation from the camera coordinate system to the world coordinate system. This parameterization of the normalized camera is based on the rigid transformation \mathbf{T}_{cw} , while Equation (8.17) is based on \mathbf{T}_{wc} .

Since Equations (8.17) and (8.23) both are used in the literature, it is important to recognize them as valid parameterizations of a normalized camera, and understand the geometric interpretation of the various parameters. In the following presentation, we tend to often use Equation (8.17), mainly because it is a bit more compact in text and in mathematical expressions. In certain applications, however, Equation (8.23) can be a more useful parameterization of the normalized camera, and we will have reasons to come back to this issue in Chapter 21.

- 8.8** A normalized camera can be parameterized in different ways. Choose a parameterization that is useful for your application.

8.3 The general pinhole camera

The last section presents a general form of a normalized camera matrix, e.g., as in Equation (8.17) or in Equation (8.23). For a normalized camera, the coordinate system of the image plane is aligned with the 3D camera coordinate system, and $f = 1$. In general, we want to be able to use an arbitrary coordinate system for the image plane, and also use an arbitrary focal length. As we have already seen, the latter parameter can be seen as a scaling parameter of the image and can therefore be included in the coordinate system of the image plane by determining the unit length of the ON-basis. In practice, however, the unit length of the image plane is often determined by the pixel-to-pixel distance, i.e., one length unit in the image is the distance from one pixel to the next neighboring pixel. This length unit is fixed, but the focal length of the camera can be variable, so we need to be able to manage a variable image scale caused by variable focal length without changing the length unit.

Apart from a variable focal length, there are other issues that can be addressed if we introduce a separate *image coordinate system*, which does not have to be aligned with the camera coordinate system. In practice an image is not a full infinite plane, but rather restricted to a truncated plane of finite extension, typically a rectangle. In most image based coordinate systems, the origin is located at the upper left corner instead of at the image center, where we otherwise typically find the principal point. This calls for some type of transformation from the C-normalized image coordinates produced by a normalized camera to the coordinates that describe the positions of pixels in the image. The latter set of coordinates are referred to as *pixel coordinates*.

Another issue that calls for a more general description of the camera matrix, which goes beyond the normalized camera, is that in some cases the world coordinate system is not really well-defined in terms of a specific ON-basis. For example, in Section 10.3.4, \mathbf{C} is determined based on the epipolar geometry between two images, but only up to an unknown homography transformation of 3D space. This means that there are multiple choices of the camera matrix that are consistent with available data, but this data may not be able to specify a world coordinate system. Without additional information about the scene, there is no way to determine which choice of \mathbf{C} that is produced by a rigid transformation relative to the world coordinate system. All we can say is that there is a 3D homography transformation that relates the coordinate systems of the camera and of the world.

As it turns out, the application of non-rigid transformations in 3D space can almost always be described as relatively simple transformations of the images space. Before looking at how to do this, we consider the general case of transforming the camera matrix, both relative to the 3D space and relative to the image space.

8.3.1 Camera transformations

Let \mathbf{C} be a camera matrix that describes the projection of a particular pinhole camera relative to a well-defined world coordinate system and relative to a well-defined image coordinate system. What if we want to describe the projection of the same camera but now use another world coordinate system or another image coordinate system? Let us consider the second case first.

Transformations of the image space

Up to now, all image coordinates have been described relative to a coordinate system that is aligned with the 3D camera coordinate system, and has the same scale as the world coordinate system, corresponding to C-normalized image coordinates. In general, we can be interested in describing the same point in the image plane but relative to some arbitrary coordinate system. The most general transformation that is represented as a linear transformation on homogeneous coordinates is a homography. Hence, we use a homography \mathbf{H}_{2D} as a coordinate transformation, from one coordinate system to another. Due to the general character of \mathbf{H}_{2D} , however, this transformation does not necessarily preserve things that we normally associate with a coordinate system, e.g., perpendicular axes, but it always preserves straight lines as straight lines. This generality is sometimes useful and, since \mathbf{H}_{2D} is invertible, there is no risk of mixing up points before and after the transformation. At worst, points at infinity and proper points may switch character.

Let \mathbf{y} be the image coordinates of some particular 3D point \mathbf{x} relative to the coordinate system defined for \mathbf{C} . The new coordinates of \mathbf{y} after the transformation of \mathbf{H}_{2D} is given as $\mathbf{y}' \sim \mathbf{H}_{2D} \mathbf{y}$. Consequently,

$$\mathbf{y}' \sim \mathbf{H}_{2D} \mathbf{y} \sim \underbrace{\mathbf{H}_{2D} \mathbf{C} \mathbf{x}}_{:=\mathbf{C}_2}, \quad (8.24)$$

and we conclude that the camera matrix $\mathbf{C}_2 = \mathbf{H}_{2D} \mathbf{C}$ represents the projection from world coordinates to the transformed image coordinate system. In general, \mathbf{C} and \mathbf{C}_2 are two distinct matrices, but they represent the same camera. If \mathbf{n} is the camera center, i.e., $\mathbf{C}\mathbf{n} = \mathbf{0}$, then it is the center also of \mathbf{C}_2 since $\mathbf{C}_2\mathbf{n} = \mathbf{0}$.

Transformations of 3D space

In a similar way, let \mathbf{H}_{3D} be a homography transformation of 3D coordinates, transforming \mathbf{x} to $\mathbf{x}' \sim \mathbf{H}_{3D}\mathbf{x}$. Here, \mathbf{x} and \mathbf{x}' represent the same 3D point, as coordinates relative to two different world coordinate systems. If we know the transformed coordinates \mathbf{x}' , what is its image coordinates, projected through the camera? Since \mathbf{x}' and \mathbf{x} represent the same 3D point, the image coordinates must be the same as before: \mathbf{y} in the original image coordinates. This can be described as

$$\mathbf{y} \sim \mathbf{C}\mathbf{x} \sim \underbrace{\mathbf{CH}_{3D}^{-1}\mathbf{x}'}_{:=\mathbf{C}_3} = \mathbf{C}_3\mathbf{x}'. \quad (8.25)$$

Here, we have expanded \mathbf{x} in terms of the inverse transformation of \mathbf{H}_{3D} on \mathbf{x}' , and conclude that the new camera matrix \mathbf{C}_3 projects from the transformed world coordinate system to the original image coordinates. Again, \mathbf{C}_3 represent the same camera as \mathbf{C} but now relative to a different world coordinate system. The camera center in the new world coordinate system is given as $\mathbf{n}' \sim \mathbf{H}_{3D}\mathbf{n}$, and it follows that $\mathbf{C}_3\mathbf{n}' = \mathbf{C}\mathbf{n} = \mathbf{0}$.

General transformations

We summarize these results. Let \mathbf{C} be a camera that describes the projection of a particular pinhole camera relative to a well-defined world coordinate system and relative to a well-defined image coordinate system. Let \mathbf{H}_{2D} be a transformation of the image coordinates to a new coordinate system, and let \mathbf{H}_{3D} be a transformation of 3D coordinates to a new world coordinate system. The same camera is then described as the matrix

$$\mathbf{C}' \sim \mathbf{H}_{2D} \mathbf{CH}_{3D}^{-1}, \quad (8.26)$$

relative to the new coordinate systems in the image and in 3D space. In general, \mathbf{C} and \mathbf{C}' are distinct, but they represent *the same camera*, relative to two different sets of coordinate systems, in the image and in the 3D space.

8.9 For one and the same camera, its algebraic representation, the camera matrix \mathbf{C} , depends on the choice of coordinate systems, both in 3D space and in the image plane.

8.3.2 The camera at infinity

In the derivations made so far, we have not really made any particular assumptions about the coordinate systems. In particular we have not said explicitly that the camera center is a proper 3D point, or excluded the possibility that it may lie at infinity. There are at least two reasons for why the camera center may end up at infinity.

First, assuming that \mathbf{n} is a proper point relative to the world coordinate system, we may design a homography transformation \mathbf{H}_{3D} of the 3D coordinates such that $\mathbf{n}' \sim \mathbf{H}_{3D}\mathbf{n}$ is a point at infinity. This is a result of the discussion in Section 7.1.2, and simply means that we have chosen a more or less outlandish coordinate system that places the camera center at infinity. This may seem like a peculiar thing to do, but can actually happen. In some application, the camera matrix \mathbf{C} may be only partially determined. For example, as described in Section 10.3.4, \mathbf{C} can only be determined up to an unknown homography transformation of 3D space. Depending on which homography transformation is chosen, the center of the camera may or may not be a proper point.

Second, it may be the case that the camera observes some objects in a scene at a large distance from the camera, and we want to use a world coordinate system that is local to the scene. In this case, the camera center ends up at a large distance from the origin, although not at infinity. If the distance is large enough, however, it may as well be approximated as infinitely large. For example, if we try to determine \mathbf{C} by means of some estimation method, such as the one described in Section 13.3, then the errors in the data lead to errors also in the estimated \mathbf{C} , and this can make the camera center end up at a large distance from the scene. In fact, the center can end up at a distance that is very much larger than the correct distance simply because moving the center further away only corresponds to small changes in the projection of the scene. In principle, the center may even end up at infinity, even if it lies a finite distance from the scene, due to the estimation errors.

Algebraic characterization

A camera that has its center at infinity relative to the 3D coordinate system is easy to recognize. In this case the homogeneous coordinates of the center is given as

$$\mathbf{n} \sim \begin{pmatrix} \hat{\mathbf{n}} \\ 0 \end{pmatrix}, \quad \hat{\mathbf{n}} \in S^2. \quad (8.27)$$

The camera matrix \mathbf{C} can be formulated as $\mathbf{C} = (\mathbf{C}_1 | \mathbf{c}_4)$, where \mathbf{C}_1 consists of the first 3 columns of \mathbf{C} and \mathbf{c}_4 is the fourth columns of \mathbf{C} . This gives

$$\mathbf{0} = \mathbf{C}\mathbf{n} = (\mathbf{C}_1 | \mathbf{c}_4) \begin{pmatrix} \hat{\mathbf{n}} \\ 0 \end{pmatrix} = \mathbf{C}_1 \hat{\mathbf{n}}. \quad (8.28)$$

Consequently, it must be the case that $\det(\mathbf{C}_1) = 0$. Conversely, if $\det(\mathbf{C}_1) = 0$ then \mathbf{n} is a point at infinity, unless the camera matrix \mathbf{C} is rank deficient. \mathbf{C} cannot be rank deficient for a pinhole camera, and we conclude that

$$\text{camera } \mathbf{C} = (\mathbf{C}_1 | \mathbf{c}_4) \text{ has its center at infinity, } \Leftrightarrow \det(\mathbf{C}_1) = 0. \quad (8.29)$$

This result means that a normalized camera, Equation (8.17), cannot be at infinity. It also means that any camera where the center is a proper point relative to the chosen 3D coordinate system can be characterized by $\det(\mathbf{C}_1) \neq 0$. In practice, when the numerical resolution is limited, and also the scaling of \mathbf{C} as a projective element can be arbitrary, this type of test must be done with some care. It is advisable to first normalize \mathbf{C} , e.g. by scaling it such that $\|\mathbf{C}\|_F = 1$, before determining $\det(\mathbf{C}_1)$. If it is sufficiently small, the center lies effectively at infinity.

8.3.3 Internal and external camera parameters

We said earlier that we want to be able to transform from C-normalized image coordinates to pixel coordinates, in order to take care of translations of the origin, scale changes, and possibly also the directions of the coordinate axes. To this end, we assume that there is a homography transformation between the C-normalized image coordinates \mathbf{y}_n and the pixel coordinates \mathbf{y}_p :

$$\mathbf{y}_p = \mathbf{H}\mathbf{y}_n, \quad (8.30)$$

where \mathbf{H} is a homography. The use of a homography here may appear as overkill if we are only talking about uniform scaling due to variable focal length, translations of the origin, and possibly reflections due to changing the handedness of the coordinate system. In a practical applications, however, where a lens or a system of lenses are used to focus the light onto the image plane, the pinhole camera is only an approximate model of how 3D points are projected onto the image, and by using a homography transformation we can include some of the imperfections of this approximation into the transformation from normalized image coordinates to pixel coordinates.

Based on Equation (8.30) we now have an expression for how a 3D point in the world coordinate system, \mathbf{x}_w is projected onto an image point \mathbf{y}_p , expressed in pixel coordinates:

$$\mathbf{y}_p \sim \mathbf{H}\mathbf{y}_n = \mathbf{H}(\mathbf{R} | \bar{\mathbf{t}})\mathbf{x}_w, \quad (8.31)$$

and we arrive at a general formulation of the camera matrix:

$$\mathbf{C} = \mathbf{H}(\mathbf{R} | \bar{\mathbf{t}}) = \mathbf{H}\mathbf{C}_0 \begin{pmatrix} \mathbf{R} & \bar{\mathbf{t}} \\ \mathbf{0} & 1 \end{pmatrix}. \quad (8.32)$$

This camera is a composition of a 3D rigid transformation, given by \mathbf{R} and $\bar{\mathbf{t}}$, the camera centered normalized camera \mathbf{C}_0 that performs the actual projection from 3D to 2D, and a homography \mathbf{H} that transforms normalized image coordinates to pixel coordinates. We have chosen a very general type of transformation between normalized and pixel coordinates, a homography.

Notice that when the camera is at infinity relative to the world coordinate system, then $\bar{\mathbf{t}}$ is a vector of infinite norm, which cannot be plugged into Equation (8.32). Thus, whenever we decompose \mathbf{C} into a rigid transformation relative to the world coordinate system, $(\mathbf{R}, \bar{\mathbf{t}})$, and a transformation \mathbf{H} between image coordinate systems, this assumes that the camera is *not* at infinity.

Next, we will show that \mathbf{H} does not have to be a general homography. Instead, \mathbf{H} can always be replaced by a special type of affine transformation.

Reducing \mathbf{H} to \mathbf{K}

A camera matrix is 3×4 , and since it is an element of a projective space it has $12 - 1 = 11$ degrees of freedom, i.e., an arbitrary camera projection can be parameterized by 11 parameters. In accordance with Section 7.1, a general 2D homography has 8 parameters, and we have 6 more parameters to determine the rigid transformation in Equation (8.32). This implies that the camera matrix in Equation (8.32) has $8 + 6 = 14$ parameters, and this is 3 parameters to many. This happens because there are multiple choices of \mathbf{H} , \mathbf{R} , and $\bar{\mathbf{t}}$ that produce the same camera matrix \mathbf{C} . A common way to eliminate redundant parameters is to write \mathbf{H} in terms of a special RQ-factorization¹: $\mathbf{H} = \mathbf{K}\mathbf{Q}$, where \mathbf{K} is an upper triangular matrix and $\mathbf{Q} \in SO(3)$. We insert this into Equation (8.32) and get

$$\mathbf{C} = \mathbf{K}\mathbf{Q}(\mathbf{R} | \bar{\mathbf{t}}) = \mathbf{K}(\mathbf{R}' | \bar{\mathbf{t}}'), \quad \text{where } \mathbf{R}' = \mathbf{Q}\mathbf{R} \in SO(3), \quad \text{and } \bar{\mathbf{t}}' = \mathbf{Q}\bar{\mathbf{t}} \in \mathbb{R}^3. \quad (8.33)$$

Here, \mathbf{K} is 3×3 upper triangular and also an element of a projective space, which means that it has only five degrees of freedom. Together with the additional degrees of freedom from the rigid transformation defined by \mathbf{R}' and $\bar{\mathbf{t}}'$ we end up with 11 degrees of freedom. This means that the representation of a general camera matrix \mathbf{C} described in Equation (8.33) is a *minimal parameterization* in terms of number of parameters. It also means that we can replace the homography \mathbf{H} with the affine transformation \mathbf{K} when normalized image coordinates are transformed to pixel coordinates:

$$\mathbf{y}_p = \mathbf{K}\mathbf{y}_n. \quad (8.34)$$

Equation (8.33) suggests that \mathbf{C} depends on a 3D rigid transformation, defined by \mathbf{R}' and $\bar{\mathbf{t}}'$. These parameters describe where the camera center is located and how the camera centered coordinate system is oriented relative to the world coordinates system. These parameters are referred to as *external camera parameters*, the *exterior orientation*, or the *pose* of the camera. The right triangular matrix \mathbf{K} describes the transformation from C-normalized image coordinates to pixel coordinates, and is referred to as the *internal camera parameters* or the *interior orientation* of the camera. In most applications it makes sense to separate between the external and internal parameters of a camera since usually one of them is fixed. For example, if we use a fixed focal length of the camera, then \mathbf{K} is fixed although it can have variable \mathbf{R}' and $\bar{\mathbf{t}}'$ if the camera changes position and orientation in 3D space.

Internal Parameters

\mathbf{K} is a projective element, but it can be normalized by dividing it with $[\mathbf{K}]_{33}$ to the following canonical form:

$$\mathbf{K} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ 0 & k_{22} & k_{23} \\ 0 & 0 & 1 \end{pmatrix}. \quad (8.35)$$

From the outset, the five parameters in $b\mathbf{K}$ are just scalars, but they carry geometric information. A common formulation of \mathbf{K} is as follows:

$$\mathbf{K} = \begin{pmatrix} fs & \gamma & u_0 \\ 0 & fs\sigma & v_0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (8.36)$$

We will now make a geometric interpretation of the parameters in Equation (8.36). As a start we see that

$$\begin{pmatrix} u_0 \\ v_0 \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \quad (8.37)$$

Consequently, (u_0, v_0) are the pixel coordinates of the principal point, the origin of the C-normalized coordinate system. Normally, we expect that this point is located at the center of the image. If the image has r rows and c columns, this means that the principal point should end up at $(u_0, v_0) = (r, c)/2$, or close to this point.

The parameter f is the focal length of the camera and it is a parameter that can be variable in many cameras, by moving the aperture closer or farther away from the image plane. The parameter s is the *pixel density*, i.e., the number of pixels per length unit. This is definitely a fixed parameter, and it is sometimes available in the technical documentation of a specific camera, or sensor chip. The total scale factor, which transforms from units in the world coordinate system to pixel units, is then determined by the product fs . Thus, in the ideal case: $k_{11} = k_{22} = fs$.

¹The special RQ-decomposition is described in Toolbox Section 8.7.1.

The uniform scaling specified by $k_{11} = k_{22}$ is sometimes not an accurate formulation of the mapping to pixel coordinates. In practice, imperfections in the camera lenses, and in the manufacturing of the sensor chip, can lead to a small difference in the horizontal and vertical scalings. This difference is described by the scaling parameter σ , which represents a relative scale of the horizontal and vertical orientations. In the ideal case, $\sigma = 1$, but in practice it can have a small deviation from this value. Here, we assume that both f and s are positive, that the coordinate system in 3D is right-handed, and that the image coordinate system is counter-clockwise. This implies that $k_{11}, k_{22} > 0$, which has certain consequences that are discussed in Section 8.5.5.

The lens system of the camera should ideally project light rays such that a rectangle that lies in a plane parallel to the image plane is mapped by the camera to a rectangle in the image. In practice this assumption is only approximately correct, and as a consequence the rectangle may appear slightly sheared. To some extent this imperfection can be described by the shearing parameter γ , which in the ideal case should vanish.

If $\gamma \neq 0$ this indicates a situation where the two coordinate axes in the pixel coordinate system are not perpendicular. This is mainly an optical effect related to a geometric distortion² caused by the camera lens, in combination with other geometric inaccuracies such as the image plane not being exactly perpendicular to the optical axis of the lens. It *does not* mean that the sensor chip or the pixel grid are sheared.

The minimal parameterization of \mathbf{C} described here implies that the homography transformation \mathbf{H} in Equation (8.32), which transforms from normalized image coordinates to pixel coordinates, can be simplified to an affine transformation that has a triangular form. This is just one of infinitely many ways of decomposing the camera matrix, and it should not be taken as an indication that \mathbf{H} must be of a triangular form. Instead, it means that if \mathbf{H} is not triangular, we can always rotate the camera centered coordinate system relative to the world coordinate system by means of \mathbf{Q} in Equation (8.33), such that the homography becomes triangular. It is possible to instead use an arbitrary matrix \mathbf{K} that has five independent parameters, five degrees of freedom, as long as it is non-singular. The one proposed here in Equation (8.36), however, has a geometric interpretation that is intuitive to understand and is therefore very common in the literature.

8.10 The standard representation of internal parameters, \mathbf{K} in Equation (8.36), has five degrees of freedom.

External parameters

As already described in Section 8.2.1, the camera center of the normalized camera ($\mathbf{R} | \bar{\mathbf{t}}$) is given as $\bar{\mathbf{n}} = -\mathbf{R}^\top \bar{\mathbf{t}}$. The multiplication by the internal parameter matrix \mathbf{K} onto this normalized camera does not change the camera center, i.e., the camera center of \mathbf{C} in Equation (8.33) is $\bar{\mathbf{n}} = -\mathbf{R}^\top \bar{\mathbf{t}}$.

The rotation \mathbf{R} describes how Cartesian coordinates, relative the world coordinate system, are rotated before the translation by $\bar{\mathbf{t}}$ is made, to become coordinates relative to the camera coordinate system. Consequently, $\mathbf{R}^{-1} = \mathbf{R}^\top$ describes the rotation of the inverse transformation. The matrix \mathbf{R} can be represented in terms of its rows:

$$\mathbf{R} = \begin{pmatrix} - & \bar{\mathbf{r}}_1^\top & - \\ - & \bar{\mathbf{r}}_2^\top & - \\ - & \bar{\mathbf{r}}_3^\top & - \end{pmatrix}, \quad (8.38)$$

The three axes of the camera coordinate system, in the camera centered coordinate system, are expressed as

$$\hat{\mathbf{e}}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \hat{\mathbf{e}}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \hat{\mathbf{e}}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \quad (8.39)$$

The transformation of these axes to the world coordinate system is given as

$$\hat{\mathbf{e}}'_1 = \mathbf{R}^\top \hat{\mathbf{e}}_1 = \bar{\mathbf{r}}_1, \quad \hat{\mathbf{e}}'_2 = \mathbf{R}^\top \hat{\mathbf{e}}_2 = \bar{\mathbf{r}}_2, \quad \hat{\mathbf{e}}'_3 = \mathbf{R}^\top \hat{\mathbf{e}}_3 = \bar{\mathbf{r}}_3. \quad (8.40)$$

²In this presentation, we will sometimes refer to these effects as deformations, i.e., deformation and geometric distortion are synonymous.

8.11 We can find the three camera axes, expressed in the world coordinate system, directly in the rows of \mathbf{R} . In particular, the viewing direction is in principle determined by the third row of \mathbf{R} .

In the next section we will see that $\hat{\mathbf{e}}_3$ may also point opposite to the viewing direction depending on the handedness of the image coordinate system, and on the signs of k_{11} and k_{22} .

8.3.4 Camera resectioning

The idea of expressing the camera matrix as a product of the internal parameters in \mathbf{K} and the external parameters $(\mathbf{R}, \bar{\mathbf{t}})$, is based on transforming the image space of a given normalized camera. What if \mathbf{C} represents a general camera, with 11 free parameters? Can \mathbf{C} somehow be decomposed into a minimal set of internal and external parameters, as in Equation (8.33)? And if so, is the decomposition unique? The answer is: if \mathbf{C} is not at infinity then we can always find a unique decomposition of \mathbf{C} into its internal and external parameters. This decomposition is referred to as *camera resectioning*, it splits the camera matrix \mathbf{C} into its internal parameters in terms of the upper triangular 3×3 matrix \mathbf{K} , and external parameters in terms of $\mathbf{R} \in SO(3)$ and $\bar{\mathbf{t}} \in \mathbb{R}^3$.

To derive the resectioning of a general \mathbf{C} , we write $\mathbf{C} = (\mathbf{C}_1 | \mathbf{c}_4)$, where \mathbf{C}_1 is the leftmost 3×3 submatrix of \mathbf{C} and \mathbf{c}_4 is the fourth column of \mathbf{C} , and then compute a special RQ-decomposition of \mathbf{C}_1 , as described in Toolbox Section 8.7.1. This gives $\mathbf{C}_1 = \mathbf{K}\mathbf{Q}$, where \mathbf{K} is upper triangular and $\mathbf{Q} \in SO(3)$, and leads to

$$\mathbf{C} = (\mathbf{K}\mathbf{Q} | \mathbf{c}_4) = \mathbf{K}(\mathbf{Q} | \bar{\mathbf{t}}), \quad (8.41)$$

where $\mathbf{K} = \mathbf{C}_1\mathbf{Q}^\top$ and $\bar{\mathbf{t}} = \mathbf{K}^{-1}\mathbf{c}_4$. Since the camera is not at infinity, it follows that $\det(\mathbf{K}) \neq 0$ and \mathbf{K} can be inverted. To turn \mathbf{K} into the canonical form in Equation (8.36), we divide \mathbf{K} by its lower right element $[\mathbf{K}]_{33}$.

As a final step, we need to take care of the signs in \mathbf{K} . In accordance with Section 8.2, we remind that changing the signs of both image coordinates corresponds to a 180° rotation of the coordinate system. Furthermore, we recall from Section 4.4.2 that if \mathbf{K} includes a reflection, this is equivalent to $\det(\mathbf{K}) < 0$. It is a straightforward exercise to show that $\det(\mathbf{K}) = k_{11}k_{22}$, assuming that $k_{33} = 1$.

If the image coordinate axes $\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2$ have the same handedness as $\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2$, and we assume that $\hat{\mathbf{e}}_3$ points in the viewing direction, then $\det(\mathbf{K}) > 0$ since no reflection is involved in the transformation from C-normalized image coordinates to pixel coordinates. This leaves two possible outcomes for k_{11} and k_{22} : either they are both positive, or both are negative. In the latter case, the negative signs of k_{11} and k_{22} can be interpreted as a 180° rotation of the image plane, and can be compensated for by rotating the camera coordinate system 180° about the $\hat{\mathbf{e}}_3$ axis.

The other case is that the basis $\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2$ has opposite handedness relative to $\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2$, when $\hat{\mathbf{e}}_3$ points in the viewing direction. In this case, $\det(\mathbf{K}) < 0$, since \mathbf{K} includes a reflection relative to either of coordinate axes. This cannot be undone by a rotation in the image plane, but it is possible to instead rotate the 3D camera coordinate system such that $\hat{\mathbf{e}}_3$ now points opposite to the viewing direction. This changes the handedness of the bases $\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2$ so that it now is the same as for $\hat{\mathbf{b}}_1, \hat{\mathbf{b}}_2$. After this rotation, possibly in combination with a 180° rotation, k_{11} and k_{22} are both positive again. However, this has been accomplished at the expense of flipping the direction of $\hat{\mathbf{e}}_3$.

In this presentation, we always assume that any 3D coordinate system is right-handed. This includes the camera coordinate system with $\hat{\mathbf{e}}_1$ and $\hat{\mathbf{e}}_2$ clockwise in the image plane, for example \mathbf{e}_1 right and \mathbf{e}_2 down. If we choose to rotate the camera coordinate system, or not, depends on whether we are comfortable with $\hat{\mathbf{e}}_3$ in the negative viewing direction. In combination, these observations give us four possible outcomes of an operation that manipulates the signs of k_{11}, k_{22} and of $\hat{\mathbf{e}}_3$:

- The image coordinate system is clockwise, k_{11} and k_{22} are both positive, and $\hat{\mathbf{e}}_3$ points in the viewing direction of the camera.
- The image coordinate system is clockwise, exactly one of k_{11} or k_{22} is negative, and $\hat{\mathbf{e}}_3$ points in the negative viewing direction of the camera.
- The image coordinate system is counter-clockwise, k_{11} and k_{22} are both positive, and $\hat{\mathbf{e}}_3$ points in the negative viewing direction of the camera.

Algorithm 8.1: Camera resectioning

```

Input:  $3 \times 4$  camera matrix  $\mathbf{C}$ 
Output: Upper triangular  $3 \times 3$  matrix  $\mathbf{K}$ , Equation (8.35), where  $k_{11}, k_{22} > 0$ 
Output:  $\mathbf{R} \in SO(3)$ , and  $\bar{\mathbf{t}} \in \mathbb{R}^3$  such that  $\mathbf{C} \sim \mathbf{K}(\mathbf{R}|\bar{\mathbf{t}})$ 
1 With  $\mathbf{C} = (\mathbf{A}|\mathbf{b})$ : compute a special RQ-decomposition3 of  $\mathbf{A} = \mathbf{U}\mathbf{Q}$ , where  $\mathbf{Q} \in SO(3)$ 
2 Set:  $\bar{\mathbf{t}} = \mathbf{U}^{-1}\mathbf{b}$ 
3 Normalize:  $\mathbf{U} = \mathbf{U}/[\mathbf{U}]_{33}$  // Assures  $[\mathbf{U}]_{33} = 1$ 
4 /* Assure that  $k_{11}, k_{22} > 0$  by applying transformations on  $\mathbf{K}, \mathbf{R}, \bar{\mathbf{t}}$  */
5 Set:  $\mathbf{D} = 3 \times 3$  diagonal matrix, where  $[\mathbf{D}]_{ii} = \text{sign}[\mathbf{U}]_{ii}$ 
6 Set  $\mathbf{K} = \mathbf{U}\mathbf{D}$  // Now  $k_{11}, k_{22} > 0$ 
7 if  $\det \mathbf{D} = +1$  then
8   Set  $\mathbf{R} = \mathbf{D}\mathbf{Q}$  and  $\bar{\mathbf{t}} = \mathbf{D}\bar{\mathbf{t}}$  // Compensates for the multiplication by  $\mathbf{D}$  on  $\mathbf{K}$ 
9 else
10  Set  $\mathbf{R} = -\mathbf{D}\mathbf{Q}$  and  $\bar{\mathbf{t}} = -\mathbf{D}\bar{\mathbf{t}}$  // Assures that  $\mathbf{R} \in SO(3)$ 
11 Return  $\mathbf{K}, \mathbf{R}, \bar{\mathbf{t}}$ 

```

- The image coordinate system is counter-clockwise, exactly one of k_{11} or k_{22} is negative, and $\hat{\mathbf{e}}_3$ points in the viewing direction of the camera.

An algorithm that resects a camera matrix \mathbf{C} such that $k_{11}, k_{22} > 0$ is presented in Algorithm 8.1. Given the above discussion, we need to know the handedness of the image coordinate system in order to make a correct interpretation of the camera's viewing direction from the resulting rotation \mathbf{R} .

We will continue this discussion in Section 8.5.5.

Camera calibration

The numerical value of the matrix \mathbf{C} for a particular camera is relevant for solving many practical problems. For example, if we know the 3D coordinates of a point in \mathbb{E}^3 and want to find the image point the camera projects it to, we need to know the corresponding camera matrix in order to use Equation (8.4). An important observation, which has already been mentioned, is that one and the same physical camera can be described by different matrices depending on which coordinate systems that are used in 3D space and in the image plane. This means that we need to specify the coordinate systems that a particular camera matrix refers to in order to use its numerical representation, and also in order to apply it in numerical computations to project 3D points to image points.

The process of determining the camera matrix \mathbf{C} for a specific pinhole camera is referred to as *camera calibration*. Since this theory behind this process requires additional material, we will not discuss this issue here. Instead, we will return to camera calibration in Chapter 18, and to some extent already in Section 13.3.

Concluding remarks

We conclude this section by summarizing the general form of the camera matrix as:

$$\mathbf{C} \sim \mathbf{K}(\mathbf{R}|\bar{\mathbf{t}}). \quad (8.42)$$

Here, \mathbf{K} is 3×3 upper triangular containing the internal parameters. $(\mathbf{R}, \bar{\mathbf{t}})$ represent the rigid transformation from the world coordinate system, which we use to express coordinates of 3D points, to the camera coordinate system. Finally, \mathbf{K} transforms C-normalized image coordinates to pixel coordinates. The inverse transformation \mathbf{K}^{-1} transforms pixel coordinates to C-normalized image coordinates:

$$\mathbf{y}_n \sim \mathbf{K}^{-1}\mathbf{y}_p. \quad (8.43)$$

Incidentally, also \mathbf{K}^{-1} is upper triangular. The resectioning of \mathbf{C} in Equation (8.42) is unique if $k_{11}, k_{22} > 0$, but to make a correct interpretation of the coordinate systems, we also need to know their handedness.

³See Toolbox Algorithm 3 on page 122.

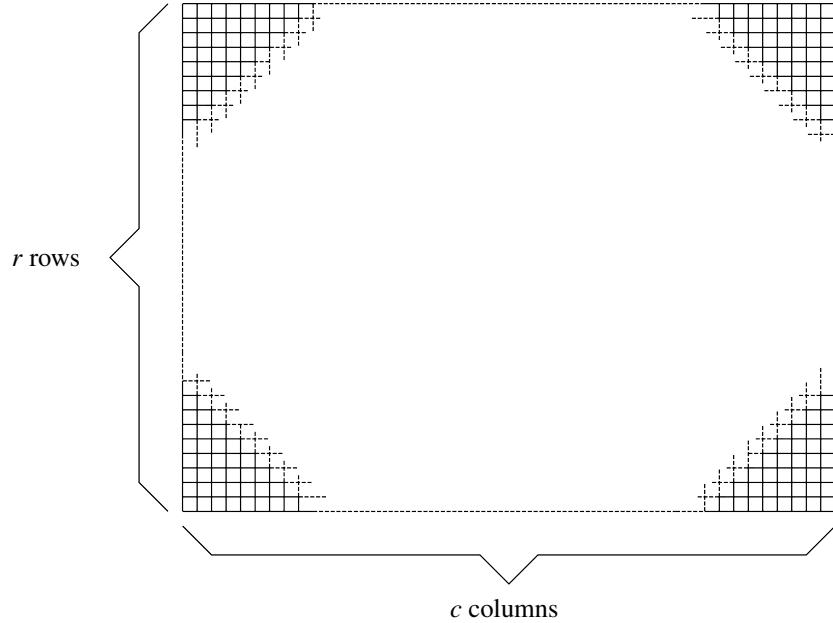


Figure 8.7: The pixel array, consisting of an $r \times c$ matrix of pixels.

The form in Equation (8.42) can only be accomplished if the camera center is not at infinity. Apart from this restriction, however, \mathbf{C} can be any general camera matrix. This implies that even if we apply a general homography transformation \mathbf{H}_{3D} to a camera \mathbf{C} , in accordance with Equation (8.25), not moving the center to infinity, the transformed matrix \mathbf{C}' can be resectioned as in Equation (8.42). In the case that the coordinate systems of the world and of the camera are related by a general homography, however, we can expect to see k_{11} and k_{22} with large difference in magnitude and k_{12} that is large.

8.4 The digital image

Until now we have discussed a pinhole camera as a more or less abstract geometric device that projects 3D space onto the image plane, be it virtual or not. Although this description is useful, there are some details missing that sometimes must be taken into consideration when the results presented here are implemented in practice, e.g., in software, and in particular when we are dealing with digital images. In this section we discuss some of these issues.

The bounding box

We have already mentioned that the image produced by a camera must be of finite extension, i.e., it is defined only within a region in the image plane. In principle, this region can have any reasonable shape, but in practice it is almost always a rectangle, often referred to as the *bounding box* of the image. Only a small number of rather exotic cameras devices produce images that are not rectangular, and we do not consider such options here. In a digital camera the bounding box is defined by the light sensitive area of the sensor chip of the camera, and for a film based camera it is defined by the size of the film. Only light rays that pass through the camera center and intersect the image plane within the bounding box can be detected by the camera.

The pixel grid

In a digital camera, the bounding box rectangle is tessellated into a *pixel grid*, a two-dimensional array or matrix, where each entry corresponds to a pixel. This array has a certain number of rows, r , and columns, c , forming an $r \times c$ matrix of pixels. We can think of each pixel as a small square, such that $r \times c$ of these squares make up the entire image, see Figure 8.7. This tessellation means that we can express the size of the bounding box, in pixel coordinates, as $r \times c$. Using Equations (8.34) and (8.36), the size of the bounding box can also be expressed in

C-normalized coordinates, as a height, h , and a width w , which are related to r and c as

$$w = r f s, \quad h = c f s. \quad (8.44)$$

Here, f and s are the camera's focal length and pixel scale, respectively, as defined in Section 8.3.3. For simplicity, the relative pixel scale σ and the shearing parameter γ are here set to one and zero, respectively.⁴

The pixel value

Each pixel holds information about the intensity or color at the corresponding position in the image, the *pixel value*. For a greyscale image, the pixel value represents as a single measurement of intensity, where a larger value corresponds to a brighter intensity of the image at the pixel's position. In practice, the pixel value is an integer within some range that depends on b = how many bits are used to represent the pixel value. For a color image, the pixel value is typically holds three such integers, representing intensity in three wavelength bands of light. These are often called "red", "green", and "blue", but other bands are also used. We will not go deeper into this topic here, other than to define the range of possible pixel values. For a greyscale image of b bits per pixel, the range is

$$R_0 = \{x \in \mathbb{Z}, \text{ where } 0 \leq x \leq 2^b - 1\}, \quad (8.45)$$

and for a color image of intensities in three different bands per pixel, the range is

$$R = R_0 \times R_0 \times R_0. \quad (8.46)$$

Exactly where in the image plane is the value of a specific pixel measured? Is at a specific point related to the pixel position? Or has it been integrated over the square of the pixel? A full answer to this question falls outside the scope of this presentation, but it lies somewhere between the two proposals; the intensity or color measurements has been integrated over a smaller area inside the square represented by the pixel. In practice, it is sufficient to think of the pixel value as a measurement of light within an area which is so small that, for most practical situations, it can be approximated as a point. Furthermore, these points lie sufficiently close to make it possible to resolve any details in the image that the camera lens or system of lenses is able to project onto the image plane.⁵

Coordinate axes

A digital image is here described a pixel grid, where each pixel holds a value, but in order to manipulate the image we need to able to address a particular pixel in the array. In pixel coordinates, this can be done simply by specifying the row and column of the pixel in the array. However, different software that supports manipulation of images have different approaches to how to do this. There are some issues that need to be observed to get this right:

- When we specify a pixel coordinate as (u, v) it can either be that u refers to the row and v to the column, or the other way around. In this compendium we normally use the second option, where coordinates are specified as (column, row). There are, however, literature and software that use the alternative convention.
- When we talk about the first row, it must be clarified if we mean the top-most row or the one at the bottom of the image. In the first case, increasing the row index moves us downward, and in the second case it moves us upward. Similarly, is the first column the left-most or the right-most one? In the first case, increasing the column index moves us right, and in the second case it moves us left. By far, the alternative where the first row is at the top and first column is the left-most is the most common, but some software uses other options.

These points means specifying the order of the two axes of the pixel coordinate system, and in which directions they point. In this presentation, the first axis points right, along the columns, and the second axis points down, along the rows. We refer to this type of coordinate system as (right, down). Once the directions of the two coordinate axes are specified, this sets the origin of the pixel coordinate system to one of the four corners of the bounding box. For our pixel coordinate system, the origin is located at the upper left corner of the image.

⁴If $\gamma \neq 0$, the bounding box in C-normalized coordinates would be in the form of a parallelepiped rather than a rectangle.

⁵Due to diffraction effects caused by the light's wave-nature, as well as imperfections in the lens system, there is a limit to how small details can be projected by the camera lenses onto the image plane. In a well-designed camera there is a balance between the pixel resolution, i.e., how dense the pixels lie, and the optical resolution of the lenses. For a fixed lens system, adding more pixels and thereby increasing the pixel resolution does not make the image sharper, it just gives you more pixels.

The origin

The enumeration of rows and columns can be based on either of two principles. In one approach, the first row or column in the array is labeled as “0”, which means that the rows and columns are indexed by integers in the ranges

$$\mathbb{Z}_r = \{i \in \mathbb{Z}, \text{ where } 0 \leq i \leq r - 1\}, \quad \mathbb{Z}_c = \{i \in \mathbb{Z}, \text{ where } 0 \leq i \leq c - 1\}, \quad (8.47)$$

respectively. This convention is used in many programming languages, such as C, Java, Python, etc. We refer to this convention as *zero-first enumeration of coordinates*.

Alternatively, the first row or column is labeled as “1”, which means that the rows and columns are indexed by integers in the ranges

$$\mathbb{Z}_r = \{i \in \mathbb{Z}, \text{ where } 1 \leq i \leq r\}, \quad \mathbb{Z}_c = \{i \in \mathbb{Z}, \text{ where } 1 \leq i \leq c\}, \quad (8.48)$$

respectively. This convention is often used in software for mathematical computations, such as Matlab, Octave, Mathematica, etc. We refer to this convention as *one-first enumeration of coordinates*.

The labeling of the first row and first column of the image, together with the order in which row and column indices are listed as pixel coordinates, imply a domain D for the pixels’ coordinates such that they lie within the bounding box of the image. For example, in the (right,down) type of coordinate system, and with the zero-first enumeration convention, the domain of the pixel coordinate (u, v) is given by

$$D = \mathbb{Z}_c \times \mathbb{Z}_r = \{(u, v) \in \mathbb{Z}^2, \text{ where } 0 \leq u \leq c - 1, \text{ and } 0 \leq v \leq r - 1\} \quad (8.49)$$

The pixel coordinate system

Defining the coordinate axes and specifying the numeric index of the first row and column is equivalent to establishing the pixel coordinate system in the image plane. Even though the value related to a pixel in practice is integrated over some area inside the “pixel square”, the coordinate of the same pixel is a point. Intuitively, we can think of this point as the center of the square, but since it is connected to an area that refers to the pixel value, we should also acknowledge that the position of a pixel must be associated with some degree of uncertainty. In fact, this uncertainty is part of what will be called *measurement noise* in Part II. This uncertainty could be in the order of half the distance between two neighboring pixels, i.e., 0.5 coordinate units in the pixel coordinate system.⁶

If we use the (right,down) type of coordinate system, and the zero-first enumeration convention, the origin of the pixel coordinate system is centered on the pixel in the upper left corner of the image. With the same type of coordinate system, but using the one-first enumeration convention, the origin is instead located one pixel unit above and to the left of this corner pixel. In both cases, the first coordinate axis points right from the origin and the second points down, see Figure 8.8.

Summary

The practical issues discussed in this section can be summarized by stating that an image can be represented either as a function or as a geometrical object. In the first case, we can represent the image as a function I :

$$I : D \rightarrow R, \quad (8.50)$$

where D is the domain of the pixels’ coordinates and R is the range of the pixel value, as previously defined. When we say that D is the domain of the pixels’ coordinates, D should be interpreted as the set of coordinates of the pixels in the bounding box of the image. For each coordinate (u, v) in D there is a corresponding pixel in the image, with an associated pixel value $I(u, v)$. And for each pixel in the image, we find its coordinate in D .

However, we are also free to consider arbitrary points in the image plane, with coordinates that are not in D . For example, points outside the bounding box or points with non-integer coordinates. A point outside the bounding box is well-defined as a point, but its pixel value is undefined. A point inside the bounding box but with non-integer pixel coordinate, is also well-defined as a point, but which pixel value does it have? If \bar{y} is a non-integer pixel coordinate inside the bounding box, there are integer valued coordinates nearby that can be used

⁶This value should be seen as lower bound. The measurement noise is also affected by the optical resolution of the lens system, and can be significantly larger than 0.5 if the lenses do not project a sharp image onto the image plane.

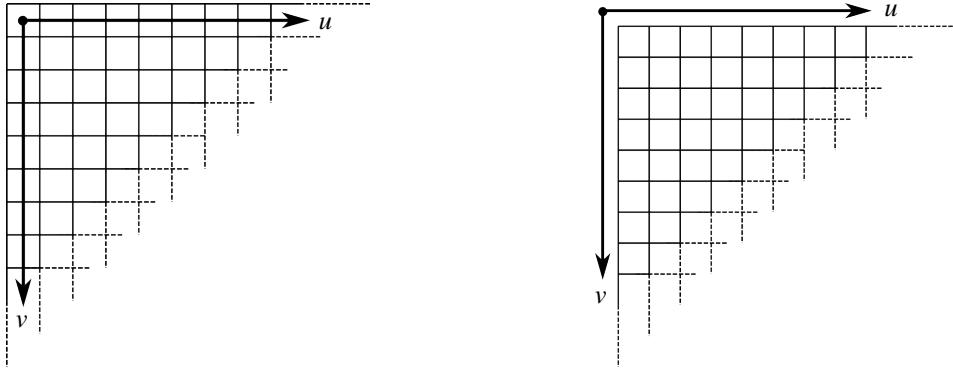


Figure 8.8: The coordinate system of type (right, down) for pixel coordinates (u, v) . Left: with zero-first enumeration. Right: with one-first enumeration.

to interpolate a pixel value $I(\bar{y})$. The simplest approach to determine $I(\bar{y})$ in this case is to use the closest integer valued point, given as $\text{round}(\bar{y})$ and its associated pixel value $I(\text{round}(\bar{y}))$. This method of assigning a value for $I(\bar{y})$ is referred to as *nearest neighbor interpolation*. Other, more advanced, interpolation schemes can also be used, but may require more calculations to determine $I(\bar{y})$ [24].

To see an image as a function is relevant in situations where we discuss transformations of images, when we sometimes need to characterize the pixel value at pixel coordinate (u, v) , where u and v are not integers but still within the bounding box. This representation is not very common in this presentation, which mostly deals with geometry, but see Chapter 19 for an example.

As a geometric entity the image appears as the image grid, an array or matrix filled with “pixel squares”, which lies in the image plane. Connected to this array is the pixel coordinate system, where the origin lies at one of the four corners, either spot on the corner pixel (zero-first enumeration) or displaced one unit both vertically and horizontally (one-first enumeration). The coordinate axes point either along the rows or columns, up or down, and left or right, depending on the type of coordinate system used. You need to be fully aware of which of the different options that are used for the particular pixel coordinate system defined by your software.

8.5 The geometry of a pinhole camera

Having examined the pinhole camera itself, we now turn to a set of geometric and algebraic consequences of this construction, e.g., in terms of how various types of geometric objects are projected into the image or how various geometric objects in the image can be projected out in 3D space.

8.5.1 Projection lines

Let \mathbf{x} a 3D point, where $\mathbf{x} \not\sim \mathbf{n}$. It generates a 3D line \mathbf{L}_x that passes through both \mathbf{x} and \mathbf{n} , with Plücker coordinates:

$$\mathbf{L}_x \sim \mathbf{x}\mathbf{n}^\top - \mathbf{n}\mathbf{x}^\top. \quad (8.51)$$

This line intersects the image plane at the image point \mathbf{y} , see Figure 8.9. In that respect, \mathbf{x} is not unique. All points on the line \mathbf{L}_x , except for \mathbf{n} , project onto \mathbf{y} , and they all generate the same line. The line \mathbf{L}_x is referred to as a *projection line* or an *optical ray*. The set of all projection lines of a specific camera consists precisely of the lines that intersect the camera center \mathbf{n} .

A natural consequence of the previous observation is that we cannot define an inverse camera mapping, producing the 3D point \mathbf{x} from its projection \mathbf{y} in the image. Instead, to each image point we can construct a projection line \mathbf{L}_y that intersects \mathbf{y} in the image plane and passes through the camera center \mathbf{n} , see Figure 8.9. This projection line contains all 3D points that project to the image point \mathbf{y} . To construct the Plücker coordinates of the projection

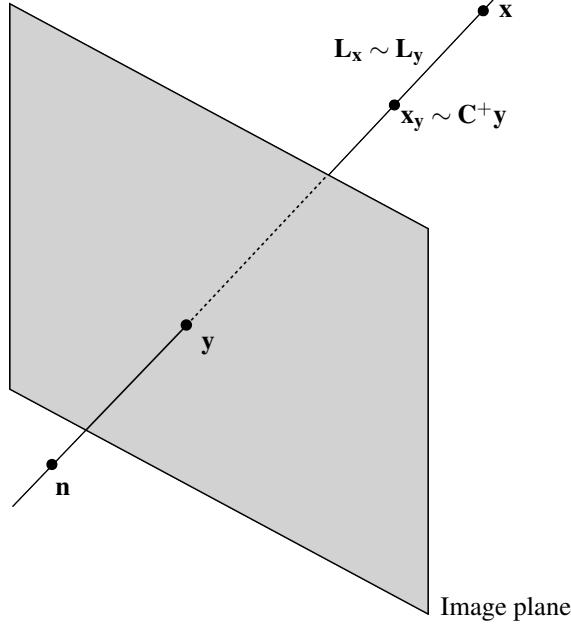


Figure 8.9: A 3D point \mathbf{x} and the corresponding projection line \mathbf{L}_x , intersecting the camera center \mathbf{n} and the image plane at image point \mathbf{y} . The same line can also be seen as the projection line \mathbf{L}_y generated by image point \mathbf{y} containing all 3D points that project to \mathbf{y} . The 3D point \mathbf{x}_y is generated from \mathbf{y} . It is distinct from \mathbf{n} and lies on \mathbf{L}_y .

line, we need two points on the line, where one can be \mathbf{n} . A second point is given as $\mathbf{x}_y \sim \mathbf{C}^+ \mathbf{y}$. The point \mathbf{x}_y lies on the projection line \mathbf{L}_y , since it projects to \mathbf{y} by the camera⁷:

$$\mathbf{C} \mathbf{x}_y \sim \mathbf{C} \mathbf{C}^+ \mathbf{y} \sim \mathbf{I} \mathbf{y} \sim \mathbf{y}. \quad (8.52)$$

Furthermore⁸, $\mathbf{x}_y \neq \mathbf{n}$ since \mathbf{n} is a null vector of \mathbf{C} and \mathbf{x}_y lies in the range of \mathbf{C}^\top . In short, two distinct points on the projection line are given by $\mathbf{x}_y \sim \mathbf{C}^+ \mathbf{y}$ and \mathbf{n} . Consequently, the Plücker coordinates of the projection line is given as

$$\mathbf{L}_y \sim (\mathbf{C}^+ \mathbf{y}) \mathbf{n}^\top - \mathbf{n} (\mathbf{C}^+ \mathbf{y})^\top = \mathbf{C}^+ \mathbf{y} \mathbf{n}^\top - \mathbf{n} \mathbf{y}^\top \mathbf{C}^{+\top}. \quad (8.53)$$

In summary, a projection line of a camera always intersects the camera center. It may be defined either in terms of an additional 3D point \mathbf{x} , or in terms of an 2D image point \mathbf{y} , which the line must intersect. Which of the two cases a projection line refers to should be clear from the context in which it appears.

8.5.2 The image of a line

A pinhole camera projects a line in 3D space to a line in the image space. There are some special lines for which this statement is not true, as will be discussed shortly, but for general lines the statement is correct. To see this, let \mathbf{L} be the Plücker coordinates of a 3D line:

$$\mathbf{L} = \mathbf{x}_1 \mathbf{x}_2^\top - \mathbf{x}_2 \mathbf{x}_1^\top, \quad (8.54)$$

where $\mathbf{x}_1, \mathbf{x}_2$ are two distinct 3D points on the line. Using the parametric representation of a 3D line described in Section 5.3.1, in terms of a point $\mathbf{x}(\lambda)$ on the line, is given as

$$\mathbf{x}(\lambda) \sim \lambda \mathbf{x}_1 + (1 - \lambda) \mathbf{x}_2, \quad \lambda \in \mathbb{R}. \quad (8.55)$$

The projection of these 3D points onto the camera image plane is the image of the 3D line:

$$\mathbf{y}(\lambda) \sim \mathbf{C} (\lambda \mathbf{x}_1 + (1 - \lambda) \mathbf{x}_2) = \lambda \mathbf{y}_1 + (1 - \lambda) \mathbf{y}_2, \quad (8.56)$$

⁷Here we are using the full rank pseudo-inverse defined in Toolbox Section 3.4.2.

⁸Here we are using the relation between the null space of \mathbf{C} and range of \mathbf{C}^\top described in Toolbox Section 3.2.4, Equation (3.40).

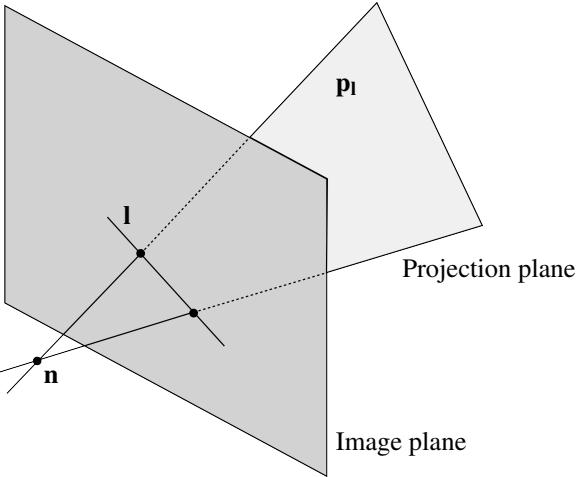


Figure 8.10: A line \mathbf{I} in the image and its corresponding projection plane $\mathbf{p}_\mathbf{I}$, which intersects both \mathbf{I} in the image plane and the camera center \mathbf{n} .

where $\mathbf{y}_1 \sim \mathbf{C}\mathbf{x}_1$ and $\mathbf{y}_2 \sim \mathbf{C}\mathbf{x}_2$. Taking into account that the right-hand side of the previous relation is a projective element, i.e., it can be multiplied by an arbitrary non-zero scalar, this represents a 2-dimensional subspace of \mathbb{R}^3 . In accordance with Section 3.7, it has 2D Plücker coordinates represented by $\mathbf{y}_1\mathbf{y}_2^\top - \mathbf{y}_2\mathbf{y}_1^\top$. Furthermore, these Plücker coordinates represent a 2D line \mathbf{I} given as

$$\mathbf{I} \sim [\mathbf{y}_1\mathbf{y}_2^\top - \mathbf{y}_2\mathbf{y}_1^\top]^\times \sim [\mathbf{C}\mathbf{x}_1\mathbf{x}_2^\top \mathbf{C}^\top - \mathbf{C}\mathbf{x}_2\mathbf{x}_1^\top \mathbf{C}^\top]^\times \sim [\mathbf{C}\mathbf{L}\mathbf{C}^\top]^\times. \quad (8.57)$$

This expression describes a mapping of the 3D line \mathbf{L} as defined by the projection of the pinhole camera \mathbf{C} , and implies that the result is a 2D line \mathbf{I} .

The line \mathbf{I} is well-defined except when \mathbf{L} intersects the camera center \mathbf{n} . In this case, we can choose $\mathbf{x}_1 = \mathbf{n}$, and inserted into Equation (8.57) this gives

$$\mathbf{I} \sim [\underbrace{\mathbf{C}\mathbf{n}}_{=0} \mathbf{x}_2^\top \mathbf{C}^\top - \mathbf{C}\mathbf{x}_2 \underbrace{\mathbf{n}^\top \mathbf{C}^\top}_{=0}]^\times = \mathbf{0}. \quad (8.58)$$

The explanation is that when \mathbf{L} intersects the camera center it acts as a projection line of the camera: all points on \mathbf{L} are projected onto a single image point, given by $\mathbf{C}\mathbf{x}_2$.

8.5.3 Projection planes

Section 8.5.1 constructs a projection line \mathbf{L}_y corresponding to the image point \mathbf{y} , containing all 3D points that project onto \mathbf{y} by the camera \mathbf{C} . Given the results of the previous section, we can extend this result from the point \mathbf{y} to a line \mathbf{I} in the image, producing a *projection plane* $\mathbf{p}_\mathbf{I}$ that contains all points in 3D space that are projected onto points on the line \mathbf{I} or, equivalently, all 3D lines that are projected onto the line \mathbf{I} .

We want to find an algebraic connection between $\mathbf{p}_\mathbf{I}$ and \mathbf{I} , and a suitable candidate is:

$$\mathbf{p}_\mathbf{I} \sim \mathbf{C}^\top \mathbf{I}. \quad (8.59)$$

From the outset, we may not even be sure that the right-hand side of this expression should be interpreted as a plane in 3D space, but it is a vector in \mathbb{R}^4 so we can always think of it as the dual homogeneous coordinates of some plane, denoted $\mathbf{p}_\mathbf{I}$. Let \mathbf{x} be a 3D point in this plane: $\mathbf{x} \cdot \mathbf{p}_\mathbf{I} = 0$, from which follows:

$$0 = \mathbf{x}^\top \mathbf{p}_\mathbf{I} = \mathbf{x}^\top \mathbf{C}^\top \mathbf{I} = (\mathbf{C}\mathbf{x})^\top \mathbf{I}. \quad (8.60)$$

Consequently, $\mathbf{C}\mathbf{x}$, the projection of \mathbf{x} into the camera image, is a point on the line \mathbf{I} , and it follows that \mathbf{x} lies in the projection plane of \mathbf{I} . Since \mathbf{x} is any arbitrary point in the projection plane, it follows that Equation (8.59) is a correct expression for the dual homogeneous coordinates of this plane. The relation between the image line \mathbf{I} and the projection plane $\mathbf{p}_\mathbf{I}$ is illustrated in Figure 8.10.

8.5.4 Camera projection constraints

Based on the previous results in this chapter, we will here make explicit the types of constraints that appear when a point or a line in 3D space are projected by a pinhole cameras \mathbf{C} , to a point or a line, respectively, in the camera image. We begin with the point case.

Constraints from projecting a 3D point

Let \mathbf{x} be a 3D point. Its projection in the camera image is given as the point $\mathbf{y} \sim \mathbf{C}\mathbf{x}$, Equation (8.4). Using a technique that later⁹ will be introduced as *direct linear transformation*, we can also formulate an equivalent relation between the camera \mathbf{C} , the 3D point \mathbf{x} , and the image point \mathbf{y} as

$$\mathbf{0} = [\mathbf{y}]_{\times} \mathbf{C} \mathbf{x}. \quad (8.61)$$

This amounts to 3 equations in the elements of \mathbf{C} , \mathbf{x} , and \mathbf{y} , but since the cross product operator¹⁰ has rank 2, one of the three equations is linearly dependent on the other two. In general, we can simply remove any of the three equations in Equation (8.61) and use the remaining two as constraints. In summary: Equation (8.61) corresponds to two linearly independent constraints in the elements of \mathbf{C} , \mathbf{x} , and \mathbf{y} . In fact, these two constraints are also *algebraically independent*, i.e., each can be satisfied or not independent of the other.

The constraints that emerge in this way can be given different interpretations. For example, if \mathbf{x} and \mathbf{y} are known but \mathbf{C} is unknown, we can see Equation (8.61) as constraints in the elements of \mathbf{C} . This is the basis for the discussion in Section 13.3, where \mathbf{C} is determined from a set of corresponding 3D points and image points. If instead \mathbf{C} and \mathbf{y} are known, Equation (8.61) can be seen as a constraint on \mathbf{x} . This what we do in Section 8.5.2, where a projection line is formulated from \mathbf{y} and \mathbf{C} , and \mathbf{x} must lie somewhere along this line. Furthermore, if we can observe \mathbf{x} in two distinct camera views, generated by cameras \mathbf{C}_1 and \mathbf{C}_2 that produce image points

$$\mathbf{y}_1 \sim \mathbf{C}_1 \mathbf{x}, \quad \text{and} \quad \mathbf{y}_2 \sim \mathbf{C}_2 \mathbf{x}, \quad (8.62)$$

then \mathbf{x} can be completely determined by *triangulation*. Triangulation implies that we use the $2 \times 2 = 4$ constraints that emerge from Equation (8.62) to solve for 3 unknowns, represented by the Cartesian coordinates of \mathbf{x} . To do so we need, in fact, only 3 constraints, and from the fourth constraint we can instead formulate a condition on \mathbf{y}_1 and \mathbf{y}_2 that necessary must be satisfied if they are the projections of the same 3D point \mathbf{x} . This is the so-called *epipolar constraint*, which will be the main topic of Chapter 10.

Constraints from projecting a 3D line

Let \mathbf{L} be a 3D line. Its projection in the camera image is given as the 2D line \mathbf{l} . In accordance with Section 8.5.3, this line generates a projection plane \mathbf{p} , given as

$$\mathbf{p} \sim \mathbf{C}^{\top} \mathbf{l}. \quad (8.63)$$

This plane must intersect the line \mathbf{L} which, according to Section 5.4.1, is equivalent to

$$\mathbf{0} = \mathbf{L} \mathbf{p} = \mathbf{L} \mathbf{C}^{\top} \mathbf{l}. \quad (8.64)$$

This amounts to 4 equations in the elements of \mathbf{C} , \mathbf{L} , and \mathbf{l} . From Section 5.3.2 we know that \mathbf{L} has rank 2, however, which means that two of the four equations are linearly dependent on the other two. In general, we can simply remove any two of the four equations in Equation (8.61) and use the remaining two as constraints. In summary: Equation (8.64) corresponds to two linearly independent constraints in the elements of \mathbf{C} , \mathbf{L} , and \mathbf{l} . These two constraints are algebraically independent.

⁹See Section 13.1.2

¹⁰See Toolbox Section 3.7.3 about the rank of the cross product operator

8.5.5 Geometric interpretation of \mathbf{C}

The camera matrix \mathbf{C} represents the mapping from 3D space to the image of a camera. The numerical representation of this matrix depends on which coordinate systems are used for the 3D space and for the image space. Once these coordinate systems are specified, and \mathbf{C} is known relative to them, we can analyze \mathbf{C} to derive geometric information about the camera. In the general case, we can do this by applying the resectioning procedure on \mathbf{C} , as described in Section 8.3.4. Some geometric information can, however, be derived directly from the matrix \mathbf{C} . The following results assume that the camera is not at infinity.

For example, since the camera center \mathbf{n} satisfies Equation (8.5), we can directly extract the homogeneous coordinates of \mathbf{n} as a null vector of \mathbf{C} , since $\mathbf{C}\mathbf{n} = \mathbf{0}$. The camera center specifies the position of the camera in 3D space, relative to the world coordinate system in which \mathbf{C} is defined. It is the primary geometric descriptor of the camera since, in principle, all cameras that share a common center are equivalent in accordance with Section 9.3.

Consider a camera matrix \mathbf{C} that is defined in terms of its three rows:

$$\mathbf{C} = \begin{pmatrix} - & \mathbf{c}_1 & - \\ - & \mathbf{c}_2 & - \\ - & \mathbf{c}_3 & - \end{pmatrix}. \quad (8.65)$$

Based on the relation $\mathbf{C}\mathbf{n} = \mathbf{0}$, we can interpret all three rows of \mathbf{C} as the dual homogeneous coordinates of three planes that all intersect at \mathbf{n} . Since \mathbf{C} is of full rank, this intersection point is unique. Furthermore, \mathbf{c}_3 can be identified as the principal plane of the camera, parallel to the image plane. To see this, we can always find three arbitrary but distinct and not co-linear 3D points, $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$, distinct also from the camera center, such that

$$\mathbf{c}_3 \cdot \mathbf{x}_1 = \mathbf{c}_3 \cdot \mathbf{x}_2 = \mathbf{c}_3 \cdot \mathbf{x}_3 = 0. \quad (8.66)$$

The three 3D points project to homogeneous image coordinates:

$$\mathbf{y}_i \sim \mathbf{C}\mathbf{x}_i = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{pmatrix} \mathbf{x}_i = \begin{pmatrix} \mathbf{c}_1 \cdot \mathbf{x}_i \\ \mathbf{c}_2 \cdot \mathbf{x}_i \\ 0 \end{pmatrix}, \quad i = 1, 2, 3. \quad (8.67)$$

This means that all three points project to points at infinity in the image. Since we assume that the transformation from C-normalized image coordinates to pixel coordinates, \mathbf{K} , is affine, the projections of the three points lie at infinity also in the C-normalized image coordinate system. As described in the beginning of Section 8.2, the principal plane consists of exactly those points that are projected to points at infinity, with the exception of the camera center, which means that $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ lie in the principal plane. Since they are arbitrary points in this plane, it follows that we can identify \mathbf{c}_3 as the dual homogeneous coordinates of the principal plane. The principal axis is a normal to this plane, and its orientation can thus be determined by normalizing the first three elements of \mathbf{c}_3 .

As already mentioned, the columns \mathbf{c}_1 and \mathbf{c}_2 represent two planes that intersect \mathbf{c}_3 uniquely at the center \mathbf{n} . Hence, \mathbf{c}_1 and \mathbf{c}_2 define the dual Plücker coordinates of a 3D line $\tilde{\mathbf{L}} \sim \mathbf{c}_1\mathbf{c}_2^\top - \mathbf{c}_2\mathbf{c}_1^\top$, which intersects the plane \mathbf{c}_3 at \mathbf{n} . Based on Section 5.4.1, we can express the camera center as

$$\mathbf{n} \sim \mathbf{L}\mathbf{c}_3 = \overbrace{\left(\mathbf{c}_1\mathbf{c}_2^\top - \mathbf{c}_2\mathbf{c}_1^\top \right)}^{} \mathbf{c}_3. \quad (8.68)$$

Here, we use the duality mapping described in Section 5.3.5. Equation (8.68) provides a method for determining \mathbf{n} directly from the elements of \mathbf{C} , instead of solving the homogeneous equation $\mathbf{C}\mathbf{n} = \mathbf{0}$ using standard techniques.

8.5.6 Field of view

The pinhole camera model in its basic form describes the image plane as infinite, i.e., any point that lies in the scene in front of the camera is projected through the camera center to a point somewhere in the image plane. In fact, this mathematical model of a camera even allows points lying behind the principal plane to be projected onto the image plane. In practice however, the image plane is restricted to a rectangular area, the bounding box, see Section 8.4. The light rays that fall within the bounding box emanate from a volume in the scene in front of the camera, in the form of a square pyramid that has its apex at the camera center and its base at infinity, see Figure 8.11. This infinite volume is referred to as the *field of view* of the camera.

The field of view is an important characterization of a real camera, since it defines “how large part” of the scene that is visible in the camera image. The field of view is often measured in terms of the angle between two opposite sides of the pyramid and since the image is rectangular, there are two such angles, the vertical field of view α_v and the horizontal field of view α_h , given by:

$$\alpha_v = 2 \tan^{-1} \left(\frac{r}{2fs} \right) = 2 \tan^{-1} \left(\frac{r}{2k_{11}} \right), \quad \alpha_h = 2 \tan^{-1} \left(\frac{c}{2fs} \right) = 2 \tan^{-1} \left(\frac{c}{2k_{22}} \right). \quad (8.69)$$

Here, r and c are the number of rows and columns in the image, f is the focal length measured in some length unit, and s is the number of pixels per length unit in the image plane. k_{11} and k_{22} are the internal camera parameters defined in Section 8.3.3, and we assume that the first image coordinate is in the vertical direction and the second in the horizontal direction.

Some camera specifications give both vertical and horizontal field of view, while other only provide the larger of the two (often the horizontal). For cameras with a small field of view, an alternative is to report the average field of view per pixel, which approximately is obtained by dividing α_v or α_h by the number of pixels in the corresponding direction. In practice, the field of view is also affected by the lens distortion, described in Chapter 18. In summary: although the field of view sometimes is a relevant parameter, its value is often not specified with high accuracy.

If the field of view is known, it can be used to complement the feasibility constraint of a real camera described in observation 8.5.

8.12 A camera can only observe 3D points that have positive depth and lie within its field of view.

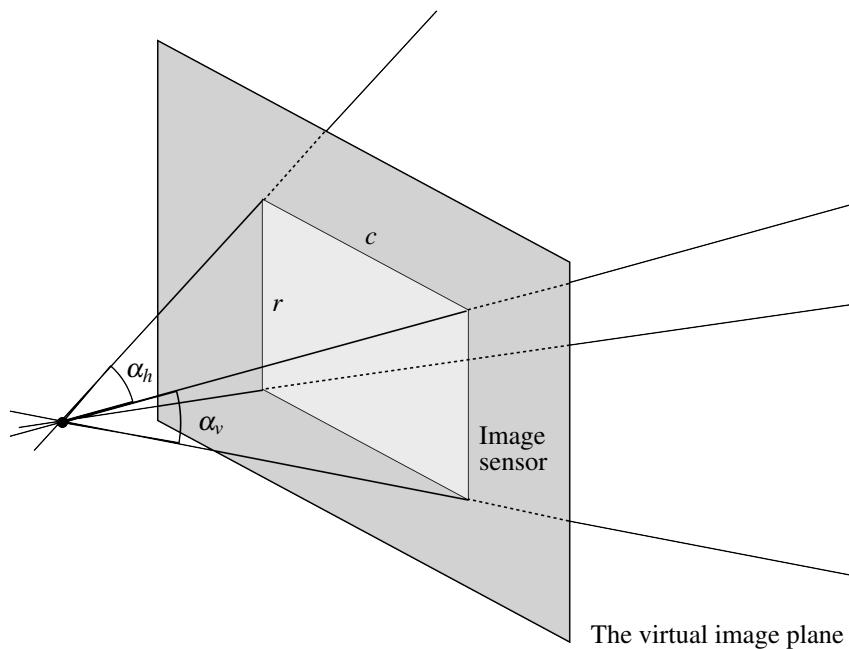


Figure 8.11: The field of view of a pinhole camera.

Chapter 9

Prelude to Two-View Geometry

Before you read this chapter, you should have a thorough understanding of the pinhole camera, presented in Chapter 8, and of homography transformations in the plane, described in Section 7.1. Some of the results from the EVD of a general matrix in Toolbox Section 3.3.6, and on the eigensystem of a rotation matrix in Toolbox Section 8.6.4 will be used. Cholesky decomposition is described in Toolbox Section 8.3.

In the previous chapter we looked at the pinhole camera, which projects the 3D space to an image, and now move on to the next interesting case: what happens when *two* cameras observe the same scene? Or, equivalently, when a single camera takes two images of the same scene with some motion in between. In this chapter, we investigate two special cases of two-view geometry: when the observed 3D points lie in a plane, and when the two images are taken by cameras with coinciding centers.

There are two reasons why these two cases end up here, in a chapter of its own. One is that in both cases, the geometric relation between corresponding points in the two images can be described by a homography transformation. Consequently, if we have a point in one image, the corresponding point in the other image can be uniquely determined by a simple transformation of the image coordinates. The homography depends on the motion of the camera between the two images, in combination with additional parameters. In Section 13.1, we will look at methods that estimate a homography from a set of corresponding points in two images. This leads to the possibility of working with a known homography in the two cases studied here.

The general case of two-view geometry, *epipolar geometry*, is presented in Chapter 10. It is general in the sense that the two cameras have general positions and orientations in 3D space. More specifically, epipolar geometry requires the two camera centers *to not coincide* in order to derive useful results. Another assumption is that the 3D points in the observed scene are at general positions. For example, they *do not* lie in a plane. Both these two cases, which are explicitly excluded from epipolar geometry, occur in practical applications. Therefore, a second reason for this chapter is to derive some results that can be used for these two cases, outside epipolar geometry.

We start the presentation by defining more in detail what we mean by corresponding points, a useful concept in this and subsequent chapters. The case of two cameras that observe points in a plane is then investigated Section 9.2. In principle, the *planar homography* \mathbf{H} , which relates corresponding points in the two images in this case, depends only on the pose of the plane relative to both cameras. If \mathbf{H} is known, we should therefore be able to say something about this pose. Finally, the case of two cameras with coinciding centers is described in Section 9.3. Here, \mathbf{H} depends on the rotation between the cameras and on their internal parameters. If this *rotational homography* \mathbf{H} is known, we should be able to say something about the rotation, or about the internal parameters, or possibly both.

9.1 Correspondences

If we want to relate two images, a principal assumption is that both images are projections of the same scene. This implies that it should be possible to find some geometric objects in 3D space that can be seen in both images. The most common implementation of this idea is to say that there exist a set of 3D points that are visible in both

images. Each such 3D point then has a projection in the first image, denoted \mathbf{y}_1 , and a corresponding projection in the second image, denoted \mathbf{y}_2 .

Definition 9.1: Corresponding points

Image points in two (or more) images that are the projections of the same 3D points are called *corresponding points* or a *point correspondence*. Let \mathbf{x} be a 3D point, and \mathbf{C}_1 and \mathbf{C}_2 be two cameras. If

$$\mathbf{y}_1 \sim \mathbf{C}_1 \mathbf{x}, \quad \text{and} \quad \mathbf{y}_2 \sim \mathbf{C}_2 \mathbf{x}, \quad (9.1)$$

then \mathbf{y}_1 and \mathbf{y}_2 are corresponding points, they form a (point) correspondence.

In the literature, corresponding points are also referred to as *matching points* or simply a *match*. In this context it also makes sense to talk about \mathbf{x} and \mathbf{y}_1 (or \mathbf{x} and \mathbf{y}_2) in Equation (9.1) as corresponding points, although one is a 3D point and the other is a 2D point. We can also extend the concept of correspondence to 3D lines and their projections into two or more images, and refer to them as *corresponding lines*.

In some applications, correspondence can be defined in an alternative way, based on transformations. For example, for points in the plane and a transformation \mathbf{T} , we use:

9.1 Given \mathbf{T} , a transformation in the plane, \mathbf{y}_1 and \mathbf{y}_2 are *corresponding points* (relative \mathbf{T}) if $\mathbf{T}\{\mathbf{y}_1\} \sim \mathbf{y}_2$.

This idea extends in the straight-forward way also to \mathbf{T} as a transformation in 3D space. In fact, we have already seen the application of this concept in Chapters 4 and 6.

9.1.1 Interest points

Before we continue the topic on correspondences, we need to better understand what we mean by *points* in this discussion. In particular when they are used in the context of correspondence, points must meet certain requirements. As a start, not every point that is visible in one image is visible in another image, even if they depict a common scene. For example, a point in the first image may be occluded or outside the boundary in the second image.

But even a pair of corresponding points that are visible in both images may be problematic. For example, a point on a flat untextured area will have a visual appearance that is the same as all other points that lie around it. In this case, and without additional information besides the images, it is therefore not possible to identify with certainty which two points in the two image regions that correspond. Or, formulated in another way, in this case the positions of corresponding points cannot be determined with precision. The same can be said about points that lie on linear structures, e.g., lines or edges. Their positions along the line or edge is affected by uncertainty, or inaccuracy. This issue is illustrated in Figure 9.1.

Consequently, if we want to be able to reidentify a point in two or more images it must not lie in a flat region, or on a linear structure. In general it should be characterized as a distinct point already from its visual appearance, i.e., it should have a well-defined position. Such a point is colloquially sometimes referred to as a *corner point*, since a corner between two edges is one possible manifestation of this type of point. But this class is more general, and a better name is *interest point* or *point of interest*. An interest point can be characterized as “standing out” from other points in a region around the point.

Definition 9.2: Interest point

An interest point (or point of interest) is a distinct image point that can be reidentified in another image of the same scene with high probability.

Figure 9.2 shows an example of interest points detected in two images of a poster wall. Many of these, but not all, can be brought into correspondence.

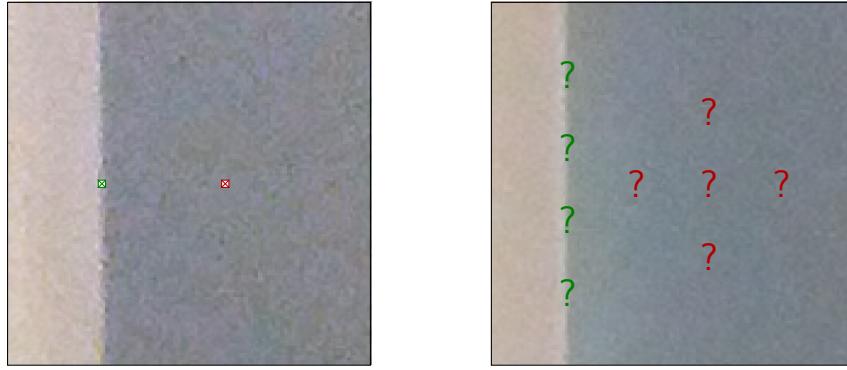


Figure 9.1: Details from two different images of approximately the same region in a 3D scene but taken from distinct camera positions. Two points are indicated by white pixels with a cross in the left image. Both points are difficult to reidentify in the right image; one lies in a region of homogeneous intensity, and the other one lies on an edge.

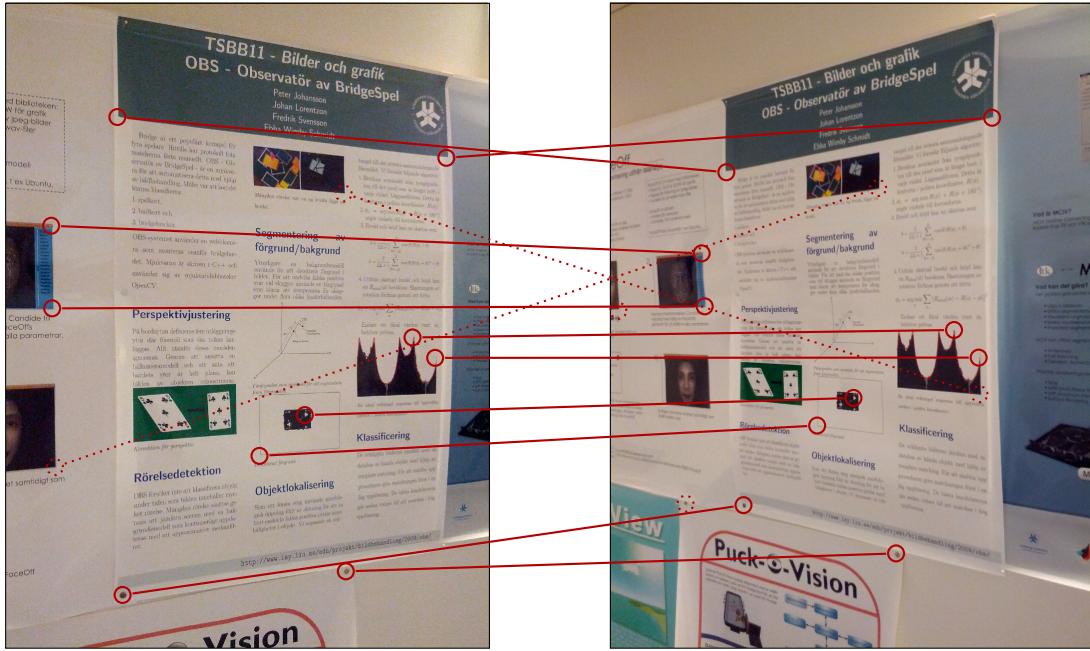


Figure 9.2: Images of a planar surface taken from two different viewpoints. Red circles are examples interest points, some of which can be brought into correspondence. As will be shown in Section 9.2, these points are related by a homography. A few interest points in the right image (single dotted circles) have no corresponding point in the left image. When correspondences are determined automatically, there is a small chance that incorrect correspondences (dotted lines) occur. This can happen when the local regions around each point are very similar.

Interest point detection

The general strategy for extracting interest points from an image is to look at a small region around each pixel in the image, and analyze the intensity variation within that region. If the region meets whatever criteria that is used for defining an interest point, the center pixel is classified as such. The automated version of this process is called *interest point detection*, and typically produces a set of interest points from an image. Depending on the specific method, on the parameters that control the detection process, and on the image itself, this can be a smaller or a larger set. Popular methods for interest point detection include the Harris-Stevens detector [27], SIFT [48], SURF [5] and ORB [57], but this is an active area within computer vision, and new methods are published every year.

The coordinates of detected interest points can often be given as a pair of integers, which refer to the column and row in the pixel grid where the interest point was detected. More sophisticated detection methods can output non-integer coordinates, but then based on various interpolation schemes. Therefore, any method for interest point detection produce coordinates that form *estimates* of the “true” point coordinates. This means that if correspondence between points y_1 and y_2 in two different images has been established, the coordinates will not exactly satisfy Equation (9.1), or observation 9.1.

Inaccuracies in coordinates

The limited accuracy relates partly also to the discussion about the formation of a digital image in Section 8.4. This description indicates that there are some limitations in the accuracy by which a digital image¹ can describe the 3D scene depicted in the image. For example, a detail in the image cannot be resolved if it is smaller than the size of a pixel. And differences in light intensities cannot be detected, if they are smaller than the steps implied by the quantization of the pixel values.

There are also additional sources of inaccuracies. The camera lenses introduce some amount of geometric distortions, to be further discussed in Section 18.2. And even in ideal circumstances, there is only a certain part of the scene that is projected in accordance with the pinhole camera model. The remaining scene points will be more or less blurred or out of focus. There are also various effects related to the light measurement process in the camera, which add noise to the pixel values. As a consequence, two images of a static scene from a fixed camera may not hold identical values in the same pixel. Furthermore, digital images are often compressed, for example using JPEG or other compression methods, and this introduces even more noise and artifacts in the image.

These effects can often be small, perhaps not even visible to the eye. But relative to the pinhole camera model, they introduce deviations in the ideal image, which holds an exact reproduction of the scene. These deviations imply that measurements made in the image, for example of coordinates, contain inaccuracies.

As an example of the inaccuracy discussed is illustrated in Figure 9.3. The two images show a detail in a 3D scene, observed from two different viewpoints by a digital camera. A specific point, an interest point, is indicated in the left image. It cannot be identified in the other image with an accuracy better than a few pixels.

9.2 Any method for interest point detection produce point coordinates that are affected by inaccuracies.

We will have reasons to return to this observation in Part II, where estimation is the main topic. But for now, we just make a note, and stay in the realm of pure geometry, where points or other geometric objects can be represented by parameters without inaccuracies.

9.1.2 The correspondence problem

To find a valid set of correspondences in two or more images, or between 3D points and 2D projected image points, is a task often referred to as solving the *correspondence problem*. As we will see in the rest of this presentation, many methods for solving particular problems, or calculating specific quantities, implicitly assume that a set of correspondences exists in one form or another. This means that solving the correspondence problem is an important task in most applications based on geometry.

¹The same limitations apply also to the “analog” film based image, although for different reasons.



Figure 9.3: A detail from the two images in Figure 9.2, showing corresponding regions in the two images. The white pixel with a cross in the left image indicates an interest point. The position of the corresponding point in the right image cannot be determined with an accuracy better than a few pixels.

Towards solving the correspondence problem

The correspondence problem can often be solved by a human without effort, using visual inspection. In computer vision, we are instead interested in solving the correspondence problem automatically, using computer software and based only on the intensity values in the images. But regardless of how it is solved, the correspondence problem has some practical limitations.

Interest point detection applied to two images of the same scene is often the first step in solving the correspondence problem. When applied to two images of the same scene, a set of points is generated for each image. In the overlapping part of the images, and where occlusion does not occur, there is a high probability that a point detected in one image also appears in the other image. But there is no guarantee. In general, the two point-sets will not even have the same number of detected points. The following step, where the actual correspondences are formed, must take this into account.

In principle, the correspondence problem can be addressed by testing combinations of hypothetical correspondences and choose the optimal one. If a small number of points have been detected in each image, *exhaustive search*, i.e., checking all possible combinations, may be a possibility. But for most applications, the number of combinations is too large to make exhaustive search a practical approach. Instead, various types of heuristics or indeterministic algorithms must be used. These methods are based on the fact that correct correspondences match certain algebraic relations, while, in general, incorrect correspondences do not. In special cases, such as for the two types of homography relations described in the rest of this chapter, the relation can be described as in observation 9.1. In the general case, correspondence of points is instead often based on definition 9.1, which leads to the so-called *epipolar constraint* to be described in Section 10.2.

A reasonable approach to this problem is to base the solution not only on geometry, but also on the visual appearance of the regions around each detected interest point. A point that lies in a region that is mostly bright, with some green texture, is unlikely to be in correspondence with a point in the other image which has a dark region with a red tone around it. Thus, a direct comparison of some type of characterization based on the intensity values, e.g., using color, is an effective way to remove incorrect correspondences.

Before we continue

We have seen that interest points in practice are affected by inaccuracies. But, also correspondences can be inaccurate. In most images, each interest point has more than one candidate in the other image, even when visual appearance is used. This is illustrated by dotted correspondences in Figure 9.2. It happens when a region in one image has good match to several regions in the other image, e.g., when they contain a repetitive pattern. If we

take also geometric constraints into account, this problem reduces, but not completely. In these cases, automatic methods for solving the correspondence problem may produce some incorrect correspondences or give ambiguous results.

Practical methods for solving the correspondence problem will have to wait until Section 17.4. For now, it is a “chicken and egg problem”. Given a sufficiently large set of known correspondences, e.g., between points before and after a homography transformation, the homography can be estimated, e.g., as described in Section 13.1. If, instead, the homography is known, we can determine correspondences by finding pairs of points that reasonably well match the condition in observation 9.1. In many applications, neither a set of known correspondences nor a known homography are initially available.

As we will see, however, there are effective methods for solving the correspondence problem. They are not 100% reliable, but for many applications they are sufficiently correct to be practically useful. This means, for example, that from a set of correspondences in two images, it is possible to estimate a homography transformation that relates them, given that such a relation exists. In the following two sections, we will discuss two cases of two-view geometry where such a scenario is possible. Consequently, we can often assume that the homography that appears is known.

9.2 Planar homography

As a special case of two-view geometry, we will here discuss what happens when two *cameras observe a plane*, or more specifically, they observe 3D points that lie on a plane. This discussion applies also to the equivalent case where two images are taken by a single camera that moves in 3D space while observing points on a plane. In fact, this case generalizes to multiple cameras, or multiple images from a single camera in motion, as long as the 3D points are restricted to a single plane. As we will see, corresponding points in the images are in this case related by a homography transformation, a *planar homography*.

9.2.1 Simple derivation of a planar homography

Let \mathbf{p} be a plane in the scene observed by a camera \mathbf{C}_1 . We assume that \mathbf{p} is not \mathbf{p}_∞ , the plane at infinity². In accordance with Section 5.2.1, let \mathbf{P} be a 4×3 matrix that defines a coordinate system for the plane such that $\mathbf{x} \sim \mathbf{P}\mathbf{y}$. Here, \mathbf{x} is the homogeneous coordinates of a 3D point in the plane, and \mathbf{y} is its Cartesian coordinates relative to the coordinate system in the plane. When \mathbf{x} is projected into camera \mathbf{C}_1 , it ends up at image point $\mathbf{y}_1 \sim \mathbf{C}_1\mathbf{x} \sim \mathbf{C}_1\mathbf{P}\mathbf{y}$. From this follows that \mathbf{y}_1 and \mathbf{y} are related by a homography \mathbf{H}_1 :

$$\mathbf{y}_1 \sim \mathbf{H}_1\mathbf{y}, \quad \text{where } \mathbf{H}_1 = \mathbf{C}_1\mathbf{P}. \quad (9.2)$$

This is nothing new, already in Section 7.1 it was established that the mapping between points on two planes is a homography, when they are related by a projection through a fixed point. In this case the two planes are \mathbf{p} and the image plane of camera \mathbf{C}_1 , and the projection point is the camera center of \mathbf{C}_1 .

Let \mathbf{C}_2 be a second camera that also observes \mathbf{p} . With the same arguments as above, the mapping from coordinates in the plane, \mathbf{y} , to image coordinates, \mathbf{y}_2 , is given as

$$\mathbf{y}_2 \sim \mathbf{H}_2\mathbf{y}, \quad \text{where } \mathbf{H}_2 = \mathbf{C}_2\mathbf{P}. \quad (9.3)$$

Since homographies are invertible, Equation (9.2) leads to $\mathbf{y} \sim \mathbf{H}_1^{-1}\mathbf{y}_1$, and together with Equation (9.3) we get

$$\mathbf{y}_2 \sim \mathbf{H}\mathbf{y}_1, \quad \text{where } \mathbf{H} = \mathbf{H}_2\mathbf{H}_1^{-1}. \quad (9.4)$$

This means that the image produced in the second camera is just a geometric transformation of the first image, in the form of the homography \mathbf{H} .

Definition 9.3: Planar homography

Two images of a planar surface in 3D space are related by a homography transformation. Such a homography is called a *planar homography*.

Note that the concept of a planar homography only gives a context to how the transformation arises from a geo-

²The case when $\mathbf{p} \sim \mathbf{p}_\infty$ is discussed in Section 9.3.4

metric point of view. It does not distinguish between homographies in any other sense.

In some applications, Equation (9.4) is useful only when both \mathbf{y}_1 and \mathbf{y}_2 lie inside the bounding box of each image. But, given that \mathbf{H} is known, we can also transform one of the two images to the coordinate system of the other, and produce a new image as the combination of the two. A more detailed discussion on this application is found in Chapter 19. See Figures 9.4 and 9.5 for an illustration.

9.2.2 The calibrated case

The result we derived in Equation (9.4) does not give much information about how a planar homography depends on the parameters of the plane, or the configuration of the two cameras. This can be done for general cameras, but here we derive an expression for the calibrated case. With the 3D coordinate system specified by the first camera, and all image coordinates C-normalized, the two cameras and the plane parameters are:

$$\mathbf{C}_1 \sim (\mathbf{I} | \mathbf{0}), \quad \mathbf{C}_2 \sim (\mathbf{R} | \bar{\mathbf{t}}), \quad \mathbf{p} = \begin{pmatrix} \hat{\mathbf{p}} \\ -\Delta \end{pmatrix}. \quad (9.5)$$

Let \mathbf{y}_1 be a point in the first image. The center of the first camera, \mathbf{n}_1 , and a 3D point \mathbf{x}_1 that is projected to \mathbf{y}_1 by the first camera, are then given as

$$\mathbf{n}_1 \sim \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}, \quad \mathbf{x}_1 \sim \begin{pmatrix} \mathbf{y}_1 \\ 1 \end{pmatrix}. \quad (9.6)$$

Both points lie on the projection line generated by \mathbf{y}_1 . Using Equation (5.37), the intersection point \mathbf{x}_0 of this line with the plane \mathbf{p} is given as the Plücker coordinates of the line multiplied with the vector \mathbf{p} :

$$\mathbf{x}_0 \sim (\mathbf{n}_1 \mathbf{x}_1^\top - \mathbf{x}_1 \mathbf{n}_1^\top) \mathbf{p} \sim \mathbf{n}_1 (\mathbf{x}_1 \cdot \mathbf{p}) - \mathbf{x}_1 (\mathbf{n}_1 \cdot \mathbf{p}) \sim -(\Delta - \mathbf{y}_1 \cdot \hat{\mathbf{p}}) \mathbf{n}_1 + \Delta \mathbf{x}_1 \sim \begin{pmatrix} \Delta \mathbf{y}_1 \\ \mathbf{y}_1 \cdot \hat{\mathbf{p}} \end{pmatrix}. \quad (9.7)$$

We find the corresponding point \mathbf{y}_2 in the second image by projecting the intersection point \mathbf{x}_0 by camera \mathbf{C}_2 :

$$\mathbf{y}_2 \sim \mathbf{C}_2 \mathbf{x}_0 \sim \Delta \mathbf{R} \mathbf{y}_2 + \bar{\mathbf{t}} \mathbf{y}_1 \cdot \hat{\mathbf{p}} \sim (\Delta \mathbf{R} + \bar{\mathbf{t}} \hat{\mathbf{p}}^\top) \mathbf{y}_1. \quad (9.8)$$

This result leads to the following result:

9.3 In the calibrated case, with C-normalized image coordinates, and when the two cameras and the plane parameters are given as in Equation (9.5), the corresponding planar homography \mathbf{H} is given as

$$\mathbf{H} \sim \Delta \mathbf{R} + \bar{\mathbf{t}} \hat{\mathbf{p}}^\top. \quad (9.9)$$

A few observations can be made based on this result. First, we recall that Δ is the distance of the plane to the origin, in this case the center of the first camera. As homographies must have full rank, observation 9.3 implies that Δ cannot be zero. In practical applications, this limitation is normally not an issue.

The vector $\bar{\mathbf{t}}$ represents the displacement between the two cameras. A second observation is that if $\bar{\mathbf{t}}$ is scaled by some factor, and Δ by the same factor, then the matrix \mathbf{H} is also scaled by this factor. But since \mathbf{H} is a projective element, this means that if we change the distance to the plane by some factor, while keeping its orientation fixed, and change the distance between the two cameras by the same factor, we get the same planar homography. This can also be expressed in the following way.

9.4 A planar homography is invariant to a common scale change in the distance between both the two camera centers and in their distances to the plane, while keeping the orientations of the cameras and the plane fixed.

9.2.3 Solving for the plane and relative camera pose

Given that a planar homography \mathbf{H} is known, observation 9.3 suggests that it may be possible to determine the relative camera pose $(\mathbf{R}, \bar{\mathbf{t}})$ and the parameters of the plane, $\hat{\mathbf{p}}$ and Δ , from \mathbf{H} . We have just seen that the absolute



Figure 9.4: The facade of a building taken from two different viewpoints. The facade is approximately planar, which means that the two images are approximately related by a homography transformation in the overlapping region.



Figure 9.5: A combination of the two images in Figure 9.4, where the right image is transformed to the same coordinate system as the left image by a suitable homography. Notice the discrepancies caused by the roof of the building, and the tiles on the ground, which do not lie in the facade plane.

scale of the 3D configuration that includes the cameras and the plane cannot be determined. But, except for a few degenerate cases, it is actually possible to determine the other parameters, although with multiple solutions. In this section we will look at how to do this, based on an article by Longuet-Higgins [46].

Since $\Delta > 0$, we start by rewriting Equation (9.9) as

$$\mathbf{H} \sim \mathbf{R} + \gamma \hat{\mathbf{t}} \hat{\mathbf{p}}^\top, \quad \text{where } \hat{\mathbf{t}} = \frac{\bar{\mathbf{t}}}{\|\bar{\mathbf{t}}\|}, \quad \text{and } \gamma = \frac{\|\bar{\mathbf{t}}\|}{\Delta} > 0. \quad (9.10)$$

We can always find a unit vector $\hat{\mathbf{u}}$ that is orthogonal to both $\hat{\mathbf{p}}$ and $\mathbf{R}^\top \hat{\mathbf{t}}$. When the right-hand side of Equation (9.10) is applied to $\hat{\mathbf{u}}$, we get

$$(\mathbf{R} + \gamma \hat{\mathbf{t}} \hat{\mathbf{p}}^\top) \hat{\mathbf{u}} = \mathbf{R} \hat{\mathbf{u}} = \hat{\mathbf{v}}. \quad (9.11)$$

Also $\hat{\mathbf{v}}$ is a unit vector, and multiplied from left into the same right-hand side, we get

$$\hat{\mathbf{v}}^\top (\mathbf{R} + \gamma \hat{\mathbf{t}} \hat{\mathbf{p}}^\top) = \hat{\mathbf{u}}^\top \mathbf{R}^\top \mathbf{R} + \gamma \hat{\mathbf{u}}^\top \mathbf{R}^\top \hat{\mathbf{t}} \hat{\mathbf{p}}^\top = \hat{\mathbf{u}}^\top. \quad (9.12)$$

In summary, the last two equations imply that $\hat{\mathbf{u}}$ is a right singular vector, and $\hat{\mathbf{v}}$ is a left singular vector³ of $\mathbf{R} + \gamma \hat{\mathbf{t}} \hat{\mathbf{p}}^\top$, with a corresponding singular value = 1.

This result suggests that we should apply the SVD on the matrix $\mathbf{H} = \mathbf{U} \mathbf{S} \mathbf{V}^\top$, where \mathbf{S} is a diagonal matrix that holds the singular values of \mathbf{H} , $\sigma_1, \sigma_2, \sigma_3 > 0$. To simplify for the rest of this derivation, we assume already here that \mathbf{H} is normalized⁴ such that $\mathbf{U}, \mathbf{V} \in SO(3)$ and $\sigma_2 = 1$. This means that we can express Equation (9.10) as an equality, rather than an equivalence relation between projective elements, and we get

$$\mathbf{S} = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \sigma_3 \end{pmatrix} = \mathbf{U}^\top \mathbf{H} \mathbf{V} = \mathbf{U}^\top \mathbf{R} \mathbf{V} + \gamma (\mathbf{U}^\top \hat{\mathbf{t}}) (\mathbf{V}^\top \hat{\mathbf{p}})^\top = \tilde{\mathbf{R}} + \gamma \tilde{\mathbf{t}} \tilde{\mathbf{p}}^\top, \quad (9.13)$$

where $\tilde{\mathbf{R}} = \mathbf{U}^\top \mathbf{R} \mathbf{V} \in SO(3)$, and both $\tilde{\mathbf{t}} = \mathbf{U}^\top \hat{\mathbf{t}}$ and $\tilde{\mathbf{p}} = \mathbf{V}^\top \hat{\mathbf{p}}$ are unit vectors in \mathbb{R}^3 . From this we form an expression for $\mathbf{S}^2 = \mathbf{S}^\top \mathbf{S}$ as

$$\mathbf{S}^2 = (\tilde{\mathbf{R}} + \gamma \tilde{\mathbf{t}} \tilde{\mathbf{p}}^\top)^\top (\tilde{\mathbf{R}} + \gamma \tilde{\mathbf{t}} \tilde{\mathbf{p}}^\top) = \tilde{\mathbf{R}}^\top \tilde{\mathbf{R}} + \gamma \tilde{\mathbf{R}}^\top \tilde{\mathbf{t}} \tilde{\mathbf{p}}^\top + \gamma \tilde{\mathbf{p}} (\tilde{\mathbf{R}}^\top \tilde{\mathbf{t}})^\top + \gamma^2 \tilde{\mathbf{p}} \tilde{\mathbf{p}}^\top. \quad (9.14)$$

With the substitutions $\tilde{\mathbf{r}} = \tilde{\mathbf{R}}^\top \tilde{\mathbf{t}} = (r_1, r_2, r_3)$ and $\tilde{\mathbf{q}} = \gamma \tilde{\mathbf{p}} + \tilde{\mathbf{r}} = (q_1, q_2, q_3)$, this expression simplifies to

$$\mathbf{S}^2 = \mathbf{I} + \tilde{\mathbf{q}} \tilde{\mathbf{q}}^\top - \tilde{\mathbf{r}} \tilde{\mathbf{r}}^\top. \quad (9.15)$$

In this matrix equation, we focus first on element (2,2), which reads

$$1 = 1 + q_2^2 - r_2^2, \quad \Rightarrow \quad r_2^2 = q_2^2. \quad (9.16)$$

At this point, it is a straight-forward exercise to show that the combination of Equations (9.16) and (9.15) gives us two choices. Either $r_2 = q_2 = 0$ or $\mathbf{S} = \mathbf{I}$. The latter case is not interesting since then $\gamma = 0$ in Equation (9.10), which is not allowed here.

Thus, $r_2 = q_2 = 0$, which means that we only need to deal with the “corner elements” in Equation (9.15):

$$\sigma_1^2 = 1 + q_1^2 - r_1^2, \quad 0 = q_1 q_3 - r_1 r_3, \quad \sigma_3^2 = 1 + q_3^2 - r_3^2, \quad 1 = r_1^2 + r_3^2. \quad (9.17)$$

The last equation comes from the fact that $\tilde{\mathbf{t}}$, and therefore also $\tilde{\mathbf{r}}$, are unit vectors. Again, it is a straight-forward exercise to show that these equations have solutions of the following form:

$$\tilde{\mathbf{r}} = s_1 s_3 \begin{pmatrix} \sigma_3 a \\ 0 \\ s_2 \sigma_1 b \end{pmatrix}, \quad \tilde{\mathbf{q}} = s_1 \begin{pmatrix} \sigma_1 a \\ 0 \\ s_2 \sigma_3 b \end{pmatrix}, \quad \text{where } a = \sqrt{\frac{\sigma_1^2 - 1}{\sigma_1^2 - \sigma_3^2}}, \quad b = \sqrt{\frac{1 - \sigma_3^2}{\sigma_1^2 - \sigma_3^2}}. \quad (9.18)$$

Here, $s_1, s_2, s_3 = \pm 1$ are three independent sign variables. This means that there are eight possible solutions for the parameters that we are solving for. Note that a and b are well-defined and real, and $\sigma_1 \geq \sigma_2 \geq \sigma_3$, as stipulated

³Singular vectors and singular values are described in Toolbox Section 8.2.

⁴If $\det \mathbf{H} < 0$, change the sign of \mathbf{H} first. After that, we can always choose $\mathbf{U}, \mathbf{V} \in SO(3)$.

in the SVD, only if $\sigma_1 > \sigma_3$, together with $\sigma_1 \geq 1$ and $\sigma_3 \leq 1$. This works well when \mathbf{H} is normalized such that $\sigma_2 = 1$, if we also can avoid the degenerate case $\mathbf{S} = \mathbf{I}$.

We can now start to back-track the substitutions that have been made, to get

$$\begin{aligned}\tilde{\mathbf{R}} &= \mathbf{S}^{-1}(\mathbf{I} + (\tilde{\mathbf{q}} - \tilde{\mathbf{r}})\tilde{\mathbf{r}}^\top) = \begin{pmatrix} p & 0 & s_2 s_3 q \\ 0 & 1 & 0 \\ -s_2 q & 0 & s_3 p \end{pmatrix}, \quad \text{where } p = \frac{1 + s_3 \sigma_1 \sigma_3}{\sigma_1 + s_3 \sigma_3}, \quad q = \frac{\sqrt{(\sigma_1^2 - 1)(1 - \sigma_3^2)}}{\sigma_1 + s_3 \sigma_3}, \\ \gamma &= \|\tilde{\mathbf{q}} - \tilde{\mathbf{r}}\| = \sigma_1 - s_3 \sigma_3, \\ \tilde{\mathbf{t}} &= \tilde{\mathbf{R}} \tilde{\mathbf{r}} = \mathbf{S}^{-1} \tilde{\mathbf{q}} = s_1 \begin{pmatrix} a \\ 0 \\ s_2 b \end{pmatrix}, \quad \tilde{\mathbf{p}} = \frac{\tilde{\mathbf{q}} - \tilde{\mathbf{r}}}{\gamma} = s_1 \begin{pmatrix} a \\ 0 \\ -s_2 s_3 b \end{pmatrix}.\end{aligned}\tag{9.19}$$

From these expressions, we finally get $\mathbf{R} = \mathbf{U} \tilde{\mathbf{R}} \mathbf{V}^\top$, $\hat{\mathbf{t}} = \mathbf{U} \tilde{\mathbf{t}}$, and $\hat{\mathbf{p}} = \mathbf{V} \tilde{\mathbf{p}}$.

Feasibility of solutions

We have shown that, in principle, there are eight solutions to the problem of finding the parameters \mathbf{R} , $\hat{\mathbf{t}}$, $\hat{\mathbf{p}}$, and γ . Not all of these are feasible in practice. For example, it is a straight-forward exercise to show that $s_3 = -1$ always leads to a solution where the two camera centers are on opposite sides of the resulting plane \mathbf{p} , while they are on the same side when $s_3 = 1$. This normally reduces the number of feasible solutions to four, and means that we only need to compute solutions for $s_3 = 1$.

In addition to this, we can also apply observation 8.5 on page 101. In many practical applications, corresponding points have been detected in the two images, and are used to estimate the planar homography \mathbf{H} . For each solution, which remains after the initial reduction mentioned above, we can project the points in each image out in 3D space to find where the projections intersect with the corresponding plane \mathbf{p} . For a point \mathbf{y}_1 in the first image, this is point \mathbf{x}_0 in Equation (9.7), and it is a straight-forward exercise to show that the corresponding expression⁵ for a point \mathbf{y}_2 in the second image is given as

$$\mathbf{x}'_0 \sim \left(\frac{\gamma \mathbf{y}_2}{(1 + \gamma \hat{\mathbf{t}}^\top \mathbf{R} \hat{\mathbf{p}})} (\mathbf{y}_2^\top \mathbf{R} \hat{\mathbf{p}}) \right). \tag{9.20}$$

For each camera, these intersecting points must be in front of the camera, i.e., with positive depth. This feasibility check often removes two or three solutions and leaves only one or two feasible solutions.

This approach to determine \mathbf{R} , $\hat{\mathbf{t}}$, $\hat{\mathbf{p}}$, and γ is summarized in Algorithm 9.1. Notice that the number of feasible solutions that are generated vary, it depends both on \mathbf{H} and exactly which interest points in the two images that are used for checking feasibility. There is not even a guarantee that a single feasible solution exists, given that correspondences are *putative* and may sometimes be incorrect, an idea further explored in Section 17.4.

9.2.4 Applications

Planar homographies appear in many practical applications. For example, in Chapter 19 image mosaics are constructed from two or more images of a planar surface, such as the facade of a building. Other examples of this application are mosaics constructed from aerial images of the ground, which can be approximated as a plane if images are taken from sufficiently high altitude. In the case of mosaics, the plane and the relative pose between two cameras is often not interesting to recover, it suffices that the homography exists and can be determined.

Another example of an application where planar homographies can be used is to consider an autonomous vehicle that is moving along a road. To operate autonomously, it uses a forward-looking camera for extracting and processing information about happens in front of the vehicle. While moving along the road, the camera produce a sequence of images, and we can consider two images that are taken close in time. The road, and in general the ground around the vehicle, can often be approximated as a plane. By restricting the processing to the road, the two images are then related by a planar homography \mathbf{H} .

If \mathbf{H} is known, automatic processing can determine where specific interest points, detected in one image, are expected to appear in the other image. By extracting the parameters of the plane, and of the relative camera pose, it

⁵Here, \mathbf{x}'_0 is expressed in the coordinate system of the second camera!

Algorithm 9.1: Determine \mathbf{R} , $\hat{\mathbf{t}}$, $\hat{\mathbf{p}}$, and γ from a planar homography \mathbf{H} .

```

Input: A homography  $\mathbf{H}$ , assumed to be a planar homography:  $\mathbf{H} \sim \mathbf{R} + \gamma \hat{\mathbf{t}} \hat{\mathbf{p}}^\top$ 
Output: A set of  $\mathbf{R}, \hat{\mathbf{t}}, \hat{\mathbf{p}}, \gamma$  that are consistent with this assumption
1 Apply SVD on  $\mathbf{H} = \mathbf{U} \mathbf{S} \mathbf{V}^\top$ 
2 Normalize:  $\sigma_1 = [\mathbf{S}]_{11}/[\mathbf{S}]_{22}$ ,  $\sigma_3 = [\mathbf{S}]_{33}/[\mathbf{S}]_{22}$ , if  $\det \mathbf{U} = -1$ , change sign of  $\mathbf{U}$ , and same with  $\mathbf{V}$ 
3 if  $\sigma_1 = \sigma_3$  then return no solution found
4 Set feasible solutions =  $\emptyset$ 
5 Set  $a = \sqrt{\frac{\sigma_1^2 - 1}{\sigma_1^2 - \sigma_3^2}}$  and  $b = \sqrt{\frac{1 - \sigma_3^2}{\sigma_1^2 - \sigma_3^2}}$ 
6 Set  $s_3 = 1$ , alternatively include  $s_3 \in \{-1, 1\}$  in the loop below if  $s_3 = -1$  is a feasible case
7 foreach  $s_1, s_2 \in \{-1, 1\}$  do
8     Set  $p = \frac{1 + s_3 \sigma_1 \sigma_3}{\sigma_1 + s_3 \sigma_3}$  and  $q = \frac{\sqrt{(\sigma_1^2 - 1)(1 - \sigma_3^2)}}{\sigma_1 + s_3 \sigma_3}$ 
9     Set  $\mathbf{R} = \mathbf{U} \begin{pmatrix} p & 0 & s_2 s_3 q \\ 0 & 1 & 0 \\ -s_2 q & 0 & s_3 p \end{pmatrix} \mathbf{V}^\top$ ,  $\hat{\mathbf{t}} = s_1 \mathbf{U} \begin{pmatrix} a \\ 0 \\ s_2 b \end{pmatrix}$ ,  $\hat{\mathbf{p}} = s_1 \mathbf{V} \begin{pmatrix} a \\ 0 \\ -s_2 s_3 b \end{pmatrix}$  and  $\gamma = \sigma_1 - s_3 \sigma_3$ 
10    if interest points exist, in C-normalized coordinates then
11        foreach  $\mathbf{y}_1$  in the first image do
12            Determine corresponding point  $\mathbf{x}_0$  on the plane, Equation (9.7)
13            if  $\mathbf{x}_0$  has negative third coordinate then discard this solution
14        end
15        foreach  $\mathbf{y}_2$  in the second image do
16            Determine corresponding point  $\mathbf{x}'_0$  on the plane, Equation (9.20)
17            if  $\mathbf{x}'_0$  has negative third coordinate then discard this solution
18        end
19    end
20    Add  $\mathbf{R}, \hat{\mathbf{t}}, \hat{\mathbf{p}}, \gamma$  to feasible solutions
21 end
22 Return feasible solutions
```

is possible to determine where image points are located on the ground, as well as the motion of the vehicle between the images, so-called *ego-motion estimation*. It is often possible to determine the ego-motion between two images by other means, e.g., based on the speed and how the vehicle is turning. In these cases, \mathbf{H} can be determined from that information, image correspondences are not necessary. Alternatively, if \mathbf{H} is determined from image correspondences can complement the ego-motion estimation, by providing higher accuracy in the estimate.

9.3 Rotational homography

The second interesting case of two-view geometry that we investigate here is the case of a *rotating camera*, which in this context has the specific meaning that it takes images while it rotates about its center. In practice, we can also use this case when two or more cameras, pointing in different directions, are observing general points in 3D space, if the distance between camera centers is negligible compared to the depth of the 3D points. Also in the case of rotating cameras, corresponding points are related by a homography transformation, a *rotational homography*.

9.3.1 Derivation of a rotational homography

Consider two cameras with distinct image planes, but which have a common camera center, \mathbf{n} . A 3D point \mathbf{x} generates a projection line, which includes \mathbf{n} , and it intersects with both of the image planes, at \mathbf{y}_1 and \mathbf{y}_2 , respectively. See Figure 9.6 for an illustration. This type of mapping is exactly the one that appears in Section 7.1, defining a homography transformation. Hence, with \mathbf{y}_1 and \mathbf{y}_2 as the homogeneous coordinates relative to a coordinate

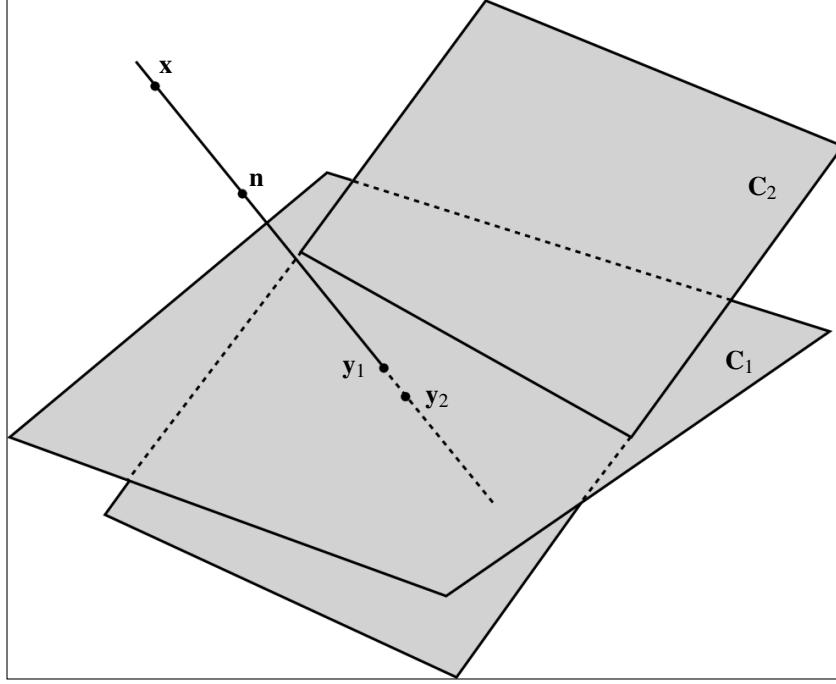


Figure 9.6: The image planes corresponding to two cameras, \mathbf{C}_1 and \mathbf{C}_2 , which have a common center \mathbf{n} .

system in each of the two image planes:

$$\mathbf{y}_2 = \mathbf{H}\mathbf{y}_1, \quad (9.21)$$

where \mathbf{H} is a homography that only depends on the two image planes, on the camera center \mathbf{n} , and on the two coordinate systems, in accordance with Equation (7.6).

If the two cameras are known, it is possible to express \mathbf{H} in the two camera matrices, \mathbf{C}_1 and \mathbf{C}_2 . Let the common center be \mathbf{n} , and consider an image point \mathbf{y}_1 in the first camera image. We can then form a 3D point $\mathbf{x}_1 \sim \mathbf{C}_1^+ \mathbf{y}_1$, which is distinct from the camera center \mathbf{n} . Therefore, it can be projected into the second camera to get the corresponding image point:

$$\mathbf{y}_2 = \mathbf{C}_2 \mathbf{x}_1 = \mathbf{C}_2 \mathbf{C}_1^+ \mathbf{y}_1. \quad (9.22)$$

9.5 Corresponding points in two images taken by a rotating camera are related by a homography $\mathbf{H} = \mathbf{C}_2 \mathbf{C}_1^+$, where $\mathbf{C}_1, \mathbf{C}_2$ are the two camera matrices, sharing a common center.

This homography can even be made a bit more explicit if we assume that both cameras are defined relative to the 3D camera centered coordinate system of the first camera, \mathbf{C}_1 . This means:

$$\mathbf{C}_1 = \mathbf{K}_1(\mathbf{I} | \mathbf{0}), \quad \mathbf{C}_2 = \mathbf{K}_2(\mathbf{R} | \mathbf{0}), \quad (9.23)$$

where \mathbf{K}_1 and \mathbf{K}_2 are the internal parameters of each camera, and \mathbf{R} and the rotation between the two cameras. Using the expression for a right pseudo-inverse⁶, we get

$$\mathbf{C}_1^+ = \mathbf{C}_1^\top (\mathbf{C}_1 \mathbf{C}_1^\top)^{-1} = \begin{pmatrix} \mathbf{K}_1^{-1} \\ \mathbf{0} \end{pmatrix}. \quad (9.24)$$

Finally, from observation 9.5, we get the homography as

$$\mathbf{H} = \mathbf{K}_2(\mathbf{R} | \mathbf{0}) \begin{pmatrix} \mathbf{K}_1^{-1} \\ \mathbf{0} \end{pmatrix} = \mathbf{K}_2 \mathbf{R} \mathbf{K}_1^{-1}. \quad (9.25)$$

⁶See Toolbox Section 3.4.2

This means that in the case of rotating cameras, it suffices that we know the relative rotation \mathbf{R} between the cameras, and their internal parameters, to determine the homography \mathbf{H} between the two images.

The type of homography discussed here, and in observation 9.5, may be referred to as a *rotational homography*. But we will normally use this concept in a more restricted context. It occurs when a single camera with fixed internal parameters \mathbf{K} rotates between taking the two images, a relatively common case in practical applications.

Definition 9.4: Rotational homography

The homography that relates the images from a camera that rotates about its center by \mathbf{R} , with fixed internal parameters \mathbf{K} , is given as

$$\mathbf{H} = \mathbf{K} \mathbf{R} \mathbf{K}^{-1}. \quad (9.26)$$

This \mathbf{H} is called a *rotational homography*.

Similar to the planar homography presented in Section 9.2, the concept of a rotational homography is primary used to give geometric context to the transformation. However, as we will see in the next section, a rotational homography in accordance to definition 9.4, needs to observe a particular constraint, which makes rotational homographies somewhat special.

Transformation matrices applied to homogeneous coordinates, including the homography \mathbf{H} , are projective elements in $P(\mathbb{R}^{3 \times 3})$. Yet, we choose to use an equality sign in Equation (9.26). This is motivated by the fact that $\det(\mathbf{K} \mathbf{R} \mathbf{K}^{-1}) = 1$, which leads to the following observation:

9.6 In algebraic relations where a rotational homography \mathbf{H} appears, it makes sense to assume that \mathbf{H} is normalized such that $\det \mathbf{H} = 1$.

If \mathbf{H}' is a rotational homography where $\det \mathbf{H}' \neq 1$, we can set $\mathbf{H} = (\det \mathbf{H}')^{-\frac{1}{3}} \cdot \mathbf{H}'$. This results in $\det \mathbf{H} = 1$.

9.3.2 A constraint on rotational homographies

In accordance with Section 7.1, a general homography \mathbf{H} that transforms between two planes also has eight degrees of freedom. This is still true if we set $\det \mathbf{H} = 1$. According to Section 6.2, a 3D rotation \mathbf{R} has three degrees of freedom, and observation 8.10 on page 109 states that the internal parameters \mathbf{K} has five degrees of freedom. This suggests that the rotational homography \mathbf{H} in Equation (9.26) has a total of eight degrees of freedom, the same as a general homography.

These observations also suggest that any $\mathbf{H} \in P(\mathbb{R}^{3 \times 3})$ in Equation (9.26) can be achieved for a suitable choice of \mathbf{K} and \mathbf{R} . In this section we will see that this is not correct. As a rotational homography, \mathbf{H} must meet a specific condition to be a rotational homography, besides $\det \mathbf{H} = 1$.

Starting with Equation (9.26), we can expand \mathbf{R} into its eigenvalue decomposition⁷, which leads to

$$\mathbf{H} = \mathbf{K} \mathbf{R} \mathbf{K}^{-1} = \mathbf{K} \underbrace{\mathbf{E} \mathbf{D} \mathbf{E}^*}_{= \mathbf{R}} \mathbf{K}^{-1} = \mathbf{K} \mathbf{E} \mathbf{D} \mathbf{E}^{-1} \mathbf{K}^{-1} = (\mathbf{K} \mathbf{E}) \mathbf{D} (\mathbf{K} \mathbf{E})^{-1}. \quad (9.27)$$

Here \mathbf{D} is a diagonal matrix that holds the eigenvalues of \mathbf{R} , and $\mathbf{E} \in U(3)$ is a 3×3 matrix holding an orthogonal basis of corresponding eigenvectors in its columns. This results means⁸ that \mathbf{H} has the same eigenvalues as \mathbf{R} , i.e., $1, e^{i\alpha}, e^{-i\alpha}$. Furthermore, a set of corresponding eigenvectors of \mathbf{H} is found in the columns of matrix $\mathbf{K} \mathbf{E}$. This result implies that \mathbf{H} cannot be a general homography, not even if we take into account $\det \mathbf{H} = 1$.

9.7 A homography \mathbf{H} must meet an *internal constraint* in order to be a rotational homography, Equation (9.26). It can be formulated as: \mathbf{H} must have eigenvalues $1, e^{i\alpha}, e^{-i\alpha}$, when $\det \mathbf{H} = 1$.

This internal constraint can be reformulated as an algebraic expression that involves the elements of \mathbf{H} . However,

⁷See Toolbox Section 8.6.4.

⁸Here we are using the fact that $\mathbf{E} \in U(3)$, i.e., it is its own inverse.

⁹See Toolbox Section 3.3.6.

this expression is realized as a high order polynomial in the elements of \mathbf{H} , and therefore of little practical use.

In the case that \mathbf{H} is estimated from a set of corresponding points, a consequence of these observations is that the internal constraint should be observed in this estimation, i.e., the estimated \mathbf{H} should satisfy the constraint in observation 9.7. Another consequence is that the internal constraint makes a rotational homography lose one degree of freedom compared to general homographies.

9.8 The rotational homography in Equation (9.26) has seven degrees of freedom, not eight as suggested initially in this section.

9.3.3 Solving for \mathbf{R} and \mathbf{K}

If a rotational homography \mathbf{H} is known, Equation (9.26) suggests the possibility to determine, or at least say something about, \mathbf{K} and \mathbf{R} . In this section, we will investigate this idea more in detail. Notice that in this discussion we assume that the constraints mentioned in observation 9.7 are satisfied.

As a first step, we can solve \mathbf{R} from Equation (9.26):

$$\mathbf{R} = \mathbf{K}^{-1}\mathbf{H}\mathbf{K}. \quad (9.28)$$

This means that if we are able to first determine \mathbf{K} from \mathbf{H} , we can always find \mathbf{R} from Equation (9.28). Since $\mathbf{R} \in SO(3)$, it also means that

$$\mathbf{I} = \mathbf{R}^\top \mathbf{R} = \mathbf{K}^\top \mathbf{H}^\top \mathbf{K}^{-\top} \mathbf{K}^{-1} \mathbf{H} \mathbf{K}, \quad (9.29)$$

and, after multiplying from the left by $\mathbf{K}^{-\top}$, and from the right by \mathbf{K}^{-1} , we get

$$\mathbf{H}^\top \mathbf{K}^{-\top} \mathbf{K}^{-1} \mathbf{H} = \mathbf{K}^{-\top} \mathbf{K}^{-1}. \quad (9.30)$$

Initially, Equation (9.30) looks a bit messy, but if we define the matrix $\boldsymbol{\omega} = \mathbf{K}^{-\top} \mathbf{K}^{-1}$ it becomes less so:

$$\mathbf{H}^\top \boldsymbol{\omega} \mathbf{H} = \boldsymbol{\omega}. \quad (9.31)$$

The symmetric and positive definite 3×3 matrix $\boldsymbol{\omega}$ occurs frequently in some derivations of the subsequent chapters, and it has a name of its own.

Definition 9.5: Image of the absolute conic (IAC)

The matrix $\boldsymbol{\omega} = \mathbf{K}^{-\top} \mathbf{K}^{-1}$, where \mathbf{K} is the internal camera parameters, is called the *image of the absolute conic*, often abbreviated as *IAC*.

Although this name does not make much sense here, there is actually a reasonable motivation for it. But that goes beyond the scope of this presentation, and the interested reader can find more details about this in, e.g., the book by Hartley & Zisserman [30].

Before trying to solve Equation (9.31), we focus first on how to determine \mathbf{K} from $\boldsymbol{\omega}$. Since \mathbf{K} , and therefore also \mathbf{K}^{-1} , are upper triangular, it follows that $\mathbf{K}^{-\top}$ is lower triangular. As it is symmetric and positive definite, $\boldsymbol{\omega}$ has a well-defined Cholesky decomposition¹⁰, as a product of a lower triangular matrix times its transpose. Consequently, $\mathbf{K}^{-\top}$ is the Cholesky factor of $\boldsymbol{\omega}$.

9.9 With the IAC $\boldsymbol{\omega}$ known, we can determine the internal camera parameters \mathbf{K} by first making a Cholesky decomposition of $\boldsymbol{\omega}$, followed by a matrix inverse and transpose.

We return now to the problem of determining the IAC $\boldsymbol{\omega}$ from Equation (9.31) when \mathbf{H} is known. Since both sides of the equation describe linear expressions in the matrix $\boldsymbol{\omega}$, it can be formulated as a homogeneous linear equation in $\boldsymbol{\omega}$. This means that $\boldsymbol{\omega}$ only can be determined up to an unknown scaling. This scaling propagates also to \mathbf{K} , but can now be determined from $[\mathbf{K}]_{33} = 1$ in accordance with Equation (8.35).

¹⁰Cholesky decomposition is described in Toolbox Section 8.3.

Unfortunately, Equation (9.31) leaves more than the scale of ω undetermined. To see this, we expand \mathbf{H} in terms of its EVD that is derived¹¹ in Equation (9.27):

$$\omega = \mathbf{H}^* \omega \mathbf{H} = (\mathbf{K} \mathbf{E} \mathbf{D} \mathbf{E}^* \mathbf{K}^{-1})^* \omega \mathbf{K} \mathbf{E} \mathbf{D} \mathbf{E}^* \mathbf{K}^{-1} = \mathbf{K}^{-\top} \mathbf{E} \mathbf{D}^* \mathbf{E}^* \mathbf{K}^\top \omega \mathbf{K} \mathbf{E} \mathbf{D} \mathbf{E}^* \mathbf{K}^{-1}. \quad (9.32)$$

Multiplication by $\mathbf{E}^* \mathbf{K}^\top$ from left, and by $\mathbf{K} \mathbf{E}$ from right, on both sides of the previous equation gives us

$$\mathbf{E}^* \mathbf{K}^\top \omega \mathbf{K} \mathbf{E} = \mathbf{D}^* \mathbf{E}^* \mathbf{K}^\top \omega \mathbf{K} \mathbf{E} \mathbf{D}. \quad (9.33)$$

This expression simplifies considerably by the substitution $\hat{\omega} = \mathbf{E}^* \mathbf{K}^\top \omega \mathbf{K} \mathbf{E}$:

$$\hat{\omega} = \mathbf{D}^* \hat{\omega} \mathbf{D}, \quad \text{or} \quad \mathbf{D}^* \hat{\omega} \mathbf{D} - \hat{\omega} = \mathbf{0}. \quad (9.34)$$

Notice that $\hat{\omega}$, just like ω , is a real and symmetric 3×3 matrix.

To finalize this discussion, we expand $\hat{\omega}$ in terms of its elements, and use the fact that \mathbf{D} holds the eigenvalues of the rotational homography \mathbf{H} in its diagonal:

$$\hat{\omega} = \begin{pmatrix} \hat{\omega}_{11} & \hat{\omega}_{12} & \hat{\omega}_{13} \\ \hat{\omega}_{12} & \hat{\omega}_{22} & \hat{\omega}_{23} \\ \hat{\omega}_{13} & \hat{\omega}_{23} & \hat{\omega}_{33} \end{pmatrix}, \quad \text{and} \quad \mathbf{D} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & e^{i\alpha} & 0 \\ 0 & 0 & e^{-i\alpha} \end{pmatrix}. \quad (9.35)$$

When $\hat{\omega}$ and \mathbf{D} are inserted into Equation (9.34), we get

$$\begin{pmatrix} 0 & (e^{i\alpha} - 1)\hat{\omega}_{12} & (e^{-i\alpha} - 1)\hat{\omega}_{13} \\ (e^{-i\alpha} - 1)\hat{\omega}_{12} & 0 & (e^{-2i\alpha} - 1)\hat{\omega}_{23} \\ (e^{i\alpha} - 1)\hat{\omega}_{13} & (e^{2i\alpha} - 1)\hat{\omega}_{23} & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \quad (9.36)$$

In general, a necessary and sufficient condition to make this equation valid is to set $\hat{\omega}_{12} = \hat{\omega}_{13} = \hat{\omega}_{23} = 0$. However, the diagonal elements of $\hat{\omega}$ can be chosen arbitrarily. This shows that Equation (9.31) has no unique solution ω when \mathbf{H} is a rotational homography.

9.10 The IAC ω cannot be uniquely determined from Equation (9.31). This means that neither \mathbf{K} nor \mathbf{R} can be uniquely determined from Equation (9.26).

This may look like a sad ending to this story, but it simply means that if we want to obtain ω , or \mathbf{K} , it is necessary to use more than one rotational homography. We will return to this problem in Section 16.4., where it is shown how ω can be *estimated* from two or more homographies.

9.3.4 Before we continue

If we return to the discussion in Section 9.2.2, on the calibrated case of a planar homography, recall that we made the explicit assumption that $\mathbf{p} \not\sim \mathbf{p}_\infty$. The distance to the plane \mathbf{p} is Δ , and if it goes toward infinity, then γ in Equation (9.10) vanishes. Thus, the corresponding planar homography goes towards $\mathbf{H} = \mathbf{R}$ at the same time. This is also the expected expression for the rotational homography in Equation (9.26), when C-normalized image coordinates are used. This summarizes to:

9.11 A rotational homography can be seen as a degenerate case of a planar homography, where the plane is \mathbf{p}_∞ .

This case is degenerated in the sense that, it is not possible to determine \mathbf{R} from \mathbf{H} , neither if we see \mathbf{H} as a rotational homography, nor as a planar homography, when $\mathbf{p} \sim \mathbf{p}_\infty$.

Although this relation between rotational and planar homographies is mainly a theoretical observation, it does have a practical consequence. We have seen that there is a requirement that the camera does not translate between the images, it only rotates about its center. This requirement can be relaxed if the distances to the observed 3D

¹¹Here, and elsewhere, we are using the fact that the operations of transpose and conjugate transpose are equivalent on real valued matrices.

points are much longer than that of any translation of the center. This means that there is no reason to assure, by mechanical or other means, that the camera center stays fixed during the rotation.

9.12 As long as the translation of a rotating camera is small, compared to the distance from the camera to the scene, corresponding points are in practice related by a rotational homography.

A consequence of this observation is that we can take images of a panoramic scene by a hand-held camera, and the images are related by rotational homographies. The only requirement is that the scene is at a distance that makes any motion of the camera center negligible.

9.3.5 Applications

In the same way as for cameras observing a plane, images from a rotating camera can be used to build image mosaics, since the images are related by homographies. In the case of a rotating camera, the resulting mosaic is often referred to as a panorama, and specific techniques for building panorama images are discussed in Section 19.2. An example of a camera rig for this particular purpose is shown in Figure 9.7.

Another application area of rotational homographies is camera calibration, i.e., finding the internal calibration \mathbf{K} of a specific camera. As we have seen in Equation (9.31), a rotational homography has a close relation to the image of the absolute conic ω , which in turn is directly linked to \mathbf{K} . In Section 16.4, we will discuss how to determine ω from two or more rotational homographies. The application of this approach to camera calibration is discussed more in detail in Section 18.1.1.



Figure 9.7: The Ladybug 2 camera rig from the Canadian company Point Grey Research, Inc. It has 5 cameras with optical axes 72° apart in the horizontal plane, and a 6th camera pointing upwards. The centers of the 6 cameras are approximately a decimeter apart, and there is an overlap in the field of view between neighboring cameras so that the total field of view of the entire camera system covers more than a hemisphere. This means that corresponding points in two images are (approximately) related by a rotational homography.

Chapter 10

Epipolar Geometry

Before you read this chapter you should have a thorough understanding of the homogeneous representations for \mathbb{E}^2 and \mathbb{E}^3 that are presented in Chapters 3 and 5. Furthermore, the pinhole camera model that is described in Chapter 8 will be used extensively in this chapter.

Epipolar geometry is the geometry of two views, i.e., two images that depict the same scene but from *two* distinct viewpoints, for example as illustrated in Figure 10.1. Such an arrangement of the cameras is referred to as *stereo cameras*. As long as there is a sufficient amount of overlap in two images of a single scene, a human has usually no problem to identify corresponding points in the two images, and to reconstruct the 3D structure in the scene. In this chapter we will develop a mathematical foundation to this feat: the *epipolar constraint*. Two points, y_1 in image 1 and y_2 in image 2, are referred to as *corresponding points*, or being *in correspondence*, or forming a *correspondence*, when they are projections of the same 3D point x . The epipolar constraint provides a condition on any pairs of points that must be satisfied if they are in correspondence, and it allows us to test if a pair of points from the two images are in correspondence or not. More precisely, the epipolar constraint is a necessary condition, but is not sufficient to determine correspondence. Consequently, it can only be used to falsify the hypothesis of correspondence between a pair of points from the two images.

The epipolar constraint is defined in terms of a 3×3 matrix, the *fundamental matrix* \mathbf{F} , which can be determined from the two camera matrices, \mathbf{C}_1 and \mathbf{C}_2 , defining the projection from 3D space to the stereo images. If \mathbf{C}_1 and \mathbf{C}_2 are known and both refer to the same 3D coordinate system, \mathbf{F} is fully determined and can be used for testing correspondences. This situation is referred to as *calibrated epipolar geometry*. An alternative is to determine \mathbf{F} directly from the epipolar constraint. Given a pair of points in the two images that we know are in correspondence, the epipolar constraint can be interpreted as a constraint on \mathbf{F} , specified by the two points. With a sufficiently large set of corresponding points has been found, it is then possible to estimate \mathbf{F} based on these constraints. This approach to determining \mathbf{F} is referred to as *uncalibrated epipolar geometry*, and has the advantage of not requiring the camera matrices to be known. Uncalibrated estimation of \mathbf{F} is further developed in Section 16.2.

The fundamental matrix depends only on the relative transformation between the corresponding stereo cameras. In the calibrated case, and assuming that we use camera normalized coordinates in both images, this implies that the fundamental matrix only depends on the rigid transformation between the two cameras. This type of fundamental matrix is particularly simple and useful, and is referred to as an *essential matrix*, denoted as \mathbf{E} . Since the essential matrix turns out to be independent to a uniform scaling of the 3D space, the case of calibrated epipolar geometry is extended to cameras with known internal calibration, known relative poses up to an unknown scaling. In the calibrated case, the essential matrix is the main representation of the epipolar geometry, as described in Section 10.5.1.

Given a pair of corresponding image points, y_1 and y_2 , it may be interesting to determine x from y_1 and y_2 . In theory this is straightforward. To each of the two points there is a projection line that intersects the point in the image plane and the corresponding camera center. These lines can be computed from y_1, y_2 and $\mathbf{C}_1, \mathbf{C}_2$, and they intersect at the 3D point x . Thus, x can be determined from relatively simple geometrical reasoning and calculations. In practice, measurement noise perturbs the image coordinates, and implies that the epipolar constraint is never satisfied exactly, not even for corresponding points. As a consequence, the two projection lines

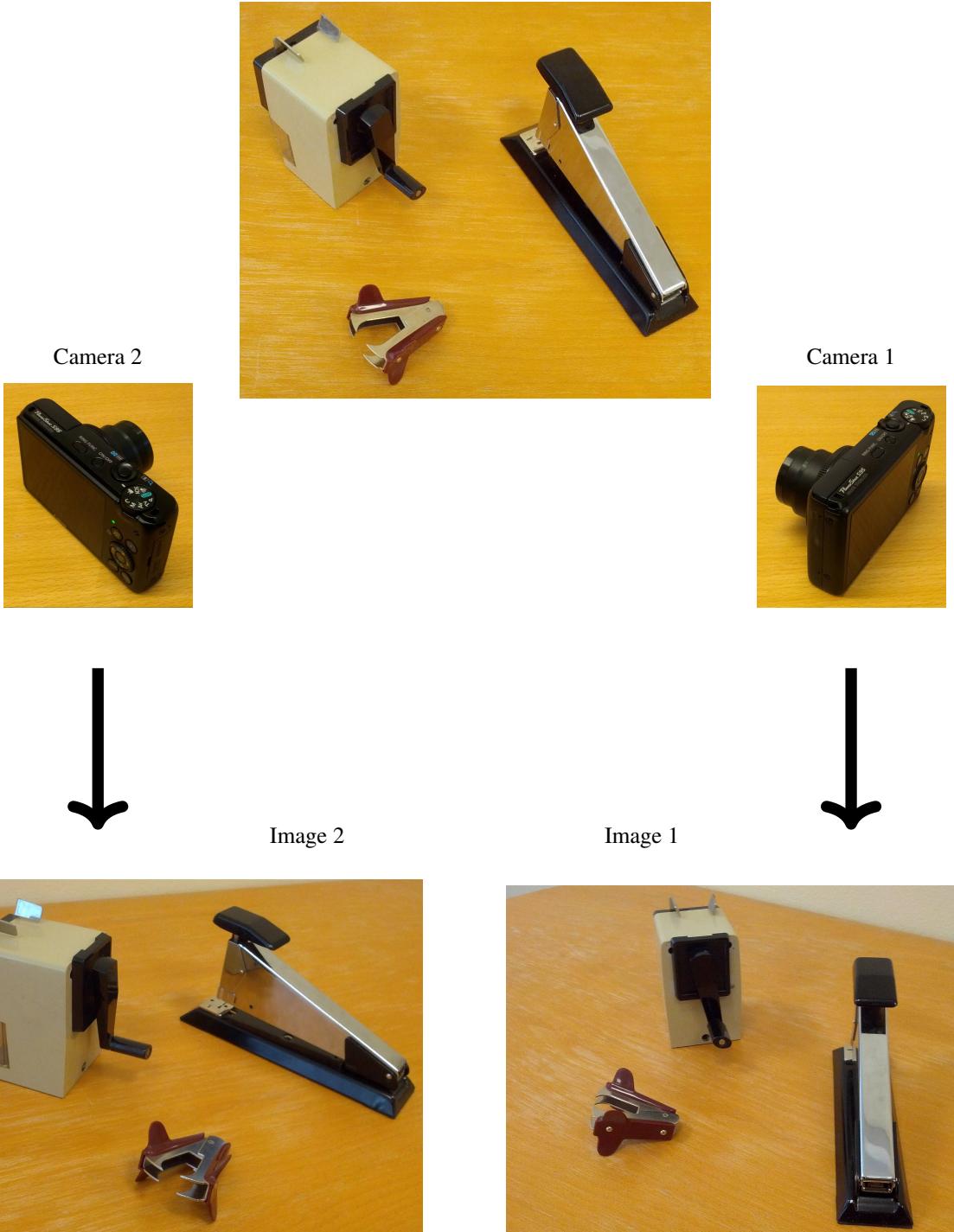


Figure 10.1: Top: a scene that contains some 3D objects. Two cameras with distinct centers depict the scene and each of them produce an image, at bottom. In the two images we can find a number of corresponding points, connected by lines.

do not intersect exactly, only approximately. The issue of determining \mathbf{x} then becomes an estimation problem that can be defined and solved in several ways. The general problem of determining \mathbf{x} from corresponding points $\mathbf{y}_1, \mathbf{y}_2$ is called *triangulation* or *3D reconstruction*, and is discussed in Section 10.4

10.1 Stereo Cameras

Stereopsis is our (and most other animals) ability to sense depth in our visual field by combining information provided by our two eyes. Stereoscopy is a technique that enables us to perceive depth in an image, by projecting two slightly different images onto our two eyes. Similarly, stereophonic sound aims to give a sense of direction to different sound sources. This is done by recording, transmitting, and reproducing two separate sound channels, one for each of our ears. The word *stereo* comes from the Greek language and means *solid*, and here we use it as in geometry, do describe something of volume, as opposed to flat.

Given this background, a stereo camera or a pair of stereo cameras refers to a system that produces images of a scene in such a way that it is possible to determine the 3D structure of the scene from these images. Such a system typically includes two or more cameras, but may also use a single camera in combination with special light sources that illuminate the scene in such a way that its 3D structure can be inferred.

In this presentation, we will use stereo cameras in a more restricted sense, based on geometry and aiming at the topic of this chapter: epipolar geometry.

Definition 10.1: Stereo cameras

A pair of *stereo cameras* corresponds to two pinhole cameras with distinct centers. The images from stereo cameras are referred to as *stereo images*.

In practice, stereo cameras are manifested mainly in two different ways. They can be implemented as a single system that consists of two distinct cameras, each with its own optical system and independent control of exposure and other parameters. Such a system is referred to as a *stereo rig*. A stereo rig is characterized by being able, at least potentially, to produce the stereo images as a single, simultaneous, exposure by both cameras. This means that the two images always correctly represent the geometry of the scene, at least at a specific instance of time. Even if objects in the scene are moving, their 3D poses can be inferred from the stereo images.

An alternative is to use a single camera that takes two images, from two distinct view-points by moving it from one pose to the other. This type of stereo cameras is referred to as *motion stereo*. As a consequence, the two images generated by motion stereo are not from the same time point. To make a correct interpretation of the 3D scene, it is therefore necessary that it includes only stationary objects. Objects in motion and surrounding parts of the scene will not be possible to reconstruct accurately from stereo images produced by motion stereo.

The geometrical relations that exist in stereo images are the same regardless of whether they are produced by a stereo rig or by motion stereo, in the latter case if we assume a stationary scene. Consequently, we will not distinguish between these two options for stereo cameras.

- 10.1** Stereo cameras can be implemented as a stereo rig, with two distinct cameras that depict a common scene. Alternatively, by motion stereo, where a single camera takes two images of the scene, moving from one pose to the other. Both approaches generate stereo images where the geometrical relations between the two images have the same mathematical form.

For brevity, we often refer to the two images as image 1 and image 2, but this labeling is arbitrary. In some contexts, it may be more appropriate to refer to the images as “left” and “right”, or “upper” and “lower”, depending on how the two camera poses are related. Furthermore, the assumption that the images depict a common scene is often used in practice since it enables us to find points in the stereo images that corresponds to the same 3D point¹. It is necessary since real images always are of limited extension, typically restricted by bounding boxes,

¹The overlap in fields of view is very practical, since it allows corresponding points to be visible in both images, but it is not necessary in a strict sense. Stereo vision can also be based on corresponding lines, which can be visible in both images without overlapping fields of view. But this approach has limited practical applications.

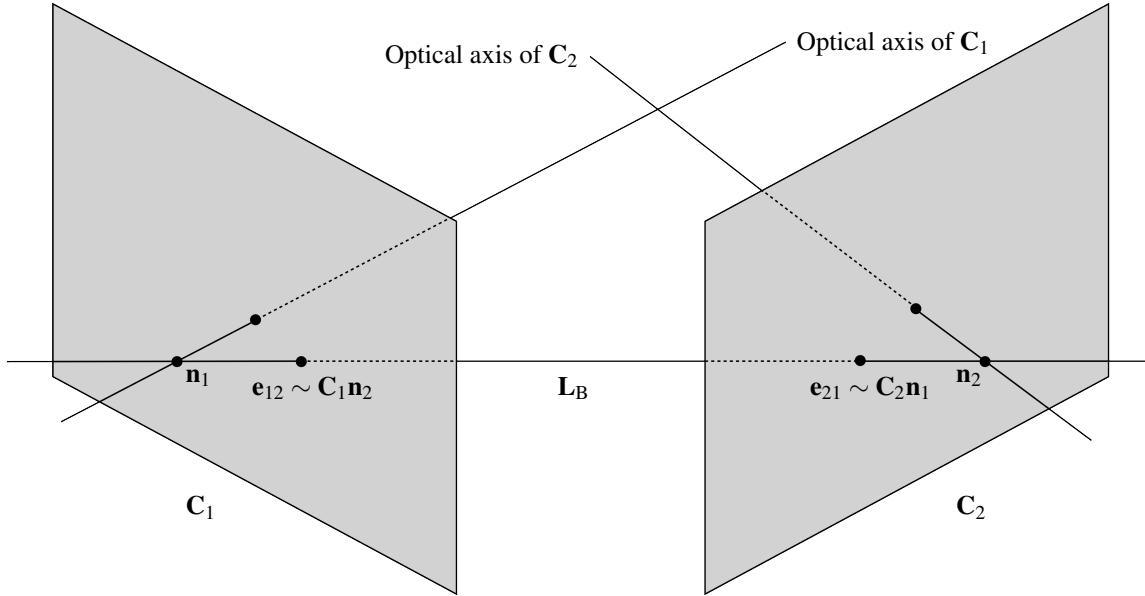


Figure 10.2: Stereo cameras with the two epipoles \mathbf{e}_{12} and \mathbf{e}_{21} , and the baseline \mathbf{L}_B . The two optical axes point in general directions, they do not need to intersect.

see Section 8.4. But from a geometric point of view, the image planes are of infinite extension and, therefore, the mathematical relations to be described in this chapter are valid also for points outside these bounding boxes.

The main result of this investigation is the epipolar constraint, presented in Section 10.2, but to get there we first need some preliminary results.

10.1.1 Epipolar points

Figure 10.2 shows a stereo camera that is represented by the two camera matrices $\mathbf{C}_1, \mathbf{C}_2$. For convenience, it uses virtual image planes, placed in front of the camera centers, \mathbf{n}_1 and \mathbf{n}_2 . One of the main issue in this chapter is to determine correspondences between the two images. From the figure, it should be clear that there is one specific point in each of the two images for which correspondence can be established directly: $\mathbf{e}_{12} \sim \mathbf{C}_1 \mathbf{n}_2$ in image 1 and $\mathbf{e}_{21} \sim \mathbf{C}_2 \mathbf{n}_1$ in image 2. These special points are therefore given a name of their own:

Definition 10.2: Epipole

The projection of one camera center into the other image, is referred to as an *epipolar point*, or simply *epipole*. The epipolar point in image 1, as the projection of the center of camera 2, is denoted as \mathbf{e}_{12} . The epipole \mathbf{e}_{21} is the projection of camera 1 into image 2.

The 3D line passing through the two epipolar points is the *baseline*² of the stereo cameras, denoted \mathbf{L}_B . The baseline \mathbf{L}_B intersects also both of the camera centers. This means that the baseline includes all points that are projected onto both of the two epipolar points, i.e., the two epipolar points are in correspondence.

The epipolar points are in a one-to-one correspondence, but shortly we will see that they represent an exception. A general point in one image can potentially be in correspondence with any point lying on a specific line in the other image.

²In some contexts, the stereo camera baseline refers to the line segment that starts and ends at the camera centers, i.e., its length is the distance between the centers. In other contexts, it is used to refer to the infinite line that passes through these two points.

10.1.2 Epipolar line

Figure 10.3 shows the same stereo cameras as before, but now with an additional point in the second image, \mathbf{y}_2 . This can be any point except \mathbf{e}_{21} , the epipolar point in the second image, and it has a corresponding projection line \mathbf{L}_2 that contains all 3D points that are projected onto \mathbf{y}_2 by camera \mathbf{C}_2 . In accordance with Section 8.5.1, this line can be defined uniquely in terms of two distinct points that lie on the line. For example, we can use the center of the second camera, \mathbf{n}_2 , and the 3D point $\mathbf{C}_2^+ \mathbf{y}_2$. In accordance with Section 8.5.2, this 3D line is projected by \mathbf{C}_1 to a line \mathbf{l}_1 in the first image, and this line intersects the projections of the two aforementioned points. Their projections in the second image are given as \mathbf{e}_{12} and $\mathbf{C}_1 \mathbf{C}_2^+ \mathbf{y}_2$:

$$\mathbf{l}_1 \sim \mathbf{e}_{12} \times (\mathbf{C}_1 \mathbf{C}_2^+ \mathbf{y}_2) = [\mathbf{e}_{12}]_\times \mathbf{C}_1 \mathbf{C}_2^+ \mathbf{y}_2. \quad (10.1)$$

The line \mathbf{l}_1 has the interesting property that it includes all points in image 1 that are in correspondence with point \mathbf{y}_2 . In a similar way, we could instead have constructed a line in image 2 that includes all points that are in correspondence with a particular point in image 1. A consequence of this observation is that, although point correspondences in stereo images cannot be established based on geometric relations between image points alone, it is possible to relate any point in one image to a line in the other image. This property renders the lines constructed here a special position in the geometry of stereo images.

Definition 10.3: Epipolar line

To each point in an image of a stereo pair, there is a line in the other image that contains all corresponding points. This line is the *epipolar line* generated by the point.

A property which follows directly from the construction of the epipolar line \mathbf{l}_1 is that it must intersect with the epipole \mathbf{e}_{12} , regardless of which point \mathbf{y}_2 it was generated from. Similarly, an epipolar line in image 2 must intersect

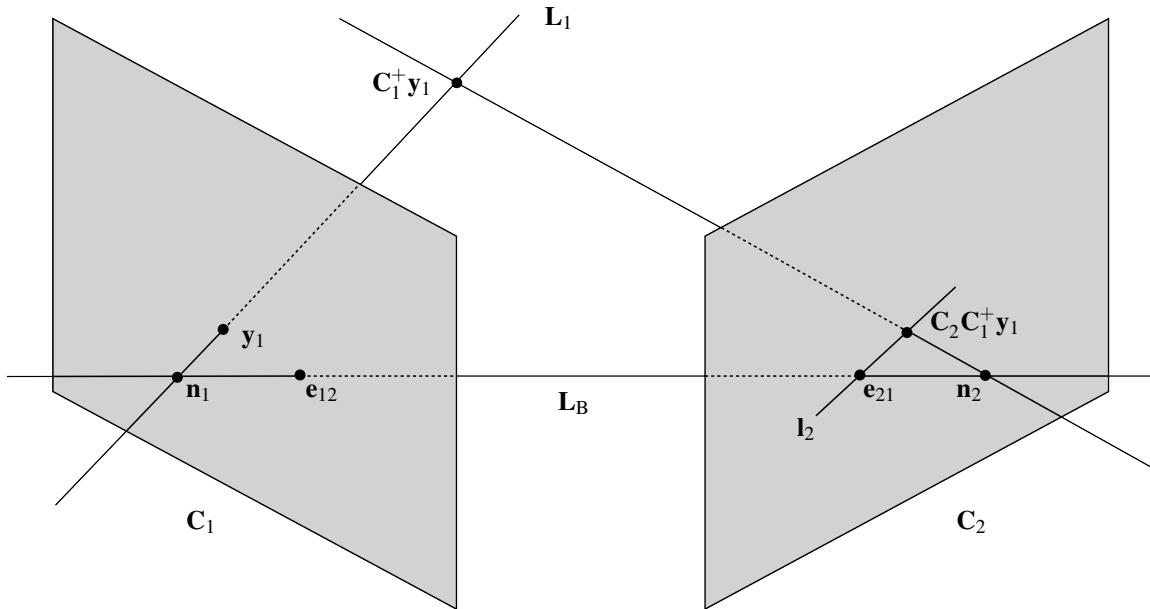


Figure 10.3: Two cameras depicting the same scene as in Figure 10.2. The line \mathbf{l}_1 is the epipolar line generated by point \mathbf{y}_2 .

with epipole \mathbf{e}_{21} .

10.2 Any epipolar line in one of the stereo images must intersect with the corresponding epipole in that image.

10.1.3 Epipolar plane

With epipolar points and epipolar lines established, we also make the following definition:

Definition 10.4: Epipolar plane

Any plane in 3D space that intersects both camera centers, \mathbf{n}_1 and \mathbf{n}_2 , is an *epipolar plane*. Equivalently, any plane that includes the baseline \mathbf{L}_B of a stereo camera is an epipolar plane.

Epipolar planes can be generated by 3D points. More precisely, any 3D point \mathbf{x} , not lying on the baseline \mathbf{L}_B , specifies a unique plane together with the two camera centers. This is the epipolar plane generated by \mathbf{x} .

Alternatively, an epipolar plane can be generated by a point in any of the two stereo images. This point generates an epipolar line in the other image. In accordance with Section 8.5.3, this line has a projection plane \mathbf{p} . This plane includes both camera centers, and also any projection line \mathbf{L}_2 that projects to \mathbf{y}_2 . Therefore, \mathbf{p} is an epipolar plane, generated by the image point \mathbf{y}_1 . See Figure 10.4 for an illustration.

As seen in the figure, an epipolar plane intersects with the two image plane along epipolar lines. In fact, the plane itself consists of 3D points that project into the images to end up on these lines. This means that an epipolar plane includes both 3D points, as well as their corresponding image points.

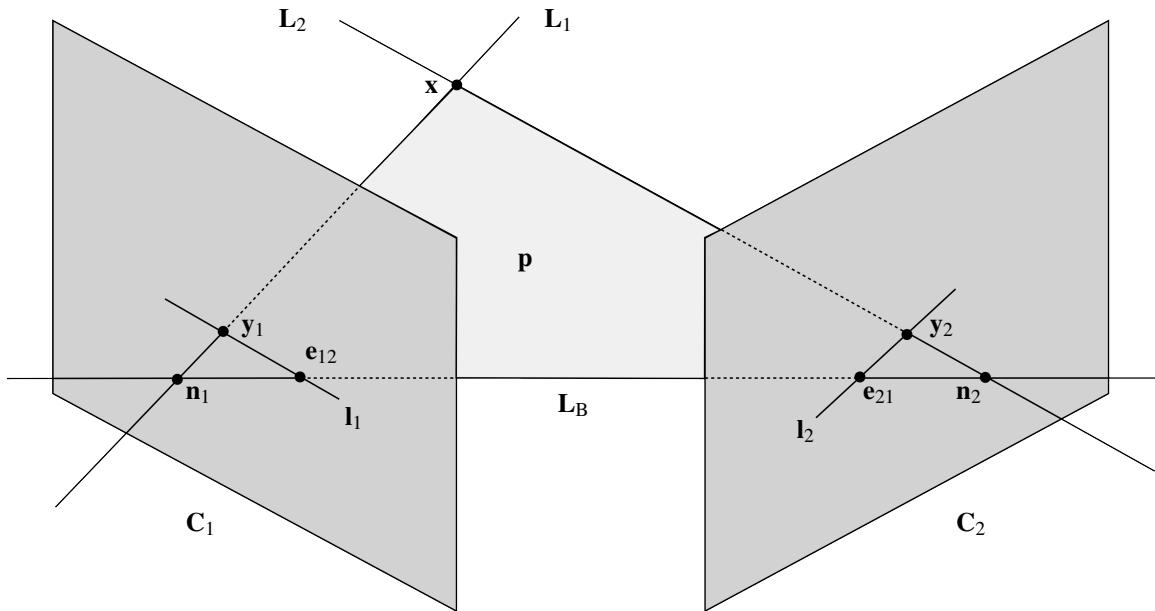


Figure 10.4: An epipolar plane generated by a 3D point \mathbf{x} . Alternatively, \mathbf{p} is the epipolar plane generated by either of the two image points \mathbf{y}_1 or \mathbf{y}_2 .

10.2 The fundamental matrix and the epipolar constraint

In the previous section, we looked at the geometric aspects of stereo images, and now we will instead turn to the algebraic consequences. Based on the epipolar lines that just have been introduced, we first define a matrix: the fundamental matrix. It forms a compact representation about more or less all geometric aspects of a pair of stereo images, given that no other information is available. For example, it specifies the epipolar constraint, which allows us to test if two points in the stereo images can be in correspondence.

The fundamental matrix was introduced simultaneously³ by Faugeras [17] and Hartley [28]. It can be seen as a generalization of the essential matrix, introduced by Longuet-Higgins [45] already in 1981, to the uncalibrated case. The essential matrix is described in Section 10.5.2.

10.2.1 The fundamental matrix

In Equation (10.1) we defined the epipolar line \mathbf{l}_1 , which is generated from point \mathbf{y}_2 in image 2. Assuming that the two cameras are fixed, the epipolar line \mathbf{l}_1 depends only on which point \mathbf{y}_2 we started with in the second image. More precisely, it is produced from \mathbf{y}_2 in terms of a linear transformation \mathbf{F} , defined as

$$\mathbf{F} \sim [\mathbf{e}_{12}]_{\times} \mathbf{C}_1 \mathbf{C}_2^+, \quad (10.2)$$

As we will see, the 3×3 matrix \mathbf{F} is a rich source of information about both geometric and algebraic relations in the context of stereo images.

Definition 10.5: Fundamental matrix

The matrix \mathbf{F} in Equation (10.3) is the the *fundamental matrix* related to the stereo images generated by cameras \mathbf{C}_1 and \mathbf{C}_2 .

Formally, \mathbf{F} represents a linear transformation $P(\mathbb{R}^3) \rightarrow P(\mathbb{R}^3)$, which makes \mathbf{F} a projective element in $P(\mathbb{R}^{3 \times 3})$.

As a start, we insert \mathbf{F} back into Equation (10.1) to get:

$$\mathbf{l}_1 \sim \mathbf{F} \mathbf{y}_2. \quad (10.3)$$

Given the general properties of epipolar lines, described in definition 10.3, we can make the following observation:

10.3 The fundamental matrix maps a point in one image to an epipolar line in the other image. This line contains all points that can be in correspondence with the first points.

Not any 3×3 matrix can be a fundamental matrix. Given its definition in Equation (10.2), it follows immediately that

$$\mathbf{e}_{12}^\top \mathbf{F} = \underbrace{\mathbf{e}_{12}^\top [\mathbf{e}_{12}]_{\times} \mathbf{C}_1 \mathbf{C}_2^+}_{=0} = 0. \quad (10.4)$$

This means that \mathbf{F} cannot have full rank. Since it maps points to epipolar lines, and these pass through a single point: the epipole, it also follows that \mathbf{F} must have rank 2. This shows that \mathbf{F} must satisfy an *internal constraint*:

10.4 A 3×3 matrix \mathbf{F} is a fundamental matrix that relates a pair of stereo images, if and only if

$$\text{rank } \mathbf{F} = 2. \quad (10.5)$$

More precisely, it must satisfy

$$\det \mathbf{F} = 0. \quad (10.6)$$

As a 3×3 matrix, \mathbf{F} has 9 degrees of freedom, but loses one degree of freedom since it is a projective element,

³Both papers appear at the 1992 ECCV conference.

and one more due to the internal constraint in Equation (10.6).

10.5 A fundamental matrix has 7 degrees of freedom.

10.2.2 The epipolar constraint

Let \mathbf{x} be a 3D point that projects onto \mathbf{y}_1 in image 1 and onto \mathbf{y}_2 in image 2, i.e., \mathbf{y}_1 and \mathbf{y}_2 are corresponding points. In many practical applications \mathbf{x} is not known, and if we find some point in one of the two images, we cannot know with certainty which is the corresponding point in the other image. However, if \mathbf{F} is known, there is a way to test if two points are in correspondence.

Thus, we slightly reformulate the situation: we know the fundamental matrix \mathbf{F} and have two image points \mathbf{y}_1 and \mathbf{y}_2 and want to know if they are in correspondence. From observation 10.3, we know that $\mathbf{l}_1 \sim \mathbf{F}\mathbf{y}_2$ is an epipolar line in image 1 that contains all possible points that are in correspondence with \mathbf{y}_2 . Therefore, if \mathbf{y}_1 is one of them, it must lie on \mathbf{l}_1 , i.e., it must be the case that

$$0 = \mathbf{y}_1 \cdot \mathbf{l}_1 = \mathbf{y}_1^\top \mathbf{l}_1 = \mathbf{y}_1^\top \mathbf{F}\mathbf{y}_2. \quad (10.7)$$

An important observation to make here is that we have showed that if \mathbf{y}_1 and \mathbf{y}_2 are corresponding points, then Equation (10.7) is valid. But it is not necessary the other way around. Each epipolar line includes infinitely many points, and all of them will satisfy Equation (10.7). Therefore, this constraints is a necessary but not sufficient condition for two points in stereo images to be in correspondence.

Definition 10.6: Epipolar constraint

The constraint defined by \mathbf{F} in Equation (10.7) is the *epipolar constraint*. The epipolar constraint provides a necessary but not sufficient condition for two points in stereo images to be in correspondence.

An alternative way to think about the epipolar constraint is to look at the projection lines \mathbf{L}_1 and \mathbf{L}_2 that are generated by the two points \mathbf{y}_1 and \mathbf{y}_2 , respectively. If the two points satisfy the epipolar constraint there exists a 3D point that projects onto both of \mathbf{y}_1 and \mathbf{y}_2 . This case therefore implies that the projection lines intersect at this point, and also that the two lines lie in the same epipolar plane. If the epipolar constraint is not satisfied, there is no common 3D point on the two projection lines, and they do not lie in a common plane.

These results are illustrated in Figure 10.5, where \mathbf{y}_1 and \mathbf{y}_2 are the projections of two distinct 3D points, \mathbf{x}_1 and \mathbf{x}_2 , respectively. This means that \mathbf{y}_1 and \mathbf{y}_2 are not corresponding points. But since \mathbf{x}_1 and \mathbf{x}_2 lies in the same epipolar plane, \mathbf{y}_1 and \mathbf{y}_2 satisfy the epipolar constraint.

A third alternative to think about the epipolar constraint is as a constraint on the fundamental matrix \mathbf{F} . If \mathbf{F} is not known, given two corresponding points, \mathbf{y}_1 and \mathbf{y}_2 , Equation (10.7) provides one constraint on the elements of \mathbf{F} . In principle, since \mathbf{F} has 7 degrees of freedom it should be possible to determine \mathbf{F} from 7 (or more) point correspondences. This means that if the camera matrices are unknown, \mathbf{F} can still be determined from a sufficiently large set of corresponding points in the two images.

10.6 If the camera matrices are unknown, \mathbf{F} can instead be determined from a sufficiently large set of correspondences.

This observation forms a basis for the methods that estimate \mathbf{F} , presented in Section 16.2.

10.2.3 Symmetry

The derivation of the epipolar constraint and the construction of \mathbf{F} in the previous sections start with the point \mathbf{y}_2 in the second image, then shows that it corresponds to an epipolar line in the first image, and conclude that any corresponding point \mathbf{y}_1 in this image must lie on this line. However, the labeling “first” and “second” image is completely arbitrary, and we can as well start with the point \mathbf{y}_1 in the first image.

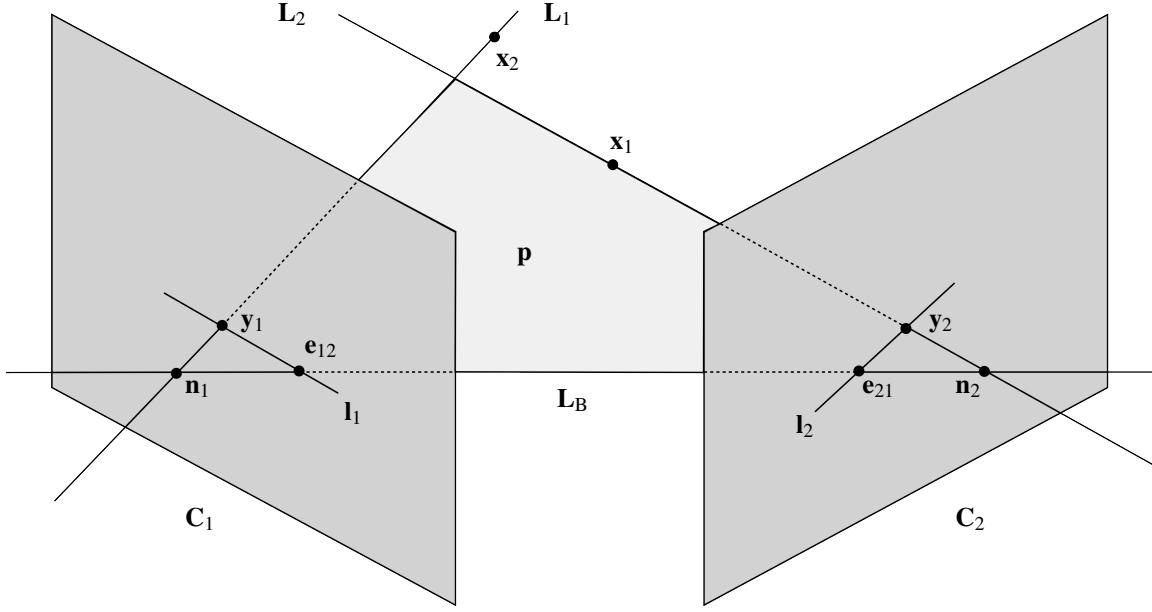


Figure 10.5: Two distinct 3D points, \mathbf{x}_1 and \mathbf{x}_2 , project onto the image points \mathbf{y}_1 and \mathbf{y}_2 , respectively. This means that the image points are not corresponding to a common 3D point, but since \mathbf{x}_1 and \mathbf{x}_2 lie in a common epipolar plane \mathbf{p} , the two image points satisfy the epipolar constraint.

If we do this, in the same way as before, \mathbf{y}_1 has a corresponding projection line \mathbf{L}_1 that contains all 3D points which camera C_1 projects onto \mathbf{y}_1 . The image of this 3D line is a 2D line \mathbf{l}_2 in the second image, again an *epipolar line*, given by

$$\mathbf{l}_2 \sim [\mathbf{e}_{21}]_{\times} \mathbf{C}_2 \mathbf{C}_1^+ \mathbf{y}_1. \quad (10.8)$$

By defining

$$\tilde{\mathbf{F}} \sim [\mathbf{e}_{21}]_{\times} \mathbf{C}_2 \mathbf{C}_1^+, \quad (10.9)$$

we can write the epipolar line \mathbf{l}_2 more compactly as

$$\mathbf{l}_2 \sim \tilde{\mathbf{F}} \mathbf{y}_1. \quad (10.10)$$

Finally, using the same arguments as before, any corresponding point \mathbf{y}_2 relative to \mathbf{y}_1 must lie on this epipolar line:

$$0 = \mathbf{l}_2 \cdot \mathbf{y}_2 = \mathbf{l}_2^\top \mathbf{y}_2 = \mathbf{y}_1^\top \tilde{\mathbf{F}}^\top \mathbf{y}_2. \quad (10.11)$$

Again, we arrive at a necessary condition for correspondence between \mathbf{y}_1 and \mathbf{y}_2 , the epipolar constraint.

From the outset, it is not obvious that Equation (10.7) and Equation (10.11) are equivalent. It could be the case that they represent two independent constraints, in particular since each of them only provides a necessary but not sufficient condition for correspondence. As it turns out, however, $\tilde{\mathbf{F}}^\top \sim \mathbf{F}$, i.e., it is the case that

$$([\mathbf{e}_{12}]_{\times} \mathbf{C}_1 \mathbf{C}_2^+)^T \sim [\mathbf{e}_{21}]_{\times} \mathbf{C}_2 \mathbf{C}_1^+, \quad (10.12)$$

but we leave the formal proof of this statement as an exercise. This means that there is only one epipolar constraint, and it is defined in terms of the fundamental matrix $\mathbf{F} \sim \tilde{\mathbf{F}}^\top$.

From this alternative expression of the fundamental matrix:

$$\mathbf{F} \sim \mathbf{C}_1^{+T} \mathbf{C}_2^\top [\mathbf{e}_{21}]_{\times}, \quad (10.13)$$

follows immediately the counterpart of Equation (10.4):

$$\mathbf{F} \mathbf{e}_{21} = \mathbf{0}. \quad (10.14)$$

10.2.4 The epipolar constraint in practice

In practice, neither \mathbf{F} nor the image coordinate of the two corresponding points $\mathbf{y}_1, \mathbf{y}_2$ are determined with full accuracy. If \mathbf{F} is computed from Equation (10.2), \mathbf{C}_1 and \mathbf{C}_2 represent only an ideal model of how 3D space is projected onto the images. The calibration process, which determines the parameters of the cameras, is correct only to a certain level accuracy. Furthermore, image coordinates contain some degree of measurement errors. Consequently, we cannot expect Equation (10.7) to be satisfied exactly for corresponding points.

A simple approach is to treat the right-hand side of Equation (10.7) as an algebraic error between the fundamental matrix \mathbf{F} and the pair of points $\mathbf{y}_1, \mathbf{y}_2$. As long as this error is below some threshold, we assume that the points and the fundamental matrix match each other sufficiently well. This approach, however, is *not recommended*. For one thing, this algebraic error is dependent on the scaling of the vectors $\mathbf{y}_1, \mathbf{y}_2$ and of the matrix \mathbf{F} , which means that we, somehow, must normalize the vectors and the matrix to avoid an arbitrary scaling of the algebraic error. It is also not trivial to choose the threshold, even though it may be done by observing the algebraic error for some representative set of corresponding points to see how large it may become. Finally, even if we can determine a threshold in this way, it does not correlate with any geometric quantity. This means that the algebraic error may be small even if a geometric error is large, and vice versa.

A better approach is to define a geometric error between \mathbf{F} and $\mathbf{y}_1, \mathbf{y}_2$. Each of the two points corresponds to an epipolar line, \mathbf{l}_2 or \mathbf{l}_1 , in the other image, and the other point should lie on, or close to, this line. We can measure the distances between the two points and the corresponding epipolar lines and, for example, formulate a simple geometric error as

$$\varepsilon_{\text{EPI}} = d_{\text{PD}}(\mathbf{y}_1, \mathbf{l}_1) + d_{\text{PD}}(\mathbf{y}_2, \mathbf{l}_2) = d_{\text{PD}}(\mathbf{y}_1, \mathbf{F} \mathbf{y}_2) + d_{\text{PD}}(\mathbf{y}_2, \mathbf{F}^\top \mathbf{y}_1). \quad (10.15)$$

Here, d_{PD} is the distance function between a point and a line in 2D, defined in Equation (3.34). This error has an intuitive interpretation in terms of distances in the image plane, and can typically have *pixels* as unit. A threshold for ε_{EPI} , which allows us to determine when the epipolar constraint is satisfied or not, should then be related to the measurement noise of the coordinates, as well as to the inaccuracies in \mathbf{F} . For example, we could choose a threshold of the same magnitude as the measurement noise of the image coordinates, times a factor of 2-4 to accommodate for estimation errors also in \mathbf{F} .

10.7 How well two points are consistent with the fundamental matrix in the epipolar constraint should always be quantified by some type of geometric error.

10.2.5 Transfer

When the epipolar geometry is known, specified by the fundamental matrix \mathbf{F} , this does not allow us to uniquely relate general points or lines in the two images. It does, however, offer the possibility of *transfer*, in which information about specific geometric object in one image can say something about a corresponding geometric object in the other image.

Transfer of points

In this context, there is only one pair of points for which correspondence is clear: the epipoles. They can be extracted as the left and right null spaces of \mathbf{F} , so this correspondence is well-defined whenever \mathbf{F} is known.

With the exception of the epipoles, points in one image are transferred to epipolar lines in the other image, in accordance with Equation (10.3). Notice that this point-to-line transfer is *not* a symmetric relation: if we know an epipolar line in one image, we cannot say from which point in the other image it was transferred. We can, however, specify the epipolar line that contains the point. This is referred to as transfer of epipolar lines.

Transfer of epipolar lines

In Section 10.1.3 we learned that any point \mathbf{y}_1 in the first image, which is distinct from epipole \mathbf{e}_{12} and also lies on the epipolar line \mathbf{l}_1 , has a projection line that lies in the epipolar plane \mathbf{p} . And the projection of this projection line into \mathbf{C}_2 is the epipolar line \mathbf{l}_2 . Vice versa, any point \mathbf{y}_2 in the second image, distinct from epipole \mathbf{e}_{21} and that lies on the epipolar line \mathbf{l}_2 , has a projection line that lies in \mathbf{p} . Its projection into \mathbf{C}_1 is the epipolar line \mathbf{l}_1 .

Consequently, there is a one-to-one correspondence between the epipolar lines \mathbf{l}_1 and \mathbf{l}_2 . \mathbf{l}_1 can be transferred to \mathbf{l}_2 , and vice versa. This one-to-one relation applies only to epipolar lines, not to general lines in the two images.

The relation between epipolar lines can also be given an algebraic formulation. Let \mathbf{l}_1 and \mathbf{l}_2 be epipolar lines, in image 1 and in image 2 respectively, corresponding to the same epipolar plane \mathbf{p} . From Equation (8.59) we then get the following two equations:

$$\mathbf{p} \sim \mathbf{C}_1^\top \mathbf{l}_1, \quad \text{and} \quad \mathbf{p} \sim \mathbf{C}_2^\top \mathbf{l}_2. \quad (10.16)$$

From this follows, for example, $\mathbf{l}_2 \sim \mathbf{C}_2^{+\top} \mathbf{p}$, which leads to

$$\mathbf{l}_2 \sim \mathbf{H}_{21} \mathbf{l}_1, \quad \text{where} \quad \mathbf{H}_{21} \sim \mathbf{C}_2^{+\top} \mathbf{C}_1^\top. \quad (10.17)$$

Here, \mathbf{H}_{21} is a homography transformation that transfers an epipolar line from image 1 to the corresponding epipolar line in image 2.

Definition 10.7: Line transfer mapping

A line transfer mapping maps an epipolar line in one image to the corresponding epipolar line in the other image.

As already mentioned, a line transfer mapping does not transfer arbitrary lines in an orderly fashion, only epipolar lines can be brought into correspondence in this way.

The previous observation is important to remember when we look at transfer of epipolar lines in the other direction. On the one hand, this can be done by \mathbf{H}_{21}^{-1} , the inverse of \mathbf{H}_{21} . On the other hand, we can also use the relations in Equation (10.16), in the opposite order as before, and get

$$\mathbf{l}_1 \sim \mathbf{H}_{12} \mathbf{l}_2, \quad \text{where} \quad \mathbf{H}_{12} \sim \mathbf{C}_1^{+\top} \mathbf{C}_2^\top. \quad (10.18)$$

This means that \mathbf{H}_{12} is a homography that transfer epipolar line \mathbf{l}_2 in image 2 to its corresponding epipolar line \mathbf{l}_1 in image 1.

Clearly, we expect that $\mathbf{H}_{12} \sim \mathbf{H}_{21}^{-1}$, but this is not correct in the general case. We have also seen that line transfer mappings can be homographies, but this does not mean that every mapping of this type is a homography. These issues, and more, will be more thoroughly discussed in Section 10.3.5.

10.3 More properties of the fundamental matrix

Now that the fundamental matrix \mathbf{F} and the epipolar constraint have been defined, and analyzed to some depth, it is time to go deeper into \mathbf{F} . This includes: how the numerical representation of the matrix \mathbf{F} is affected by transformations of 3D space, as well as of the images. We will also have a look at how \mathbf{F} can be parameterized in different ways, and how camera matrices that are consistent with \mathbf{F} can be recovered from \mathbf{F} itself. Finally, we will look at a factorization of \mathbf{F} as a product of two matrices, where one is a cross product operator defined by one of the epipoles, and the other one is a mapping that can be used for various types of transfer between the two images.

10.3.1 Invariance to homography transformations of 3D space

The epipolar geometry that is represented by the fundamental matrix \mathbf{F} depends on the two cameras \mathbf{C}_1 and \mathbf{C}_2 . This means that, in general, if some type of transformation is applied to the cameras, the fundamental matrix changes too. There is, however, one type of transformation that leaves the epipolar geometry invariant: when both cameras are subject to the same homography transformation of the 3D space.

To derive this result, we form two new camera matrices as

$$\mathbf{C}'_1 \sim \mathbf{C}_1 \mathbf{H}, \quad \mathbf{C}'_2 \sim \mathbf{C}_2 \mathbf{H}, \quad (10.19)$$

where \mathbf{H} is an arbitrary 3D homography transformation: $\mathbf{H} \in P(GL(4))$. We can interpret this as: \mathbf{C}'_1 and \mathbf{C}'_2 are the same cameras as \mathbf{C}_1 and \mathbf{C}_2 but represented relative to a 3D coordinate system that is transformed by \mathbf{H}^{-1} . Alternatively, the poses of cameras $\mathbf{C}_1, \mathbf{C}_2$ are transformed to new poses represented by $\mathbf{C}'_1, \mathbf{C}'_2$.

The fundamental matrix \mathbf{F}' , derived from \mathbf{C}'_1 and \mathbf{C}'_2 , is determined in accordance with Equation (10.13):

$$\begin{aligned}\mathbf{F}' &= \mathbf{C}'_1^{+\top} \mathbf{C}'_2^\top [\mathbf{e}'_{21}]_\times = (\mathbf{C}_1 \mathbf{H})^{+\top} \mathbf{H}^\top \mathbf{C}_2^\top [\mathbf{e}_{21}]_\times \stackrel{4}{=} (\mathbf{H}^{-1}(\mathbf{I} - \mathbf{n}_1 \mathbf{a}^\top) \mathbf{C}_1^+)^\top \mathbf{H}^\top \mathbf{C}_2^\top [\mathbf{e}_{21}]_\times = \\ &= \mathbf{C}_1^{+\top} (\mathbf{I} - \mathbf{a} \mathbf{n}_1^\top) \mathbf{H}^{-\top} \mathbf{H}^\top \mathbf{C}_2^\top [\mathbf{e}_{21}]_\times = \mathbf{C}_1^{+\top} (\mathbf{I} - \mathbf{a} \mathbf{n}_1^\top) \mathbf{C}_2^\top [\mathbf{e}_{21}]_\times = \\ &= \mathbf{C}_1^{+\top} \mathbf{C}_2^\top [\mathbf{e}_{21}]_\times - \underbrace{\mathbf{C}_1^{+\top} \mathbf{a} \mathbf{n}_1^\top \mathbf{C}_2^\top [\mathbf{e}_{21}]_\times}_{=\mathbf{e}_{21}^\top} = \mathbf{C}_1^{+\top} \mathbf{C}_2^\top [\mathbf{e}_{21}]_\times - \mathbf{C}_1^{+\top} \mathbf{a} \underbrace{\mathbf{e}_{21}^\top [\mathbf{e}_{21}]_\times}_{=0} = \mathbf{C}_1^{+\top} \mathbf{C}_2^\top [\mathbf{e}_{21}]_\times \sim \mathbf{F}\end{aligned}\quad (10.20)$$

In these expressions, \mathbf{a} is an arbitrary vector in \mathbb{R}^3 . A consequence of this results is that the fundamental matrix, the two epipolar points, and all epipolar lines remain the same when 3D space is transformed by a homography.

We summarize these results:

10.8 The epipolar geometry related to a pair of stereo cameras, represented by the fundamental matrix, epipolar points, and epipolar lines, is invariant to homography transformations of 3D space.

10.3.2 Transformations of the image coordinates

In contrast to transformations of the 3D space, the numerical representation of the epipolar geometry, represented by the fundamental matrix, change when transformations are applied to the two image spaces.

Let \mathbf{H}_1 be a homography that transforms the image coordinates in the first image and, similarly, \mathbf{H}_2 transforms image coordinates in the second image:

$$\mathbf{y}'_1 \sim \mathbf{H}_1 \mathbf{y}_1, \quad \mathbf{y}'_2 \sim \mathbf{H}_2 \mathbf{y}_2. \quad (10.21)$$

Here, \mathbf{y}_1 are the homogeneous coordinates of some image point in the first image and \mathbf{y}'_1 are the homogeneous coordinates of the same point after the homography \mathbf{H}_1 is applied, and similarly for \mathbf{y}'_2 and \mathbf{y}_2 . If \mathbf{y}_1 and \mathbf{y}_2 are corresponding points, they satisfy the epipolar constraint:

$$\mathbf{y}_1^\top \mathbf{F} \mathbf{y}_2 = 0. \quad (10.22)$$

If the homographies \mathbf{H}_1 and \mathbf{H}_2 refer to transformations of the coordinate systems in two images, the actual positions of the image points remain the same. This means that if \mathbf{y}_1 and \mathbf{y}_2 are corresponding points before the homographies are applied, they remain so also after the transformation. Consequently:

$$\mathbf{y}'_1^\top \mathbf{F}' \mathbf{y}'_2 = 0, \quad (10.23)$$

where \mathbf{F}' is the fundamental matrix that describes the epipolar geometry in the transformed coordinate systems.

One way to determine \mathbf{F}' is to transform the camera matrices to represent the new image coordinates:

$$\mathbf{C}'_1 \sim \mathbf{H}_1 \mathbf{C}_1, \quad \mathbf{C}'_2 \sim \mathbf{H}_2 \mathbf{C}_2. \quad (10.24)$$

We plug these new camera matrices into the usual expression for the fundamental matrix, Equation (10.2), and get \mathbf{F}' as

$$\begin{aligned}\mathbf{F}' &\sim [\mathbf{e}'_{12}]_\times \mathbf{C}'_1 \mathbf{C}'_2^+ = [\mathbf{H}_1 \mathbf{e}_{12}]_\times \mathbf{H}_1 \mathbf{C}_1 (\mathbf{H}_2 \mathbf{C}_2)^+ \stackrel{5}{=} \mathbf{H}_1^{-\top} \mathbf{H}_1^\top [\mathbf{H}_1 \mathbf{e}_{12}]_\times \mathbf{H}_1 \mathbf{C}_1 \mathbf{C}_2^+ \mathbf{H}_2^{-1} \sim \\ &\sim \mathbf{H}_1^{-\top} [\mathbf{e}_{12}]_\times \mathbf{C}_1 \mathbf{C}_2^+ \mathbf{H}_2^{-1} = \mathbf{H}_1^{-\top} \mathbf{F} \mathbf{H}_2^{-1}\end{aligned}\quad (10.25)$$

Consequently, Equation (10.25) describes the transformation of the fundamental matrix when the image coordinates in the two images are transformed in accordance with Equation (10.21).

⁴Here we are using Equation (3.122) that describes how the right pseudo-inverse is affected by multiplication from right prior to the inverse.

⁵Here we are using Equation (3.121) that describes how the right pseudo-inverse is affected by multiplication from left prior to the inverse.

10.3.3 Parameterization of \mathbf{F}

The fundamental matrix \mathbf{F} is represented by a 3×3 matrix, but it has only 7 degrees of freedom since it is projective element and has to satisfy $\det \mathbf{F} = 0$. How can we parameterize the matrix \mathbf{F} with only 7 parameters such that it always represents a valid fundamental matrix? This would correspond to a *minimal parameterization* of \mathbf{F} .

There are several answers to this question, and here we present a very simple parameterization of \mathbf{F} , based on SVD. Any fundamental matrix \mathbf{F} can be decomposed as

$$\mathbf{F} = \mathbf{U} \mathbf{S} \mathbf{V}^\top, \quad (10.26)$$

where $\mathbf{U}, \mathbf{V} \in SO(3)$ and \mathbf{S} holds the singular values $\sigma_1, \sigma_2, 0$ in its diagonal. This means that we should use the special version of SVD described in Toolbox Section 8.2.7 to assure $\mathbf{U}, \mathbf{V} \in SO(3)$. In this particular case, since one singular value is zero, this does not have to affect the two non-zero singular values. Furthermore, $\sigma_1 \geq \sigma_2 > 0$ since \mathbf{F} has rank 2. To obtain a minimal parameterization of \mathbf{F} we must normalize σ_1, σ_2 , in one way or another. For example, we can express \mathbf{F} as

$$\mathbf{F} = \mathbf{U} \begin{pmatrix} \cos v & 0 & 0 \\ 0 & \sin v & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{V}^\top, \quad \text{where} \quad \tan v = \frac{\sigma_2}{\sigma_1}. \quad (10.27)$$

In section Chapter 11, a number of minimal parameterizations of $SO(3)$ are presented and we can choose any of them to parameterize $\mathbf{U}, \mathbf{V} \in SO(3)$. Additionally, we have the parameter v that controls the relation between the non-zero singular values of \mathbf{F} . In total this gives $3 + 3 + 1 = 7$ parameters to parameterize \mathbf{F} in Equation (10.27), and therefore it is a minimal parameterization.

Other minimal parameterizations exist in the literature, but the one presented here offers the possible advantage of always guaranteeing that $\|\mathbf{F}\|_F = 1$, which may be advantageous in an optimization process.

10.3.4 Camera matrices from \mathbf{F}

The derivation of \mathbf{F} in the previous section explicitly uses the two camera matrices \mathbf{C}_1 and \mathbf{C}_2 to construct projection lines that can be projected from one camera into the other camera, and as a result we produce epipolar lines. Once the existence of \mathbf{F} is proven, however, it exists independent of whether we actually know the two camera matrices or not, i.e., independent of whether the stereo cameras are calibrated or not. In most practical situations, the two camera matrices are not known, and \mathbf{F} can then instead be determined from a set of corresponding points, for example using any of the methods described in the Section 16.2. This is referred to as *uncalibrated epipolar geometry*.

Once \mathbf{F} have been determined or estimated for the uncalibrated case, it may in some cases be relevant to establish two camera matrices \mathbf{C}_1 and \mathbf{C}_2 that are consistent with this \mathbf{F} . Each such camera matrix has 11 degrees of freedom, in total 22 degrees of freedom for 2 cameras. The fundamental matrix, however, has only 7 degrees of freedom, which implies that there must be many choices of camera matrices that produce the same \mathbf{F} . In fact, there are infinitely many such choices. In Section 10.3.1 is shown that \mathbf{F} is invariant to any 3D transformation of 3D space. In accordance with Section 7.2, a 3D homography transformation has 15 degrees of freedom, which means that from the 22 degrees of freedom for choosing the two camera matrices, there are only $22 - 15 = 7$ degrees of freedom left to determine the epipolar geometry. These are the 7 degrees of freedom for the fundamental matrix.

Since \mathbf{F} is independent of a 3D transformation, we can choose one of the two camera matrices, for example \mathbf{C}_1 , to have an as simple form as possible, and then try to determine \mathbf{C}_2 that is consistent with a given \mathbf{F} . All other pairs of camera matrices that are consistent with the same \mathbf{F} are then produced by applying an arbitrary homography transformation of the 3D space, as in Equation (10.19).

We set

$$\mathbf{C}_1 = (\mathbf{I} | \mathbf{0}), \quad \mathbf{C}_2 = (\mathbf{A} | \mathbf{b}). \quad (10.28)$$

where $\mathbf{A} \in \mathbb{R}^{3 \times 3}$ and $\mathbf{b} \in \mathbb{R}^3$. The goal is to determine \mathbf{A} and \mathbf{b} such that we can compute a given \mathbf{F} from Equation (10.2). It follows trivially that $\mathbf{C}_1^{+\top} = \mathbf{C}_1$ and also that

$$\mathbf{n}_1 \sim \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}, \quad \Rightarrow \quad \mathbf{e}_{21} \sim \mathbf{C}_2 \mathbf{n}_1 = \mathbf{b}. \quad (10.29)$$

Algorithm 10.1: Determine $\mathbf{C}_1, \mathbf{C}_2$ that are consistent with \mathbf{F} , based on epipoles

- Input:** A fundamental matrix \mathbf{F} , where $\det \mathbf{F} = 0$.
Output: A camera matrix \mathbf{C}_2 which together with $\mathbf{C}_1 = (\mathbf{I} | \mathbf{0})$ are consistent with \mathbf{F} .
- 1 Determine epipolar point \mathbf{e}_{21} from $\mathbf{F}\mathbf{e}_{21} = \mathbf{0}$
 - 2 Determine $\lambda_1, \lambda_2 \neq 0$ and $\mathbf{v} \in \mathbb{R}^3$
 - 3 $\lambda_1 = \lambda_2 = 1$ and $\mathbf{v} = \mathbf{0}$ work fine unless additional constraints must be satisfied
 - 4 Set $\mathbf{C}_2 = (\lambda_1[\mathbf{e}_{21}]_\times \mathbf{F}^\top + \mathbf{b} \mathbf{v}^\top | \lambda_2 \mathbf{e}_{21})$

The fundamental matrix \mathbf{F} can therefore be expressed as

$$\mathbf{F} \sim \mathbf{C}_1^{+\top} \mathbf{C}_2^\top [\mathbf{e}_{21}]_\times = (\mathbf{I} | \mathbf{0}) \begin{pmatrix} \mathbf{A}^\top \\ \mathbf{b}^\top \end{pmatrix} [\mathbf{b}]_\times = \mathbf{A}^\top [\mathbf{b}]_\times. \quad (10.30)$$

The problem, then, is to determine \mathbf{A} and \mathbf{b} for a given \mathbf{F} such that Equation (10.30) is satisfied. Several options exist, and we review a pair of approaches that are practically useful.

Based on epipolar points

By simply looking at Equation (10.30) we can guess values for \mathbf{A} and \mathbf{b} that satisfy the equation for a given \mathbf{F} . For example, we can try

$$\mathbf{A} = [\mathbf{e}_{21}]_\times \mathbf{F}^\top, \quad \mathbf{b} = \mathbf{e}_{21}. \quad (10.31)$$

The right-hand side of Equation (10.30) then becomes:

$$\mathbf{A}^\top [\mathbf{b}]_\times = \mathbf{F} [\mathbf{e}_{21}]_\times^\top [\mathbf{e}_{21}]_\times \stackrel{\text{Equation (10.13)}}{\sim} \mathbf{C}_1^{+\top} \mathbf{C}_2^\top [\mathbf{e}_{21}]_\times^3 \stackrel{\text{Equation (3.147)}}{\sim} \mathbf{C}_1^{+\top} \mathbf{C}_2^\top [\mathbf{e}_{21}]_\times \sim \mathbf{F}. \quad (10.32)$$

A closer look at these relations reveal that we can multiply either of \mathbf{A} or \mathbf{b} in Equation (10.31) with an arbitrary non-zero scalar, and the result is, again, a pair of camera matrices that are consistent with \mathbf{F} . Also, we can add $\mathbf{b} \mathbf{v}^\top$ to \mathbf{A} , for arbitrary $\mathbf{v} \in \mathbb{R}^3$, since this term vanishes when it meets the cross product with \mathbf{b} in Equation (10.30).

Consequently, the two camera matrices

$$\mathbf{C}_1 = (\mathbf{I} | \mathbf{0}), \quad \mathbf{C}_2 = (\lambda_1[\mathbf{e}_{21}]_\times \mathbf{F}^\top + \mathbf{b} \mathbf{v}^\top | \lambda_2 \mathbf{e}_{21}), \quad (10.33)$$

are consistent with the fundamental matrix \mathbf{F} for arbitrary $\lambda_1, \lambda_2 \neq 0$ and $\mathbf{v} \in \mathbb{R}^3$. This method of determining the two camera matrices from \mathbf{F} is summarized in Algorithm 10.1.

Based on SVD of \mathbf{F}

An alternative approach to determine \mathbf{A} and \mathbf{b} is to transform \mathbf{F} to a simpler form. This can be done by means of SVD: $\mathbf{A}^\top [\mathbf{b}]_\times \sim \mathbf{F} = \mathbf{U} \mathbf{S} \mathbf{V}^\top$. Here $\mathbf{U}, \mathbf{V} \in O(3)$ and \mathbf{S} is diagonal and holds the singular values of \mathbf{F} , where one of them is zero. The equality in Equation (10.30) can then be reformulated as

$$\mathbf{S} \sim \mathbf{U}^\top \mathbf{A}^\top [\mathbf{b}]_\times \mathbf{V}. \quad (10.34)$$

Set $\mathbf{A}' = \mathbf{V}^\top \mathbf{A} \mathbf{U}$ and $\mathbf{b}' = \mathbf{V}^\top \mathbf{b}$, which inserted into the last equation leads to

$$\mathbf{A}'^\top [\mathbf{b}']_\times \sim \mathbf{A}'^\top \mathbf{V}^\top [\mathbf{V} \mathbf{b}']_\times \mathbf{V} \stackrel{\text{Equation (3.152)}}{\sim} \mathbf{U}^\top \mathbf{A}^\top [\mathbf{b}]_\times \mathbf{V} \sim \mathbf{S} = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad (10.35)$$

where $\sigma_1 \geq \sigma_2 > 0$ are the two largest singular values of \mathbf{F} . This equation is satisfied by

$$\mathbf{A}' = \begin{pmatrix} 0 & -\sigma_2 & 0 \\ \sigma_1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{b}' = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \quad (10.36)$$

Algorithm 10.2: Determine $\mathbf{C}_1, \mathbf{C}_2$ that are consistent with \mathbf{F} , based on SVD

Input: A minimal parameterization of \mathbf{F} in accordance with Section 10.3.3

Output: A camera matrix \mathbf{C}_2 which together with $\mathbf{C}_1 = (\mathbf{I} | \mathbf{0})$ are consistent with \mathbf{F} .

- 1 The minimal parameterization of \mathbf{F} implies: $\mathbf{F} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$, where $\mathbf{U}, \mathbf{V} \in SO(3)$ and $\mathbf{S} = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$
- 2 Set $\mathbf{C}_2 = \mathbf{V} \begin{pmatrix} 0 & -\sigma_2 & 0 & 0 \\ \sigma_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{U}^\top & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}$

We can now formulate \mathbf{C}_2 as:

$$\mathbf{C}_2 = (\mathbf{A} | \mathbf{b}) = \left(\mathbf{V} \mathbf{A}' \mathbf{U}^\top | \mathbf{V} \mathbf{b}' \right) = \mathbf{V} \underbrace{\left(\mathbf{A}' | \mathbf{b}' \right)}_{:= \mathbf{C}'_2} \begin{pmatrix} \mathbf{U}^\top & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix}. \quad (10.37)$$

Notice that

$$\mathbf{C}'_2 = (\mathbf{A}' | \mathbf{b}') = \begin{pmatrix} 0 & -\sigma_2 & 0 & 0 \\ \sigma_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (10.38)$$

is a 3×4 matrix of full row rank, which means that also \mathbf{C}_2 is of full rank.

This method for determining a pair of matrices that are consistent with a given \mathbf{F} is summarized in Algorithm 10.2. Once a specific pair of camera matrices is computed from \mathbf{F} , the full set of consistent camera matrices is given by Equation (10.19) for an arbitrary 3D homography \mathbf{H} . This particular approach to determine the camera matrices from \mathbf{F} is useful in the case that \mathbf{F} is parameterized in accordance with Section 10.3.3. In this case, the minimal parameterization in terms of $\mathbf{U}, \mathbf{V} \in SO(3)$ and $v \in \mathbb{R}$, directly generates both \mathbf{F} from Equation (10.27) and \mathbf{C}_2 from Equation (10.37) and Equation (10.38).

10.3.5 Factorization of \mathbf{F}

In this section we will discuss the particular factorization of \mathbf{F} as

$$\mathbf{F} \sim [\mathbf{e}_{12}]_\times \mathbf{M}, \quad \text{alternatively, } \mathbf{F} \sim \tilde{\mathbf{M}} [\mathbf{e}_{21}]_\times, \quad (10.39)$$

where \mathbf{M} and $\tilde{\mathbf{M}}$ are full rank matrices. In accordance with Equation (10.2), we choose $\mathbf{M} \sim \mathbf{C}_1 \mathbf{C}_2^+$. But this is only one possible choice. \mathbf{M} cannot be unique, since from Equation (10.39) follows

$$\mathbf{F} \sim [\mathbf{e}_{12}]_\times \mathbf{M} = [\mathbf{e}_{12}]_\times (\mathbf{M} + \mathbf{e}_{12} \mathbf{a}^\top), \quad (10.40)$$

for any $\mathbf{a} \in \mathbb{R}^3$. The alternative form is $\mathbf{F} \sim (\tilde{\mathbf{M}} + \tilde{\mathbf{a}} \mathbf{e}_{21}^\top) [\mathbf{e}_{21}]_\times$, where, e.g., $\tilde{\mathbf{M}} = \mathbf{C}_1^{+\top} \mathbf{C}_2^\top$, and for any $\tilde{\mathbf{a}} \in \mathbb{R}^3$.

10.9 The fundamental matrix can always be factorized in accordance with Equation (10.39), but \mathbf{M} or $\tilde{\mathbf{M}}$ are not unique.

In this context, the matrices \mathbf{M} or $\tilde{\mathbf{M}}$ have some interesting and useful properties, several of which are described in an article by Hartley [29]. For example, from Equation (10.39) follows that $\mathbf{0} = \mathbf{F} \mathbf{e}_{21} \sim [\mathbf{e}_{12}]_\times \mathbf{M} \mathbf{e}_{21}$. This implies that $\mathbf{e}_{12} \times (\mathbf{M} \mathbf{e}_{21}) = \mathbf{0}$, i.e., $\mathbf{M} \mathbf{e}_{21} \sim \mathbf{e}_{12}$. We conclude:

10.10 The matrix \mathbf{M} (and $\tilde{\mathbf{M}}^{-\top}$) in Equation (10.39) transfers epipole \mathbf{e}_{21} to \mathbf{e}_{12} . Consequently, \mathbf{M}^{-1} (and $\tilde{\mathbf{M}}^\top$) transfers \mathbf{e}_{12} to \mathbf{e}_{21} .

With this observation at hand, we see that $\mathbf{M}^\top \mathbf{F} \sim \mathbf{M}^\top [\mathbf{e}_{12}]_\times \mathbf{M}$. Based on Equation (3.152) and observa-

tion 10.10, $\mathbf{M}^\top \mathbf{F}$ can then reformulated as $\mathbf{M}^\top \mathbf{F} \sim [\mathbf{M}^{-1} \mathbf{e}_{12}]_\times \sim [\mathbf{e}_{21}]_\times$. In summary, we get

10.11 When \mathbf{F} can be factorized as in Equation (10.39), it also factorizes as $\mathbf{F} \sim \mathbf{M}^{-\top} [\mathbf{e}_{21}]_\times \sim [\mathbf{e}_{12}]_\times \tilde{\mathbf{M}}^{-\top}$.

We also see that when \mathbf{y}_2 is a point in image 2, then $\mathbf{l}_1 = \mathbf{F}\mathbf{y}_2$ is an epipolar line in image 1, and we can interpret $\mathbf{y}'_1 = \mathbf{M}\mathbf{y}_2$ as a point, also in image 1. In fact, from $\mathbf{l}_1^\top \mathbf{y}'_1 = (\mathbf{F}\mathbf{y}_2)^\top \mathbf{M}\mathbf{y}_2 = \mathbf{y}_2^\top \mathbf{M}^\top [\mathbf{e}_{12}]_\times \mathbf{M}\mathbf{y}_2 = 0$, follows:

10.12 If \mathbf{y}_2 is a point in image 2, then $\mathbf{M}\mathbf{y}_2$ is a point in the first image that lies on the epipolar line given by $\mathbf{F}\mathbf{y}_2$. By symmetry, if \mathbf{y}_1 is a point in image 1, then $\tilde{\mathbf{M}}^\top \mathbf{y}_1$ is a point in the second image that lies on the epipolar line given by $\mathbf{F}^\top \mathbf{y}_1$.

Let \mathbf{l}_1 and \mathbf{l}_2 be two corresponding epipolar lines, in image 1 and image 2, and let \mathbf{y}_2 be a point on \mathbf{l}_2 . It then follows that $\mathbf{l}_1 \sim \mathbf{F}\mathbf{y}_2 \sim \mathbf{M}^{-\top} [\mathbf{e}_{21}]_\times \mathbf{y}_2 = \mathbf{M}^{-\top} (\mathbf{e}_{21} \times \mathbf{y}_2) \sim \mathbf{M}^{-\top} \mathbf{l}_2$. This summarizes as:

10.13 $\mathbf{M}^{-\top}$ (and $\tilde{\mathbf{M}}$) transfers an epipolar line in image 2 to the corresponding epipolar line in image 1. Conversely, \mathbf{M}^\top (and $\tilde{\mathbf{M}}^{-1}$) transfers an epipolar line in image 1 to the corresponding epipolar line in image 2.

Finding \mathbf{M} or $\tilde{\mathbf{M}}$ from \mathbf{F}

In many practical situations, we can determine \mathbf{F} between two camera views, but the camera matrices \mathbf{C}_1 and \mathbf{C}_2 are unknown. This means that we cannot set $\mathbf{M} \sim \mathbf{C}_1 \mathbf{C}_2^+$, but it is still possible to determine \mathbf{M} or $\tilde{\mathbf{M}}$ directly from \mathbf{F} . We will now discuss how this can be done, but focus only on \mathbf{M} .

The fundamental matrix has a range that is orthogonal to \mathbf{e}_{12} , and $(\mathbf{I} - \hat{\mathbf{e}}_{12} \hat{\mathbf{e}}_{12}^\top)$ is the projection operator onto this range, where $\hat{\mathbf{e}}_{12}$ is the homogeneous coordinates of the epipole in image 1, with unit norm as a vector in \mathbb{R}^3 . Consequently:

$$\mathbf{F} = (\mathbf{I} - \hat{\mathbf{e}}_{12} \hat{\mathbf{e}}_{12}^\top) \mathbf{F} \sim -[\mathbf{e}_{12}]_\times^2 \mathbf{F}. \quad (10.41)$$

This suggests that we can choose $\mathbf{M} \sim [\mathbf{e}_{12}]_\times \mathbf{F}$, but then we do not get a \mathbf{M} with of full rank. However, Equation (10.40) means that we can add a term $\mathbf{e}_{12} \mathbf{a}^\top$ to $[\mathbf{e}_{12}]_\times \mathbf{F}$, where \mathbf{a} is chosen to give full rank to the result. For example, we can choose \mathbf{M} as

$$\mathbf{M} \sim [\mathbf{e}_{12}]_\times \mathbf{F} + \lambda \hat{\mathbf{e}}_{12} \hat{\mathbf{e}}_{21}^\top. \quad (10.42)$$

Here, $\lambda \neq 0$ is a scalar that is chosen to make the singular values of \mathbf{M} to have approximately equal order of magnitude. For example, if the singular values of \mathbf{F} in Equation (10.26) are $\sigma_1, \sigma_2, 0$, we can set $\lambda = \frac{\sigma_1 + \sigma_2}{2}$. In fact, an equivalent approach to choosing \mathbf{M} is to first apply an SVD on \mathbf{F} , in accordance with Equation (10.26), and then set

$$\mathbf{M} \sim \mathbf{U} \mathbf{S}' \mathbf{V}^\top, \quad \text{where} \quad \mathbf{S}' = \begin{pmatrix} 0 & \sigma_2 & 0 \\ -\sigma_1 & 0 & 0 \\ 0 & 0 & \lambda \end{pmatrix}. \quad (10.43)$$

The corresponding expression for $\tilde{\mathbf{M}}$ is

$$\tilde{\mathbf{M}} \sim \mathbf{U} \tilde{\mathbf{S}} \mathbf{V}^\top, \quad \text{where} \quad \tilde{\mathbf{S}} = \begin{pmatrix} 0 & \sigma_1 & 0 \\ -\sigma_2 & 0 & 0 \\ 0 & 0 & \lambda \end{pmatrix}. \quad (10.44)$$

The derivation of these results is left as an exercise.

10.3.6 Summary of properties related to \mathbf{F}

In the list below, we summarize the facts derived so far about the fundamental matrix \mathbf{F} .

1. \mathbf{F} can be computed directly from the camera matrices either as $\mathbf{F} \sim [\mathbf{e}_{12}]_\times \mathbf{C}_1 \mathbf{C}_2^+$ or as $\mathbf{F} \sim \mathbf{C}_1^{+\top} \mathbf{C}_2^\top [\mathbf{e}_{21}]_\times$.

2. \mathbf{F} is a 3×3 matrix of rank 2. As a projective element it has 7 degrees of freedom, since it must satisfy the internal constraint $\det \mathbf{F} = 0$.
3. The left and right null spaces of \mathbf{F} are determined by the epipoles: $\mathbf{F}\mathbf{e}_{21} = \mathbf{F}^\top\mathbf{e}_{12} = \mathbf{0}$.
4. \mathbf{F} generates a necessary but not sufficient condition, the epipolar constraint, which must be satisfied for corresponding points from a pair of stereo images: $\mathbf{y}_1^\top \mathbf{F} \mathbf{y}_2 = 0$.
5. \mathbf{F} maps a point in one image to an epipolar line in the other image: $\mathbf{l}_1 \sim \mathbf{F}\mathbf{y}_2, \mathbf{l}_2 \sim \mathbf{F}^\top\mathbf{y}_1$.
6. \mathbf{F} is invariant to transformations of 3D space.
7. Transformations of the two image coordinates, by the homographies \mathbf{H}_1 and \mathbf{H}_2 , transform the fundamental matrix to $\mathbf{H}_1^{-\top} \mathbf{F} \mathbf{H}_2^{-1}$.
8. A minimal parameterization of \mathbf{F} (7 parameters) can be defined in accordance with Section 10.3.3.
9. Given \mathbf{F} , a pair of camera matrices \mathbf{C}_1 and \mathbf{C}_2 , consistent with \mathbf{F} , can be determined by means of Algorithm 10.1 on page 150 or Algorithm 10.2. There are infinitely many camera matrices \mathbf{C}_1 and \mathbf{C}_2 that produce the same \mathbf{F} .
10. \mathbf{F} can be factorized as $\mathbf{F} \sim [\mathbf{e}_{12}]_\times \mathbf{M}$, where \mathbf{M} is a 3×3 full rank matrix. This \mathbf{M} is not unique, and an example of how \mathbf{M} can be chosen is given by Equation (10.43). Alternatively, $\mathbf{F} \sim \tilde{\mathbf{M}} [\mathbf{e}_{21}]_\times$, where $\tilde{\mathbf{M}}$ is given, e.g., by Equation (10.44).

10.4 Triangulation

A major application of the fundamental matrix is to establish correspondences between pairs of points in stereo images. It allows us to verify the hypothesis that the two points are in correspondence. Once a correspondence is established, we can treat the two points as projections of the same point in 3D space. This leads to the problem of determining the 3D coordinates of this point, given that the two corresponding image points are known, a problem known as *triangulation* or *reconstruction*.

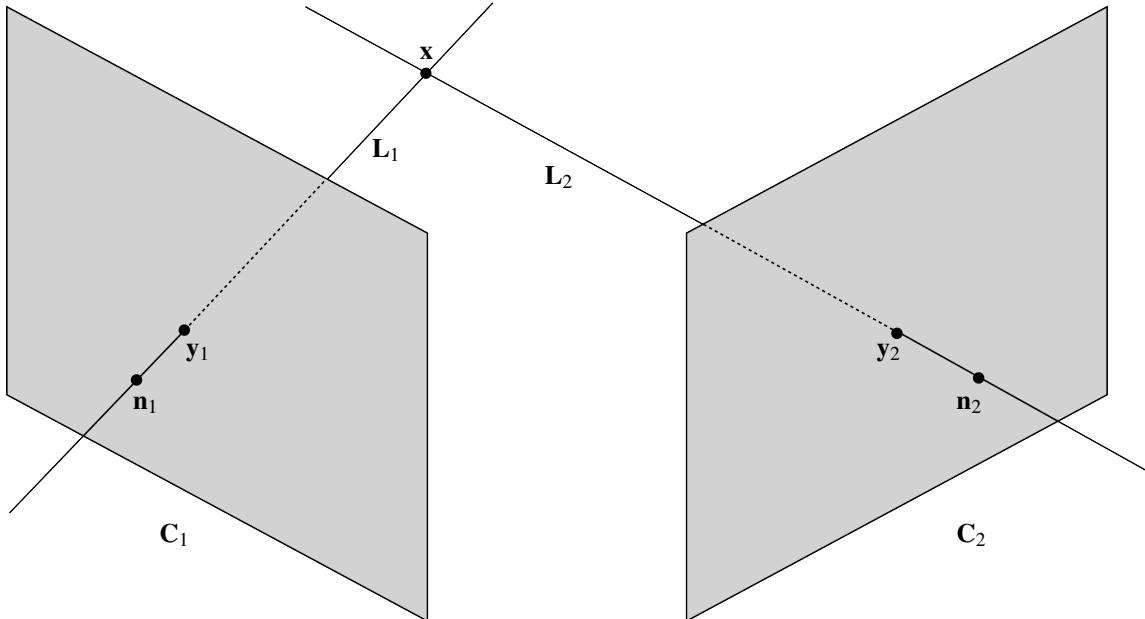


Figure 10.6: Triangulation of a 3D point \mathbf{x} from two image points \mathbf{y}_1 and \mathbf{y}_2 .

The triangulation problem is illustrated geometrically in Figure 10.6. To each of the two image points, \mathbf{y}_1 and \mathbf{y}_2 , there is a projection line, \mathbf{L}_1 and \mathbf{L}_2 , which pass through the corresponding image point and camera center. Since the two image points are in correspondence, the two projection lines intersect at the 3D point \mathbf{x} . If we can determine the two projection lines it should then be possible to work out their intersection. Although this appears relatively straightforward from the outset, there are complications.

To begin with, the approach of determining \mathbf{x} as the intersection of the two projection lines only works in principle. Any practical application implies that there is some amount of noise involved in the coordinates of our image points. Regardless of whether the camera matrices are derived from a calibration process or from the fundamental matrix, where the latter is estimated from noisy data, we expect also some amount of inaccuracy in the camera matrices. As a consequence, there is some noise also on the two projection lines and in general, therefore, they do not intersect. Triangulation then amounts to an estimation problem, where we determine the 3D point \mathbf{x} that optimally fits our data in the form of two image points. As we have seen, estimation problems can be formulated in many different ways, for example in terms of either algebraic or geometric errors. We return to this issue in Section 16.1, where some of the more common approaches to this estimation problem are presented.

10.4.1 Ambiguities of the world coordinate system

The image coordinates that we use are assumed to be the result of projecting 3D points through one or several cameras, where they are observed as coordinates relative to well-defined coordinate systems in the images. The 3D points, however, refer to a world coordinate system that may or may not be well-defined. What we can assume is that there exists a well-defined, Cartesian coordinate system of the 3D space, a world coordinate system, and that the 3D coordinates that we want to determine in triangulation refer to this “true” world coordinate system.

In some cases, we can assume that the camera matrices are fully calibrated, i.e., they are known both in terms of internal and external parameters, and the external parameters refer to the true 3D world coordinate system. In such a case, we can reconstruct true 3D coordinates, i.e., they refer to the designed world coordinate system. In practice, however, this situation is not very common. What is possible, then, is to reconstruct 3D coordinates that are correct but only after some unknown transformation has been applied. Depending on the character of this transformation, we end up in different types of reconstructions.

Euclidean reconstruction

In the case that we know the internal calibration of the cameras, it may still be possible that we know the relative pose of the cameras. If the relative pose is known, we can triangulate 3D points relative to a world coordinate system that is rigidly transformed relative to the “true” coordinate system. A typical choice is to use the camera centered 3D coordinate system of one of the cameras as world coordinate system. We refer to this type of reconstruction as a *Euclidean reconstruction* to reflect the fact that there is an unknown Euclidean transformation (a rigid transformation) between the coordinate system used in the reconstruction and the true coordinate system.

Similarity reconstruction

A more common situation is that we know the relative pose between the cameras, but only up to an unknown scaling of the translation. The motivation behind this case is discussed more in detail in Section 10.5, but it implies that also the absolute scale in 3D space is unknown. The unknown scale can often be resolved by setting some specific distance in 3D space to have unit length. For example, we can define the baseline between two cameras as having unit length. As a consequence, there is a similarity transformation (rigid transformation and scaling) between the world coordinate system that is used in the reconstruction and the true coordinate system. We refer to this as *similarity reconstruction*.

Projective reconstruction

In the case of uncalibrated epipolar geometry, we may determine the fundamental matrix between two views, but the corresponding camera matrices can only be determined up to an unknown homography transformation (projective transformation) of 3D space, as discussed in Section 10.3.4. This means that if we want to do reconstruction in the uncalibrated case, and unless additional information is available, the reconstruction can only be done relative to a world coordinate system that is transformed by an unknown projective transformation relative to the true

coordinate system. We refer to this type of reconstruction as *projective reconstruction*. This implies that the geometry of the points that result from a projective reconstruction may be more or less deformed: angles, distances, areas, volumes can be significantly different from a Euclidean reconstruction, but it is also possible that lines or planes that are parallel in a Euclidean reconstruction do not remain parallel in the projective reconstruction. As a consequence, proper points in the correct coordinate system may end up as points at infinity in the projective reconstruction, and vice versa.

This character of a projective reconstruction may be a problem, in particular if we are interested in the quality of the 3D reconstruction. But if we are mainly interested in the projection of 3D points into two or more cameras, the specific coordinate system of the 3D space is of no importance. As long as all camera matrices refer to one and the same world coordinate system, the projected image coordinates behave as expected.

10.5 Calibrated epipolar geometry

There are two ways we can use epipolar geometry: the uncalibrated case and the calibrated case. In the uncalibrated case we have no explicit knowledge of the two stereo cameras. Instead we can estimate a fundamental matrix \mathbf{F} from a set of corresponding points, using methods that will be developed in Section 16.2. In the calibrated case, we know the internal parameters of each camera, the \mathbf{K} matrix discussed in Section 8.3.3, often as a result of a calibration process, described in Chapter 18. We may even have the external parameters of each camera, i.e., also \mathbf{R} and $\mathbf{\bar{t}}$ in Section 8.3.3. \mathbf{F} can then be computed directly from the camera matrices, Equation (10.2.1).

The difference between the two cases is not limited to how \mathbf{F} is determined, and it is not necessary to make a full calibration of the two cameras. If \mathbf{K} is known, each camera matrix can be represented as a normalized camera, Section 8.2.1, which only accounts for the external parameters $\mathbf{R}, \mathbf{\bar{t}}$. In general, image coordinates are often represented relative to a pixel-based coordinate system. But when \mathbf{K} is known, these coordinates can be transformed to a normalized coordinate system, which removes the effect of the internal parameters. As a result, we can define a simpler type of epipolar geometry, which only depends on the relative rotation and translation (pose) of the two cameras.

Of particular interest is the epipolar constraint, which in the calibrated case is represented by the *essential matrix*, denoted as \mathbf{E} . The essential matrix is a special case of fundamental matrix, adapted to the calibrated case where image coordinates refer to the normalized coordinates system. While \mathbf{F} in the uncalibrated case can be associated to an infinite set of camera matrices, see Section 10.3.4, in the calibrated case each \mathbf{E} corresponds to only one principal configurations of the two cameras. Hence, the calibrated case simplifies the problem of determining the relative pose between two cameras.

Precursors to the essential matrix have been described within the photogrammetry vision already in the early 1900s. These remained unknown to the computer community field which, instead, base its work on Longuet-Higgins [45]. However, it took about a decade before the more general form, the fundamental matrix, came into fashion.

10.5.1 Normalized stereo cameras

A general pinhole camera can be described as a normalized camera combined with an affine transformation that converts C-normalized image coordinates to pixel coordinates, see Section 8.3. In the case of stereo cameras, each with an absolute pose given by the rigid transformations $(\mathbf{R}_1, \mathbf{\bar{t}}_1)$ and $(\mathbf{R}_2, \mathbf{\bar{t}}_2)$, respectively, and with internal parameters \mathbf{K}_1 and \mathbf{K}_2 , the two camera matrices can thus be expressed as

$$\mathbf{K}_1(\mathbf{R}_1 | \mathbf{\bar{t}}_1), \quad \text{and} \quad \mathbf{K}_2(\mathbf{R}_2 | \mathbf{\bar{t}}_2). \quad (10.45)$$

The image coordinates produced by these cameras are referred to as pixel coordinates, were the length unit typically is defined by the pixel-to-pixel distance in the image and the origin is located at the upper left corner.

If both cameras are normalized, this corresponds to $\mathbf{K}_1 = \mathbf{K}_2 = \mathbf{I}$. In practice, we cannot assume that our cameras are normalized, but in some applications we can assume that \mathbf{K}_1 and \mathbf{K}_2 are known, for example from a calibration of both cameras. In this case, in each image we can transform the pixel coordinates, \mathbf{y}_{1p} and \mathbf{y}_{2p} , to C-normalized coordinates, \mathbf{y}_{1n} and \mathbf{y}_{2n} :

$$\mathbf{y}_{1n} \sim \mathbf{K}_1^{-1} \mathbf{y}_{1p}, \quad \mathbf{y}_{2n} \sim \mathbf{K}_2^{-1} \mathbf{y}_{2p}. \quad (10.46)$$

These C-normalized image coordinates correspond to what would have been produced by the normalized cameras

$$\mathbf{C}'_1 \sim (\mathbf{R}_1 | \bar{\mathbf{t}}_1), \quad \mathbf{C}'_2 \sim (\mathbf{R}_2 | \bar{\mathbf{t}}_2). \quad (10.47)$$

In accordance with the result in Section 10.3.1, the epipolar geometry related to a pair of stereo cameras is independent of a homography transformation of 3D space, we can therefore simplify the description of the two normalized cameras, by assuming that the world coordinate system is aligned with the camera centered coordinate system of the first camera:

$$\mathbf{C}_1 \sim (\mathbf{I} | \mathbf{0}), \quad \mathbf{C}_2 \sim (\mathbf{R} | \bar{\mathbf{t}}), \quad (10.48)$$

where

$$\mathbf{R} = \mathbf{R}_2 \mathbf{R}_1^\top, \quad \bar{\mathbf{t}} = \bar{\mathbf{t}}_2 - \mathbf{R}_2 \mathbf{R}_1^\top \bar{\mathbf{t}}_1. \quad (10.49)$$

As a consequence of this description of the two cameras, it follows that the epipolar geometry of two normalized cameras is only dependent of the relative rigid transformation $(\mathbf{R}, \bar{\mathbf{t}})$, from the first to the second camera.

10.5.2 The essential matrix

The epipolar constraint in Equation (10.7) does not make any particular assumption about what coordinate system is used to described the coordinates of points in the two images. It simply assumes that there is some coordinate system for each of the two image planes, given by the internal parameters in \mathbf{K}_1 and \mathbf{K}_2 , respectively. If we want to express the fundamental matrix explicitly in C-normalized image coordinates, \mathbf{y}_{1n} and \mathbf{y}_{2n} , we must then transform the matrix \mathbf{F} , related to the pixel coordinate systems of the two images, to a new matrix \mathbf{E} that refers to the C-normalized coordinates:

$$\mathbf{E} \sim \mathbf{K}_1^\top \mathbf{F} \mathbf{K}_2. \quad (10.50)$$

This is done in accordance with the results in Section 10.3.2, based on Equation (10.46). The matrix \mathbf{E} then generates an epipolar constraint of the form

$$0 = \mathbf{y}_{1n}^\top \mathbf{E} \mathbf{y}_{2n}, \quad (10.51)$$

if \mathbf{y}_{1n} and \mathbf{y}_{2n} are corresponding points. So far, this means that \mathbf{E} is nothing but a sort of fundamental matrix, one that refers explicitly to the case of C-normalized image coordinates.

If the internal camera calibrations are available, we can compute \mathbf{E} from \mathbf{F} using Equation (10.50), but how is \mathbf{E} related to the two normalized cameras in Equation (10.48)? Since \mathbf{E} is just a particular instance of a fundamental matrix, we can compute it from the general expression in Equation (10.13), using the cameras in Equation (10.48). These two cameras have a camera center \mathbf{n}_1 and epipole \mathbf{e}_{21} given as

$$\mathbf{n}_1 \sim \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}, \quad \text{and} \quad \mathbf{e}_{21} \sim \mathbf{C}_2 \mathbf{n}_1 \sim \bar{\mathbf{t}}. \quad (10.52)$$

Thus, \mathbf{E} is computed as

$$\mathbf{E} \sim \mathbf{C}_1^{+\top} \mathbf{C}_2^\top [\mathbf{e}_{21}]_\times = (\mathbf{I} | \mathbf{0}) \begin{pmatrix} \mathbf{R}^\top \\ \bar{\mathbf{t}}^\top \end{pmatrix} [\bar{\mathbf{t}}]_\times = \mathbf{R}^\top [\bar{\mathbf{t}}]_\times. \quad (10.53)$$

In the following, we refer to \mathbf{E} as the *essential matrix* related to the two normalized cameras $\mathbf{C}_1, \mathbf{C}_2$. An alternative, but equivalent, form for \mathbf{E} is given as:

$$\mathbf{E} \sim \mathbf{R}^\top [\bar{\mathbf{t}}]_\times \stackrel{\text{Equation (3.152)}}{=} \mathbf{R}^\top [\mathbf{R} \mathbf{R}^\top \bar{\mathbf{t}}]_\times \mathbf{R} \mathbf{R}^\top = [\mathbf{R}^\top \bar{\mathbf{t}}]_\times \mathbf{R}^\top. \quad (10.54)$$

The construction of \mathbf{E} from the normalized cameras deserves some contemplation. First, it means that the essential matrix depends only on the relative pose of the two cameras, i.e., the rigid transformation of the second camera relative to the first camera. Second, as a projective element, \mathbf{E} is independent of the length and direction of the translation represented by $\bar{\mathbf{t}}$. This implies that \mathbf{E} depends on the three parameters that are necessary to represent an arbitrary $\mathbf{R} \in SO(3)$, and on two additional parameters that describe the *direction* of $\bar{\mathbf{t}} \in \mathbb{R}^3$. In short, the essential matrix has only 5 degrees of freedom, compared to 7 degrees of freedom for the fundamental matrix.

The reduced degrees of freedom for the essential matrix implies that the parameterization of \mathbf{F} described in Section 10.3.3 is not really adequate for \mathbf{E} . Furthermore, there must be additional internal constraints beside

$\det \mathbf{F} = 0$ that are applicable for \mathbf{E} , to account for the fewer degrees of freedom. To address the last issue, consider the symmetric matrix $\mathbf{E}^\top \mathbf{E}$:

$$\mathbf{E}^\top \mathbf{E} \sim [\bar{\mathbf{t}}]_\times \mathbf{R} \mathbf{R}^\top [\bar{\mathbf{t}}]_\times = [\bar{\mathbf{t}}]_\times^2 \sim \mathbf{I} - \hat{\mathbf{t}}\hat{\mathbf{t}}^\top, \quad (10.55)$$

where $\bar{\mathbf{t}} = |\bar{\mathbf{t}}| \cdot \hat{\mathbf{t}}$, and $\hat{\mathbf{t}} \in S^2$. This implies that the 3×3 matrix $\mathbf{E}^\top \mathbf{E}$ is proportional to the projection operator onto the orthogonal complement of the 1-dimensional subspace spanned by $\hat{\mathbf{t}}$. Consequently, $\mathbf{E}^\top \mathbf{E}$ has two eigenvalues that are equal and one that vanish. The eigenvalues of $\mathbf{E}^\top \mathbf{E}$ equal the squares of the singular values of \mathbf{E} , and it follows that \mathbf{E} has two singular values that are equal and one that vanish:

$$\mathbf{E} = \mathbf{U} \begin{pmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{V}^\top. \quad (10.56)$$

Since \mathbf{E} is a projective element, we can set $\sigma = 1$, and \mathbf{E} is then determined only by $\mathbf{U}, \mathbf{V} \in SO(3)$. We need $3 + 3 = 6$ degrees of freedom to specify the two rotation matrices, \mathbf{U} and \mathbf{V} , whose columns represent the left and right singular vectors of \mathbf{E} , respectively. However, since \mathbf{E} has two singular values that are equal, it must be the case any linear combination of the first two columns in \mathbf{U} is, again, a left singular vector with singular value σ . Similarly, any linear combination of the first two columns in \mathbf{V} forms a right singular vector corresponding to the same singular value. This means that \mathbf{U} and \mathbf{V} are not uniquely determined: for a given \mathbf{U}, \mathbf{V} that hold the left and right singular vectors of \mathbf{E} , we can rotate the first two singular vectors in \mathbf{U} in the orthogonal complement of the third left singular vector. This corresponds to a rotation also of the first two right singular vectors in \mathbf{V} in the orthogonal complement of the third right singular vector. In summary: there is one redundant degree of freedom when we determine $\mathbf{U}, \mathbf{V} \in SO(3)$, related to this rotation in a 2-dimensional subspace. Again, this leads to \mathbf{E} having $3 + 3 - 1 = 5$ degrees of freedom.

The fact that \mathbf{E} has two identical singular values leads to the following equations:

$$\mathbf{E} \mathbf{E}^\top \mathbf{E} = \sigma^2 \mathbf{E}, \quad \text{trace}(\mathbf{E}^\top \mathbf{E}) \mathbf{E} = 2\sigma^2 \mathbf{E}, \quad (10.57)$$

which we can summarize as an *internal constraint* for the essential matrix:

$$\mathbf{E} \mathbf{E}^\top \mathbf{E} - \frac{1}{2} \text{trace}(\mathbf{E}^\top \mathbf{E}) \mathbf{E} = \mathbf{0}. \quad (10.58)$$

Both sides of this equation are a 3×3 matrix, which means that Equation (10.58) represents 9 internal constraints in \mathbf{E} . Formally, there should only be 3 internal constraints to account for the 5 degrees of freedom in \mathbf{E} , which means that these 9 constraints are not algebraically independent. Since \mathbf{E} in principle is a fundamental matrix relative to a particular set of image coordinates, it must be the case that $\det \mathbf{E} = 0$, and this constraint is embedded into the 9 constraints in Equation (10.58).

10.5.3 Camera poses from \mathbf{E}

The main reason for using the essential matrix, instead of fundamental matrix, in the calibrated case is that from \mathbf{E} it is straightforward to determine a pair of *normalized* camera matrices that are consistent with \mathbf{E} . In fact, there are four principal pairs of normalized camera matrices consistent with any \mathbf{E} , i.e., which satisfy Equation (10.58). Each of them represents a relative pose of the stereo cameras that produce the epipolar geometry represented by this \mathbf{E} .

As before we assume that the world coordinate system is aligned with the first camera, so the cameras are given as in Equation (10.48). The essential matrix \mathbf{E} is in this case given in accordance with Equation (10.53). The question is now: if \mathbf{E} is given, typically it is estimated from a set of C-normalized image points, what can we say about \mathbf{R} and $\bar{\mathbf{t}}$?

An SVD of \mathbf{E} leads to

$$\mathbf{E} = \mathbf{U} \mathbf{S} \mathbf{V}^\top, \quad (10.59)$$

where $\mathbf{U}, \mathbf{V} \in SO(3)$ and \mathbf{S} is diagonal with two equal and positive singular values, and a third singular value equal to zero, as in Equation (10.56). To get this particular result, we apply the special version of SVD discussed in

Toolbox Section 8.2.7, and negate the sign of the third column in \mathbf{U} or \mathbf{V} if the original SVD gives either of them a negative determinant. The epipole \mathbf{e}_{21} is a null vector of \mathbf{E} :

$$\mathbf{0} = \mathbf{E}\mathbf{e}_{21} = \mathbf{U} \underbrace{\begin{pmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{pmatrix}}_S \underbrace{\begin{pmatrix} - & \hat{\mathbf{v}}_1^\top & - \\ - & \hat{\mathbf{v}}_2^\top & - \\ - & \hat{\mathbf{v}}_3^\top & - \end{pmatrix}}_{\mathbf{V}^\top} \mathbf{e}_{21}, \quad (10.60)$$

where $\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \hat{\mathbf{v}}_3$ are the columns of \mathbf{V} . This implies that $\mathbf{e}_{21} \sim \hat{\mathbf{v}}_3$. Furthermore, Equation (10.53) leads to

$$\mathbf{0} = \mathbf{E}\mathbf{e}_{21} \sim \mathbf{R}^\top [\bar{\mathbf{t}}]_\times \mathbf{e}_{21}, \quad (10.61)$$

from which follows that $\mathbf{e}_{21} \sim \bar{\mathbf{t}}$. Consequently, the translation part of the rigid transformation between the two cameras is given as

$$\bar{\mathbf{t}} \sim \hat{\mathbf{v}}_3. \quad (10.62)$$

From this expression alone, we can only determine the orientation of the translation vector $\bar{\mathbf{t}}$, not its direction or length.

To factor out also the rotation component \mathbf{R} , we need some intermediate results:

$$[\bar{\mathbf{t}}]_\times \sim [\hat{\mathbf{v}}_3]_\times = [\mathbf{V}\hat{\mathbf{b}}_3]_\times = \underbrace{\mathbf{V}\mathbf{V}^\top}_I [\mathbf{V}\hat{\mathbf{b}}_3]_\times \underbrace{\mathbf{V}\mathbf{V}^\top}_I = \mathbf{V}[\hat{\mathbf{b}}_3]_\times \mathbf{V}^\top, \quad \text{where } \hat{\mathbf{b}}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad (10.63)$$

and

$$\mathbf{S} \sim \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \mathbf{W}[\hat{\mathbf{b}}_3]_\times = -\mathbf{W}^\top[\hat{\mathbf{b}}_3]_\times, \quad \text{where } \mathbf{W} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \in SO(3). \quad (10.64)$$

We can express \mathbf{E} either as in Equation (10.53) or in terms of its SVD:

$$\mathbf{E} \sim \mathbf{R}^\top [\bar{\mathbf{t}}]_\times \sim \mathbf{R}^\top \mathbf{V}[\hat{\mathbf{b}}_3]_\times \mathbf{V}^\top, \quad \text{or} \quad \mathbf{E} = \mathbf{U}\mathbf{S}\mathbf{V}^\top \sim \mathbf{U}\mathbf{W}^\top[\hat{\mathbf{b}}_3]_\times \mathbf{V}^\top. \quad (10.65)$$

These two expansions of \mathbf{E} are equivalent only if

$$\mathbf{R}^\top \mathbf{V} = \mathbf{U}\mathbf{W}^\top, \quad \Rightarrow \quad \mathbf{R} = \mathbf{V}\mathbf{W}\mathbf{U}^\top. \quad (10.66)$$

Since $\mathbf{V}, \mathbf{W}, \mathbf{U} \in SO(3)$, this guarantees that $\mathbf{R} \in SO(3)$.

As is shown in Equation (10.64) we can change \mathbf{W} to \mathbf{W}^\top and only change the sign of \mathbf{E} . Consequently, as a projective element, the rigid transformations that define \mathbf{E} can be summarized as:

$$\mathbf{R} = \mathbf{V}\mathbf{W}\mathbf{U}^\top, \quad \text{or} \quad \mathbf{R} = \mathbf{V}\mathbf{W}^\top\mathbf{U}^\top, \quad \text{and} \quad \bar{\mathbf{t}} \sim \hat{\mathbf{v}}_3. \quad (10.67)$$

Since $\bar{\mathbf{t}}$ cannot be determined in terms of sign or length relative to \mathbf{v}_3 , there is an infinite set of rigid transformations $\mathbf{R}, \bar{\mathbf{t}}$ that are consistent with any given \mathbf{E} . In practice, these can be condensed into a set of four principal rigid transformations:

1. $\mathbf{R} = \mathbf{V}\mathbf{W}\mathbf{U}^\top, \hat{\mathbf{t}} = +\hat{\mathbf{v}}_3,$
 2. $\mathbf{R}' = \mathbf{V}\mathbf{W}^\top\mathbf{U}^\top, \hat{\mathbf{t}} = +\hat{\mathbf{v}}_3,$
 3. $\mathbf{R}' = \mathbf{V}\mathbf{W}\mathbf{U}^\top, \hat{\mathbf{t}} = -\hat{\mathbf{v}}_3,$
 4. $\mathbf{R} = \mathbf{V}\mathbf{W}^\top\mathbf{U}^\top, \hat{\mathbf{t}} = -\hat{\mathbf{v}}_3,$
- (10.68)

where the distance of the translation part here is normalized to unit length. Any other rigid transformation that is consistent with \mathbf{E} can be obtained by picking one of the four transformations in Equation (10.68) and by scaling the translation component with a positive real number.

Although all of the four principal rigid transformations in Equation (10.68) are consistent with \mathbf{E} from an algebraic point of view, as described by Equation (10.53), only one of them is realistic from a geometrical point of view. In any practical case, we assume that the stereo cameras have a relative pose such that there exists a region of 3D space that lies in the field of view of both cameras. In particular, this means that any pair of corresponding points that can be established from two images must refer to a 3D point that lies *in front of* both cameras. This implies that the third coordinate of the 3D point relative to the camera centered coordinate system always is positive for *both* cameras. In general, this is the case only for one of the four relative poses in Equation (10.68).

To see this, we determine the difference in the rotation generated by \mathbf{R} and by \mathbf{R}' , given as

$$\mathbf{R}(\mathbf{R}')^\top = (\mathbf{V}\mathbf{W}\mathbf{U}^\top) (\mathbf{V}\mathbf{W}^\top\mathbf{U}^\top)^\top = \mathbf{V}\mathbf{W}^2\mathbf{V}^\top = \mathbf{V} \underbrace{\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{W}^2} \mathbf{V}^\top. \quad (10.69)$$

The right-hand side is a 180° rotation about the axis $\hat{\mathbf{v}}_3$. This result characterizes \mathbf{R} and \mathbf{R}' as a pair of *twisted rotations*, see Section 11.7.2. As a consequence, the pose of the second camera, represented by \mathbf{R}' , is the same as if it was determined from \mathbf{R} , but with a 180° rotation about the translation vector $\hat{\mathbf{t}}$. Given that $\hat{\mathbf{t}} \sim \mathbf{v}_3$, we can schematically illustrate the four different principal rigid transformations as in Figure 10.7. In all four cases, we assume that \mathbf{y}_1 and \mathbf{y}_2 are a pair of corresponding image points, and determine the location of the 3D point \mathbf{x} that projects onto these image points. In three of the cases, the 3D point lies “behind” the image plane one or both of the cameras, and only in one of the cases is it “in front of” both cameras.

Which of the four cases that represents the single geometrically feasible camera pose can thus be determined from a single pair of corresponding points. To do so, we compute all four of the relative camera poses in Equation (10.68) and take a pair of corresponding points, represented by their C-normalized coordinates \mathbf{y}_{1n} and \mathbf{y}_2 , and triangulate \mathbf{x} based on the 4 pairs of cameras. For each of the four 3D positions of \mathbf{x} , we determine its third coordinate in the camera centered coordinate systems of *both* cameras. Both third coordinates produced this way should be positive for the pose that we seek. This method of determining the relative camera pose from an estimated essential matrix is summarized in Algorithm 10.3.

10.5.4 Transformations of the image coordinates

Section 10.3.2 discusses what happens with the fundamental matrix if the image coordinates are modified by some transformations in each of the two images. A similar discussion can be applied also for an essential matrix, but since

Algorithm 10.3: Determine a relative camera pose $(\mathbf{R}, \hat{\mathbf{t}})$ that is consistent with \mathbf{E}

Input: An essential matrix \mathbf{E} .

Input: A pair of corresponding points $\mathbf{y}_1, \mathbf{y}_2$, in C-normalized coordinates.

Output: A relative camera pose $(\mathbf{R}, \hat{\mathbf{t}})$ that is consistent with \mathbf{E} , and for which the 3D point that corresponds to $\mathbf{y}_1, \mathbf{y}_2$ lies in front of both cameras.

1 Compute SSVD of $\mathbf{E} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$, Toolbox Algorithm 1 on page 109. This guarantees $\mathbf{U}, \mathbf{V} \in SO(3)$

2 From \mathbf{W} in Equation (10.64), and \mathbf{U}, \mathbf{V} , compute the 4 principal camera poses in Equation (10.68)

3 **foreach** of the 4 poses $(\mathbf{R}, \hat{\mathbf{t}})$ **do**

4 Triangulate the 3D point $\bar{\mathbf{x}}$ corresponding to $\mathbf{y}_1, \mathbf{y}_2$

5 $\bar{\mathbf{x}} = (x_1, x_2, x_3)$ relative to the camera centered coordinates system of the first camera

6 Compute the same point in the camera centered coordinate system of the second camera:

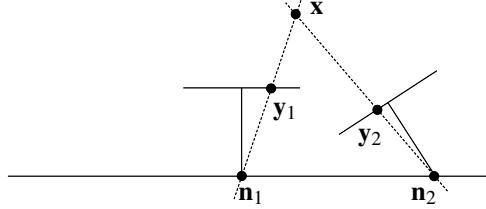
$$\bar{\mathbf{x}}' = \mathbf{R}\bar{\mathbf{x}} + \hat{\mathbf{t}} = (x'_1, x'_2, x'_3)$$

7 Return $(\mathbf{R}, \hat{\mathbf{t}})$ if $x_3 > 0$ and $x'_3 > 0$

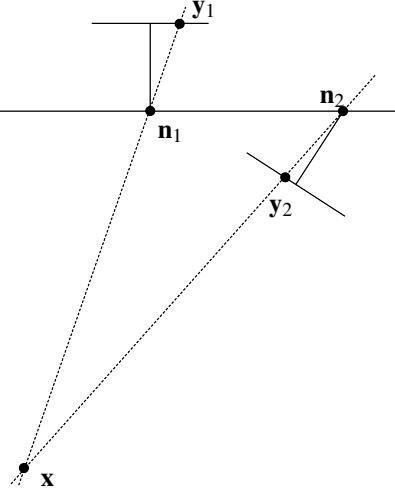
8 /* This condition happens exactly for one of the four poses */

9 **end**

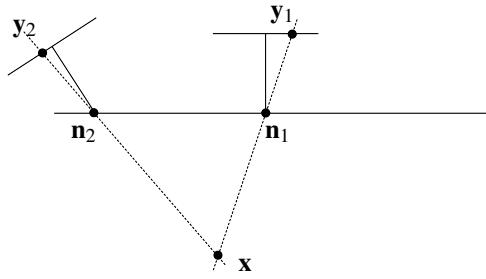
Case 1): \mathbf{R} and $\bar{\mathbf{t}} = +\hat{\mathbf{v}}_3$.



Case 2): \mathbf{R}' and $\bar{\mathbf{t}} = +\hat{\mathbf{v}}_3$.



Case 3): \mathbf{R} and $\bar{\mathbf{t}} = -\hat{\mathbf{v}}_3$.



Case 4): \mathbf{R}' and $\bar{\mathbf{t}} = -\hat{\mathbf{v}}_3$.

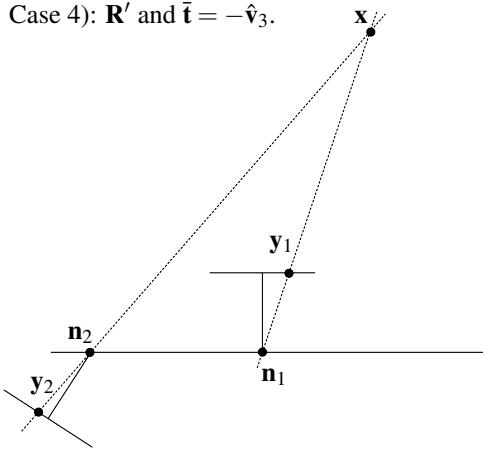


Figure 10.7: An illustration of the four principal rigid transformation between two cameras that are consistent with a given \mathbf{E} .

an essential matrix is a particular type of fundamental matrix we need to be careful about what transformations are applied to the image coordinates.

In Section 9.3 it is shown that if a camera rotates about its center, there is a corresponding transformation of its image coordinates in terms of a homography \mathbf{H} , Equation (9.25), which depends on the rotation and on the internal calibration in \mathbf{K} . We are now dealing with normalized cameras, which means that $\mathbf{K} = \mathbf{I}$, and the homography that transforms image coordinates is then given as $\mathbf{H} = \mathbf{R}_0$ – the rotation that transforms the 3D camera coordinate system. Notice that $\mathbf{R}_0 \in SO(3)$, i.e., it is a 3D rotation, but it appears here in the form of a 2D homography that transforms homogeneous image coordinates. Such a homography includes 2D rotations in the image plane, but also other more general 2D transformations that cannot be characterized in any simpler way than: a transformation in the image plane corresponding to a general 3D rotation of the camera about its center.

A consequence of this observation is that even if some specific \mathbf{E} refers to particular pair of normalized cameras, with a particular relative pose, $(\mathbf{R}, \bar{\mathbf{t}})$, it is perfectly valid to consider the epipolar geometry that appears if one, or both, of the cameras is rotated about its center. In accordance with the results that are mentioned above, such a rotation implies that the normalized image coordinates of an image that referred to a camera that has been rotated, are transformed by a homography \mathbf{R}_0 . Let \mathbf{R}_1 and \mathbf{R}_2 rotate the first and second camera, respectively. This means that \mathbf{R}_1 and \mathbf{R}_2 also transform the corresponding homogeneous image coordinates, as homographies. The epipolar

constraint for the transformed images is then represented by the transformed essential matrix

$$\mathbf{E}' \sim \mathbf{R}_1^{-\top} \mathbf{E} \mathbf{R}_2^{-1} = \mathbf{R}_1 \mathbf{E} \mathbf{R}_2^{\top}. \quad (10.70)$$

If \mathbf{E} satisfies the internal constraints in Equation (10.58) for an essential matrix, then so does \mathbf{E}' .

10.5.5 Minimal parameterization of \mathbf{E}

A straightforward approach to parameterize \mathbf{E} is in terms of a rotation \mathbf{R} and unit norm translation $\hat{\mathbf{t}}$, where

$$\mathbf{E} = \mathbf{R}^{\top} [\hat{\mathbf{t}}]_{\times}. \quad (10.71)$$

$\mathbf{R} \in SO(3)$ has 3 parameters and $\hat{\mathbf{t}} \in S^2$ has 2 more parameters, in total $3 + 2 = 5$. Since \mathbf{E} has 5 degrees of freedom, this constitutes a minimal parameterization of the essential matrix.

10.5.6 Summary of properties related to \mathbf{E}

In the list below, we summarize the facts derived so far about the essential matrix \mathbf{E} . These properties should be compared to those of the fundamental matrix in Section 10.3.6, where some are identical, and some are different or new.

1. The essential matrix refers only to C-normalized image coordinates.
2. \mathbf{E} can be computed directly from the camera matrices as $\mathbf{E} \sim \mathbf{R}^{\top} [\bar{\mathbf{t}}]_{\times}$ or, equivalently, as $\mathbf{E} \sim [\mathbf{R}^{\top} \bar{\mathbf{t}}]_{\times} \mathbf{R}^{\top}$, where $(\mathbf{R}, \bar{\mathbf{t}})$ is the rigid transformation from camera 1 to camera 2.
3. The essential matrix \mathbf{E} related to the fundamental matrix \mathbf{F} , defined for pixel coordinates, as $\mathbf{E} \sim \mathbf{K}_1^{\top} \mathbf{F} \mathbf{K}_2$, or $\mathbf{F} \sim \mathbf{K}_1^{-\top} \mathbf{E} \mathbf{K}_2^{-1}$.
4. \mathbf{E} is a 3×3 matrix of rank 2. As a projective element it has 5 degrees of freedom, since it must satisfy the internal constraint $\mathbf{E} \mathbf{E}^{\top} \mathbf{E} - \frac{1}{2} \text{trace}(\mathbf{E}^{\top} \mathbf{E}) \mathbf{E} = \mathbf{0}$.
5. The left and right null spaces of \mathbf{E} are determined by the epipoles: $\mathbf{E} \mathbf{e}_{21} = \mathbf{E}^{\top} \mathbf{e}_{12} = \mathbf{0}$.
6. \mathbf{E} generates a necessary but not sufficient condition, the epipolar constraint, which must be satisfied for corresponding points from a pair of stereo images: $\mathbf{y}_1^{\top} \mathbf{E} \mathbf{y}_2$. The image coordinates must in this case be C-normalized.
7. \mathbf{E} maps a point in one image to an epipolar line in the other image: $\mathbf{l}_1 \sim \mathbf{E} \mathbf{y}_2, \mathbf{l}_2 \sim \mathbf{E}^{\top} \mathbf{y}_1$. Again, all image coordinates are C-normalized.
8. \mathbf{E} is invariant to transformations of 3D space.
9. An essential matrix \mathbf{E} can be transformed by homographies in each of the two images, represented by rotation matrices \mathbf{R}_1 and \mathbf{R}_2 , to a new essential matrix $\mathbf{E}' \sim \mathbf{R}_1 \mathbf{E} \mathbf{R}_2^{\top}$.
10. Given \mathbf{E} , a pair of normalized camera matrices \mathbf{C}_1 and \mathbf{C}_2 , consistent with \mathbf{E} , can be determined by means of Algorithm 10.3 on page 159.
11. A minimal parameterization of \mathbf{E} (5 parameters) can be implemented in accordance with Section 10.5.5.

Chapter 11

Representations of 3D Rotations

Before you read this chapter, you may want to have a look at Toolbox Section 8.6.2 and Toolbox Section 8.6.4 that describe how $so(3)$ is mapped to $SO(3)$, i.e., how 3D rotation matrices can be parameterized by 3×3 anti-symmetric matrices. Toolbox Section 7.2 presents an overview of quaternions, which are used in Section 11.4.

Rigid transformations is a class of transformations that preserve the distance between any pair of point. As a special case of rigid transformations, rotations have two more properties. (1) There exist fixed points, i.e., points that are not affected by the transformation. (2) They preserve handedness (or chirality). For a 2D rotation, there is a unique fixed point, for 3D rotations the fixed points form a line: the rotation axis. In both 2D and 3D, the exception is the identity rotation that leaves all points unchanged.

In this chapter we discuss various types of algebraic representations of rotations. An *algebraic representation* implies that we use some type of algebraic objects, such as scalars, vectors, or matrices to represent the rotations. To simplify the presentation, the algebraic representations discussed here always rotate with the origin as a fixed point. This assumption enables us to describe a rotation as a linear transformation \mathbf{R} , a rotation matrix, which acts on the Cartesian coordinates of points. We can form more general types of rotation transformations by combining these rotations with translations, as described in Section 4.2 and Section 6.2.

Rotations in 2D can be seen as a special case of 3D rotations, where the rotation takes place relative to a fixed axis, or in a fixed plane. We need only by a single parameter, the rotation angle, to characterize such rotations. So, 2D rotations are relatively simple and straightforward to represent. For example, we can represent a 2D rotation using a 2×2 matrix in $SO(2)$, as in Equation (4.3). We can also use a complex number on the unit circle, $z = e^{i\alpha}$, where the rotation angle α is given as the argument of z . Although there exist different options for the representation of 2D rotations, these do not differ that much and it is mainly a matter of taste which one we use in a practical application. Therefore, 2D representations are not discussed further in this chapter.

In contrast to 2D rotations, $SO(3)$, the group of 3D rotations, has 3 degrees of freedom, i.e., to specify an arbitrary rotation we need 3 independent parameters. This group it is also non-commutative, i.e., applying two 3D rotation in sequence in general gives different results depending on which order they are applied. More importantly, there exists many algebraic representations of 3D rotations, and it sometimes matters a lot which representation we use.

In the following sections, we will present some of the more common algebraic representations for 3D rotations, and also compares them. In some situations, it is natural to represent a 3D rotation as a 3×3 matrix \mathbf{R} that transforms Cartesian coordinates. In other cases, it is more relevant to talk about the same rotation in terms of a rotation axis $\hat{\mathbf{n}}$ and a rotation angle α about the axis. There are also less intuitive ways to represent 3D rotations. Being able to comfortably move back and forth between these representations is a requisite for effective calculations and algebraic manipulations of 3D rotations. Understanding the advantages and limitations of these algebraic representations, and taking them into account in numerical methods is a key to successful implementations of various computations and estimation problems that involve rotations.

• Handedness of coordinate systems is described in Toolbox Section 3.7.1.

• Groups are described in Toolbox Section 2.2.

11.1 Rotation matrices

- The group $SO(3)$ is defined in Toolbox Section 3.7.6

A convenient algebraic representation of rotations in \mathbb{R}^3 is to consider the group $SO(3)$. $SO(3)$ consists of 3×3 rotation matrices, and has the advantage of being unique. To each rotation in \mathbb{R}^3 there is exactly one rotation matrix $\mathbf{R} \in SO(3)$ that implements this rotation as a linear transformation on the Cartesian coordinates of points in \mathbb{R}^3 . This allows us to identify $SO(3)$ with the geometric concept of a rotation about an axis that passes through the origin in \mathbb{R}^3 . In the rest of this chapter we use *rotation* to refer to an element of $SO(3)$. So, instead of algebraic representations of rotations in \mathbb{R}^3 , we can talk about algebraic representations of $SO(3)$.

Let us begin by asking if $SO(3)$ itself is a suitable algebraic representation of 3D rotations. We have already seen that it is unique and that it has a simple interpretation as a coordinate transformation. As such, it appears in many mathematical expressions and derivations, for example describing the rotation part of the external camera parameters discussed in Section 8.3.3. Furthermore, if we want to combine two rotations, \mathbf{R}_1 and \mathbf{R}_2 , by first applying \mathbf{R}_1 to some coordinates and then \mathbf{R}_2 , the result is the rotation matrix $\mathbf{R}_2\mathbf{R}_1$. The computational complexity for computing this product is relative low: 27 multiplications and 18 additions. Finally, we get the inverse of $\mathbf{R} \in SO(3)$ as \mathbf{R}^\top , an operation that has a low computational cost.

Internal constraints

Unfortunately, there are also some disadvantages related to this representation that make it less useful in some situations. First, \mathbf{R} is a 3×3 matrix which means that it is represented by the 9 elements of such a matrix. These elements must satisfy a set of *internal constraints*, specified by:

$$\mathbf{R}^\top \mathbf{R} = \mathbf{I}, \quad \text{and} \quad \det(\mathbf{R}) = 1. \quad (11.1)$$

Consequently, not any 3×3 matrix is a rotation matrix and this observation has some consequences that we may have to deal with if rotation matrices are used in practical calculations. Here we discuss some of them.

For one thing, Equation (11.1) implies that if we change the elements of a matrix in $SO(3)$ in an arbitrary way, the result may not remain in $SO(3)$. In short: a 3×3 matrix has 9 free parameters while $SO(3)$ only has 3 parameters. This difference in number of parameters is formalized in terms of the internal constraints in Equation (11.1). But it is not trivial to describe how to do a modification of $\mathbf{R} \in SO(3)$ so that it stays in $SO(3)$. This observation makes $SO(3)$ less suited as an algebraic representation of $SO(3)$, e.g., when we optimize some function over the set of rotations. This observation is further developed in Section 15.3.

Storing all 9 elements of a rotation matrix is actually not necessary. It is enough to keep only the elements of two columns or rows of the matrix, in total 6 elements. If needed we can compute the missing column or row as a cross product of the remaining two. This reduction of elements, however, does not fix the problem of how to change the remaining 6 elements such that they are the columns or rows of a rotation matrix.

- The vector cross product is defined in Toolbox Section 3.7.2.

Interpolation and averaging

A second difficulty in using $SO(3)$ as a representation of rotations relates to interpolation. Let us assume that we have two rotation matrices, \mathbf{R}_1 and \mathbf{R}_2 . In some applications we can be interested in describing a set of intermediate rotations, a sequence of rotations starting at \mathbf{R}_1 and ending at \mathbf{R}_2 . Intuitively, this corresponds to the idea of *interpolating* a set of rotations between \mathbf{R}_1 and \mathbf{R}_2 , and we may initially want to try something like defining a parameterized set of matrices like

$$\mathbf{R}(\lambda) = \lambda \mathbf{R}_2 + (1 - \lambda) \mathbf{R}_1. \quad (11.2)$$

We get $\mathbf{R}(0) = \mathbf{R}_1$ and $\mathbf{R}(1) = \mathbf{R}_2$, but what about λ between 0 and 1? Unfortunately, $\mathbf{R}(\lambda)$ does not satisfy Equation (11.1) for general values of $\lambda \in \mathbb{R}$. Only in the case that $\mathbf{R}_1 \approx \mathbf{R}_2$ is $\mathbf{R}(\lambda)$ approximately in $SO(3)$ for $\lambda \in [0, 1]$.

A similar observation applies to averaging. Given a set of m rotation matrices $\mathbf{R}_k \in SO(3)$, which all are estimates of the same rotation \mathbf{R} , we expect their mean to be a better estimate of \mathbf{R} . Just computing something like

$$\mathbf{R}_{\text{aver}} = \frac{1}{m} \sum_{k=1}^m \mathbf{R}_k, \quad (11.3)$$

in general does *not* produce $\mathbf{R}_{\text{aver}} \in SO(3)$. At best, \mathbf{R}_{aver} lies close to $SO(3)$ if the variation in the set of rotations is small.

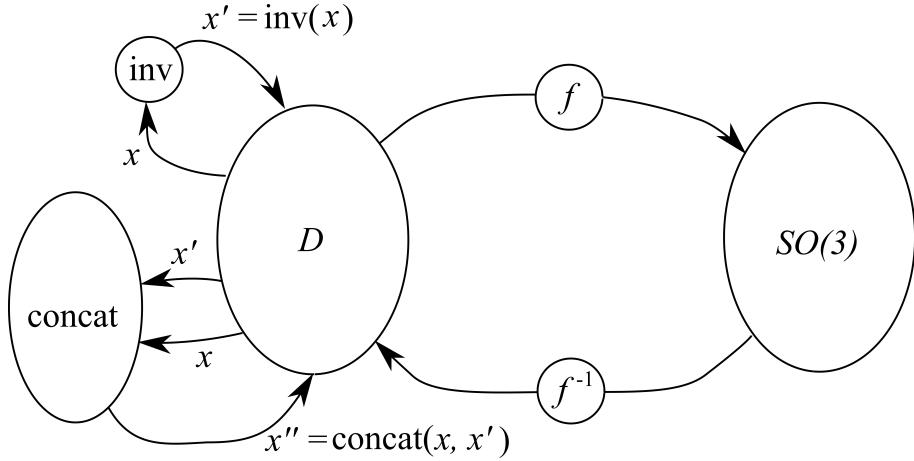


Figure 11.1: A representation of 3D rotations is a function f from some domain D to $SO(3)$. The two functions inv and concat are defined below.

Random rotations

A third issue that can emerge in some applications is that of selecting random rotations. For example, we may want to select random rotations with uniform probability. Clearly, we cannot select a random 3×3 matrix and hope that it satisfies Equation (11.1). And if it was possible to produce a random selection of matrices from $SO(3)$, can we be certain that they are uniformly distributed in $SO(3)$, i.e., any rotation has the same probability as any other? This question can be addressed by choosing alternative representations for $SO(3)$, other than matrices.

Conclusions

In summary: if we want to update $\mathbf{R} \in SO(3)$ to optimize some function over $SO(3)$, the result is a 3×3 matrix but not necessarily an element in $SO(3)$. The same situation appears when we interpolate, or compute averages of rotation matrices, by updating, interpolating, or averaging the individual elements of the matrices. These are some of the reasons why alternative representations of $SO(3)$ may be interesting.

11.1.1 Representations of $SO(3)$

Let us first define more in detail what we mean by an *algebraic representation* of $SO(3)$. Formally, a representation of $SO(3)$ is a function from some domain D to $SO(3)$, $f : D \rightarrow SO(3)$, see Figure 11.1. This means that we can take any $x \in D$ and apply f to this x and we always get $f(x) \in SO(3)$ as a result. The domain D is sometimes also referred to as the *parameter space* of the representation f , since it contains the free parameters of the function f . If $\mathbf{R} = f(x)$ this means that $x \in D$ is a *representation* of \mathbf{R} and that \mathbf{R} is *represented* by $x \in D$. To understand a representation, the inverse function f^{-1} , from $SO(3)$ back to D , is also important.

When a particular representation f is discussed, the following aspects of f are often relevant:

- **Completeness:** A *complete representation* f maps onto $SO(3)$, i.e., $f(D) = SO(3)$. This means that any element in $SO(3)$ is the image of at least one $x \in D$. As we will see, some representations are complete while others are incomplete. In the latter case, there is a small set of specific rotations that cannot be represented by any element in D . This may appear like a severe flaw, but incomplete representations are useful as long as we can avoid the rotations that are missing. This can be done if we are aware of which rotations are missing in an incomplete representation.

• $f(D)$ and “onto” are defined in Toolbox Section 1.5.

- **Uniqueness:** f is *unique representation* if there is exactly one $x \in D$ such that $f(x) = \mathbf{R}$ for every $\mathbf{R} \in f(D)$. As we will see, some representations are unique while others are not. Lack of uniqueness is often not a major problem, but can be an issue if we need to compute f^{-1} . In practice, we can often avoid this problem by restricting D to some subset that makes f^{-1} unique.

• This can also be described as: f is a one-to-one mapping, see Toolbox Section 1.5.

- **Singularities:** A representation f can have a *singularity*, a rotation $\mathbf{R} \in SO(3)$ that behaves in a specific way relative to f . \mathbf{R} is then a singularity if almost all other rotations close to \mathbf{R} do not have this property. For example, representations that are incomplete typically lack representation for a smaller set of singular rotations. Another example are representations that are unique for all $SO(3)$, except for a small set of rotations. These rotations constitute singularities.
- **Closed form inverse rotation:** Given $x \in D$ that represents the rotation $f(x) \in SO(3)$, we are sometimes interested in the inverse rotation $f(x)^\top \in SO(x)$. In general, this rotation is represented by some $x' \in D$, and we may want to be able to determine $x' \in D$ directly from $x \in D$. This means that x' is given as a function $x' = \text{inv}(x) = f^{-1}(f(x)^\top)$, see Figure 11.1. For some representations, it is possible to write $\text{inv}(x)$ as a closed form expression in x , sometimes even a simple expression in x . For other representations, $\text{inv}(x)$ can be a very complicated expression.
- **Closed form concatenation of rotations:** Given that $x, x' \in D$ represent the two rotations $f(x), f(x') \in SO(3)$, the combined rotation $f(x')f(x) \in SO(3)$ is sometimes of interest. In general this rotation is represented by some $x'' \in D$, and we may want to be able to determine $x'' \in D$ directly from $x, x' \in D$, as a function $\text{concat}(x, x') = f^{-1}(f(x')f(x))$, see Figure 11.1. For some representations, it is possible to write $\text{concat}(x, x')$ as a closed form expression, sometimes even a simple expression in x and x' . For other representations, $\text{concat}(x, x')$ can be a complicated expression.
- **Internal constraint:** In some cases, the representation f has a domain D that is a vector space. This means that any vector of this space is a valid x in the sense that it can be mapped to $f(x) \in SO(3)$. In other cases, the representation f uses a domain D that is a subset of a vector space, restricted by some type of internal constraint. Only vectors which meet this constraint can then be mapped to $f(x) \in SO(3)$. In practice, the constraint is often much simpler than Equation (11.1). For example, it may be as simple as the normalization of a vector.

Different representations deal with these aspects in different ways. It is often these differences that makes one representation more suitable than another for a certain application. A thorough understanding of these differences is a key to a successful implementation of computations that includes 3D rotations. In the following sections, we will review some of the more common representations of $SO(3)$, and discuss how they relate to the above aspects. We will see that it is possible to define f on a variety of domains D , consisting of scalars, vectors, or matrices, or even combinations of these objects.

11.2 Axis-angle representation

Besides rotation matrices, the most intuitive representation of a rotation in \mathbb{R}^3 is to specify a rotation axis and an angle that describes how much to rotate about this axis. To do this properly, we need to describe more precisely what it means to rotate a certain angle about a certain axis. One way to do that is to specify the rotation axis as a vector, e.g., a normalized vector $\hat{\mathbf{n}} \in S^2$. We then use the right-hand rule to describe what we mean by a positive rotation angle α . The axis-angle representation is based on using the pair $(\hat{\mathbf{n}}, \alpha)$, a normalized vector $\hat{\mathbf{n}} \in S^2$ and an angle $\alpha \in \mathbb{R}$, and mapping them to $SO(3)$. Therefore, we can initially set the domain $D = S^2 \times \mathbb{R}$ for the axis-angle representation of $SO(3)$. We need 3 scalars to specify $\hat{\mathbf{n}}$, and one more to specify the angle. Hence, there are $3 + 1 = 4$ parameters in this representation, but the constraint $\|\hat{\mathbf{n}}\| = 1$ reduces the degrees of freedom for the whole representation to 3.

Since any 3D rotation can be described with a rotation axis and a rotation angle, the axis-angle representation is complete. But it also has some issues that make it less suitable in some situations. This representation has a singularity for the *identity rotation* $\mathbf{R} = \mathbf{I}$, i.e., when $\alpha = 0$, which leaves all 3D points as they are. In this case, $\hat{\mathbf{n}}$ becomes completely unspecified: any rotation axis is valid for this specific rotation. This singularity may or may not be a problem, depending on the application. For example, we can use this representation to find a rotation that optimizes some function. If we also use a non-linear optimization method, for example described in [54] Chapter 9, the singularity may cause problems in the optimization process. In particular, this happens if the optimal rotation

• The right-hand rule is described in Toolbox Section 3.7.2.

lies at, or close to, the identity rotation.

11.1 The axis-angle representation has a singularity for $\mathbf{R} = \mathbf{I}$, in which case the rotation axis is unspecified.

Disregarding the singularity, the axis-angle representation is not unique unless we restrict D in some way or another relative to the initial domain $S^2 \times \mathbb{R}$. For example, the representation is 2π -periodic in α , and it is also the case that $(\hat{\mathbf{n}}, \alpha)$ represents the same rotation as $(-\hat{\mathbf{n}}, -\alpha)$. By restricting the ranges of $\hat{\mathbf{n}}$ and α in different ways, we can avoid this ambiguity. For example, we can restrict α to the interval $[0, \pi]$. If we want to rotate a larger angle than π , given by $\alpha = \pi + \beta$ relative to the axis $\hat{\mathbf{n}}$, this is the same as rotating $\pi - \beta$ about the axis $-\hat{\mathbf{n}}$. The restriction $\alpha \in [0, \pi]$ makes the axis-angle representation almost unique. A rotation by π about $\hat{\mathbf{n}}$ is still the same as rotating π about $-\hat{\mathbf{n}}$. To overcome this ambiguity, we can divide the unit sphere S^2 into two hemispheres: if $\hat{\mathbf{n}}$ lies in one of the hemispheres then $-\hat{\mathbf{n}}$ lies in the other. With this division made, we let α have the range $[0, \pi]$ for one hemisphere and the range $[0, \pi[$ for the other hemisphere.

11.2 The axis angle representation is not unique. For example, for all rotation axes $\hat{\mathbf{n}}$ and rotation angles α : $\mathbf{R}(\hat{\mathbf{n}}, \alpha) = \mathbf{R}(\hat{\mathbf{n}}, \alpha + 2\pi k) = \mathbf{R}(-\hat{\mathbf{n}}, -\alpha)$.

An alternative restriction is to let $\hat{\mathbf{n}}$ lie on a hemisphere of S^2 , and let $\alpha \in [0, 2\pi]$. This illustrates that there are several choices for a restricted domain that lead to a unique representation. But regardless of which one we choose, there will always exist two rotations that are very similar but are represented by rotation axes pointing in almost opposite directions or by rotations angles that differ by approximately 2π . So, f^{-1} in these cases is discontinuous.

The axis-angle representation has an inverse rotation function, but, unless a restricted domain is used, the representation is not unique and this function can be implemented in different ways. Two simple examples are:

$$\text{inv}(\hat{\mathbf{n}}, \alpha) = (-\hat{\mathbf{n}}, \alpha), \quad \text{alternatively} \quad \text{inv}(\hat{\mathbf{n}}, \alpha) = (\hat{\mathbf{n}}, -\alpha). \quad (11.4)$$

The concatenation operation, however, is non-trivial and beyond the scope of most practical application. It is therefore not discussed here.

11.2.1 Rodrigues' rotation formula

So far, we have not given the function f , which maps $D = S^2 \times \mathbb{R}$ (or some restriction thereof) to $SO(3)$, an explicit form for the axis-angle representation. This function is defined by the *Rodrigues' rotation formula*. As a preliminary result for the derivation of this formula, we will first dissect a rotation in \mathbb{R}^3 into its basic parts. Let $\hat{\mathbf{n}}$ be the rotation axis that defines a positive rotation angle α according to the right-hand rule. Let $\hat{\mathbf{p}}$ and $\hat{\mathbf{q}}$ be two other vectors in \mathbb{R}^3 such that $\hat{\mathbf{p}}, \hat{\mathbf{q}}, \hat{\mathbf{n}}$ form a right-handed ON-basis. Finally, let $\bar{\mathbf{x}}$ be an arbitrary vector in \mathbb{R}^3 . Since $\hat{\mathbf{p}}, \hat{\mathbf{q}}, \hat{\mathbf{n}}$ is a basis, we can write

$$\bar{\mathbf{x}} = c_1 \hat{\mathbf{n}} + c_2 \hat{\mathbf{p}} + c_3 \hat{\mathbf{q}}. \quad (11.5)$$

Since the basis is ON, the coordinates of $\bar{\mathbf{x}}$ relative to the basis vectors are given as

$$c_1 = \hat{\mathbf{n}} \cdot \bar{\mathbf{x}}, \quad c_2 = \hat{\mathbf{p}} \cdot \bar{\mathbf{x}}, \quad c_3 = \hat{\mathbf{q}} \cdot \bar{\mathbf{x}}. \quad (11.6)$$

• Right-handed coordinate systems in \mathbb{R}^3 are defined in Toolbox Section 3.7.1.

The rotation defined by the axis $\hat{\mathbf{n}}$ and the angle α is applied to $\bar{\mathbf{x}}$, and the result is $\bar{\mathbf{x}}' \in \mathbb{R}^3$. The coordinates of $\bar{\mathbf{x}}'$ relative to the same ON-basis are given as

$$c'_1 = c_1, \quad c'_2 = c_2 \cos \alpha - c_3 \sin \alpha, \quad c'_3 = c_3 \cos \alpha + c_2 \sin \alpha, \quad (11.7)$$

or

$$c'_1 = \hat{\mathbf{n}} \cdot \bar{\mathbf{x}}, \quad c'_2 = \cos \alpha (\hat{\mathbf{p}} \cdot \bar{\mathbf{x}}) - \sin \alpha (\hat{\mathbf{q}} \cdot \bar{\mathbf{x}}), \quad c'_3 = \cos \alpha (\hat{\mathbf{q}} \cdot \bar{\mathbf{x}}) + \sin \alpha (\hat{\mathbf{p}} \cdot \bar{\mathbf{x}}). \quad (11.8)$$

The resulting vector $\bar{\mathbf{x}}'$ is then reconstructed as

$$\begin{aligned} \bar{\mathbf{x}}' &= c'_1 \hat{\mathbf{n}} + c'_2 \hat{\mathbf{p}} + c'_3 \hat{\mathbf{q}} = \\ &= \hat{\mathbf{n}} (\hat{\mathbf{n}} \cdot \bar{\mathbf{x}}) + \cos \alpha (\hat{\mathbf{p}} (\hat{\mathbf{p}} \cdot \bar{\mathbf{x}}) + \hat{\mathbf{q}} (\hat{\mathbf{q}} \cdot \bar{\mathbf{x}})) + \sin \alpha (\hat{\mathbf{q}} (\hat{\mathbf{p}} \cdot \bar{\mathbf{x}}) - \hat{\mathbf{p}} (\hat{\mathbf{q}} \cdot \bar{\mathbf{x}})) = \\ &= \left(\hat{\mathbf{n}} \hat{\mathbf{n}}^\top + \cos \alpha (\hat{\mathbf{p}} \hat{\mathbf{p}}^\top + \hat{\mathbf{q}} \hat{\mathbf{q}}^\top) + \sin \alpha (\hat{\mathbf{q}} \hat{\mathbf{p}}^\top - \hat{\mathbf{p}} \hat{\mathbf{q}}^\top) \right) \bar{\mathbf{x}}. \end{aligned} \quad (11.9)$$

Figure 11.2 illustrates the different vectors that are included in this last expression. It leads to a formula for the corresponding rotation matrix \mathbf{R} as a function of $\hat{\mathbf{n}}$ and α :

$$\mathbf{R}(\hat{\mathbf{n}}, \alpha) = \hat{\mathbf{n}}\hat{\mathbf{n}}^\top + \cos \alpha (\hat{\mathbf{p}}\hat{\mathbf{p}}^\top + \hat{\mathbf{q}}\hat{\mathbf{q}}^\top) + \sin \alpha (\hat{\mathbf{q}}\hat{\mathbf{p}}^\top - \hat{\mathbf{p}}\hat{\mathbf{q}}^\top). \quad (11.10)$$

Since $\hat{\mathbf{p}}, \hat{\mathbf{q}}, \hat{\mathbf{n}}$ form an ON-basis for \mathbb{R}^3 , we can write¹

$$\hat{\mathbf{p}}\hat{\mathbf{p}}^\top + \hat{\mathbf{q}}\hat{\mathbf{q}}^\top = \mathbf{I} - \hat{\mathbf{n}}\hat{\mathbf{n}}^\top. \quad (11.11)$$

Together with Equation (3.142), this allow us to rewrite Equation (11.10) as

$$\mathbf{R}(\hat{\mathbf{n}}, \alpha) = \hat{\mathbf{n}}\hat{\mathbf{n}}^\top + \cos \alpha (\mathbf{I} - \hat{\mathbf{n}}\hat{\mathbf{n}}^\top) + \sin \alpha [\hat{\mathbf{n}}]_\times. \quad (11.12)$$

or, alternatively,

$$\mathbf{R}(\hat{\mathbf{n}}, \alpha) = \mathbf{I} + (1 - \cos \alpha)(\hat{\mathbf{n}}\hat{\mathbf{n}}^\top - \mathbf{I}) + \sin \alpha [\hat{\mathbf{n}}]_\times. \quad (11.13)$$

A third expression for $\mathbf{R}(\hat{\mathbf{n}}, \alpha)$ is obtained by applying Equation (3.146) to the last expression:

$$\mathbf{R}(\hat{\mathbf{n}}, \alpha) = \mathbf{I} + (1 - \cos \alpha)[\hat{\mathbf{n}}]^2_\times + \sin \alpha [\hat{\mathbf{n}}]_\times. \quad (11.14)$$

Any of the last three expressions for \mathbf{R} , Equation (11.12), Equation (11.13), or Equation (11.14), are referred to as *Rodrigues' rotation formula*.² They define the function f that maps a rotation axis $\hat{\mathbf{n}}$ and a rotation angle α to the corresponding rotation matrix $\mathbf{R} \in SO(3)$.

11.3 Rodrigues' rotation formula maps the rotation axis $\hat{\mathbf{n}}$ and rotation angle α to the corresponding rotation matrix $\mathbf{R}(\hat{\mathbf{n}}, \alpha)$:

$$\begin{aligned} \mathbf{R}(\hat{\mathbf{n}}, \alpha) &= \hat{\mathbf{n}}\hat{\mathbf{n}}^\top + \cos \alpha (\mathbf{I} - \hat{\mathbf{n}}\hat{\mathbf{n}}^\top) + \sin \alpha [\hat{\mathbf{n}}]_\times \\ &= \cos \alpha \mathbf{I} + (1 - \cos \alpha)\hat{\mathbf{n}}\hat{\mathbf{n}}^\top + \sin \alpha [\hat{\mathbf{n}}]_\times \\ &= \mathbf{I} + (1 - \cos \alpha)(\hat{\mathbf{n}}\hat{\mathbf{n}}^\top - \mathbf{I}) + \sin \alpha [\hat{\mathbf{n}}]_\times \\ &= \mathbf{I} + (1 - \cos \alpha)[\hat{\mathbf{n}}]^2_\times + \sin \alpha [\hat{\mathbf{n}}]_\times \end{aligned}$$

¹Here we are expanding the identity matrix in accordance with Toolbox Section 3.3.7, Equation (3.95).

²Named after the French mathematician Olinde Rodrigues (1795–1851).

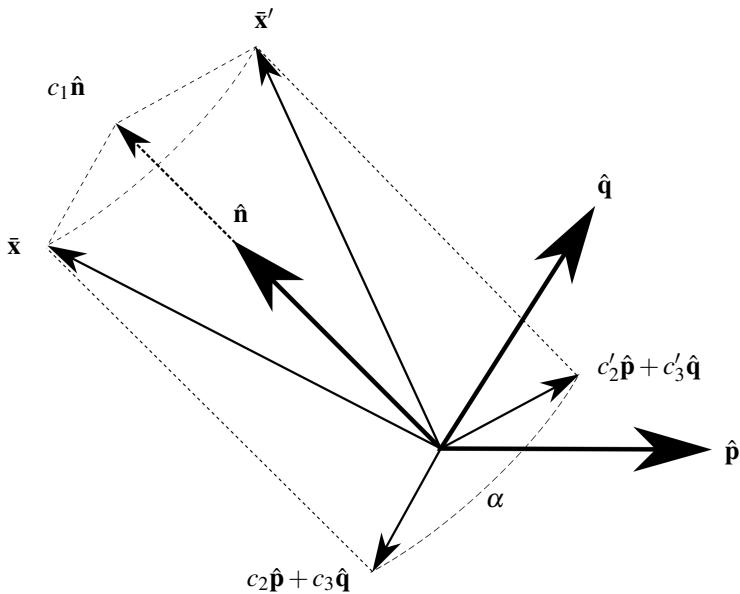


Figure 11.2: The vector $\bar{\mathbf{x}}$ is rotated the angle α about $\hat{\mathbf{n}}$. The result is the vector $\bar{\mathbf{x}}'$.

Small rotations

Rodrigues' rotation formula allows us to express a useful approximation of a rotation matrix for small α . In this case “small” means that we can make the following approximations

$$\sin \alpha \approx \alpha, \quad \text{and} \quad \cos \alpha \approx 1. \quad (11.15)$$

Inserted into Rodrigues' rotation formula, these approximations lead to

$$\mathbf{R}(\hat{\mathbf{n}}, \alpha) \approx \mathbf{I} + \alpha [\hat{\mathbf{n}}]_{\times}. \quad (11.16)$$

So, if \mathbf{R} corresponds to a small rotation angle α , i.e., $\mathbf{R} \approx \mathbf{I}$, \mathbf{R} can be approximated as \mathbf{I} plus an anti-symmetric matrix $\alpha [\hat{\mathbf{n}}]_{\times} \in so(3)$. The latter matrix represents both the rotation axis $\hat{\mathbf{n}}$ and the rotation angle α .

11.2.2 Axis-angle from $SO(3)$

Rodrigues' rotation formula provides the mapping f for the axis-angle representation. What about f^{-1} , the inverse mapping that takes $\mathbf{R} \in SO(3)$ and determines $\hat{\mathbf{n}}$ and α ? In this section we present a computational approach for f^{-1} . We recall that the discussion in the first part of Section 11.2 concluded that there is a singularity for $\mathbf{R} = \mathbf{I}$. We also saw that $\hat{\mathbf{n}}$ and α are not unique unless they are restricted in some way or another.

A first step to finding f^{-1} is to look at the trace of \mathbf{R} . It can be computed from Equation (11.10) as

$$\begin{aligned} \text{trace}(\mathbf{R}) &= \text{trace}\left(\hat{\mathbf{n}}\hat{\mathbf{n}}^{\top} + \cos \alpha (\hat{\mathbf{p}}\hat{\mathbf{p}}^{\top} + \hat{\mathbf{q}}\hat{\mathbf{q}}^{\top}) + \sin \alpha (\hat{\mathbf{q}}\hat{\mathbf{p}}^{\top} - \hat{\mathbf{p}}\hat{\mathbf{q}}^{\top})\right)^3 \\ &= \text{trace}\left(\underbrace{\hat{\mathbf{n}}^{\top}\hat{\mathbf{n}}}_{=1} + \cos \alpha (\underbrace{\hat{\mathbf{p}}^{\top}\hat{\mathbf{p}}}_{=1} + \underbrace{\hat{\mathbf{q}}^{\top}\hat{\mathbf{q}}}_{=1}) + \sin \alpha (\underbrace{\hat{\mathbf{p}}^{\top}\hat{\mathbf{q}}}_{=0} - \underbrace{\hat{\mathbf{q}}^{\top}\hat{\mathbf{p}}}_{=0})\right) = 1 + 2 \cos \alpha. \end{aligned} \quad (11.17)$$

This means that $\cos \alpha$ can be written as

$$\cos \alpha = \frac{\text{trace}(\mathbf{R}) - 1}{2}. \quad (11.18)$$

The next step is to look at the anti-symmetric part of \mathbf{R} , given by⁴

$$\frac{\mathbf{R} - \mathbf{R}^{\top}}{2} = \sin \alpha [\hat{\mathbf{n}}]_{\times}, \quad \Rightarrow \quad \sin \alpha \hat{\mathbf{n}} = [\mathbf{R}]^{\times}. \quad (11.19)$$

From the last two expressions there are multiple ways to proceed, depending on what type of restrictions on $\hat{\mathbf{n}}$ and α we use. For example, since $\|\hat{\mathbf{n}}\| = 1$ it follows that

$$\|[\mathbf{R}]^{\times}\| = \sin \alpha, \quad \text{with } \alpha \in [0, \pi]. \quad (11.20)$$

This together with Equation (11.18) allow us to uniquely determine $\alpha \in [0, \pi]$. Furthermore, combining Equation (11.19) with Equation (11.20) leads to

$$\hat{\mathbf{n}} = \frac{[\mathbf{R}]^{\times}}{\|[\mathbf{R}]^{\times}\|}, \quad \text{assuming } \sin \alpha \neq 0. \quad (11.21)$$

To correctly recover $\hat{\mathbf{n}}$ when $\sin \alpha = 0$, we need to distinguish between two cases. The first case refers to the singularity when $\alpha = 0$, corresponding to $\cos \alpha = 1$ and $\mathbf{R} = \mathbf{I}$. In accordance with observation 11.1 on page 167, we can use any $\hat{\mathbf{n}}$ as the rotation axis in this situation. The second case is $\alpha = \pi$, which leads to $\sin \alpha = 0$ and $\cos \alpha = -1$. In this case Rodrigues' formula reduces to

$$\mathbf{R}(\hat{\mathbf{n}}, \pi) = 2\hat{\mathbf{n}}\hat{\mathbf{n}}^{\top} - \mathbf{I}, \quad \Rightarrow \quad \hat{\mathbf{n}}\hat{\mathbf{n}}^{\top} = \frac{\mathbf{R}(\hat{\mathbf{n}}, \pi) + \mathbf{I}}{2}. \quad (11.22)$$

This implies that $\hat{\mathbf{n}}$ can be recovered from $\mathbf{R}(\hat{\mathbf{n}}, \pi)$, e.g., as a normalized row or column of the matrix $\mathbf{R}(\hat{\mathbf{n}}, \pi) + \mathbf{I}$. The sign of $\hat{\mathbf{n}}$ is not important in this case since $\mathbf{R}(\hat{\mathbf{n}}, \pi) = \mathbf{R}(-\hat{\mathbf{n}}, \pi)$.

This method of implementing f^{-1} for the axis-angle representation is summarized in Algorithm 11.1. Notice that the numerical resolution must be taken into account here, to avoid using the “normal branch” if s is too small to give a numerically stable estimate of $\hat{\mathbf{n}}$.

³Here we are using the cyclic property of the trace, described in Toolbox Section 3.3.1, Equation (3.53).

⁴Here we are using the inverse cross product operator, defined in Toolbox Section 3.7.3.3.

Algorithm 11.1: Finding rotation axis $\hat{\mathbf{n}}$ and rotation angle α given a rotation matrix \mathbf{R} . No EVD.

```

Input:  $\mathbf{R} \in SO(3)$ 
Output: A rotation axis  $\hat{\mathbf{n}} \in S^2$ , and a rotation angle  $\alpha$ 
1 Compute 3D vector  $\mathbf{v} = [\mathbf{R}]^\times$  and scalars  $c = (\text{trace}(\mathbf{R}) - 1)/2$ ,  $s = \|\mathbf{v}\|$ 
2 if  $s = 0$  then /* Check if  $\sin \alpha = 0$  */
3   if  $c > 0$  then /* Check if we have the singular case  $\alpha = 0$  */
4     Return arbitrary  $\hat{\mathbf{n}} \in S^2$  and  $\alpha = 0$ 
5   else /* This the case  $\alpha = \pi$  */
6     Set  $\mathbf{n}_k = \text{column } k \text{ in matrix } (\mathbf{R} + \mathbf{I}) \text{ for } k = 1, 2, 3$ 
7     /* Alternatively use the columns instead of the rows */
8     Set  $\hat{\mathbf{n}} = \mathbf{n}_k / \|\mathbf{n}_k\|$  where  $\|\mathbf{n}_k\|$  is the largest of  $\|\mathbf{n}_1\|, \|\mathbf{n}_2\|, \|\mathbf{n}_3\|$ 
9     Return  $\hat{\mathbf{n}}$  and  $\alpha = \pi$ 
10  end
11 else /* This is the normal case */
12   Return  $\hat{\mathbf{n}} = \mathbf{v}/s$  and  $\alpha = \arctan(s/c)$ 
13 end
```

11.2.3 Summary

We summarize the properties of the axis-angle representation of $SO(3)$ that we have observed here. The representation function f is given by Rodrigues' rotation formula, defined on the domain $D = S^2 \times \mathbb{R}$, and its inverse f^{-1} is given by Algorithm 11.1. The representation has a singularity for $\mathbf{R} = \mathbf{I}$, when the rotation axis becomes undefined. Moreover, the representation has ambiguities on both $\hat{\mathbf{n}}$ and α . We can change the signs of both $\hat{\mathbf{n}}$ and α at the same time, and add arbitrary integer multiples of 2π to α . These ambiguities can be resolved by restricting D in various ways, but such restrictions also introduce discontinuities in f^{-1} . The inverse rotation mapping inv on $\mathbf{R}(\hat{\mathbf{n}}, \alpha)$ is implemented by $\mathbf{R}(-\hat{\mathbf{n}}, \alpha) = \mathbf{R}(\hat{\mathbf{n}}, -\alpha)$. The concatenation operation cannot be implemented in a straight-forward way for the axis-angle representation.

In conclusion, the axis-angle representation of $SO(3)$ works well in numerical computations as long as we can avoid the singularity at $\mathbf{R} = \mathbf{I}$. We can do this either by not dealing with rotations close to \mathbf{I} , or by explicitly taking care of the singularity in the computations. The issue of avoiding the singularity is further illustrated in Section 15.3.2, where optimization of functions that depend on rotations is discussed.

The cause of the singularity is the fact that we treat $\hat{\mathbf{n}}$ and α as separate entities. The singularity can be avoided by combining the vector and the angle, in one way or another. This is what we will do next.

11.3 $so(3)$

While the representations of rotations in \mathbb{R}^3 presented so far are straight-forward, there are also less intuitive, but very efficient, representations that can be used. Two of them use the domain $D = so(3)$, the set of 3×3 anti-symmetric matrices, but they map this domain in two different ways to $SO(3)$, i.e., f is different for the two representations. One uses the matrix exponential function, defined in Toolbox Section 8.5.2, and one uses a more regular matrix expression. As is described in Toolbox Section 3.7.3.3, there is a one-to-one correspondence between $so(3)$ and \mathbb{R}^3 . Therefore, both representations can equally well be defined with $D = \mathbb{R}^3$. Since $\dim(so(3)) = 3$ it means that an algebraic representation based on $D = so(3)$ provides a minimal parameterization of $SO(3)$.

11.3.1 Using the matrix exponential function

Toolbox Section 8.6.3 defines the matrix exponential function, a generalization of the usual exponential function defined on square matrices. This function maps⁵ onto $SO(3)$, so for each $\mathbf{R} \in SO(3)$ there is an $\mathbf{N} \in so(3)$ such that

$$\mathbf{R} = f(\mathbf{N}) = e^{\mathbf{N}}. \quad (11.23)$$

⁵See Toolbox Section 8.6.3, Equation (8.105).

As discussed in Toolbox Section 8.5.3, this \mathbf{N} is not unique. We will now look at the exponential mapping more in detail, to understand the connection between \mathbf{N} and the axis-angle representation.

Based on the results of Toolbox Section 3.7.3.3 and since $\mathbf{N} \in so(3)$, we can set $\mathbf{N} = [\mathbf{n}]_{\times}$ for some unique $\mathbf{n} \in \mathbb{R}^3$. Furthermore, we can write $\mathbf{n} = \alpha \hat{\mathbf{n}}$ where $\alpha \geq 0$ and $\hat{\mathbf{n}} \in S^2$. This leads to

$$\mathbf{N} = [\mathbf{n}]_{\times} = \alpha [\hat{\mathbf{n}}]_{\times}. \quad (11.24)$$

where

$$|\alpha| = \frac{1}{\sqrt{2}} \|\mathbf{N}\|_F, \quad \hat{\mathbf{n}} = \frac{[\mathbf{N}]_{\times}}{\alpha}, \quad \alpha \neq 0. \quad (11.25)$$

These expressions leave the signs of α and of $\hat{\mathbf{n}}$ unspecified, which is consistent with observation 11.2.

In Toolbox Section 8.6.2 it is shown that

$$\mathbf{N} = \alpha [\hat{\mathbf{n}}]_{\times} = \mathbf{E} \mathbf{D} \mathbf{E}^*, \quad \text{where } \mathbf{E} = \begin{pmatrix} \hat{\mathbf{n}} & \frac{\hat{\mathbf{p}} - i\hat{\mathbf{q}}}{\sqrt{2}} & \frac{\hat{\mathbf{p}} + i\hat{\mathbf{q}}}{\sqrt{2}} \end{pmatrix}, \quad \text{and } \mathbf{D} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & e^{i\alpha} & 0 \\ 0 & 0 & e^{-i\alpha} \end{pmatrix}. \quad (11.26)$$

Here, \mathbf{E} is a unitary matrix⁷, and $\hat{\mathbf{n}}, \hat{\mathbf{p}}, \hat{\mathbf{q}}$ forms a right-handed ON-basis. The eigenvalues of \mathbf{N} in the diagonal of \mathbf{D} are in general complex-valued. This applies also to the corresponding eigenvectors in the columns of \mathbf{E} . We insert Equation (11.26) into Equation (11.23) and get

$$\mathbf{R} = e^{\alpha[\hat{\mathbf{n}}]_{\times}} = e^{\mathbf{E} \mathbf{D} \mathbf{E}^*} = \mathbf{E} e^{\mathbf{D}} \mathbf{E}^*. \quad (11.27)$$

This \mathbf{R} is a rotation, an element of $SO(3)$, but which rotation is it? What is the rotation axis and the rotation angle? We expand the right-hand side of Equation (11.27):

$$\begin{aligned} e^{\alpha[\hat{\mathbf{n}}]_{\times}} &= \mathbf{E} e^{\mathbf{D}} \mathbf{E}^* = \left(\hat{\mathbf{n}} \quad \frac{\hat{\mathbf{p}} - i\hat{\mathbf{q}}}{\sqrt{2}} \quad \frac{\hat{\mathbf{p}} + i\hat{\mathbf{q}}}{\sqrt{2}} \right) \begin{pmatrix} 1 & 0 & 0 \\ 0 & e^{i\alpha} & 0 \\ 0 & 0 & e^{-i\alpha} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{n}}^T \\ \frac{\hat{\mathbf{p}}^T + i\hat{\mathbf{q}}^T}{\sqrt{2}} \\ \frac{\hat{\mathbf{p}}^T - i\hat{\mathbf{q}}^T}{\sqrt{2}} \end{pmatrix} = \\ &= \hat{\mathbf{n}} \hat{\mathbf{n}}^T + \frac{1}{2} e^{i\alpha} (\hat{\mathbf{p}} - i\hat{\mathbf{q}})(\hat{\mathbf{p}}^T + i\hat{\mathbf{q}}^T) + \frac{1}{2} e^{-i\alpha} (\hat{\mathbf{p}} + i\hat{\mathbf{q}})(\hat{\mathbf{p}}^T - i\hat{\mathbf{q}}^T) = \\ &= \hat{\mathbf{n}} \hat{\mathbf{n}}^T + \cos \alpha (\hat{\mathbf{p}} \hat{\mathbf{p}}^T + \hat{\mathbf{q}} \hat{\mathbf{q}}^T) + \sin \alpha (\hat{\mathbf{q}} \hat{\mathbf{p}}^T - \hat{\mathbf{p}} \hat{\mathbf{q}}^T) = \\ &= \hat{\mathbf{n}} \hat{\mathbf{n}}^T + \cos \alpha (\mathbf{I} - \hat{\mathbf{n}} \hat{\mathbf{n}}^T) + \sin \alpha [\hat{\mathbf{n}}]_{\times}. \end{aligned} \quad (11.28)$$

We recognize this as Rodrigues' rotation formula, Equation (11.12), so we can write

$$\mathbf{R}(\hat{\mathbf{n}}, \alpha) = e^{\alpha[\hat{\mathbf{n}}]_{\times}} = e^{[\mathbf{n}]_{\times}} = e^{\mathbf{N}}. \quad (11.29)$$

We conclude that $\hat{\mathbf{n}}$ is a normalized rotation axis of \mathbf{R} and α is the corresponding rotation angle, given by the right-hand rule.

Exponential Mapping Restricted on $so(3)$

The result in Equation (11.29) means that when the matrix exponential mapping is applied on $so(3)$, it can be implemented based on Rodrigues' rotation formula. With $\mathbf{n} \neq 0$, the rotation angle and rotation axis are given as

$$\alpha = \|\mathbf{n}\|, \quad \hat{\mathbf{n}} = \frac{\mathbf{n}}{\|\mathbf{n}\|}. \quad (11.30)$$

Inserted into Equation (11.28) and in combination with Equation (11.29), this leads to

$$e^{[\mathbf{n}]_{\times}} = \mathbf{I} + (\cos \|\mathbf{n}\| - 1) \left(\mathbf{I} - \frac{\mathbf{n} \mathbf{n}^T}{\|\mathbf{n}\|^2} \right) + \sin \|\mathbf{n}\| \frac{[\mathbf{n}]_{\times}}{\|\mathbf{n}\|} = \mathbf{I} + g(\|\mathbf{n}\|) (\mathbf{n} \mathbf{n}^T - \|\mathbf{n}\|^2 \mathbf{I}) + h(\|\mathbf{n}\|) [\mathbf{n}]_{\times}, \quad (11.31)$$

⁶The factor $\sqrt{2}$ comes from Toolbox Section 3.7.3, Equation (3.143).

⁷The group of unitary matrices, $U(n)$, is defined in Toolbox Section 3.10.4.5.

where

$$g(x) = \frac{1 - \cos(x)}{x^2}, \quad h(x) = \frac{\sin(x)}{x} = \text{sinc}(x). \quad (11.32)$$

Notice that both g and h are continuous functions on \mathbb{R} , at least if we specify $g(0) = \frac{1}{2}$ and $h(0) = \text{sinc}(0) = 1$. As a consequence, we can use Equation (11.31) also when $\mathbf{n} = \mathbf{0}$, since it correctly gives $e^{\mathbf{0}} = \mathbf{I}$. In summary, for the special case when the matrix exponential mapping is applied onto $so(3)$, it can be implemented as in Equation (11.31) for an arbitrary $\mathbf{n} \in \mathbb{R}^3$.

Inverse Mapping

Since the rotation \mathbf{R} can be obtained from f in Equation (11.23), the inverse mapping gives \mathbf{N} as:

$$\mathbf{N} = f^{-1}(\mathbf{R}) = \log \mathbf{R}. \quad (11.33)$$

Here, we use the matrix logarithm defined in Toolbox Section 8.5.3.

Numerical implementations of both the matrix exponential function and the matrix logarithm are non-trivial. If you are serious about using this particular representation, use dedicated software implementations⁸ of these functions. In particular, avoid decomposing \mathbf{N} into a rotation axis $\hat{\mathbf{n}}$ and a rotation angle α . In fact, such an operation implies that you are using the axis-angle representation instead. The axis-angle representation has a singularity for $\mathbf{R} = \mathbf{I}$, when $\mathbf{N} = \mathbf{0}$. The exponential mapping does not exhibit this singularity.

General Properties

The representation of $SO(3)$ discussed here is complete but not unique. Since the rotation angle is modulo 2π , we get

$$\mathbf{R}(\hat{\mathbf{n}}, \alpha) = e^{(\alpha + 2\pi k)[\hat{\mathbf{n}}]_{\times}}, \quad k \in \mathbb{Z}. \quad (11.34)$$

This implies, for example, that any $\mathbf{N} \in so(3)$ such that $\|\mathbf{N}\|_F = 2\sqrt{2}\pi k$ gives $e^{\mathbf{N}} = \mathbf{I}$ for any $k \in \mathbb{Z}$. More generally, for $\mathbf{N} \neq \mathbf{0}$, it means that

$$e^{\mathbf{N}} = e^{\left(1 + \frac{2\sqrt{2}\pi k}{\|\mathbf{N}\|_F}\right)\mathbf{N}}, \quad k \in \mathbb{Z}. \quad (11.35)$$

A straight-forward way to make the representation unique is to restrict D to the ball where $\|\mathbf{N}\|_F \leq \sqrt{2}\pi$, and use only a hemisphere of the surface. This makes f^{-1} well-defined but also means that this inverse mapping becomes discontinuous.

Let $\mathbf{N} \in so(3)$ be a representation of the rotation $\mathbf{R} = e^{\mathbf{N}}$, from which follows⁹ that the inverse rotation is $\mathbf{R}^{-1} = e^{-\mathbf{N}}$. This means that finding inverse rotations in D is trivial:

$$\text{inv}(\mathbf{N}) = -\mathbf{N}. \quad (11.36)$$

Concatenation in D , however, is non-trivial for the case of the exponential mapping and not further discussed here.

Using the exponential mapping to represent $SO(3)$ requires the access to efficient implementations of the exponential map on $so(3)$, and possibly also to the matrix logarithm on $SO(3)$. This is not the case for the Cayley transform, another representation that uses $D = so(3)$.

11.3.2 The Cayley transform

If we want a mapping $f : so(3) \rightarrow SO(3)$, there is an alternative to the matrix exponential function. It is simpler to implement as it uses only basic matrix operations on 3×3 matrices. The downside is that it does not give a complete representation of $SO(3)$: it cannot represent rotations by 180° .

Let $\mathbf{M} \in so(3)$ and define the so-called *Cayley transform* of \mathbf{M} as the matrix \mathbf{R} formulated as

$$\mathbf{R} = (\mathbf{I} + \mathbf{M})(\mathbf{I} - \mathbf{M})^{-1} = / \text{the two factors commute} / = (\mathbf{I} - \mathbf{M})^{-1}(\mathbf{I} + \mathbf{M}). \quad (11.37)$$

⁸For example, see chapters 10 and 11 in [35]. The exponential and the logarithm functions on matrices are implemented in Matlab/Octave as `expm` and `logm`, respectively.

⁹See Toolbox Section 8.5.2, Equation (8.77).

From this expression of \mathbf{R} follows $\mathbf{R} \in SO(3)$, although this fact may not be obvious at the moment. To see that this statement is correct, factor \mathbf{M} with EVD: $\mathbf{M} = \mathbf{E} \mathbf{D} \mathbf{E}^*$, and insert this into Equation (11.37):

$$\begin{aligned}\mathbf{R} &= (\mathbf{I} + \mathbf{E} \mathbf{D} \mathbf{E}^*) (\mathbf{I} - \mathbf{E} \mathbf{D} \mathbf{E}^*)^{-1} = \mathbf{E} (\mathbf{I} + \mathbf{D}) \mathbf{E}^* (\mathbf{E} (\mathbf{I} - \mathbf{D}) \mathbf{E}^*)^{-1} = \\ &= \mathbf{E} (\mathbf{I} + \mathbf{D}) \mathbf{E}^* \mathbf{E} (\mathbf{I} - \mathbf{D})^{-1} \mathbf{E}^* = \mathbf{E} (\mathbf{I} + \mathbf{D}) (\mathbf{I} - \mathbf{D})^{-1} \mathbf{E}^*. \end{aligned} \quad (11.38)$$

- Here we are using the fact that $\mathbf{E} \in U(3)$, i.e., $\mathbf{E}^{-1} = \mathbf{E}^*$.

This result implies that \mathbf{R} has the same eigenvectors as \mathbf{M} . But when $\lambda = ia$ is an eigenvalue of \mathbf{M} , the corresponding eigenvalue ρ of \mathbf{R} is given as

$$\rho = \frac{1 + \lambda}{1 - \lambda} = \frac{1 + ia}{1 - ia}, \quad a \in \mathbb{R}. \quad (11.39)$$

Based on the identity

$$e^{i\alpha} = \frac{e^{i\frac{\alpha}{2}}}{e^{-i\frac{\alpha}{2}}} = \frac{\cos \frac{\alpha}{2} + i \sin \frac{\alpha}{2}}{\cos \frac{\alpha}{2} - i \sin \frac{\alpha}{2}} = \frac{1 + i \tan \frac{\alpha}{2}}{1 - i \tan \frac{\alpha}{2}}, \quad \alpha \neq \pm\pi, \quad (11.40)$$

we can write the eigenvalues of \mathbf{R} as

$$\rho = \frac{1 + ia}{1 - ia} = e^{i\alpha}, \quad \text{where } a = \tan \frac{\alpha}{2}. \quad (11.41)$$

Consequently, the Cayley transform maps the three eigenvalues $0, \pm ia$ for \mathbf{M} to the eigenvalues $1, e^{\pm i\alpha}$ for \mathbf{R} . Based on the discussion in Toolbox Section 8.6.4, this result means that \mathbf{R} indeed is an element of $SO(3)$. More precisely: $\mathbf{M} \in so(3)$ can be written as $\mathbf{M} = [a \hat{\mathbf{m}}]_{\times}$ and has eigenvalues $0, \pm ia$, where 0 is an eigenvalue corresponding to eigenvector $\hat{\mathbf{m}}$.

We can summarize these results as

$$\text{The Cayley transform of } a [\hat{\mathbf{m}}]_{\times} \in so(3) = \mathbf{R}(\hat{\mathbf{m}}, 2 \tan^{-1} a) \in SO(3), \quad (11.42)$$

or, alternatively, as

$$\text{The Cayley transform of } \tan \frac{\alpha}{2} [\hat{\mathbf{m}}]_{\times} \in so(3) = \mathbf{R}(\hat{\mathbf{m}}, \alpha) \in SO(3). \quad (11.43)$$

The Cayley transformation defines an incomplete representation of $SO(3)$ based on the domain $D = so(3)$. No finite $a \in \mathbb{R}$ corresponds to $\alpha = \pm\pi$, so this representation has singularities for such rotations. There is no $\mathbf{M} \in so(3)$ that can be Cayley transformed to a 180° rotation.

From Equation (11.37) follows immediately that the Cayley transform of \mathbf{M} and of $-\mathbf{M}$ are \mathbf{R} and \mathbf{R}^{-1} , respectively. This means that inversion of a rotation corresponds to a change of sign in the parameter space.

Inverse Cayley transformation

Although the Cayley transform leads to an incomplete representation, it can still be useful if we can avoid 180° rotations. This is because we can define the inverse transformation f^{-1} in a straight-forward way.

Let $\mathbf{R} \in SO(3)$, represented by $\mathbf{M} \in so(3)$ using the Cayley transform in Equation (11.37), and form

$$\mathbf{M}' = (\mathbf{R} - \mathbf{I})(\mathbf{R} + \mathbf{I})^{-1}. \quad (11.44)$$

Using the same argumentation as above, it follows that \mathbf{M}' and \mathbf{R} share the same eigenvectors. Furthermore, if \mathbf{R} has an eigenvalue ρ , then the corresponding eigenvalue for \mathbf{M}' is

$$\frac{\rho - 1}{\rho + 1}. \quad (11.45)$$

It must also be the case that we get the eigenvalues of \mathbf{R} from Equation (11.39) where ia is the corresponding eigenvalue of \mathbf{M}' . So, we replace ρ with the right-hand side of Equation (11.39) to get:

$$\frac{\frac{1+ia}{1-ia} - 1}{\frac{1+ia}{1-ia} + 1} = \frac{1+ia - (1-ia)}{1+ia + (1-ia)} = \frac{2ia}{2} = ia. \quad (11.46)$$

This means that the right-hand side of Equation (11.44) has the same eigenvectors and eigenvalues as \mathbf{M}' has: $\mathbf{M}' = \mathbf{M}$. From this follows that Equation (11.44), with $\mathbf{M}' = \mathbf{M}$, defines the *inverse Cayley transform*.

In the case that \mathbf{R} is a 180° rotation, it has two eigenvalues $= -1$. This implies that $\mathbf{I} + \mathbf{R}$ has two eigenvalues that vanish, and the matrix inverse of $\mathbf{I} + \mathbf{R}$ does not exist. This is consistent with the incompleteness already described for this representation.

We now turn to the concatenation of two rotations and how to describe this operation in the parameters space. Consider two rotations $\mathbf{R}_1, \mathbf{R}_2$, represented by $\mathbf{M}_1, \mathbf{M}_2 \in so(3)$ using the Cayley transform. To find $\mathbf{M} \in so(3)$ that represent $\mathbf{R}_2\mathbf{R}_1$, we can first multiply the expressions for \mathbf{R}_1 and \mathbf{R}_2 and then apply the inverse Cayley transformation onto $\mathbf{R}_2\mathbf{R}_1$. This gives us an expression for how the resulting matrix in $so(3)$ depends on $\mathbf{M}_1, \mathbf{M}_2$. We leave this as an exercise for the interested reader.

11.3.3 Summary

In this section we have discussed two representations of $SO(3)$ that use the domain $D = so(3)$, the set of anti-symmetric 3×3 matrices. The representation function f is in one case defined as the matrix exponential function, and in the other case as the Cayley transform. The first representation needs an effective implementation of the matrix exponential function for matrices. If also f^{-1} is used, then the matrix logarithm is needed as well. From this perspective, the Cayley transform may be more interesting. It requires only standard operations, such as multiplication and inverse of 3×3 matrices. This latter representation is not complete, however, since 180° rotations cannot be represented. This singularity does not exist for the matrix exponential function.

In both representations, we can replace the matrices $\mathbf{N}, \mathbf{M} \in so(3)$ by the vectors in \mathbb{R}^3 , using the identities $\mathbf{n} = [\mathbf{N}]_\times$ and $\mathbf{m} = [\mathbf{M}]_\times$, respectively. This means that in both cases we can use $D = \mathbb{R}^3$ as an alternative domain of the representation. When necessary, we then replace \mathbf{N} or \mathbf{M} by

$$\mathbf{N} = [\mathbf{n}]_\times, \quad \mathbf{M} = [\mathbf{m}]_\times. \quad (11.47)$$

Furthermore, we get the inverse rotation by a negation of \mathbf{N}, \mathbf{M} , or of \mathbf{n}, \mathbf{m} . It is not possible to express the concatenating two rotations in a simple way in the parameter space for neither of the two representations.

The two $so(3)$ -representations are related. Let $\mathbf{N} = [\mathbf{n}]_\times = \alpha[\hat{\mathbf{n}}]_\times$ represent a particular rotation based on the exponential function. Let $\mathbf{M} = [\mathbf{m}]_\times = a[\hat{\mathbf{m}}]_\times$ to represent the *same* rotation based on the Cayley transformation. When $\alpha \neq \pi$, the two representations are then related as

$$\begin{aligned} \alpha &= 2 \tan^{-1} a, & a &= \tan \frac{\alpha}{2}, \\ \hat{\mathbf{n}} &= \hat{\mathbf{m}}, & \hat{\mathbf{m}} &= \hat{\mathbf{n}}, \\ \mathbf{n} &= 2 \frac{\tan^{-1} \|\mathbf{m}\|}{\|\mathbf{m}\|} \mathbf{m}, & \mathbf{m} &= \frac{\tan \frac{\|\mathbf{n}\|}{2}}{\|\mathbf{n}\|} \mathbf{n} = \tan \frac{\|\mathbf{n}\|}{2} \hat{\mathbf{n}}, \\ \mathbf{N} &= \sqrt{8} \tan^{-1} \left(\frac{\|\mathbf{M}\|}{\sqrt{2}} \right) \frac{\mathbf{M}}{\|\mathbf{M}\|_F}, & \mathbf{M} &= \sqrt{2} \tan \left(\frac{\|\mathbf{N}\|}{\sqrt{8}} \right) \frac{\mathbf{N}}{\|\mathbf{N}\|_F}, \\ \mathbf{N} &= \alpha[\hat{\mathbf{n}}]_\times, & \mathbf{M} &= \tan \frac{\alpha}{2} [\hat{\mathbf{n}}]_\times. \end{aligned} \quad (11.48)$$

Toolbox Section 8.5.2 defines the ball $D = \|\mathbf{N}\|_F < \sqrt{2}\pi$ in $so(3)$ as a possible domain for the exponential function. The result we derived just now implies that we can see the Cayley approach as a non-linear mapping of this ball, effectively stretching it to include the entirety of $so(3)$.

11.4 Unit quaternions

The quaternion field \mathbb{H} and its basic properties are described in Toolbox Section 7.2. One of the more common applications for quaternions is for the representation of 3D rotations. In this section we use the scalar-vector representation of quaternions to describe how 3D rotations can be represented by unit quaternions.

Quaternionic embedding of \mathbb{R}^3

Let $\bar{\mathbf{x}} \in \mathbb{R}^3$, and represent this vector as the pure quaternion $(0, \bar{\mathbf{x}})$, with the real part (the scalar component) equal to zero. This construction defines a so-called *quaternionic embedding of \mathbb{R}^3* . It describes a one-to-one mapping between \mathbb{R}^3 and pure quaternions.

By themselves, pure quaternions are not so interesting. This set is closed under addition, but not under quaternion multiplication. We need an operation that combines pure quaternions with other quaternions to represent more general operations. This leads to the quaternion sandwich product.

The quaternion sandwich product

Let $q = (s, \bar{\mathbf{v}})$ be a general quaternion, and let $(0, \bar{\mathbf{x}})$ be a pure quaternion¹⁰, representing $\bar{\mathbf{x}} \in \mathbb{R}^3$. The *quaternion sandwich product* that combines them is defined as¹¹

$$q' = q \circ (0, \bar{\mathbf{x}}) \circ q^* = (s, \bar{\mathbf{v}}) \circ (0, \bar{\mathbf{x}}) \circ (s, -\bar{\mathbf{v}}). \quad (11.49)$$

Since the quaternion product is associative, we can evaluate this expression from left or from right¹². Here we do it from left to right:

$$q' = (-\bar{\mathbf{v}} \cdot \bar{\mathbf{x}}, s\bar{\mathbf{x}} + \bar{\mathbf{v}} \times \bar{\mathbf{x}}) \circ (s, -\bar{\mathbf{v}}) = (-s(\bar{\mathbf{v}} \cdot \bar{\mathbf{x}}) + (s\bar{\mathbf{x}} + \bar{\mathbf{v}} \times \bar{\mathbf{x}}) \cdot \bar{\mathbf{v}}, \bar{\mathbf{x}}') = (0, \bar{\mathbf{x}}'). \quad (11.50)$$

This intermediate result shows that the quaternion sandwich product can be seen as a mapping from a pure quaternion to a pure quaternion or, equivalently, from $\bar{\mathbf{x}} \in \mathbb{R}^3$ to $\bar{\mathbf{x}}' \in \mathbb{R}^3$, where the latter vector is expressed as

$$\begin{aligned} \bar{\mathbf{x}}' &= \bar{\mathbf{v}}(\bar{\mathbf{v}} \cdot \bar{\mathbf{x}}) + s(s\bar{\mathbf{x}} + \bar{\mathbf{v}} \times \bar{\mathbf{x}}) - (s\bar{\mathbf{x}} + \bar{\mathbf{v}} \times \bar{\mathbf{x}}) \times \bar{\mathbf{v}} = \\ &= \bar{\mathbf{v}}\bar{\mathbf{v}}^\top \bar{\mathbf{x}} + s^2 \bar{\mathbf{x}} + 2s[\bar{\mathbf{v}}]_\times \bar{\mathbf{x}} + [\bar{\mathbf{v}}]_\times^2 \bar{\mathbf{x}} = \\ &= \left(\bar{\mathbf{v}}\bar{\mathbf{v}}^\top + s^2 \mathbf{I} + 2s[\bar{\mathbf{v}}]_\times + [\bar{\mathbf{v}}]_\times^2 \right) \bar{\mathbf{x}} \end{aligned} \quad (11.51)$$

This shows that the quaternion sandwich product in Equation (11.49) defines a linear transformation on $\bar{\mathbf{x}} \in \mathbb{R}^3$. This linear transformation may appear somewhat obscure, and next we will see that it can be identified as a known quantity by choosing $(s, \bar{\mathbf{v}})$ in a suitable way.

11.4.1 The quaternionic embedding of $SO(3)$

Consider a rotation in \mathbb{R}^3 that is defined by a rotation axis $\hat{\mathbf{n}}$ and a rotation angle α . We define a second type quaternionic embedding, the *quaternionic embedding of $SO(3)$* , specifically for this rotation as

$$q(\hat{\mathbf{n}}, \alpha) = \left(\cos \frac{\alpha}{2}, \hat{\mathbf{n}} \sin \frac{\alpha}{2} \right). \quad (11.52)$$

From this expression follows¹³ that $|q(\hat{\mathbf{n}}, \alpha)| = 1$. So, $q(\hat{\mathbf{n}}, \alpha)$ is a unit quaternion and therefore

$$q(\hat{\mathbf{n}}, \alpha)^* = q(\hat{\mathbf{n}}, \alpha)^{-1} = \left(\cos \frac{\alpha}{2}, -\hat{\mathbf{n}} \sin \frac{\alpha}{2} \right). \quad (11.53)$$

This means that the quaternionic embedding of $SO(3)$ is a mapping from $SO(3)$ to unit quaternions in $S^3 \in \mathbb{H}$.

We can determine the rotation angle α of an $SO(3)$ rotation only up to an additive multiple of 2π . In Equation (11.52) this additional angle is halved, leading to a sign change of $q(\hat{\mathbf{n}}, \alpha)$ for odd multiples of 2π . Thus, the same rotation in $SO(3)$ can be represented either by some quaternion $q = q(\hat{\mathbf{n}}, \alpha)$, or as $q(\hat{\mathbf{n}}, \alpha + 2\pi) = -q$. Consequently, each rotation in $SO(3)$ is represented by exactly two unit quaternions that only differ in sign. Notice that there is no singularity for $\alpha = 0$, $\mathbf{R} = \mathbf{I}$ is represented by the two unit quaternions $(\pm 1, \mathbf{0})$.

¹⁰Pure quaternions are defined in Toolbox Section 7.2.1.

¹¹ q^* denotes the conjugate of q .

¹²Using Equation (7.10) in Toolbox Section 7.2.1.

¹³Here we are using Section 7.2.1, Equation (7.12).

To see why this particular way of representing $SO(3)$ with unit quaternions is useful, let us see what happens if we substitute the general quaternion $(s, \bar{\mathbf{v}})$ with $q(\hat{\mathbf{n}}, \alpha)$ in the expression for $\bar{\mathbf{x}}'$ in Equation (11.51):

$$\begin{aligned}\bar{\mathbf{x}}' &= \frac{1}{2} \hat{\mathbf{n}} \hat{\mathbf{n}}^\top \bar{\mathbf{x}} (1 - \cos \alpha) + \frac{1}{2} \bar{\mathbf{x}} (1 + \cos \alpha) + [\hat{\mathbf{n}}]_\times \bar{\mathbf{x}} \sin \alpha + \frac{1}{2} [\hat{\mathbf{n}}]_\times^2 \bar{\mathbf{x}} (1 - \cos \alpha) = \\ &= \left(\frac{1}{2} \hat{\mathbf{n}} \hat{\mathbf{n}}^\top (1 - \cos \alpha) + \frac{1}{2} \mathbf{I} (1 + \cos \alpha) + [\hat{\mathbf{n}}]_\times \sin \alpha + \frac{1}{2} (\hat{\mathbf{n}} \hat{\mathbf{n}}^\top - \mathbf{I}) (1 - \cos \alpha) \right) \bar{\mathbf{x}} = \\ &= \left(\hat{\mathbf{n}} \hat{\mathbf{n}}^\top + \cos \alpha (\mathbf{I} - \hat{\mathbf{n}} \hat{\mathbf{n}}^\top) + [\hat{\mathbf{n}}]_\times \sin \alpha \right) \bar{\mathbf{x}}.\end{aligned}\quad (11.54)$$

We recognize the terms within the parentheses as Rodrigues' rotation formula, Equation (11.12). They define the rotation matrix $\mathbf{R}(\hat{\mathbf{n}}, \alpha)$, a rotation about $\hat{\mathbf{n}}$ with the angle α . The conclusion is that the sandwich product in Equation (11.49) implements a rotation of the vector $\bar{\mathbf{x}} \in \mathbb{R}^3$:

$$\bar{\mathbf{x}}' = \mathbf{R}(\hat{\mathbf{n}}, \alpha) \bar{\mathbf{x}}, \quad \text{or} \quad q' = (0, \mathbf{R}(\hat{\mathbf{n}}, \alpha) \bar{\mathbf{x}}).\quad (11.55)$$

So far we have seen that any 3D rotation $\mathbf{R}(\hat{\mathbf{n}}, \alpha)$ applied to any vector in \mathbb{R}^3 can be implemented as the sandwich product in Equation (11.49). In this product, $q(\hat{\mathbf{n}}, \alpha) \in S^3$ represents the rotation, and it is unique except for an undefined sign. What remains to show is that every unit quaternion represents some 3D rotation. Take an arbitrary $q = (s, \bar{\mathbf{v}}) \in S^3$. If the vector part is zero, $\bar{\mathbf{v}} = \mathbf{0}$, we have already seen that this q corresponds to the identity rotation $\mathbf{R} = \mathbf{I}$. Otherwise, the vector part can be written as $\bar{\mathbf{v}} = \hat{\mathbf{n}} \sin \frac{\alpha}{2}$ for some angle α , and we get

$$\hat{\mathbf{n}} = \frac{\bar{\mathbf{v}}}{|\bar{\mathbf{v}}|}.\quad (11.56)$$

Since $q \in S^3$, the scalar part of q must equal $\cos \frac{\alpha}{2}$ and

$$\sin \frac{\alpha}{2} = \hat{\mathbf{n}} \cdot \bar{\mathbf{v}}.\quad (11.57)$$

From these relations we can determine $\frac{\alpha}{2}$ modulo 2π .

We can summarize these observations. The quaternionic embedding of $SO(3)$ defines a two-to-one mapping $f : S^3 \rightarrow SO(3)$, i.e., the parameters space D is the set of unit quaternions S^3 . This is referred to as the *double embedding* of $SO(3)$ in $S^3 \in \mathbb{H}$. In practice the double embedding is usually not an issue as long as we are aware of it and take necessary steps to deal with it.

The inverse of some rotation $\mathbf{R}(\hat{\mathbf{n}}, \alpha) \in SO(3)$ corresponds to the rotation $\mathbf{R}(\hat{\mathbf{n}}, -\alpha) = \mathbf{R}(-\hat{\mathbf{n}}, \alpha)$. It is represented by the unit quaternion $\pm q(-\hat{\mathbf{n}}, \alpha) = \pm q(\hat{\mathbf{n}}, -\alpha) = \pm q(\hat{\mathbf{n}}, \alpha)^*$. So, inversion of rotations is implemented as complex conjugation in the parameter space D .

It is trivial to concatenate two rotations using unit quaternions. Let $q(\hat{\mathbf{n}}_1, \alpha_1)$ and $q(\hat{\mathbf{n}}_2, \alpha_2)$ be two unit quaternions, representing two rotations \mathbf{R}_1 and \mathbf{R}_2 . The rotation $\mathbf{R}_2 \mathbf{R}_1$ is then represented by the unit quaternion $q(\hat{\mathbf{n}}_2, \alpha_2) \circ q(\hat{\mathbf{n}}_1, \alpha_1)$. Furthermore, this quaternion product can be implemented using only 16 multiplications and 12 additions of real numbers. This should be compared to 27 multiplications and 18 additions for computing $\mathbf{R}_2 \mathbf{R}_1$.

11.4.2 R from unit quaternion

In the case that the pure quaternion is sandwich multiplied with the unit quaternion $(s, \bar{\mathbf{v}}) \in S^3$, Equation (11.51) describes the resulting rotation as

$$\mathbf{R}(s, \bar{\mathbf{v}}) = \bar{\mathbf{v}} \bar{\mathbf{v}}^\top + s^2 \mathbf{I} + 2s [\bar{\mathbf{v}}]_\times + [\bar{\mathbf{v}}]_\times^2.\quad (11.58)$$

With the four elements of the unit quaternion given as

$$\mathbf{q} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix} \in S^3, \quad \Rightarrow \quad s = q_1, \quad \bar{\mathbf{v}} = \begin{pmatrix} q_2 \\ q_3 \\ q_4 \end{pmatrix},\quad (11.59)$$

the rotation matrix can be expressed directly in the elements of the quaternion \mathbf{q} as

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} q_1^2 + q_2^2 - q_3^2 - q_4^2 & 2(q_2 q_3 - q_1 q_4) & 2(q_2 q_4 + q_1 q_3) \\ 2(q_2 q_3 + q_1 q_4) & q_1^2 - q_2^2 + q_3^2 - q_4^2 & 2(q_3 q_4 - q_1 q_2) \\ 2(q_2 q_4 - q_1 q_3) & 2(q_3 q_4 + q_1 q_2) & q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{pmatrix}. \quad (11.60)$$

Each element of \mathbf{R} is a quadratic form in $\mathbf{q} \in \mathbb{R}^4$, and it follows that \mathbf{R} does not depend on the sign of \mathbf{q} .

11.4.3 Unit quaternion from \mathbf{R}

The inverse mapping f^{-1} , from $SO(3)$ to unit quaternions in S^3 can be implemented in different ways, but is typically based on the following observations. The trace of \mathbf{R} is given as

$$r_{11} + r_{22} + r_{33} = 3 - 4(q_2^2 + q_3^2 + q_4^2) = 3 - 4(1 - q_1^2) = 4q_1^2 - 1. \quad (11.61)$$

In fact, the squares of all four elements in \mathbf{q} can be derived from the diagonal elements in \mathbf{R} :

$$\begin{aligned} q_1^2 &= \frac{1 + r_{11} + r_{22} + r_{33}}{4}, & q_2^2 &= \frac{1 + r_{11} - r_{22} - r_{33}}{4}, \\ q_3^2 &= \frac{1 - r_{11} + r_{22} - r_{33}}{4}, & q_4^2 &= \frac{1 - r_{11} - r_{22} + r_{33}}{4}. \end{aligned} \quad (11.62)$$

The square roots of these expressions give the magnitudes of the elements of \mathbf{q} but, due to the double embedding, not the signs. We can choose the sign of one of the four elements in an arbitrary way, but then the other elements must have signs that are consistent with this choice. To sort out the signs it helps to consider the following combinations of elements in \mathbf{R} :

$$\begin{aligned} r_{12} + r_{21} &= 4q_2 q_3, & r_{13} + r_{31} &= 4q_2 q_4, & r_{23} + r_{32} &= 4q_3 q_4, \\ r_{21} - r_{12} &= 4q_1 q_4, & r_{13} - r_{31} &= 4q_1 q_3, & r_{32} - r_{23} &= 4q_1 q_2. \end{aligned} \quad (11.63)$$

Once we have determined the sign of one of the four elements in \mathbf{q} , there is only one choice of signs of the remaining three elements that is consistent with all relations in Equation (11.63). For example, we can choose q_1 as positive and select the signs of (q_2, q_3, q_4) to make them consistent with Equation (11.63). In fact, only three of the equations are need.

This approach has one possible issue, it gives undetermined signs when $q_1 = 0$. This happens regardless of which of the four elements that is chosen as positive. A solution is to first determine which of the four elements in \mathbf{q} is the largest in magnitude, based on Equation (11.62). We can then determine a positive value of this element by taking the square root of the corresponding expression. Finally, the other three elements can be determined by picking the right triplet of equations in Equation (11.63) and factor out the element chosen first. For example, if q_1 is the largest in magnitude, then the other three elements are determined as

$$q_2 = \frac{r_{32} - r_{23}}{4q_1}, \quad q_3 = \frac{r_{13} - r_{31}}{4q_1}, \quad q_4 = \frac{r_{21} - r_{12}}{4q_1}, \quad (11.64)$$

and since $q_1 > 0$ in this case, all three fractions are well-defined. It is not critical that we first choose the largest element in \mathbf{q} , it suffices that it is not close to zero to make the factoring-out well-defined. As long as its magnitude is larger than some threshold, we can choose any of the four elements as the first one to be determined with a positive sign. The remaining three elements are then produced by factoring out the first element from three equations in Equation (11.63).

A practical implementation of this approach begins with comparing the squares of the magnitudes, Equation (11.62), against a threshold t . The first element to exceed the threshold is used for the factoring-out. We must choose $t < \frac{1}{2}$, since there must always be at least one element in $\mathbf{q} \in S^3$ with a squared magnitude $\geq \frac{1}{2}$. Smaller values lead to fewer comparisons, while a very small value risks factoring-out an element close to zero.

A complete algorithm based on this approach is presented in Algorithm 11.2. The computational complexity increases with the threshold t . This is because, in general, a larger t more often leads further into the if-statements for a general $\mathbf{R} \in SO(3)$. In practice, it may therefore be more effective to use $0 < t \ll \frac{1}{2}$.

Algorithm 11.2: Find a unit quaternion that represents a given 3D rotation.

```

Input:  $\mathbf{R} \in SO(3)$ , with elements in accordance with Equation (11.60)
Output: A unit quaternion  $\mathbf{q} = (q_1, q_2, q_3, q_4)$  satisfying Equation (11.60)

1 Set threshold  $t$  in the range  $0 < t \leq \frac{1}{2}$ . Do not choose t too close to zero
2 Set  $\gamma = \frac{1+r_{11}+r_{22}+r_{33}}{4}$  //  $= q_1^2$ , Equation (11.62)
3 if  $\gamma \geq t$  then
4   Set  $q_1 = \sqrt{\gamma}$ ,  $q_2 = \frac{r_{32}-r_{23}}{4q_1}$ ,  $q_3 = \frac{r_{13}-r_{31}}{4q_1}$ ,  $q_4 = \frac{r_{21}-r_{12}}{4q_1}$  // Equation (11.63)
5 else
6   Set  $\gamma = q_2^2 = \frac{1+r_{11}-r_{22}-r_{33}}{4}$  //  $= q_2^2$ , Equation (11.62)
7   if  $s \geq t$  then
8     Set  $q_2 = \sqrt{\gamma}$ ,  $q_1 = \frac{r_{32}-r_{23}}{4q_2}$ ,  $q_3 = \frac{r_{12}+r_{21}}{4q_2}$ ,  $q_4 = \frac{r_{13}+r_{31}}{4q_2}$  // Equation (11.63)
9   else
10    Set  $\gamma = q_3^2 = \frac{1-r_{11}+r_{22}-r_{33}}{4}$  //  $= q_3^2$ , Equation (11.62)
11    if  $s \geq t$  then
12      Set  $q_3 = \sqrt{\gamma}$ ,  $q_1 = \frac{r_{13}-r_{31}}{4q_3}$ ,  $q_2 = \frac{r_{12}+r_{21}}{4q_3}$ ,  $q_4 = \frac{r_{23}+r_{32}}{4q_3}$  // Equation (11.63)
13    else
14      // This branch implies that  $q_4$  has largest magnitude
15      Set  $\gamma = q_4^2 = \frac{1-r_{11}-r_{22}+r_{33}}{4}$  //  $= q_4^2$ , Equation (11.62)
16      Set  $q_4 = \sqrt{\gamma}$ ,  $q_1 = \frac{r_{21}-r_{12}}{4q_4}$ ,  $q_2 = \frac{r_{13}+r_{31}}{4q_4}$ ,  $q_3 = \frac{r_{23}+r_{32}}{4q_4}$  // Equation (11.63)
17    end
18  end
19 end
20 Return  $\mathbf{q} = (q_1, q_2, q_3, q_4)$ 
```

11.4.4 Relation to the Cayley representation

Section 11.3.2 presents a representation of $SO(3)$ based on the Cayley transform. Here, the rotation is represented by a matrix $\mathbf{M} \in so(3)$, corresponding to a vector $\mathbf{m} = [\mathbf{M}]^\times \in \mathbb{R}^3$. Equation (11.48) describes the vector \mathbf{m} as an expression of the rotation axis $\hat{\mathbf{n}}$ and the rotation angle α . If we compare this expression with Equation (11.52) we see that the vector \mathbf{m} can be formulated in terms of the elements of the unit quaternion q :

$$\mathbf{m} = \tan \frac{\alpha}{2}, \quad \hat{\mathbf{n}} = \frac{1}{q_1} \begin{pmatrix} q_2 \\ q_3 \\ q_4 \end{pmatrix}. \quad (11.65)$$

Furthermore, q can be computed from \mathbf{m} as

$$q = \pm \frac{q_0}{|q_0|}, \quad \text{where} \quad q_0 = \begin{pmatrix} 1 \\ \mathbf{m} \end{pmatrix} = \begin{pmatrix} 1 \\ \tan \frac{\alpha}{2} \hat{\mathbf{n}} \end{pmatrix} \sim \begin{pmatrix} \cos \frac{\alpha}{2} \\ \sin \frac{\alpha}{2} \hat{\mathbf{n}} \end{pmatrix} = q. \quad (11.66)$$

In summary, it is trivial to go back and forth between the Cayley and the quaternion representation of $SO(3)$. When doing so, we must avoid the case $\alpha = \pi$, since \mathbf{m} is undefined for these rotations.

11.4.5 Summary

In summary, the quaternion representation of $SO(3)$ uses $D = S^3 \in \mathbb{H}$. It has no singularities and it is straightforward to describe both inversion and concatenation of rotations in terms of quaternions. It is also easy to implement both f and f^{-1} , see Equation (11.60) and Algorithm 11.2 on page 178. A possible downside is the double embedding: each rotation is represented by some a unit quaternion with undetermined sign. This ambiguity can normally be handled without problems in practical implementations where the quaternions are involved.

When we compare quaternions to the matrix representation described in Section 11.1, the quaternionic embedding is more compact. It requires only 4 components, instead of 9 for the matrix representation. The only internal

constraint for the quaternion q is $|q| = 1$, which is simpler to implement than $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$ for a rotation matrix. All these properties make unit quaternions a popular representation of 3D rotations. It comes only at the cost of some additional theory needed for dealing with the \mathbb{H} algebra.

11.5 Three-angle representations

As we will see, any rotation in $SO(3)$ can be decomposed as a sequence of three rotations about specific axes. Thus $SO(3)$ can be represented by the three rotation angles. This is a relatively simple representation of $SO(3)$, and therefore also popular in many applications. But it also has some disadvantages, e.g., a large set of singularities. To see this, and make a fair judgment of its virtues and drawbacks, this so-called three-angle representation is presented here.

Let $\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \hat{\mathbf{e}}_3$ be the three axes of an ON-basis for \mathbb{R}^3 , and let $\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2, \hat{\mathbf{n}}_3$ be three rotation axes with corresponding rotation angles $\alpha_1, \alpha_2, \alpha_3$. The rotation axes and rotation angles define three rotations:

$$\mathbf{R}_1 = \mathbf{R}(\hat{\mathbf{n}}_1, \alpha_1), \quad \mathbf{R}_2 = \mathbf{R}(\hat{\mathbf{n}}_2, \alpha_2), \quad \mathbf{R}_3 = \mathbf{R}(\hat{\mathbf{n}}_3, \alpha_3). \quad (11.67)$$

We assume that the three rotation axes are fixed, and not all three of them are parallel. The sequence of rotating first by \mathbf{R}_1 , then by \mathbf{R}_2 , and finally by \mathbf{R}_3 provides a representation of an arbitrary rotation in $SO(3)$:

$$\mathbf{R} = \mathbf{R}(\alpha_1, \alpha_2, \alpha_3) = \mathbf{R}_3 \mathbf{R}_2 \mathbf{R}_1 = \mathbf{R}(\hat{\mathbf{n}}_3, \alpha_3) \mathbf{R}(\hat{\mathbf{n}}_2, \alpha_2) \mathbf{R}(\hat{\mathbf{n}}_1, \alpha_1). \quad (11.68)$$

This means that an arbitrary $\mathbf{R} \in SO(3)$ can be expanded as in Equation (11.68) for three angles $\alpha_1, \alpha_2, \alpha_3$.

The representation of $SO(3)$ defined in Equation (11.68) is sometimes referred to as a *three-angle representation*. Another name for this type of representation is *Euler angles*. This concept refers specifically to the three angles $\alpha_1, \alpha_2, \alpha_3$ that determine a specific rotation. To make practical use of a three-angle representation, the three rotation axes $\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2, \hat{\mathbf{n}}_3$ must be specified. Euler angles are sometimes used for general three angle representations, but the original idea behind this concept was developed for the case when $\hat{\mathbf{n}}_1 = \hat{\mathbf{n}}_3$ and $\hat{\mathbf{n}}_1 \perp \hat{\mathbf{n}}_2$. More precisely, Euler angles normally use $\hat{\mathbf{n}}_1 = \hat{\mathbf{n}}_3 = \hat{\mathbf{e}}_1$ and $\hat{\mathbf{n}}_2 = \hat{\mathbf{e}}_2$. The case when the three rotation axes instead form a general ON-basis is sometimes referred to as *Cardano angles*. Another name that appears in the literature is *Tait-Bryan angles*. This label is often used for the specific case when $\hat{\mathbf{n}}_1 = \hat{\mathbf{e}}_1, \hat{\mathbf{n}}_2 = \hat{\mathbf{e}}_2, \hat{\mathbf{n}}_3 = \hat{\mathbf{e}}_3$. In this presentation we refer to all these cases as three-angle representations of $SO(3)$.

Since \mathbf{R} in Equation (11.68) is a product of rotations, it follows that $\mathbf{R} \in SO(3)$. But the fact that it is possible to expand every $\mathbf{R} \in SO(3)$ as in Equation (11.68) may not be clear at this point. We present here an outline of a proof for the case that $\hat{\mathbf{n}}_1 = \hat{\mathbf{n}}_3 = \hat{\mathbf{e}}_1$ and $\hat{\mathbf{n}}_2 = \hat{\mathbf{e}}_2$, i.e., leading to what is commonly known as Euler angles. In this case we have

$$\mathbf{R}_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha_1 & -\sin \alpha_1 \\ 0 & \sin \alpha_1 & \cos \alpha_1 \end{pmatrix}, \quad \mathbf{R}_2 = \begin{pmatrix} \cos \alpha_2 & 0 & -\sin \alpha_2 \\ 0 & 1 & 0 \\ \sin \alpha_2 & 0 & \cos \alpha_2 \end{pmatrix}, \quad \mathbf{R}_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha_3 & -\sin \alpha_3 \\ 0 & \sin \alpha_3 & \cos \alpha_3 \end{pmatrix}. \quad (11.69)$$

Inserted into Equation (11.68) this gives the combined rotation

$$\mathbf{R} = \begin{pmatrix} \cos \alpha_2 & -\sin \alpha_1 \sin \alpha_2 & -\cos \alpha_1 \sin \alpha_2 \\ -\sin \alpha_2 \sin \alpha_3 & \cos \alpha_1 \cos \alpha_3 - \sin \alpha_1 \cos \alpha_2 \sin \alpha_3 & -\sin \alpha_1 \cos \alpha_3 - \cos \alpha_1 \cos \alpha_2 \sin \alpha_3 \\ \sin \alpha_2 \cos \alpha_3 & \sin \alpha_1 \cos \alpha_2 \cos \alpha_3 + \cos \alpha_1 \sin \alpha_3 & \cos \alpha_1 \cos \alpha_2 \cos \alpha_3 - \sin \alpha_1 \sin \alpha_3 \end{pmatrix}. \quad (11.70)$$

From Equation (11.70) follows that a change in the sign of α_2 is equivalent to adding (or subtracting) π to (or from) α_1 and α_3 . This means that we can restrict α_2 to an interval of length π , which includes both the lower and upper bound, e.g., $[0, \pi]$. This implies $\sin \alpha_2 \geq 0$. The other two angles, α_1 and α_3 , are defined on an interval of length 2π , which includes only one of the bounds, e.g., $[0, 2\pi]$.

The inverse rotation of $\mathbf{R} = \mathbf{R}_3 \mathbf{R}_2 \mathbf{R}_1$, represented in terms of its Euler angles ($\alpha_1, \alpha_2, \alpha_3$), is given as

$$\mathbf{R}^{-1} = \mathbf{R}_1^{-1} \mathbf{R}_2^{-1} \mathbf{R}_3^{-1} = \mathbf{R}(\hat{\mathbf{e}}_1, -\alpha_1) \mathbf{R}(\hat{\mathbf{e}}_2, -\alpha_2) \mathbf{R}(\hat{\mathbf{e}}_1, -\alpha_3). \quad (11.71)$$

This rotation is represented by the Euler angles $(-\alpha_3, -\alpha_2, -\alpha_1)$. So, inversion is straightforward to implement in the parameter space. The concatenation of two general rotations cannot be implemented as a reasonably practical expression in the parameters space. This is the case both for Euler angles and for any other three-angle representation of $SO(3)$.

11.5.1 Finding the Euler angles for \mathbf{R}

When we want to determine the angles $\alpha_1, \alpha_2, \alpha_3$ for a particular $\mathbf{R} \in SO(3)$, three distinct cases appear: (a) $[\mathbf{R}]_{11} \neq \pm 1$, (b) $[\mathbf{R}]_{11} = 1$, and (c) $[\mathbf{R}]_{11} = -1$. In case (a) it follows that α_1, α_3 can be determined from

$$\begin{pmatrix} \cos \alpha_1 \\ \sin \alpha_1 \end{pmatrix} = \frac{-1}{\sqrt{[\mathbf{R}]_{12}^2 + [\mathbf{R}]_{13}^2}} \begin{pmatrix} [\mathbf{R}]_{13} \\ [\mathbf{R}]_{12} \end{pmatrix}, \quad \begin{pmatrix} \cos \alpha_3 \\ \sin \alpha_3 \end{pmatrix} = \frac{1}{\sqrt{[\mathbf{R}]_{21}^2 + [\mathbf{R}]_{31}^2}} \begin{pmatrix} [\mathbf{R}]_{31} \\ -[\mathbf{R}]_{21} \end{pmatrix}. \quad (11.72)$$

In case (a) we have $\sin \alpha_2 > 0$, which implies that the two denominators in Equation (11.72) are non-zero.

We insert these values into \mathbf{R}_1 and \mathbf{R}_3 in Equation (11.69), and solve for \mathbf{R}_2 :

$$\mathbf{R}_2 = \mathbf{R}_3^\top \mathbf{R} \mathbf{R}_1^\top, \quad (11.73)$$

where the elements of \mathbf{R}_2 relate to the rotation angle α_2 in accordance with Equation (11.69). The rotation angle α_2 can then be derived from the “corner elements” of \mathbf{R}_2 in Equation (11.73).

For a general rotation we expect to see case (a). But, as indicated in Equation (11.72), the numerical accuracy and stability of these computations can be severely reduced when $\sin \alpha_2 \approx 0$. This happens when \mathbf{R} comes close to cases (b) or (c), which are what we investigate next.

Singularities

Case (b) occurs when $\alpha_2 = 0$, where the rotation matrix \mathbf{R} can be written as

$$\mathbf{R} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_1 + \alpha_3) & -\sin(\alpha_1 + \alpha_3) \\ 0 & \sin(\alpha_1 + \alpha_3) & \cos(\alpha_1 + \alpha_3) \end{pmatrix}. \quad (11.74)$$

From this follows that $\hat{\mathbf{e}}_1 = (1, 0, 0)$ is an eigenvector of \mathbf{R} with eigenvalue 1. Indeed, case (b) corresponds to a rotation by the angle $\alpha = \alpha_1 + \alpha_3$ about the axis $\hat{\mathbf{n}} = \hat{\mathbf{e}}_1$. Or, equivalently, it corresponds to a rotation by the angle $\alpha = -\alpha_1 - \alpha_3$ about the axis $\hat{\mathbf{n}} = -\hat{\mathbf{e}}_1$.

All Euler angles $(\alpha_1, 0, \alpha_3)$ where $\alpha_1 + \alpha_3$ is the same correspond to the same rotation matrix \mathbf{R} , representing a rotation about the axis $\hat{\mathbf{e}}_1$ by the angle $\alpha_1 + \alpha_3$. Consequently, this three-angle representation has singularities for all rotations about the rotation axis $\hat{\mathbf{e}}_1$. As a special case of case (b) we have $\mathbf{R} = \mathbf{I}$. This rotation is represented by Euler angles $(\alpha_1, 0, -\alpha_1)$ for any α_1 .

Case (c) occurs when $\alpha_2 = \pi$, where the rotation matrix \mathbf{R} simplifies to

$$\mathbf{R} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & \cos(\alpha_3 - \alpha_1) & \sin(\alpha_3 - \alpha_1) \\ 0 & \sin(\alpha_3 - \alpha_1) & -\cos(\alpha_3 - \alpha_1) \end{pmatrix}. \quad (11.75)$$

The rotation axis is in this case given by

$$\hat{\mathbf{n}} = \begin{pmatrix} 0 \\ \cos \beta / 2 \\ \sin \beta / 2 \end{pmatrix}, \quad \text{where } \beta = \alpha_3 - \alpha_1. \quad (11.76)$$

It is possible to add multiples of 2π to β , but this will only change the sign of $\hat{\mathbf{n}}$ for odd multiples. From Equation (11.18) follows that $\cos \alpha = -1$ for the corresponding rotation angle α . This gives $\alpha = \pi$ regardless of the sign we choose for $\hat{\mathbf{n}}$.

All Euler angles $(\alpha_1, \pi, \alpha_3)$, where $\alpha_3 - \alpha_1 + 2\pi n = \beta$, represent the same rotation matrix \mathbf{R} . This happens for any rotation by the angle $\alpha = \pi$ about $\hat{\mathbf{n}}$ in Equation (11.76). Consequently, this three-angle representation has singularities also for all rotations about axes in the plane perpendicular to $\hat{\mathbf{e}}_1$ if the rotation angle is π .

11.4 The representation of $SO(3)$ using Euler angles has singularities for all rotations about the axis $\hat{\mathbf{n}} = (1, 0, 0)$, and for the rotation angle π about any axis that is perpendicular to $\hat{\mathbf{n}}$.

The mapping from $\mathbf{R} \in SO(3)$ to a set of Euler angles $\alpha_1, \alpha_2, \alpha_3$, covering all three cases (a), (b), and (c), is summarized in Algorithm 11.3.

Algorithm 11.3: Find Euler angles corresponding to $\mathbf{R} \in SO(3)$.

```

Input:  $\mathbf{R} \in SO(3)$ 
Output: Euler angles  $\alpha_1, \alpha_2, \alpha_3$  that satisfy Equation (11.70)
1 if  $|\mathbf{R}_{11}| \neq 1$  then // Test if case (a)
2   Set  $s_1 = \sqrt{\mathbf{R}_{12}^2 + \mathbf{R}_{13}^2}$  and  $s_3 = \sqrt{\mathbf{R}_{21}^2 + \mathbf{R}_{31}^2}$ 
3   Determine  $\alpha_1$  from  $\cos \alpha_1 = -\frac{\mathbf{R}_{13}}{s_1}$  and  $\sin \alpha_1 = -\frac{\mathbf{R}_{12}}{s_1}$ 
4   Determine  $\alpha_3$  from  $\cos \alpha_3 = \frac{\mathbf{R}_{31}}{s_3}$  and  $\sin \alpha_3 = -\frac{\mathbf{R}_{21}}{s_3}$ 
5   Form matrices  $\mathbf{R}_1$  and  $\mathbf{R}_3$  from Equation (11.69)
6   Compute  $\mathbf{R}_2 = \mathbf{R}_3^\top \mathbf{R} \mathbf{R}_1^\top$ 
7   Determine  $\alpha_2$  from  $\cos \alpha_2 = [\mathbf{R}_2]_{11}$  and  $\sin \alpha_2 = [\mathbf{R}_2]_{31}$ 
8 else if  $[\mathbf{R}]_{11} = 1$  then // Test if case (b)
9   /* Case (b) */
10  Set  $\alpha_2 = 0$ 
11  Determine  $\alpha$  from  $\cos \alpha = [\mathbf{R}]_{33}$  and  $\sin \alpha = [\mathbf{R}]_{32}$ 
12  Set  $\alpha_1$  and  $\alpha_3$  such that  $\alpha_1 + \alpha_3 = \alpha$ . For example:  $\alpha_1 = \alpha, \alpha_3 = 0$ 
13 else // This is case (c)
14  Set  $\alpha_2 = \pi$ 
15  Determine  $\beta$  from  $\cos \beta = [\mathbf{R}]_{22}$  and  $\sin \beta = [\mathbf{R}]_{23}$ 
16  Set  $\alpha_1$  and  $\alpha_3$  such that  $\alpha_3 - \alpha_1 = \beta$ . For example:  $\alpha_1 = 0, \alpha_3 = \beta$ 
17 end
```

11.5.2 Summary

The three-angle representation uses $D = \mathbb{R}^3$, where the elements of $\mathbf{v} \in D$ are the three rotation angles about three fixed axes, $\hat{\mathbf{n}}_1, \hat{\mathbf{n}}_2$ and $\hat{\mathbf{n}}_3$. To use this type of representation in practice, we must specify the three axes. A common approach is the Euler-angle representation where $\hat{\mathbf{n}}_1 = \hat{\mathbf{n}}_3 = \hat{\mathbf{e}}_1$ and $\hat{\mathbf{n}}_2 = \hat{\mathbf{e}}_2$. In this case, Equation (11.70) defines the mapping f and Algorithm 11.3 defines the inverse mapping f^{-1} .

Three-angle representations have singularities. In the case of Euler angles, there are two sets of singularities. One set is given by any rotation about the axis $\hat{\mathbf{e}}_1$, and the other as rotations by π about any axis perpendicular to $\hat{\mathbf{e}}_1$. In both cases there are infinitely many combinations of α_1 and α_3 that all represent the same rotation \mathbf{R} .

In the case of Euler angles, we can restrict the domain for α_1 and α_3 to an interval of length 2π , for example $\alpha_1, \alpha_3 \in [0, 2\pi[$. If $\sin \alpha_2$ changes sign in Equation (11.70), this is equivalent to instead adding π to both α_1 and α_3 . This means that we can restrict $\sin \alpha_2$ to non-negative values: $\alpha_2 \in [0, \pi[$. Except for the singularities, we can use this type of restriction of the Euler angles to get an unambiguous representation. Other three angle-representations, with different choices of the three rotation axes, may have the three angles restricted to other ranges to make the representation unambiguous.

A rotation given by Euler angles $(\alpha_1, \alpha_2, \alpha_3)$ has an inverse rotation with Euler angles $(-\alpha_3, -\alpha_2, -\alpha_1)$. If we use the restriction of the Euler angles described above, the inverse can instead be represented by the Euler angles $(\pi - \alpha_3, \alpha_2, \pi - \alpha_1)$. There is no trivial way to determine the concatenation of two general rotations based on their Euler angles, or based on any other three-angle representation.

11.6 Which representation do I choose?

The basic properties of the algebraic representations for 3D rotations presented here are summarized in Table 11.1. As can be seen, there is no ideal representation, which does not have any issue that potentially can cause problems. If you need to manage 3D rotations in numerical computations, you should at least be aware of these issues, even if you have a preference of choice. But this choice cannot always be subject to taste or custom. In some cases, there are better and worse choices, and making a bad choice will influence the performance of any system that deals with rotations.

Regardless of which representation you use, make sure that you can avoid any singularity. Otherwise they can

Representation	Parameters	Ambiguities	Singularities
matrix	9 or 6 (3 DOF)	no, but must be consistent with Equation (11.1)	no
axis-angle	3+1 (2+1 DOF)	sign of both vector and angle, and modulo 2π for the angle	the identity rotation
Vector	3 DOF	modulo 2π on vector norm	no
$so(3)$, matrix exponential	3 DOF	modulo $\sqrt{8}\pi$ on Frobenius norm	no
$so(3)$, Cayley transform	3 DOF	no	incomplete: no 180° rotations
Unit quaternions	4 (3 DOF)	double embedding	no
Euler angles, as in Section 11.5	3 DOF	modulo 2π in α_1, α_3 , adding π to α_1 changes sign of α_1, α_3	rotations about $\hat{\mathbf{e}}_1$, π -rotations perpendicular to $\hat{\mathbf{e}}_1$

Table 11.1: A summary and comparison between the different algebraic representations of rotations in \mathbb{R}^3 presented in this chapter. The number of parameters in the second column reflects the number of scalars that are needed for the representation, where the number given as DOF is the true degrees of freedom for these parameters

potentially degrade the numerical computations that involve the rotations. As seen in the table, the only representations discussed there that are completely without singularities are the unit quaternions and the exponential mapping on $so(3)$. In practice, the exponential representation requires an effective implementation of the exponential mapping. This mapping is offered by many software platforms for numerical calculations, for example Matlab and Mathematica, but may not be available in most low-level libraries for computer vision, such as OpenCV.

The quaternion representation is straight-forward to use. Both the mapping f and its inverse f^{-1} are simple to implement, see Equation (11.60) and Algorithm 11.2 on page 178. It can also deal with both concatenation and inversion of rotations directly in the parameter space. This is not the case for the exponential mapping. Consequently, the unit quaternions offer an effective representation of $SO(3)$ that only comes at the cost of sometimes having to deal with the double embedding, i.e., $\pm q \in S^3$ represent the same rotation. The quaternions must also be normalized whenever they are manipulated, in particular before being mapped by f to a rotation matrix, Equation (11.60). In most cases you do not have to bother about the quaternion algebra, you just use f and f^{-1} defined by Equation (11.60) and Algorithm 11.2, respectively.

Based on these observations, the unit quaternions or the exponential mapping of $so(3)$ become the recommended representation of $SO(3)$. In particular, the quaternions can be used for all sorts of numerical computations that involve 3D rotations, e.g., in optimization problems where rotations are included. The details of this property will be further developed in Section 15.3.1, where optimization of functions that depend on rotations is discussed.

11.5 The reader is recommended to use unit quaternions, or the exponential mapping of $so(3)$, as a representation of $SO(3)$. They have no singularities.

A singularity free representation is of importance in some applications. In particular, this is the case if the rotations are completely general, and you cannot restrict them to some domain that is free of singularities. In Chapter 17 we will see that a singularity free representation is important for an efficient implementation of methods that optimize rotations. At, and close to, the singularities there is a risk of ill-defined computations for the iteration step of the optimization. This makes the Euler angle representation a particularly bad choice since it has a large set of singularities.

11.6 Do not use the Euler angle representation for 3D rotations, unless there is some very specific reason why it must be used. If used, make sure that the large set of singularities it has can be avoided.

In principle, this observation applies also to the axis-angle representation, and to the Cayley transformation.

The latter representation, at least, does not have its singularities close to \mathbf{I} , and it also has simple expressions for both f and f^{-1} .

11.7 Related topics

In the closing sections of this chapter we will discuss some topics related to representations of $SO(3)$. We will see how the different representations that have been presented in the earlier sections can be used to address specific issues related to rotations.

11.7.1 Uniformly sampled random rotations

In some applications it is necessary to generate random rotations. For example, in order to create a dataset that is used to evaluate some algorithm which deals with rotations, it may be necessary to demonstrate that a large set of random rotations are included in the set. Given the different representations of $SO(3)$ that have been presented here, it should not be too difficult to achieve a random sampling of the different parameters that are used to specify a particular rotation.

This solves the problem of randomness, but in some applications, we may even have to generate the set in such a way that the entire $SO(3)$ is sampled with a *uniform probability*. This means that every rotation is generated with the same probability as any other rotation. In general, a random sampling of the parameters that are used in a particular representation of $SO(3)$ will not produce a uniform sampling of the rotations, even if each parameter is uniformly sampled. The reason is the representation function f , which maps the parameters to $SO(3)$ in a non-linear way. As a consequence, any uniformity in the PDF of the original parameters will be lost for the PDF of the resulting matrices in $SO(3)$. At least, this is what will happen if we do not choose the representation, and also sample its parameters, with care.

In this section we present an easy approach to uniform random sampling of $SO(3)$, enabled by the quaternion representation described in Section 11.4. It is based on an article by Muller [51] about uniform sampling on the unit sphere.

An approach based on unit quaternions

In Section 11.4 we learned that $SO(3)$ has a double embedding in the set of unit quaternions, which we can identify as S^3 , the unit sphere in \mathbb{R}^4 . Consequently, a uniform sampling of S^3 produces a uniform sampling of $SO(3)$. The fact that each rotation has two possible realizations as quaternions is not relevant here, since this applies equally to all rotations.

So, how do we produce a uniform sampling of S^3 , a subset of \mathbb{R}^4 ? The first step is to sample \mathbb{R}^4 according to an *isotropic* probability density function (PDF). This means that if we look at a sample $\bar{\mathbf{u}} \in \mathbb{R}^4$, its direction has a uniform PDF. In principle, any isotropic PDF will do, but the easiest way to generate an isotropic PDF in an n -dimensional space is to form a vector $\bar{\mathbf{u}}$ where each of the n elements is independently sampled from a normal distribution, with zero mean and a common variance. The variance does not affect the isotropy, and can be arbitrarily chosen, e.g., set to 1.

In our case, we form a vector $\bar{\mathbf{u}} \in \mathbb{R}^4$ where each element is sampled from a normal distribution $N(0, 1)$. As a result, $\bar{\mathbf{u}}$ has an isotropic PDF in \mathbb{R}^4 . From this follows that $\hat{\mathbf{u}} = \bar{\mathbf{u}} / \|\bar{\mathbf{u}}\|$ also has an isotropic PDF, although it is restricted to lie in S^3 . Consequently, the normalized vector $\hat{\mathbf{u}}$ lies on the unit sphere in \mathbb{R}^4 , and is sampled with a uniform distribution on this set. In the context of rotations, we can identify $\hat{\mathbf{u}}$ with the unit quaternion that we want to generate.

This very simple approach of generating a random unit quaternion with uniform distribution is summarized in Algorithm 11.4. If a rotation matrix $\mathbf{R} \in SO(3)$ is required, rather than a quaternion, use Equation (11.60) to map the quaternion to \mathbf{R} .

11.7.2 Twisted rotations

The concept of twisted rotations appears in some of the chapters. In this section we define this concept and show its interpretation in different representations of $SO(3)$. Applications of this concept appear in Section 10.5.3.

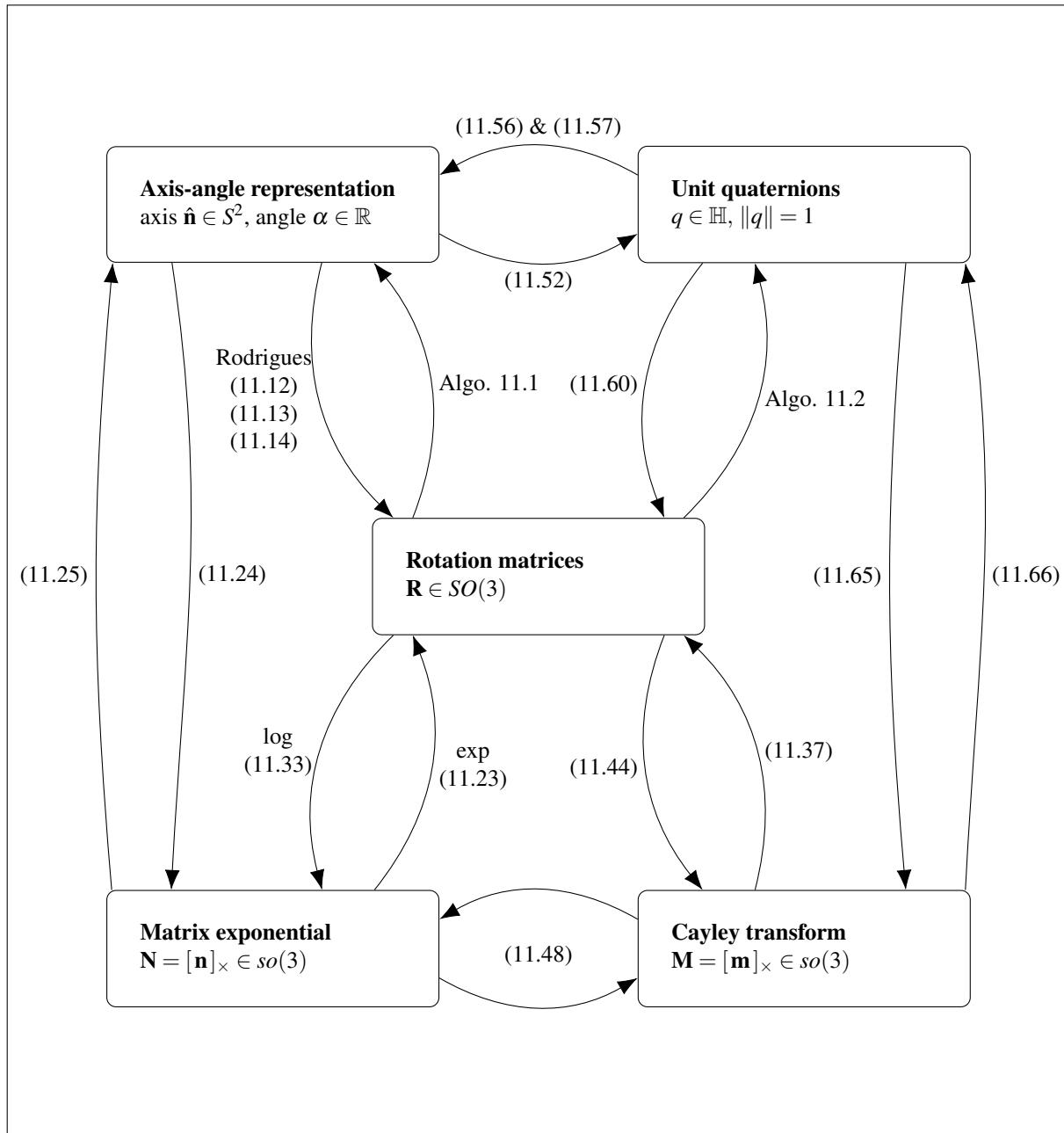


Figure 11.3: The different representations of 3D rotations presented in this chapter and how they are related.

Algorithm 11.4: Generates a random element of $SO(3)$ with a uniform PDF.

Output: A rotation \mathbf{R} , uniformly sampled from the set $SO(3)$.

- 1 Form a vector $\bar{\mathbf{u}} \in \mathbb{R}^4$, where each element is sampled from a normal distribution $N(0, 1)$
- 2 Set $\mathbf{q} = \bar{\mathbf{u}} / \|\bar{\mathbf{u}}\|$
- 3 Map the unit quaternion \mathbf{q} to \mathbf{R} , Equation (11.60)

Geometric definition

Two rotations $\mathbf{R}_1, \mathbf{R}_2 \in SO(3)$ form a *twisted pair*, or a pair of *twisted rotations*, if the combination $\mathbf{R}_2^\top \mathbf{R}_1$ is a 180° rotation about some axis $\hat{\mathbf{n}}$. We do not need to specify the two rotations in a specific order, the definition is symmetric. If $\mathbf{R}_1, \mathbf{R}_2$ form a twisted pair, also $\mathbf{R}_2, \mathbf{R}_1$ is a twisted pair. More generally, if $\mathbf{R}_1, \mathbf{R}_2$ is a twisted pair, then all the following four combinations are 180° rotations:

$$\mathbf{R}_2^\top \mathbf{R}_1, \quad \mathbf{R}_1^\top \mathbf{R}_2, \quad \mathbf{R}_2 \mathbf{R}_1^\top, \quad \mathbf{R}_1 \mathbf{R}_2^\top. \quad (11.77)$$

The first two combinations rotate about an axis $\hat{\mathbf{n}}$ and the last two about an axis $\hat{\mathbf{n}}'$ which, in general, is distinct from $\hat{\mathbf{n}}$. The two axes are related as

$$\hat{\mathbf{n}}' = \mathbf{R}_1 \hat{\mathbf{n}} = \mathbf{R}_2 \hat{\mathbf{n}}. \quad (11.78)$$

This gives an alternative characterization of a twisted pair of rotations: both \mathbf{R}_1 and \mathbf{R}_2 rotate $\hat{\mathbf{n}}$ to $\hat{\mathbf{n}}'$.

For a given rotation $\mathbf{R}_1 \in SO(3)$ there are infinitely many rotations $\mathbf{R}_2 \in SO(2)$ such that $\mathbf{R}_1, \mathbf{R}_2$ form a twisted pair. One way to form a twisted pair is to start with a fixed rotation $\mathbf{R}_1 \in SO(3)$ and a fixed rotation axis $\hat{\mathbf{n}}$, and then ask: which $\mathbf{R}_2 \in SO(3)$ together with \mathbf{R}_1 form a twisted pair such that $\mathbf{R}_2^\top \mathbf{R}_1$ is a 180° rotation about $\hat{\mathbf{n}}$? Based on Rodrigues' formula, this \mathbf{R}_2 must satisfy

$$\mathbf{R}_2^\top \mathbf{R}_1 = \mathbf{R}(\hat{\mathbf{n}}, \pi) = 2 \hat{\mathbf{n}} \hat{\mathbf{n}}^\top - \mathbf{I}. \quad (11.79)$$

From this equation we can solve for \mathbf{R}_2 and get

$$\mathbf{R}_2 = \mathbf{R}_1 (2 \hat{\mathbf{n}} \hat{\mathbf{n}}^\top - \mathbf{I}) = (2 \hat{\mathbf{n}}' (\hat{\mathbf{n}}')^\top - \mathbf{I}) \mathbf{R}_1. \quad (11.80)$$

An alternative is to solve for \mathbf{R}_1 :

$$\mathbf{R}_1 = \mathbf{R}_2 (2 \hat{\mathbf{n}}' (\hat{\mathbf{n}}')^\top - \mathbf{I}) = (2 \hat{\mathbf{n}} \hat{\mathbf{n}}^\top - \mathbf{I}) \mathbf{R}_2. \quad (11.81)$$

Here $\hat{\mathbf{n}}$ and $\hat{\mathbf{n}}'$ are related as in Equation (11.78).

Choosing two general 3D rotations requires $3 + 3 = 6$ parameters to be specified, but due to the constraint in Equation (11.82) the set of twisted rotations has 5 degrees of freedom. The geometric definition of twisted rotations can be reformulated in more algebraic terms, once a particular representation of $SO(3)$ is chosen. This what we will do next.

Matrix representation

From Equation (11.17), it follows that $\mathbf{R}_1, \mathbf{R}_2$ form a twisted pair exactly when

$$\text{trace}(\mathbf{R}_2^\top \mathbf{R}_1) = \text{trace}(\mathbf{R}_1^\top \mathbf{R}_2) = \text{trace}(\mathbf{R}_2 \mathbf{R}_1^\top) = \text{trace}(\mathbf{R}_1 \mathbf{R}_2^\top) = \cos \pi = -1. \quad (11.82)$$

In fact, all four expressions in Equation (11.82) that include a trace operation are equivalent for general \mathbf{R}_1 and \mathbf{R}_2 , and correspond to the Frobenius scalar product¹⁴ of the two rotations. The twisted pair condition therefore becomes

$$\mathbf{R}_1 \cdot \mathbf{R}_2 = -1. \quad (11.83)$$

¹⁴The Frobenius scalar product is defined in Toolbox Section 3.4.1.

Unit quaternions

Representation of rotations based on unit quaternions is described in Section 11.4. In this case, concatenation of 3D rotations is implemented in the parameter space simply as a product of quaternions. The quaternion representation of a 180° rotation about some axis $\hat{\mathbf{n}}$ is given as

$$\pm \left(\cos\left(\frac{\pi}{2}\right), \sin\left(\frac{\pi}{2}\right) \hat{\mathbf{n}} \right) = \pm(0, \hat{\mathbf{n}}). \quad (11.84)$$

Consequently, two unit quaternions $q_1, q_2 \in S^3$ form a twisted pair when the scalar part of $q_2^* \circ q_1$ vanishes. With the two quaternions represented as vectors $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{R}^4$, this relation can also be written in a more compact way as:

$$\mathbf{q}_2 \cdot \mathbf{q}_1 = 0. \quad (11.85)$$

Axis-angle representation

There are several relations between the axes and the angles of $\mathbf{R}_1 = \mathbf{R}(\hat{\mathbf{n}}_1, \alpha_1)$ and $\mathbf{R}_2 = \mathbf{R}(\hat{\mathbf{n}}_2, \alpha_2)$ that can be used to specify when the two rotations form a twisted pair. For example, inserted into Equation (11.85), two unit quaternions defined as

$$\mathbf{q}_1 = \begin{pmatrix} \cos \frac{\alpha_1}{2} \\ \sin \frac{\alpha_1}{2} \hat{\mathbf{n}}_1 \end{pmatrix}, \quad \mathbf{q}_2 = \begin{pmatrix} \cos \frac{\alpha_2}{2} \\ \sin \frac{\alpha_2}{2} \hat{\mathbf{n}}_2 \end{pmatrix}, \quad (11.86)$$

form a twisted pair if

$$0 = \cos \frac{\alpha_1}{2} \cos \frac{\alpha_2}{2} + (\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_2) \sin \frac{\alpha_1}{2} \sin \frac{\alpha_2}{2}, \quad \Leftrightarrow \quad \cot \frac{\alpha_1}{2} \cot \frac{\alpha_2}{2} = -\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_2. \quad (11.87)$$

Alternatively, we can start with Equation (11.82) and use Rodrigues' formula Equation (11.12) to express each of the two rotations. This results in

$$0 = (1 - \cos \alpha_1)(1 - \cos \alpha_2) + (\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_2) \sin \alpha_1 \sin \alpha_2. \quad (11.88)$$

Either of Equation (11.87) or Equation (11.88) can be used to define twisted rotations based on the axis-angle representation.

Cayley transform

Here, we have to assume that neither of \mathbf{R}_1 or \mathbf{R}_2 by itself is a 180° rotation. In this case we can find two matrices $\mathbf{M}_1, \mathbf{M}_2 \in so(3)$ such that their Cayley transforms are \mathbf{R}_1 and \mathbf{R}_2 , respectively:

$$\mathbf{R}_1 = \mathbf{R}(\hat{\mathbf{n}}_1, \alpha_1) = (\mathbf{I} + \mathbf{M}_1)(\mathbf{I} - \mathbf{M}_1)^{-1}, \quad \mathbf{R}_2 = \mathbf{R}(\hat{\mathbf{n}}_2, \alpha_2) = (\mathbf{I} + \mathbf{M}_2)(\mathbf{I} - \mathbf{M}_2)^{-1}. \quad (11.89)$$

where

$$\mathbf{M}_1 = \tan \frac{\alpha_1}{2} [\hat{\mathbf{n}}_1]_{\times}, \quad \mathbf{M}_2 = \tan \frac{\alpha_2}{2} [\hat{\mathbf{n}}_2]_{\times}. \quad (11.90)$$

The Frobenius scalar product of matrices \mathbf{M}_1 and \mathbf{M}_2 is

$$\mathbf{M}_1 \cdot \mathbf{M}_2 = \tan \frac{\alpha_1}{2} \tan \frac{\alpha_2}{2} ([\hat{\mathbf{n}}_1]_{\times} \cdot [\hat{\mathbf{n}}_2]_{\times})^{15} = 2 \tan \frac{\alpha_1}{2} \tan \frac{\alpha_2}{2} (\hat{\mathbf{n}}_1 \cdot \hat{\mathbf{n}}_2). \quad (11.91)$$

We combine this with Equation (11.87) to formulate as a necessary and sufficient condition that the two rotations represented by \mathbf{M}_1 and \mathbf{M}_2 form a twisted pair:

$$\mathbf{M}_1 \cdot \mathbf{M}_2 = -2. \quad (11.92)$$

¹⁵Here we are using Toolbox Section 3.7.3.4, Equation (3.155).

11.7.3 Singularities and gimbal lock

A practical example of what can go wrong if 3D rotations are represented in such a way that singularities can appear, is offered by the so-called gimbal lock, described next.

The singularities that are described for several of the representations mentioned here can have severe effects on a system that has to keep track of 3D rotations. This can be, for example, a system that performs numerical computations that involves rotations or a mechanical system that measures and adjusts rotations.

In the first case, we may be optimizing some cost function ε over all possible rotations, i.e., we want to determine a rotation in \mathbb{R}^3 that optimizes ε . Assuming that we are using a axis-angle representation, we know that there is a singularity for the identity matrix: all rotation axes are a valid representation of a rotation where the rotation angle is $\alpha = 0$. Let us assume that the optimum of the cost function lies very close to the identity rotation, i.e., it is a rotation by a very small angle. Let us also assume that the optimization process has successfully changed the rotation angle to a value equal to zero, or very close to zero. Furthermore, the rotation axis points in some direction given by the course of the optimization process. This state, the axis and the angle, then represents a rotation very close to the optimum of the cost function.

If this state does not represent the optimal rotation, we need to compute the gradient of both the rotation axis and the rotation angle in order determine how to update them to come closer to the optimum. Since we are close to the optimum, the update in the rotation angle is expected to be small. The same observation, however, does not apply to the rotation axis. The change in the rotation axis can be large, even if the resulting update of the corresponding rotation is small. A rotation about $\hat{\mathbf{e}}_1$ a very small angle is very similar to a rotation about $\hat{\mathbf{e}}_2$ a very small angle, even if $\hat{\mathbf{n}}_1$ and $\hat{\mathbf{n}}_2$ are quite different axes. Updating from one rotation to the other then means a large update in the rotation axis. As a consequence, the gradient in $\hat{\mathbf{n}}$ can be large, even very large, and can potentially cause numerical instability of the optimization process.

This observation is not unique for the axis-angle representation: any representation that has singularities can potentially become ill-behaved near these singularities. In certain practical applications it may be possible to avoid the singularities, but otherwise it is unsafe to use representations with singularities. As we will see in Chapter 15, this type of behavior in the gradients is not always observed close to the singularities of the representations described here. Another possibility is that the steepest descent process slows down close to the singularities, even if the optimal point is not reached.

A similar situation appears for a mechanical system that measures angles or rotations if it has singularities. In this case, the effect is known as *gimbal lock* and this term has spread also to other application areas as a general label of either numerical or mechanical problems that occur for isolated rotations, or for states close to these rotations. As a simple example of gimbal lock, consider a mechanical system that adjust a radar antenna that we want to lock onto aircrafts moving in the airspace of an airport. In this way, the antenna can measure the distance from its position on the ground to the vehicle. To do this, the antenna must mechanically change its elevation and azimuth angles by means of an automatic control system that tries to point the antenna onto the flying vehicle. The elevation angle refers to direction of the antenna above the horizontal plane, it can vary between 0° , when the antenna points at the horizon, and 90° when the antenna points straight up. In practice, the elevation angle can even go all the way to 180° to point in the opposite direction of the horizon relative to 0° . The azimuth angle controls in which horizontal direction the antenna is pointing, at zero elevation. It can vary from 0° to 360° .

Imagine a helicopter that approaches the airport from east, which in this example is represented by an azimuth angle of 0° . As the helicopter approaches the airport at some altitude above the ground, the elevation angle increases from close to 0° toward 90° , while the azimuth angle is constant at 0° . Assume that the helicopter flies all the way to a point straight above the radar antenna and begins to hovers there. The azimuth angle remains at 0° and the elevation angle now ends up at 90° . After a while the helicopter decides to fly due south and leave the airport. To allow the radar antenna to follow the helicopter on this route, the elevation angle has to slowly decrease from 90° all the way back to 0° . The azimuth angle, however, must instantly change from 0° to 90° in order to follow the helicopter on its new path. This discontinuous behavior in one of the two signals that control the direction of the antenna is in general difficult to accomplish, and any standard control system for locking the antenna onto an aerial vehicle will fail to do this robustly at, and close to, a singularity of this type. In this example, the singularity occurs at 90° elevation, since the azimuth angle is ambiguous for this case.

A more famous example, but also a more complicated one, of gimbal lock is given by the control system that supported the landing of the Apollo 11 Lunar Module [40].

Part II

Estimation

Chapter 12

Introduction to Estimation in Geometry

Before you read this chapter, you should have a thorough understanding of the homogeneous representations for \mathbb{E}^2 that are presented in Chapters 3 and 5. You should also have a look at the singular value decomposition, presented in Toolbox Section 8.2. Constrained optimization of quadratic forms is discussed in Toolbox Section 4.4.4. The presentation of Maximum-likelihood estimation in Section 12.6 is based on concepts from probability theory, presented in the Toolbox, Chapter 5.

This chapter presents a few basic methods for estimating geometric objects. The estimation uses data typically extracted from images, in this chapter they are in the form of point coordinates. While we present the different methods, specific issues of each method will be highlighted. These issues are important to understand when we use or select between the different methods. We will also show how assumptions about the data we use lead us to specific estimation methods. In later chapters, we will use these methods and apply them to specialized estimation problems. At that point, it is important to understand basic properties and possible limitations of the methods.

In this chapter we focus on the problem of how to estimate a line given a set of points in \mathbb{E}^2 . This is a simple problem, usually discussed already in secondary school, where it is solved by a technique called *least squares*. Still, we will look into this problem and discuss several different solution strategies. In later sections and chapters, we will return to this problem as a starting point for further discussion on the topic of estimation in geometry.

12.1 Least squares formulation

Let $\{\bar{\mathbf{y}}_i, i = 1, \dots, m\}$ be a set of m points in \mathbb{E}^2 which we assume are lying *approximately* on a straight line. For the moment, we leave aside the reason why they are not exactly on the line and return to this issue in Section 12.3. The Cartesian coordinates of the points are given as

$$\bar{\mathbf{y}}_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix}, \quad i = 1, 2, \dots, m. \quad (12.1)$$

We want to fit a line to these points. In secondary school, you probably never used dual homogeneous coordinates to describe a line in \mathbb{E}^2 , instead the line was described in terms of an equation like:

$$v = k u + l. \quad (12.2)$$

Equation (12.2) is satisfied for all points with Cartesian coordinates $\bar{\mathbf{y}} = (u, v)$ lying on this line. Here, (k, l) are the slope and intercept of the line, as described in Section 2.2, where also some warnings were raised about representing a line in this way. But we will stick to this simple and well-established formulation for now, and look at the homogeneous representation later on.

Since the points $\bar{\mathbf{y}}_i$ lie approximately on a line, we cannot expect that all of them satisfy Equation (12.2). This can also be formulated as: given the set of points in Equation (12.1), we cannot find a line with parameters (k, l)

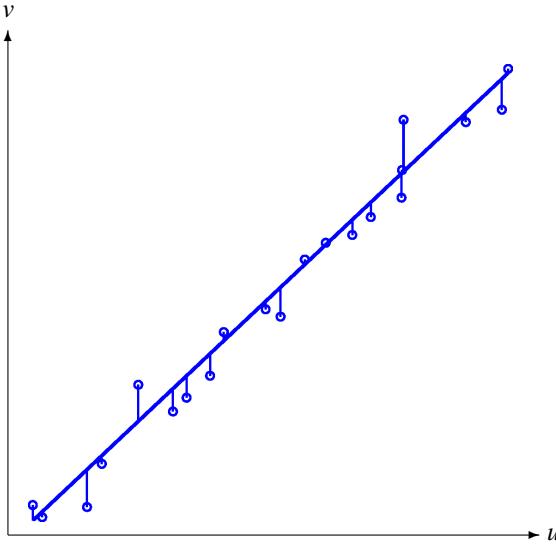


Figure 12.1: A set of points (circles) that is fitted to a line (solid) by minimizing ε_V in Equation (12.4). The figure also shows the vertical distances ε_i , Equation (12.3), for each point.

that meet Equation (12.2) for every point. To quantify how well Equation (12.2) is met, we define an error for each point \bar{y}_i as

$$\varepsilon_i = v_i - (k u_i + l), \quad i = 1, \dots, m. \quad (12.3)$$

The error ε_i is the signed distance¹ between the point \bar{y}_i and the line, but measured along the second (here vertical) coordinate axis rather than perpendicular to the line, see Figure 12.1. Each point \bar{y}_i contributes with an error ε_i , and a total or overall error can be formed as the sum of all these errors squared:

$$\varepsilon_V = \sum_{i=1}^m \varepsilon_i^2 = \varepsilon_1^2 + \dots + \varepsilon_m^2. \quad (12.4)$$

Since each ε_i is a function of (k, l) , so is ε_V too, and we refer to the latter as the *error function* of this estimation problem. In the literature *cost function*, *objective function*, or *energy function* are sometimes used as alternative terms for the error function. The problem of fitting a line to the points can now be formulated as follows: we want to determine the two parameters (k, l) that minimize the error function.

The solution to this problem is found by determining the stationary points of ε_V with respect to k and l . This implies finding (k, l) such that

$$0 = \frac{d\varepsilon_V}{dk} = -2 \sum_{i=1}^m \varepsilon_i u_i = -2 \sum_{i=1}^m (v_i - (k u_i + l)) u_i, \quad (12.5)$$

$$0 = \frac{d\varepsilon_V}{dl} = -2 \sum_{i=1}^m \varepsilon_i = -2 \sum_{i=1}^m (v_i - (k u_i + l)). \quad (12.6)$$

We solve these two linear equations in (k, l) and get the values for (k, l) that minimize ε_V as

$$k_V = \frac{s_{12}}{s_{11}}, \quad l_V = \frac{s_{11}s_2 - s_{12}s_1}{s_{11}}, \quad (12.7)$$

where

$$\begin{aligned} s_1 &= \frac{1}{m} \sum_{i=1}^m u_i, & s_2 &= \frac{1}{m} \sum_{i=1}^m v_i, \\ s_{11} &= \frac{1}{m} \sum_{i=1}^m (u_i - s_1)^2, & s_{22} &= \frac{1}{m} \sum_{i=1}^m (v_i - s_2)^2, \\ s_{12} &= \frac{1}{m} \sum_{i=1}^m (u_i - s_1)(v_i - s_2). \end{aligned} \quad (12.8)$$

¹Compare this to the signed distance between a point and a line defined in Section 3.4.2.

Here, (s_1, s_2) corresponds to the *mean* or the *centroid* of the point coordinates (u_i, v_i) , and s_{11}, s_{22} are the *variances* of the same coordinates. Finally, s_{12} is referred to as the *cross-correlation* between horizontal and vertical coordinates.

We can combine these expressions with Equation (12.7) and plug that into Equation (12.4) to get a *residual error*, the error for the optimal line, as

$$\epsilon_{V,\text{res}} = \frac{m(s_{11}s_{22} - s_{12}^2)}{s_{11}}. \quad (12.9)$$

Before we continue, let us summarize what we have done so far:

12.1 Estimation is about **fitting a model to data**.

In this example, the data is a set of points, or more specifically, their coordinates (u_i, v_i) for $i = 1, \dots, m$. The model is in this case a line, and it has some *model parameters*. Here, we have chosen the slope k and the intercept l as the model parameters. The process of fitting the model to the data implies that we seek values for the model parameters that minimize an error. In our example the error is ϵ_V , defined in Equation (12.4). We can now reformulate the previous observation in a less abstract way:

12.2 Estimation is about fitting a model to data. The model that is fitted to the data is **parameterized**. Fitting the model to the data implies **minimizing an error** between the model and the data, over all possible choices of the model parameters.

This last definition of what estimation is about describes estimation as an optimization problem. This opens the possibility of formulating estimation problems in more general ways. We can then solve these using more general optimization techniques. For special types of error functions, there are several standard optimization techniques, and we will look at some of them in this chapter. There are also methods that are applicable to more general cases, described in later chapters.

Another point to make here is that observation 12.2 describes estimation as a rather general problem: find a model that fits the data. The model does not have to be a line, and the data does not have to be a set of points. For example, we can deal with the problem of fitting a set of points (data) to a circle (model) in a similar way. We need to have some way of quantifying how well a specific point fits a circle, e.g., the smallest distance from the point to any other point on the circle. By adding the squares of these distances for all points in the set, we get a cost function ϵ . Finding the circle that minimizes ϵ , gives us an estimate of a circle that optimally fits the points.

In the general case, we denote the model as m and the parameters that can be adjusted to describe a specific model is represented as the vector $\mathbf{z} \in \mathbb{R}^p$. Since the model depends on the parameters, we write $m = m(\mathbf{z})$. The model is an element of some set or space. For example, it can be an element of the set of lines or an element of the projective space that holds dual homogeneous coordinates of lines.

Finally, observation 12.2 does not specify exactly how we should define the error function. As we will see, it can be defined in several different ways, and each new formulation also implies a new formulation of what *optimal* means. A model that is optimal relative to a specific error function may not be optimal relative to another. To verify this statement, we will look at an alternative error function that we can use for the line estimation problem.

Alternative least squares formulation

An example of an alternative error function to ϵ_V can be formulated based on the observation that, in most cases, there is no particular reason why we should measure distances between points and a line along the vertical axis. We could equally well choose to measure them along the horizontal axis:

$$\epsilon'_i = u_i - \frac{v_i - l}{m}, \quad (12.10)$$

and form an error function as

$$\epsilon_H = \sum_{i=1}^m \epsilon'^2_i = \epsilon'^2_1 + \dots + \epsilon'^2_m. \quad (12.11)$$

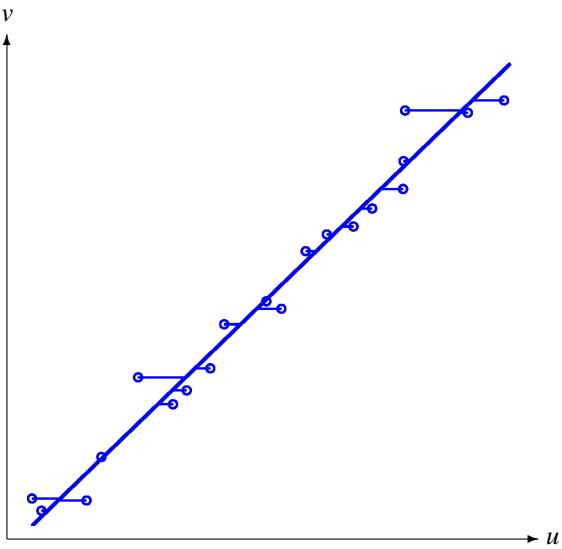


Figure 12.2: A set of points (circles) that is fitted to a line (solid) by minimizing ϵ_H in Equation (12.11). The figure also shows the horizontal distances ϵ'_i , Equation (12.10), for each point.

We minimize ϵ_H over (k, l) , and the optimal values are now given as

$$k_H = \frac{s_{22}}{s_{12}}, \quad l_H = \frac{s_{12}s_2 - s_{22}s_1}{s_{12}}, \quad (12.12)$$

with a corresponding residual error

$$\epsilon_{H,\text{res}} = \frac{m(s_{11}s_{22} - s_{12}^2)}{s_{22}}. \quad (12.13)$$

The formation of ϵ_H and the resulting line is illustrated in Figure 12.2 together with the optimally fitted line.

A comparison between the optimal parameters in Equations (12.7) and (12.12) shows that they are not the same, at least in the general case. This means that the line in Figure 12.2 is not identical to the one in Figure 12.1. Consequently, the resulting model does not only depend on the data, it also matters what error function we use:

12.3 It matters what error function is used for fitting the model to the data. In general, if we optimize two different error functions, the result are two different sets of optimal parameters. As a consequence, we should expect to **get different estimated models when different error functions are being used** in estimation.

In the next section, we will investigate yet another way of defining an error function for the line estimation problem.

12.2 Total least squares formulation

The error function ϵ_V makes sense if we can assume that the horizontal position of each point has a much higher accuracy than the vertical position. Similarly, if the vertical position has a much higher accuracy than the horizontal, we use ϵ_H . In many cases, the vertical and the horizontal position have the same degree of inaccuracy, and then we need to formulate the error in a different way, leading to a different optimal line.

The choice of error function does not only affect the resulting optimal model. It can also have large consequences for what types of computations that we need for computing the optimal model. We illustrate this with an example, again based on the line estimation problem. In this case, we may measure the error perpendicular to the line.

Finding a formulation of a solution to this problem becomes easier if we use homogeneous representations of both the points and of the estimated line. Let y_i be the homogeneous coordinates of point $i = 1, \dots, m$, and let \mathbf{l} be

the dual homogeneous coordinates of the line. We assume that \mathbf{y}_i is P-normalized and that \mathbf{l} is D-normalized:

$$\mathbf{y}_i = \begin{pmatrix} \bar{\mathbf{y}}_i \\ 1 \end{pmatrix}, \quad \mathbf{l} = \begin{pmatrix} \hat{\mathbf{l}} \\ -\Delta \end{pmatrix}, \quad \|\hat{\mathbf{l}}\| = 1. \quad (12.14)$$

This allows us to define a corresponding error as the signed distance (see Section 3.4.2) between point i and the line:

$$\varepsilon_i'' = \mathbf{y}_i \cdot \mathbf{l} = \bar{\mathbf{y}}_i \cdot \hat{\mathbf{l}} - \Delta. \quad (12.15)$$

The overall error is now given as

$$\varepsilon_{\text{TOT}} = \sum_{i=1}^m \varepsilon_i''^2 = \sum_{i=1}^m (\bar{\mathbf{y}}_i \cdot \hat{\mathbf{l}} - \Delta)^2 = \sum_{i=1}^m \left(\frac{\varepsilon_i'}{2} \right)^2 + \sum_{i=1}^m \left(\frac{\varepsilon_i'}{2} \right)^2 = \frac{\varepsilon_V + \varepsilon_H}{4}. \quad (12.16)$$

This implies that ε_{TOT} measures a deviation of each point relative to the line in both the horizontal and vertical direction. Hence, this approach is referred to as *total least squares* minimization.

We want to minimize ε_{TOT} over $\hat{\mathbf{l}} \in S^1$ and $\Delta \in \mathbb{R}$. Let us start with the latter. The optimal Δ is found where the derivative of ε_{TOT} with respect to Δ vanishes:

$$0 = \frac{d\varepsilon_{\text{TOT}}}{d\Delta} = -2 \sum_{i=1}^m (\bar{\mathbf{y}}_i \cdot \hat{\mathbf{l}} - \Delta), \quad \Rightarrow \quad \Delta = \bar{\mathbf{s}} \cdot \hat{\mathbf{l}} \quad \text{where} \quad \bar{\mathbf{s}} = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix}. \quad (12.17)$$

Here, $\bar{\mathbf{s}}$ is the centroid of the data, defined in Equation (12.8). ε_{TOT} can now be expressed as a function only of $\hat{\mathbf{l}}$:

$$\varepsilon_{\text{TOT}} = \sum_{i=1}^m ((\bar{\mathbf{y}}_i - \bar{\mathbf{s}}) \cdot \hat{\mathbf{l}})^2 = \hat{\mathbf{l}}^\top \mathbf{A}^\top \mathbf{A} \hat{\mathbf{l}}, \quad \text{where} \quad \mathbf{A}^\top = (\bar{\mathbf{y}}_1 - \bar{\mathbf{s}}, \bar{\mathbf{y}}_2 - \bar{\mathbf{s}}, \dots, \bar{\mathbf{y}}_m - \bar{\mathbf{s}}). \quad (12.18)$$

Here, \mathbf{A} is an $m \times 2$ matrix that holds the coordinates of the points in its rows, after they have been subtracted by the centroid $\bar{\mathbf{s}}$. In summary, we want to

$$\text{find } \hat{\mathbf{l}} \text{ that minimizes } \varepsilon_{\text{TOT}}(\hat{\mathbf{l}}) = \hat{\mathbf{l}}^\top \mathbf{A}^\top \mathbf{A} \hat{\mathbf{l}}, \quad \text{when} \quad c(\hat{\mathbf{l}}) = \hat{\mathbf{l}}^\top \hat{\mathbf{l}} = 1. \quad (12.19)$$

This constrained optimization problem can be solved using Lagrange's method,. In this case, the result is that the optimal $\hat{\mathbf{l}}$ is a normalized eigenvector corresponding to the smallest eigenvalue of $\mathbf{A}^\top \mathbf{A}$. The corresponding Δ is then given by Equation (12.17). The formation of this error and the resulting line is illustrated in Figure 12.3, together with the optimally fitted line.

• This optimization problem is discussed in Toolbox Section 4.4.4.

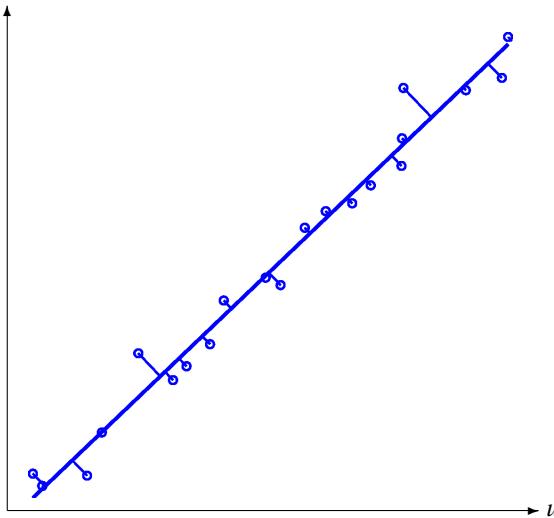


Figure 12.3: A set of points (circles) that is fitted to a line (solid) by minimizing ε_{TOT} in Equation (12.16). The figure also shows the distances ε_i'' , Equation (12.15), for each point.

Using SVD instead of EVD

Toolbox Section 8.2 describes the relation between eigenvalues of $\mathbf{A}^\top \mathbf{A}$ and the singular values of \mathbf{A} . It also discusses numerical advantages of these two operations. It means that we can reformulate our solution as: $\hat{\mathbf{l}}$ is a right singular vector corresponding to the smallest singular value of \mathbf{A} . Another reformulation is: $\hat{\mathbf{l}}$ is a left singular vector corresponding to the smallest singular value of \mathbf{A}^\top . In the following, we use SVD rather than EVD for solving this and similar types of problems. It does not mean that EVD is unsuitable for estimation problems, but rather that SVD in general is more reliable. The choice between SVD and EVD is in practice also determined by which one is easiest to use in the system where we compute the solution.

Conclusions

We now have a third formulation of the line fitting problem, when the error is defined as in Equation (12.16). In addition, we have also changed the line parameters, from (k, l) to $(\alpha, \Delta L)$. As a consequence, the computations involved in finding the solution are rather different from those in Section 12.1. When the error is defined as in Equation (12.4) or Equation (12.11), we get the line parameters by solving an ordinary linear equation. When the error instead is formulated as in Equation (12.16), we end up dealing with an SVD (or EVD) of a matrix. With these results at hand we can add to observation 12.3 on page 194:

12.4 The choice of error function also has a consequence for the **type of computations** that are used to obtain the optimal model.

Depending on what type of model that we want to estimate, there are certain choices of the error function that lead to simple computations, as those that we have seen so far. Other choices can instead lead to much more complicated and even also unreliable computations. From the outset, the first category is more attractive and often a first choice whenever possible. But, in some applications these “simple” error functions do not correspond to a realistic formulation of the problem we want to solve. Instead, the second category must be used. We will continue this discussion in Chapter 13, which discuss more complicated estimation problems.

12.3 What causes the errors? (Part I)

So far and without explanation, we have assumed that an optimal model *cannot* be fitted perfectly to all points in the data set. There may be many reasons for the mismatch between data and model, and we discuss some of them in this section. This topic is continued in Section 17.1 when more theory is established.

12.3.1 Measurement errors

Most of the estimation problems discussed in this presentation find a model that fits a data set consisting of points. These are often image points, but could also be 3D points. To be sure, there are plenty of interesting estimation problems where the observed data comes in the form of lines, or planes, or other geometric objects. But, by far, the most common types of estimation problems are based on observations of points. Very often, they are points that have been detected as interest points in one or several images, as discussed in Section 9.1.1. In some cases, image points are combined with 3D points that have been measured or estimated relative to some 3D coordinate system.

The image points have coordinates defined relative to some coordinate system in the image. The important question here is: with what accuracy can these coordinates be determined? We have already touched upon the answer to this question in Section 9.1.1, where interest points and their position accuracy were discussed. Observation 9.2 on page 124 states that the point coordinates extracted from digital images have a limited accuracy. This is motivated by the underlying measurement process, and it applies both to automatic interest point detection, as well as to manual measurements of coordinates using a ruler. In the context of model estimation, these inaccuracies translate to *measurement errors* in the data, or *data error*.

For example, already the fact that most simpler methods for interest point detection generate integer valued coordinates implies that they are determined with an error of ± 0.5 pixel, or larger. In addition to this, the interest point extraction method itself often gives errors due to assumptions that are only approximately correct in a digital

image. Furthermore, in Chapter 18, we will see that a pinhole camera model is only an approximately correct model of how a real camera depicts 3D space. So-called lens distortion displaces the positions of image points in a systematic way. The perturbations of all these effects add up, and even on a good day they result in measurement errors that can be several pixels for an interest point in an image.

Measurement noise

A common way of dealing with measurement errors, in particular when they relate to image coordinates $\bar{\mathbf{y}} \in \mathbb{R}^2$, is to assume that they can be expressed as

$$\bar{\mathbf{y}} = \dot{\mathbf{y}} + \bar{\boldsymbol{\eta}}. \quad (12.20)$$

Here, $\dot{\mathbf{y}} \in \mathbb{R}^2$ holds the ideal and accurate Cartesian coordinates. The measurement error is represented by a random vector variable $\bar{\boldsymbol{\eta}} \in \mathbb{R}^2$, generated by a corresponding *probability density function* (PDF) $P_{\text{noise}}(\bar{\boldsymbol{\eta}}) : \mathbb{R}^2 \rightarrow \mathbb{R}$. In this probabilistic context, the error $\bar{\boldsymbol{\eta}}$ is often referred to as *measurement noise*.

We will refer to $\dot{\mathbf{y}}$ as the *ideal data* or *unperturbed data*, given by a noise-free measurement that is produced when all geometric assumptions are valid, for example the pinhole camera model. The ideal coordinates cannot be determined by measurements on the image, and are in practice always unknown. Instead, $\bar{\mathbf{y}}$ is an *observation* of the same data. The observed coordinates can be obtained by measurements on the image, they are known, but they are also perturbed by the measurement noise. In this context, where $\bar{\boldsymbol{\eta}}$ is drawn in accordance with its PDF, $P_{\text{noise}}(\bar{\boldsymbol{\eta}})$, for each new measurement. The observed coordinates $\bar{\mathbf{y}}$ can even get a new value each time they are observed, or measured.

The measurement error $\bar{\boldsymbol{\eta}}$ in Equation (12.20) is additive, which makes this error model too simple in some applications. But as a first order approximation it works well for many types of measurement processes. We typically also assume that $\bar{\boldsymbol{\eta}}$ has zero mean, and that its PDF is isotropic:

$$P_{\text{noise}}(\bar{\boldsymbol{\eta}}) = P_0(\|\bar{\boldsymbol{\eta}}\|). \quad (12.21)$$

Here, $P_0 : \mathbb{R}^+ \rightarrow \mathbb{R}$, is a suitable function in the sense that its integral over all possible $\bar{\boldsymbol{\eta}} \in \mathbb{R}^2$ must equal one.

In addition, we often assume that errors related to two distinct points are *independent and have identical distributions*, often denoted as *IID*. As we will see, this set of assumptions enables us to formulate useful results about some estimation methods, but they are often a simplification of how the error distributions appear in reality.

12.5 Measurement errors of Cartesian coordinates can often be modeled as an additive and independent random variable with zero mean, which has an identical PDF at every point that is isotropic. In many applications, this is a useful first order approximation of what the error really looks like.

Notice that the noise model described here is applicable also to 3D coordinates. For certain applications, we may need to introduce further assumptions about the PDF of the measurement noise, e.g., that it has a Gaussian distribution.

12.3.2 Model errors

At the end of the day we want to fit data to a model, but what if we are using the wrong model, or if some of the data approximately fits the model and another part does not? Consider the 2D points that appear in the left part of Figure 12.4. We can fit a line to these points, using any of the approaches described earlier in this chapter, for example leading to the line in the center part. But we also see that the optimal line has a large error to many of the points, i.e., the residual error is large. Given the configuration of the points, it seems a better idea to instead fit them to a circle. This could lead to the result in the right part of the figure. The phrase “better idea” suggests that if we choose a circle, instead of a line, the residual error becomes smaller.

A variant of this theme is illustrated in Figure 12.5. To the left we see a set of points that approximately can be fitted to either of two lines. If our model is a single line, and we try to fit it to this data, the result could be the line in the center part. It is optimally fitted to all points in the data set, but forms a meaningless result in terms of model fitting. If we instead use a more advanced model that includes two lines, and fit them simultaneously to the data set, we could get the result illustrated in the right part of the figure. Clearly, the double line model is better

suit for this particular data set, compared to a single line. The former produce a lower residual error compared to the latter.

These two examples show the importance of choosing a suitable model that is fitted to our data. If we choose an unsuitable model and fit it to the data, we should expect the residual error to be large. Therefore, residual errors can often be used to quantify the quality of the resulting model. If the residual error is small, of the same magnitude as the inaccuracies of the data, we can assume that the model is suitable. If the residual error is larger, this is an indication that the model may be wrong, we have *model error*. Even if we solve the estimation problem and produce an optimally fitted model, it may be a useless result if the corresponding residual error is very large because of model error.

These last observations may lead to the conclusion that we should use a complex model, with many parameters that can be tuned to accomplish a snug fit between the model and the data. Although multi-parameter models can be useful in certain situations, they are also deceiving. For example, the points in the left part of Figure 12.6 may be fitted to a line, where the result is shown in center part. The line fits nicely to the points, but there is some amount of residual error. Alternatively, they may be fitted to a high order polynomial. If we choose a polynomial curve of sufficiently high order it can even be fitted perfectly to all points, as illustrated in the figure to the right in the figure. We will return to this curve shortly.

In practice, the choice of model is often guided by knowledge about the process that has generated the data. In fact, the more information we have about this process, the easier it is to find a useful model. In Section 12.3.1 we learned that observed data is always affected by measurement noise. Therefore, it is not reasonable to expect the model to fit the data *exactly*. The noise can be seen as part of the model. If we know that the data has been produced by some type of measurement, we should expect some amount of residual error. If we know the PDF of the measurement noise, and how the model is estimated, we may even be able to predict the PDF of the residual error.

Consequently, the data contains noise and is inherently unreliable in terms of accuracy. Ignoring this fact leads to *over-fitting*: choosing a complex model with lots of free parameters that can be optimized to fit the model almost perfectly to the data. Even though the resulting model has a good fit to the data, to a high degree it is influenced by the noise in the data. Had the same data been produced by another measurement process, which perturbs the data in a different way, the optimally fitted model would also be different. For example, minor changes in the positions of the points in Figure 12.6 would generate a completely different polynomial curve in the right part of the figure. This is not the expected behavior of a model, it should not change if only the measurement noise changes. Consequently, the polynomial curve in the right part of Figure 12.6 is in most cases not a useful model of this data set.

One way of thinking about models, which summarizes what already has been said, is that a useful model should *explain* the observed data, or be able of *generating* the data. If the model states that the observed points lie on a line but have perturbed by additive noise, then this model would explain the data seen in Figure 12.1. We can also say that this data is likely to have been generated by the stated model. However, the same model is not able to explain the data illustrated in Figures 12.4 or 12.5. It is very unlikely that this model has generated the data in these figures.

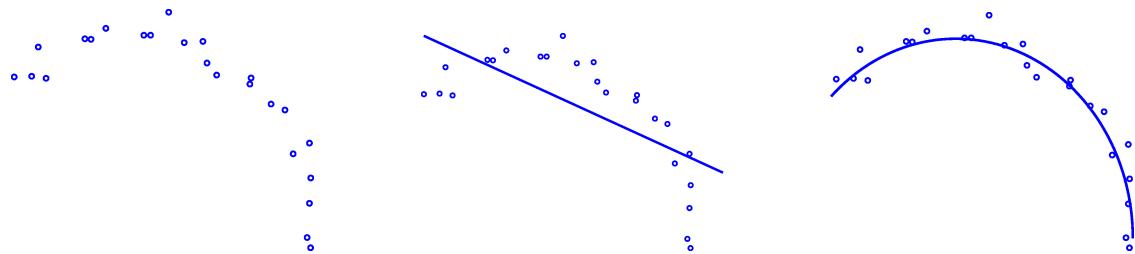


Figure 12.4: Left: a set of points. Center: A line that is optimally fitted to the points. Right: A circle that is optimally fitted to the points.

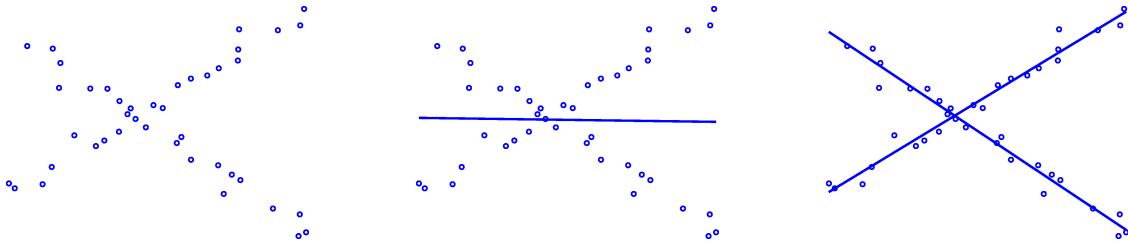


Figure 12.5: Left: a set of points. Center: A line that is optimally fitted to the points. Right: Two lines optimally fitted to the points.

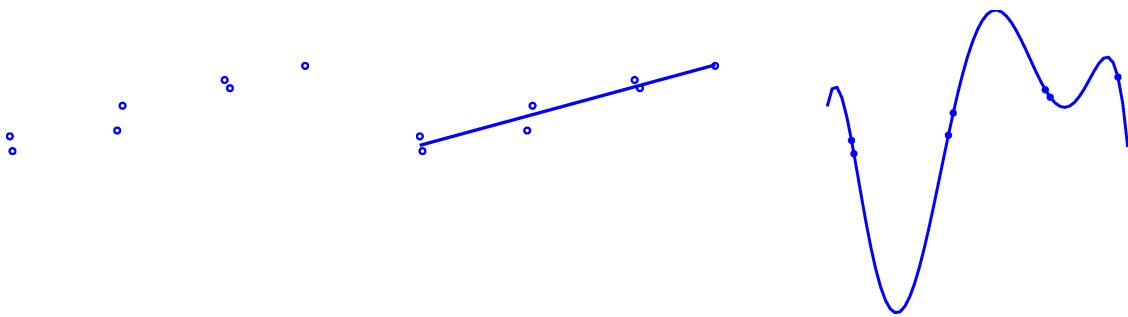


Figure 12.6: Left: a set of points. Center: A line that is optimally fitted to the points. Right: A high order polynomial that is exactly fitted to the points.

We summarize these observations as:

12.6 The model that is fitted to the data should be carefully chosen. It should be guided by as much knowledge as possible about the data generation process. A good model, including the measurement noise, should be able to generate the observed data.

and

12.7 Mind the residual error of the estimated model. It contains information about the model error. If the residual error is too large it probably means that the estimated model is not useful. If it is too small, there is a risk of over-fitting.

12.3.3 Outliers and inliers

There is another possibility for discrepancies between model and data. In some applications we can simple assume that all data is generated from some model, which normally includes some amount of measurement noise. We just need to pick the right model and then find the parameters that make the model fit optimally to the data.

In other applications, the data set may also contain points that cannot be fitted to any model what so ever, while the remaining points can. The former points are referred to as *outliers*, while the latter points are *inliers*. It should be noted already here, that the partition of the data set into inliers and outliers is often not a clear case. A more correct distinction is to say that, in general, inliers can be fitted with small errors to the model, while the outliers have large errors. The difficulty lies in specifying a threshold for the error that robustly distinguishes the inliers from the outlier. Such a threshold can be determined in a reasonable way, but there is still a small chance for

misclassification, in both directions.

Up to now, the idea of inliers and outliers may not make much sense, if you have not seen any application where these concepts appear in a natural way. However, one way that an outlier can be generated is that the measurement noise, by chance, has been given a very large value for a particular observation of the data. In this case, that observation would lie very far from the generating model and could constitute an outlier.

A more common example has to do with correspondences. As we will see later on, many applications rely on having established correspondences between points in two or more images, or between image points and 3D points. In practice, a statement about correspondence can often not be made with full certainty. Instead, we can make hypotheses about correspondence between points. Such a hypothesis can then be tested, e.g., based on the epipolar geometry between two images in the form of the fundamental matrix \mathbf{F} that exists between the images.

As we will see in Section 16.2, when this \mathbf{F} is estimated, each data comes in the form of pairs of corresponding image points. But, as was pointed out above, these pairs can only be hypothetical correspondences. They can be correct or incorrect correspondences, and we cannot know until they have been tested which one it is. Each incorrect correspondence forms an outlier, it cannot be fitted to the model at hand, in this case to \mathbf{F} .

Dealing with outliers is an issue of its own, and we will give it an extensive treatment in Chapter 17. For now, we simply notice that outliers are, in principle, a possibility that we may have to deal with. In the rest of this chapter, and in several of the following chapters, we assume that the data consists of inliers only, unless outliers are explicitly mentioned.

12.4 Geometric errors

The errors discussed in Sections 12.1 and 12.2 are *geometric errors*: they refer to distances in \mathbb{E}^2 . More precisely, they are distances between points and a line. For the line fitting problem, we have seen that this distance can be measured in different ways. For example, we can do this along the vertical axis, along the horizontal axis, or perpendicular to the line. Different ways of measuring the distances produce different types of error functions, but they all have a geometric character. In other applications, geometric errors may instead refer to distances between two points, or between points and a plane in \mathbb{E}^3 . Even angles can be used as a type of geometric error.

In particular, a geometric error is a quantity that is independent of how we choose the underlying coordinate system. We have seen different examples of how to define an error function ε . This function includes quantities derived from coordinates or parameters, e.g., of points and lines. These coordinates and parameters, in turn, depend on how we specify the coordinate system. If ε is a geometric error, it must not change its value when a rigid transformation changes the coordinate system.

Definition 12.1: Geometric error

A geometric error is a quantity that measures how well a model fits data, based on geometric features, often distance, between the model and the data. The error should be independent of rigid transformations (rotations and translations) of the coordinate system.

An estimation methods that minimize a geometric error is often referred to as performing *geometric estimation*.

Since geometric errors often have a direct interpretation, they are intuitive to use. For example, we can understand what it means that an error is twice as large as another: some distance is twice as large as another distance, or the average distance is twice as large as another average distance. We can also often express geometric errors using a physical unit of measurement. For 3D points this unit can, for example, be *meters*. For 2D points the unit can still be meters, but a more common unit is *pixels*, which refers to the pixel-to-pixel distance in a digital image.

When we use geometric errors, their magnitudes often have a direct and intuitive interpretation. For example, it makes sense to reason about how much a 1 pixel error is, compared to 100 pixels. Geometric error can also easily be related to measurement errors that we expect to see in the point coordinates, which are discussed more in detail in Section 12.3. A geometric residual error should be of the same magnitude as the total amount of measurement errors that we have in the data.²

So, why not always use geometric errors, and are there any alternative? To answer the first question, we return to the three error functions that we have discussed so far: ε_V in Equation (12.4), ε_H in Equation (12.11), and ε_{TOT}

²Although the term “total amount of measurement errors” can have different meanings for different types of geometric errors.

in Equation (12.16). These functions have in common that the error which we want to minimize is a quadratic (or second order) expression in the model parameters. As a consequence, their first order derivatives with respect to the parameters are linear functions. The optimal parameters are then found by solving linear equations, a trivial and straightforward task in most cases.

The above observation is important. It suggests that the reason why we easily can minimize our errors is not so much because that we are using geometric errors. It is rather that the particular error functions we have considered so far happen to be of a type that leads to simple computations, and they also happen to be geometric errors. This is not, however, a general character of geometric errors. Shortly, and in later chapters, we will see plenty of geometric errors that cannot be minimized by solving linear equations. Instead the minimum is found by solving non-linear equations in the unknown model parameters. To find the minimum of these geometric error, we must then use non-trivial methods: *non-linear optimization*.

12.8 Unless there are other issues involved, choose an error function that leads to simple computations of the model parameters. This is the main motivation why sums of squared distances are popular.

12.4.1 Examples of geometric errors

We have already seen three examples of how to define a geometric error for estimation of a line from a set of points. Of these, ε_{TOT} is the most commonly used, since it defines an isotropic error that is independent of the coordinate system's orientation. We now return to this geometric error, but place it in a more general context, and then present other common examples of geometric errors.

For more general estimation problems than the line fitting problem, we can use d_i to denote a distance between the estimated model and some observed data with index i . It may be the Euclidean distance between some manifestation of the model and observed data i . But it could also be a distance that is measured along some curve, or some other geometric feature that quantifies the fit between the model and observed data i . In general, we require only that $d_i \geq 0$ for each i .

L_2 error

The error ε_{TOT} is sometimes also referred to as an *L_2 error*, or a *sum of square differences (SSD)*. It can be defined in terms of the distances d_i as

$$\varepsilon_{\text{SSD}} = \sum_{i=1}^m d_i^2. \quad (12.22)$$

if we want ε_{SSD} to have the same physical dimension as the distances d_i , an alternative is to consider an error function like

$$\varepsilon_{\text{SSD}'} = \sqrt{\sum_{i=1}^m d_i^2}. \quad (12.23)$$

In practice, however, this formulation is less useful. It is minimized by the same model parameters as ε_{SSD} is, but it is also more complicated to work with in mathematical derivations.

Weighted L_2 error

In some applications, the distances related to different observations may have different weights. In these cases, each weight w_i defines how much the corresponding distance d_i should influence the error function. Formally, the error function is then defined as

$$\varepsilon_{\text{WSSD}} = \sum_{i=1}^m w_i d_i^2. \quad (12.24)$$

This is referred to as a weighted L_2 -error, or weighted sum of squared differences.

When weights appear in an error function, this typically happens in applications where each observation of data has a confidence, or certainty. This confidence translates into a weight w_i that quantifies the influence of the distance d_i in the error function. If weight w_i is relatively large, it becomes important to reduce the distance d_i to

also make the residual error small. If the weight is relatively small, it does not matter if d_i is large, the residual error can still be small.

For the line estimation problem, the introduction of weights in the error function imply that the solution, i.e., the parameters of an optimal line, need only a minor modification. We leave this as an exercise to the reader. The unweighted L_2 -error in Equation (12.22) can be seen as a special case of Equation (12.24), where the weights are equal.

L_1 error

Although L_2 -errors are common in estimation problems, from the outset it is not obvious why the distances should be squared before they are added in Equation (12.22). A more intuitive definition of a geometric error function is to just add the distances:

$$\epsilon_{\text{SAD}} = \sum_{i=1}^m d_i. \quad (12.25)$$

This error function is often referred to as an L_1 *error* or a *sum of absolute differences (SAD)*.

We will shortly compare the L_1 -error and the L_2 -error, and see which one is most useful in different situations.

Weighted L_1 error

In a similar way as for the L_2 error, we can have a weight w_i for each distance d_i also in the L_1 -error:

$$\epsilon_{\text{WSAD}} = \sum_{i=1}^m w_i d_i. \quad (12.26)$$

L_p -error and Max-error

Both the L_1 -error and the L_2 -error can be seen as special cases of the L_p -error:

$$\epsilon_{Lp} = \sum_{i=1}^m d_i^p, \quad \text{or} \quad \epsilon'_{Lp} = \left(\sum_{i=1}^m d_i^p \right)^{1/p}. \quad (12.27)$$

To make sense mathematically, we have to assume here that $p \geq 1$.

Yet another special case of the L_p -error occurs if we consider ϵ'_{Lp} when $p \rightarrow \infty$. In the limit case, this gives the maximum of all distances d_i , and defines the *max-error*:

$$\epsilon_{\text{MAX}} = \max(d_1, d_2, \dots, d_m). \quad (12.28)$$

Median

The max-error is found by going through the entire set of distances, and determining which one of them is the largest. This can be extended to a full sorting of all the distances, from which we can extract the middle value³ of all the distances. This value is the median distance, and it defines the *median-error*:

$$\epsilon_{\text{MED}} = \text{median}(d_1, d_2, \dots, d_m). \quad (12.29)$$

Analysis

To understand which one of these different options to use for geometric errors, we need to investigate their characteristic properties and differences.

If we compare ϵ_{SAD} and ϵ_{SSD} , the former is perhaps more intuitive: just accumulating the individual errors d_i is the simplest way of combining them. But ϵ_{SAD} has a downside, its gradient, consisting of derivatives with respect to the different model parameters, is non-trivial. Each derivative typically includes a sign that depends on the corresponding unknown parameter. This makes it difficult to efficiently minimize ϵ_{SAD} .

³If m is odd, the middle value is represented by the distance at position $\frac{m+1}{2} m$ after sorting. If m is even, the median can be represented either by the average of the two central elements, or by either of them.

On top of that, ε_{SAD} has a discontinuous derivate with respect to the model parameters whenever $d_i = 0$. This means that at, or near, these singularities in ε_{SAD} , components in the gradient can suddenly change their sign. As a consequence, we must expect that a non-linear optimization process that is based on these gradients will not work satisfactory for a subset of the model parameters.

These above issues are shared also by the weighted L_1 -error $\varepsilon_{\text{WSAD}}$ in Equation (12.26). But they are solved are solved by the error function ε_{SSD} in Equation (12.25). The squaring of each distance assures that the gradient is well-defined for all models⁴ and can be given a simple expression in the model parameters. The same observation can be made for the weighted L_2 -error $\varepsilon_{\text{WSSD}}$ in Equation (12.26). This is the main motivation why L_2 -errors are very common in estimation problems, with or without weights. But L_2 -errors have other potential problems, which we will return to shortly.

In most applications that uses weights for the distances d_i , each w_i is fixed, it depends only on i . In principle, each weight can be a function also of the estimated model. This enables us to make a connection between L_1 and L_2 -errors. We can see ε_{SSD} as a weighted L_1 -error, where each weight equals the corresponding distance: $w_i = d_i$.

The same observation applies to the L_p -errors: we can see each term d_i^p as the distance d_i times a weight d_i^{p-1} . The larger p is, the more will each distance influence its own weight, thereby trying to make it small for the optimal model. Already at $p = 2$ (the L_2 -error), there is a significant influence from each distance on how it is weighted by itself. As a consequence, the optimization wants to reduce the largest distances for the optimal model. In the extreme case of $p = \infty$, corresponding to the max-error, the optimization is only focused on reducing the single largest distance. As a consequence, this resulting model can have several large distances, as long as they are less than the largest one. If instead $p = 1$, the optimal model can accept one or a few large distances, as long as the majority of errors d_i are small.

Finally, the median-error can be seen as an extreme form of L_1 -error. One or a few large distances, in fact, all distances larger than the median distance, are ignored by the median-error. This makes it possible to estimate a reasonable model from data that contains one or a few points that are completely off, so-called *outliers*. We will return to this application of the median-error in Chapter 17. As a downside, this error requires that the individual errors d_i must be sorted each time ε_{MED} is evaluated, an operation with relatively high computational complexity.

It deserves to be mentioned that a gradient of the error function cannot even be formulated as a closed form expression for ε_{MAX} and ε_{MED} . As a consequence, finding the optimal model parameters is non-trivial for these error functions. As was mentioned above, also ε_{SAD} has problems with its gradient. This means that standard techniques, such as those described in [54] Chapter 9, cannot be used in the optimization. Instead, dedicated methods have to be applied if these error functions are used. The only trivial option is ε_{SSD} , which still requires non-linear optimization methods to minimize. Unless each d_i is a purely quadratic function in the model parameters, the gradient of ε_{SSD} is a non-linear function in these parameters which cannot be solved easily.

12.9 Use geometric errors, but be prepared that the corresponding minimization problem in general leads to non-trivial optimization methods.

As an illustration of how the different geometric errors behave, Figure 12.7 shows a set of five point, approximately lying on a line. Lines are fitted to these points by minimizing ε_{SSD} , ε_{SAD} , ε_{MAX} , and ε_{MED} . Notice that one of the points is perturbed significantly more than the other four points, it may be an outlier.

As a first observation from this figure, we see that the four different error functions produce four different lines. This may or may not happen, depending on the data, but for this set of points the lines are clearly distinct. We also see that the two lines that are estimated by minimizing ε_{SAD} and ε_{MED} do not try much to fit the outlier point. Instead, they focus on the four remaining points that are relatively well organized on a line. In fact, the line corresponding to the ε_{MED} ignores the outlier point completely. After sorting, the distance for this point always ends up at the top, but how large it is does not matter. Therefore, moving the outlier point farther away from the original line will not affect the estimated line in this case.

The ε_{SAD} -error instead adds all the individual errors. Therefore, the corresponding line is affected by the outlier point, which in this case “draws” the line slightly in the direction of the outlier. Even more so, this is the case for the line corresponding to the ε_{SSD} -error, where the distance is used as a weight. As we can see, this line is pulled farther towards the outlier point. Finally, the ε_{MAX} -error cares only about the largest distance and, consequently, it is affected to a very large degree by the outlier point.

⁴Unless the distance itself is defined such that ε_{SSD} has discontinuous derivatives.

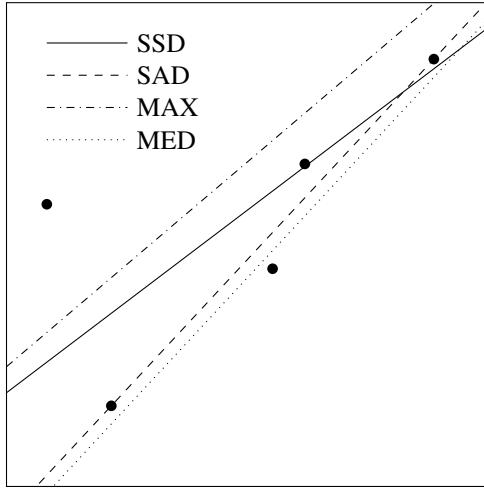


Figure 12.7: A set of points lying approximately on a line, where one of the points is a bit off from the rest. Lines are estimated from the points by minimizing different types of geometric errors.

This example should *not* be taken as evidence that the ϵ_{SAD} -error always gives a line that lies close to the unperturbed points, or that the ϵ_{MAX} always gives a poor estimate of the line. The resulting line depends both on the choice of error function and on the data to which the line is fitted. But the general observation is that, on the one hand, both ϵ_{SAD} and ϵ_{MED} accept a few large errors for the estimated line, as long as the remaining ones are small. On the other hand, the error function ϵ_{SSD} , and even more so ϵ_{MAX} , try to spread the errors out over the entire data set, thereby being more sensitive to outliers.

12.10 The error functions ϵ_{SAD} and ϵ_{MED} tend to generate optimal models that accept a few large errors, as long as the remaining ones are relatively small. The error function ϵ_{SSD} instead tends to spread out the errors more evenly, making many of the largest to be of comparable size for the optimal model. To an even larger degree, this is the case for ϵ_{MAX} .

12.4.2 Before we continue

We want estimation of models from observed data to be simple. If the optimization is done in software or hardware, it will then have a predictable behavior as far as computation time and reliability are concerned. Unfortunately, geometric errors seldom offer these properties. In fact, finding a 2D line by minimizing ϵ_{H} , ϵ_{V} , or ϵ_{TOT} are among the few estimation problems that use geometric errors and which are simple to solve. The remaining chapters present several different estimation problems that use geometric errors, and most of them have to be solved with iterative non-linear methods.

Any iterative method needs a set of model parameters from where the iterations start, an *initial solution*. A useful initial solution can often be found by solving a related estimation problem, based on so-called algebraic errors. This will be our next topic.

12.5 Algebraic errors

Part I has introduced homogeneous representations for several types of geometric objects. We have seen that these representations make certain geometric relations correspond to simple algebraic relations. This has so far been the main argument why we should care about these representations, but they have also another main application. Homogeneous representations make it possible to define error functions which are easy to optimize, based on so-called *algebraic errors*. In this section, we will describe how this is done.

12.5.1 Introductory example

As before, we return to the problem of fitting a line to a set of points. But this time we represent the points using homogeneous coordinates, \mathbf{y}_i , and the line with dual homogeneous coordinates \mathbf{l} . In Chapter 3 we learned that a point \mathbf{y}_i lies on the line \mathbf{l} if and only if $\mathbf{y}_i \cdot \mathbf{l} = 0$. Furthermore, the scalar product $\mathbf{y}_i \cdot \mathbf{l}$ is in general proportional to the distance between the point and the line. Unless we have made sure that \mathbf{y}_i and \mathbf{l} are properly normalized, the scalar product $\mathbf{y}_i \cdot \mathbf{l}$ does not have a direct interpretation as a distance. But we can still say that for the problem of fitting a line to points, in principle⁵, the smaller $\mathbf{y}_i \cdot \mathbf{l}$ is, the better does the line fit the point. As a consequence, it makes sense to define an error between the point \mathbf{y}_i and the line \mathbf{l} simply as

$$r_i = \mathbf{y}_i \cdot \mathbf{l} = \mathbf{y}_i^\top \mathbf{l}. \quad (12.30)$$

An overall error for the entire set of points is then defined as

$$\varepsilon_{\text{ALG}} = \sum_{i=1}^m r_i^2 = \sum_{i=1}^m (\mathbf{y}_i^\top \mathbf{l})^2. \quad (12.31)$$

We will call this an *algebraic error*, since it is derived from an algebraic expression instead of a geometric relation.

In the context of algebraic errors, the homogeneous representations that appear need not be normalized. In some cases, they are normalized, and an algebraic error can then at the same time also be a geometric error. For example, if we assume that all \mathbf{y}_i are P-normalized and \mathbf{l} is D-normalized, then ε_{ALG} is also a geometric error based on the Euclidean distances between the points and the line. But these normalizations can in general not be assumed, and then algebraic errors have no intuitive interpretation based on geometry.

The lack of intuitive understanding of an algebraic error implies that its magnitude has no particular interpretation. If it is twice as large in one case than in another, this does not necessarily mean that the first case is twice as bad.

The algebraic error defined for the line estimation problem in Equation (12.31) can be formulated in a more compact form. The individual errors for the different points, r_i in Equation (12.30), can be collected into a vector:

$$\mathbf{r} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix} = \begin{pmatrix} \mathbf{y}_1^\top \mathbf{l} \\ \mathbf{y}_2^\top \mathbf{l} \\ \vdots \\ \mathbf{y}_m^\top \mathbf{l} \end{pmatrix} = \begin{pmatrix} \mathbf{y}_1^\top \\ \mathbf{y}_2^\top \\ \vdots \\ \mathbf{y}_m^\top \end{pmatrix} \mathbf{l} = \mathbf{Y}^\top \mathbf{l}. \quad (12.32)$$

Here, the matrix $\mathbf{Y} = (\mathbf{y}_1 \dots \mathbf{y}_m)$ holds the homogeneous coordinates of the points in its columns. This is the same approach we took in Section 3.3.3, where relations between a set of points and a line were studied. There, we concluded that all points lie on the line if and only if $\mathbf{r} = \mathbf{0}$. What we are doing now, is to use $\|\mathbf{r}\|$ to quantify how much the points deviate from the ideal case, when all of them lie on the line. In fact, from Equation (12.32) follows that the algebraic error can be formulated as

$$\varepsilon_{\text{ALG}} = \|\mathbf{r}\|^2 = \|\mathbf{Y}^\top \mathbf{l}\|^2. \quad (12.33)$$

To summarize, for this estimation problem we want to find \mathbf{l} such that, in the ideal case, we get

$$\mathbf{r} = \mathbf{Y}^\top \mathbf{l} = \mathbf{0}. \quad (12.34)$$

When the points do not fit the line perfectly, instead we find \mathbf{l} that minimizes ε_{ALG} .

12.5.2 Model parameters, data matrix and residual

For general estimation problems, the simplest form of algebraic error can be formulated as

$$\varepsilon_{\text{ALG}}(\mathbf{z}) = \|\mathbf{r}\|^2, \quad \text{where } \mathbf{r} = \mathbf{A} \mathbf{z}. \quad (12.35)$$

The different quantities that appear in this algebraic error have their own names. The vector $\mathbf{z} \in \mathbb{R}^p$ holds the *model parameters* that we want to estimate, it represents the unknown quantity of the estimation problem. Typically,

⁵As long as the vector \mathbf{l} is not restricted so that only $\|\mathbf{l}\|$ can vary.

\mathbf{z} is a homogeneous representation of the geometric object to be determined. Solving the estimation problem corresponds to finding \mathbf{z} that minimizes our error function. For example, in Section 12.5.1 \mathbf{z} corresponds to the dual homogeneous coordinates of a 2D line \mathbf{l} .

The matrix $\mathbf{A} \in \mathbb{R}^{m \times p}$ is the *data matrix*, it represents the known or observed quantities of the problem. Although \mathbf{A} often is related to homogeneous representations, these relations may be more or less intricate. In the simplest cases, each row in \mathbf{A} is a vector which forms a homogeneous representation of some entity. This is what happened in the previous example, where we estimate a line from 2D points, Equation (12.33). In this case $\mathbf{A} = \mathbf{Y}^\top$, and each row of \mathbf{A} represents a point \mathbf{y}_k .

The product of \mathbf{A} and \mathbf{z} is the *residual vector* $\mathbf{r} = \mathbf{A}\mathbf{z}$, a vector in \mathbb{R}^m . The underlying assumption behind the algebraic error in Equation (12.35) is that $\mathbf{r} = \mathbf{A}\mathbf{z}$ quantifies how well the data in \mathbf{A} fits the model represented by \mathbf{z} . The smaller $\|\mathbf{r}\|$ is, the better is the match between data and model. The ideal situation is $\mathbf{r} = \mathbf{0}$, in this case there is an exact match. The residual need not have an intuitive interpretation in geometric terms. But the general assumption is that the smaller $\|\mathbf{r}\|$ is, the better is the fit between model and data.

In the rest of this presentation, we will focus on algebraic error formulated as in Equation (12.35). The reason is that this formulation of ε_{ALG} is very general and that it can be minimized with relatively simple computations. In fact, this can even be done in several different ways. Before we look at specific methods for minimizing algebraic errors, there are a few concepts that need to be defined and issues that need to be sorted out.

12.5.3 Internal constraints

Clearly, if we want to minimize ε_{ALG} in Equation (12.35), then $\mathbf{z} = \mathbf{0}$ produces the smallest possible error: $\varepsilon_{\text{ALG}} = 0$. But we have also said that \mathbf{z} is a homogeneous representation of some geometric object, an element of a projective space. In Toolbox Section 7.1 we learned that $\mathbf{z} = \mathbf{0}$ it is not a proper projective element. And in general, since the scaling of \mathbf{z} is irrelevant for the geometric interpretation of \mathbf{z} , we should not try to further minimize ε_{ALG} just by making $\|\mathbf{z}\|$ smaller.

To avoid $\mathbf{z} = \mathbf{0}$, we introduce a constraint on \mathbf{z} , to be included in the minimization problem. This *normalization constraint* on \mathbf{z} is specified such that further reduction of ε_{ALG} is not possible just by a scaling of \mathbf{z} . We will shortly look at two simple choices of normalization constraints on \mathbf{z} , leading to two different methods for computing the optimal model parameters.

A normalization of \mathbf{z} may not be the only constraint that has to be observed. For example, if the data is in the form of 3D points to which we want to fit a line, then \mathbf{z} would be the Plücker coordinates of this line. Plücker coordinates come in the form of anti-symmetric 4×4 matrices, elements of $so(4)$. But $so(4)$ is essentially a 6-dimensional vector space, which means that we can represent any element of $so(4)$ by $\mathbf{z} \in \mathbb{R}^6$. This trick of converting a matrix into a vector is called *vectorization*, and we will return to this idea in Chapter 13. We have also seen that not every element of $so(6)$ are the Plücker coordinates of a 3D line, it has to satisfy the internal constraint in Equation (5.29). This translates into a constraint also for the model parameters $\mathbf{z} \in \mathbb{R}^6$ that represent the line.

Another example is if we want to estimate a 3D rotation matrix \mathbf{R} from a set of points, before and after the rotation. \mathbf{R} is a 3×3 matrix, an element of the vector space $\mathbb{R}^{3 \times 3}$. But $\mathbb{R}^{3 \times 3}$ is essentially a 9-dimensional vector space, which means that any element of $\mathbb{R}^{3 \times 3}$ can be vectorized by $\mathbf{z} \in \mathbb{R}^9$. We have also seen that $\mathbf{R} \in SO(3)$ implies that $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$ and $\det(\mathbf{R}) = 1$. This property translates into a set of constraints for the model parameters $\mathbf{z} \in \mathbb{R}^9$ that represents the rotation matrix.

The type of constraints that we discuss in relation to Plücker coordinates and rotation matrices is about *consistency* of the model parameters in \mathbf{z} . In some cases, e.g., when \mathbf{z} represents points, a 2D line, or a 3D plane, there is no consistency constraint on \mathbf{z} . Any $\mathbf{z} \neq \mathbf{0}$ in the appropriate vector space represents some point, line, or plane. If this \mathbf{z} is used to define the algebraic error in Equation (12.35), we only have to worry about the normalizing constraint to avoid $\mathbf{z} = \mathbf{0}$ as a solution. In other cases, there is also a set of consistency constraints that has to be observed in the minimization of ε_{ALG} . In later chapters, we will investigate strategies for dealing with consistency constraints. Together, normalizing constraints and consistency constraints are referred to as *internal constraints*.

12.5.4 The size of \mathbf{A}

When we minimize ε_{ALG} in Equation (12.35), a second issue that needs our attention is the character the data matrix \mathbf{A} , including its size: $m \times p$. The parameter p is the width of \mathbf{A} , and depends only on the type of estimation problem that we are trying to solve. It equals the number of elements in \mathbf{z} , so it says something about the *degrees*

of freedom for the model parameters. If there are no consistency constraints on the projective element \mathbf{z} , for which a scaling does not change its interpretation, the model parameters have $p - 1$ degrees of freedom. If there are additional consistency constraints, the degrees of freedom are reduced. Each independent constraint removes one degree of freedom from $p - 1$.

The parameter m is the height of \mathbf{A} , and depends on the amount of observed data that is used in the estimation. As we will see, exactly how m depends on the amount of data can differ a lot between various estimation problems. The general conclusion is that the more observations of the data we have, the larger is m .

The cost function for an algebraic error in Equation (12.35) can equivalently be formulated as

$$\varepsilon_{\text{ALG}} = \sum_{i=1}^m (\mathbf{a}_i \cdot \mathbf{z})^2, \quad (12.36)$$

where \mathbf{a}_i is the i -th row of \mathbf{A} . This can be interpreted as: each row \mathbf{a}_i specifies a *data constraint* on \mathbf{z} . In the ideal case we want $\mathbf{a}_i \cdot \mathbf{z} = 0$ for each row \mathbf{a}_i . This is a soft constraint in the sense that it may not be possible to find such a model. Instead, the goal of our estimation process is to find \mathbf{z} that meets the data constraints as much as possible, by minimizing ε_{ALG} at the same time as it satisfies the hard internal constraints.

Solution space

In principle, we want to find \mathbf{z} such that

$$\mathbf{r} = \mathbf{A}\mathbf{z} = \mathbf{0}, \quad (12.37)$$

so we seek $\mathbf{z} \in \text{Null}(\mathbf{A})$ for $m \times p$ matrix \mathbf{A} . Consequently, we can think of $\text{Null}(\mathbf{A})$ as the *solution space* of the estimation problem. However, restricting ourselves to the space of all \mathbf{z} that satisfy Equation (12.37) exactly is problematic. The reason is that \mathbf{A} is perturbed by measurement errors, and it may also have limited numerical resolution. Therefore, we will use the concept “solution space” in a slightly broader meaning.

Definition 12.2: Solution space

The solution space is a subspace of \mathbb{R}^p that satisfy Equation (12.37) to a degree specified by the error in \mathbf{A} .

The exact construction of the solution space may vary with different estimation methods.

To form a well-defined estimation problem, we require in practice that the set of \mathbf{z} which minimize ε_{ALG} and satisfy all internal constraints is finite. If this is true or not depends on m and p , the size of \mathbf{A} , and on c , the number of internal constraints⁶.

The under-determined case

If $m < p$, then $\text{Null}(\mathbf{A})$ has at least $p - m$ dimensions. This means that the solution space contains at least some non-zero vector $\mathbf{z} \in \text{Null}(\mathbf{A})$ that minimizes the error with $\varepsilon_{\text{ALG}} = 0$. But, if $p - m < c$, where c the number of internal constraints, the solution space is “too large”. In this case \mathbf{z} is *under-determined*, we do not have sufficiently many data constraints together with internal constraints to specify a finite set of \mathbf{z} as a solution. In the example of fitting a line to a set of points, we have an under-determined solution when there is only one single point in this set.

Data degeneracy

Apart from having too few observations of the data, there is another possibility that can lead to an under-determined solution. In general, when $p > m$ we will find that $\dim(\text{Null}(\mathbf{A})) = p - m$. However, for particular choices of the data, $\dim(\text{Null}(\mathbf{A})) > p - m$. What happens here is that even if we add some new observations to \mathbf{A} , the corresponding data constraints are linearly dependent with other constraints from earlier data. For example, if we observe a set of 3D points to which a plane is fitted, and the points happen to be collinear, the plane is still under-determined.

This case is often referred to as degenerate, and we will use the term *data degeneracy* to make explicit that the cause of the problem are special configurations of the observed data, which increase the dimensions of $\text{Null}(\mathbf{A})$. Data degeneracy can often be avoided by assuming that data is in “general configuration”, i.e., it does not satisfy

⁶More precisely, the number of algebraically independent constraints.

any particular constraints that would lead to degeneracy. This, however, is only an assumption, and unless explicit measures are taken to avoid this case, real data that is generated by measurements may be degenerate.

The minimal case

If $p - m = c$, and there is no data degeneracy, the solution space has just the right size to let the internal constraints specify a finite set of solutions \mathbf{z} . This is the *minimal case* of model estimation, where a finite set of models \mathbf{z} has an exact match with the data constraints, i.e., $\mathbf{A} \mathbf{z} = \mathbf{0}$. For example, when we fit a line to a point-set, the minimal case is to use two distinct points.

The minimal case is sometimes seen as a special case of estimation. Since the residual is zero, finding \mathbf{z} corresponds to solving a set of equations, and this can often be done effectively with methods that are dedicated for the minimal case. For example, if we want to find the line \mathbf{l} that passes through two distinct points \mathbf{y}_1 and \mathbf{y}_2 , we can set $\mathbf{l} \sim \mathbf{y}_1 \times \mathbf{y}_2$. This approach cannot be generalized to matching a line with a larger set of 2D points.

A common case occurs when $c = 1$, i.e., only a normalization constraint need to be observed. Assuming that the data is not degenerate, then $m = p - 1$ implies that there is just enough data constraints (m) to specify the solution space as a one-dimensional subspace of \mathbb{R}^p . Any non-zero element of the solution space represents the same projective element, the parameters of a model that exactly fits the data. The normalization constraint will produce a specific representative of this projective element.

Since the residual error is zero, the resulting model \mathbf{z} and the data match each other exactly in the minimal case. It is then easy to get the impression that \mathbf{z} is an exact and error-free determination of the corresponding model. However, there are no reasons to assume that the data used to determine \mathbf{z} is without errors, which means that there is some amount of error also in the resulting model.

The over-determined case

When $p - m > c$, and there is no data degeneracy, we have more data constraints than degrees of freedom in the model parameters \mathbf{z} . As a consequence, we will not be able to find a \mathbf{z} that gives $\mathbf{A} \mathbf{z} = \mathbf{0}$ unless the data happens to fit a model perfectly. Instead, we have to seek a \mathbf{z} that minimizes $\|\mathbf{A} \mathbf{z}\|$, taking also the internal constraints on \mathbf{z} into account. This is the *over-determined case* of model estimation. Estimation of model parameters is normally done based on the over-determined case in the context of algebraic errors.

The notion of a solution space for the over-determined case is a bit unclear. In general, $\text{Null}(\mathbf{A})$ is empty (except for $\mathbf{0}$) for this case. But if we take inaccuracies in the data into account, it may still be reasonable to talk about a solution space also for the over-determined case. In what follows, we will present different options to form a solution space for this case.

12.5.5 General solution strategy for the over-determined case

Before looking at specific methods for dealing with algebraic errors, we summarize and make some observations. Given the previous discussion about algebraic errors, in general we define this concept as:

Definition 12.3: Algebraic error

An algebraic error is constructed by combining a data matrix \mathbf{A} , formed from observed data, and model parameters \mathbf{z} , as

$$\varepsilon_{\text{ALG}} = \|\mathbf{A} \mathbf{z}\|^2. \quad (12.38)$$

In general, this error has no intuitive interpretation based on the geometry of the original space where estimation problem is formulated.

An estimation methods that minimize an algebraic error is often referred to as performing *algebraic estimation*.

Furthermore, since x^2 is a monotonous growing function for $x > 0$, it should be clear that the same \mathbf{z} minimizes the error formulated in Equation (12.38), and also without the squaring:

$$\varepsilon_{\text{ALG}} = \|\mathbf{A} \mathbf{z}\|. \quad (12.39)$$

From a theoretical point of view, we may then instead define our algebraic error as in Equation (12.39), and get the same model parameters \mathbf{z} . This is true also if there are internal constraints on \mathbf{z} that must be observed.

12.11 We can formulate an algebraic error as $\|\mathbf{A}\mathbf{z}\|^2$ or as $\|\mathbf{A}\mathbf{z}\|$. They are equivalent in the sense that both are minimized by the same \mathbf{z} .

From a practical point of view, the squaring simplifies the derivations of the optimal \mathbf{z} . Equation (12.38) defines the error as a second order expression in the unknown model parameters \mathbf{z} . Finding the optimal model parameters requires a differentiation of ε_{ALG} with respect to the elements of \mathbf{z} . The result is a first order expression in \mathbf{z} . Consequently, finding \mathbf{z} amounts to solving a linear equation in \mathbf{z} , which normally can be done in a straightforward way. Sections 12.5.6, and 12.5.7 describe two standard approaches for how to do this.

In many applications, we are primarily interested in minimizing geometric errors. As mentioned in observation 12.9 on page 203, the minimization of such an error often requires iterative non-linear methods. The same estimation problem can then often be reformulated using algebraic errors, and solved with much simpler methods.

This is the typical use case of algebraic errors. Minimization of geometric errors often requires a high computational effort, and they are therefore substituted, approximately, by algebraic errors, which are simple to minimize. As we will see, how good an approximation they are depends on several factors and is not guaranteed. Another possibility is if we want to minimize a geometric error with some iterative method. Such a method needs an initial solution, and this can be provided by an algebraic solution.

12.12 It is often easy to formulate and minimize algebraic errors based on homogeneous representations. These can be used as approximate substitutes for geometric errors, or as initial solution when geometric errors are minimized.

In the following two sections, we present two different normalization constraints for \mathbf{z} , leading to two different methods for finding the \mathbf{z} that minimizes an algebraic error.

12.5.6 The inhomogeneous method

One common normalization constraint on $\mathbf{z} \in \mathbb{R}^p$ in Equation (12.39) is to require that a specific element in \mathbf{z} assumes a constant value, often = 1. In general, we can choose any element in \mathbf{z} designated as = 1, but we will see shortly the choice is not completely arbitrary. For example, the designated element can be the last one:

$$\mathbf{z} = \begin{pmatrix} \bar{\mathbf{z}} \\ 1 \end{pmatrix}, \quad \text{where } \bar{\mathbf{z}} \in \mathbb{R}^{p-1}. \quad (12.40)$$

This normalization of \mathbf{z} corresponds to the canonical form of homogeneous coordinates of points described in Sections 3.1 and 5.1. In general, however, \mathbf{z} does not have to be the homogeneous coordinates of some point, it can represent any geometrical object. Furthermore, the following results generalize in a straightforward way if we choose another element in \mathbf{z} than the last element to be constant.

With \mathbf{z} described by Equation (12.40), our algebraic error can be reformulated as

$$\varepsilon_{\text{ALG}} = \left\| \mathbf{A} \begin{pmatrix} \bar{\mathbf{z}} \\ 1 \end{pmatrix} \right\|^2 = \|\mathbf{A}_0 \bar{\mathbf{z}} + \mathbf{b}\|^2, \quad \text{where } \mathbf{A} = (\mathbf{A}_0 \ \mathbf{b}). \quad (12.41)$$

Here, $\mathbf{b} \in \mathbb{R}^m$ is the p -th (the right-most) column of the data matrix \mathbf{A} , and \mathbf{A}_0 is the $m \times (p-1)$ matrix holding the remaining columns. The normalization constraint which we use here implies that the last element in \mathbf{z} is set to 1. The remaining elements, corresponding to the vector $\bar{\mathbf{z}}$, are free. Consequently, we minimize ε_{ALG} in Equation (12.41) over $\bar{\mathbf{z}}$, which is a standard least square problem. Any $\bar{\mathbf{z}}$ that minimizes ε_{ALG} must solve the corresponding normal equation:

$$\mathbf{A}_0^\top \mathbf{A}_0 \bar{\mathbf{z}} = -\mathbf{A}_0^\top \mathbf{b}. \quad (12.42)$$

In general, we can describe the solution of the normal equation as

$$\bar{\mathbf{z}} = -(\mathbf{A}_0^\top \mathbf{A}_0)^{-1} \mathbf{A}_0^\top \mathbf{b} = -\mathbf{A}_0^+ \mathbf{b}, \quad (12.43)$$

• Least squares problems are discussed in Toolbox Section 3.9.

Algorithm 12.1: The inhomogeneous method for minimizing algebraic errors

Input: Data matrix \mathbf{A}
Input: $i =$ index of the designated element in \mathbf{z} , set = 1
Output: An estimate of \mathbf{z} , minimizing the algebraic error ε_{ALG} in Equation (12.39), where the i -th element of \mathbf{z} is set = 1

- 1 Set $\mathbf{b} =$ the i -th column in \mathbf{A}
- 2 Set $\mathbf{A}_0 =$ the remaining columns in \mathbf{A}
- 3 Solve $\mathbf{A}_0^\top \mathbf{A}_0 \bar{\mathbf{z}} = -\mathbf{A}_0^\top \mathbf{b}$ for $\bar{\mathbf{z}}$. // There is a risk that $\mathbf{A}_0^\top \mathbf{A}_0$ may be singular here!
- 4 Construct \mathbf{z} from $\bar{\mathbf{z}}$ by inserting a constant element = 1 at position i

where \mathbf{A}_0^+ is the pseudo-inverse of \mathbf{A}_0 . This means that the computational complexity for finding the optimal \mathbf{z} corresponds to solving a system of $p - 1$ linear equations, possibly in terms of computing the inverse of the $(p - 1) \times (p - 1)$ matrix $\mathbf{A}_0^\top \mathbf{A}_0$, or applying other techniques such as Gaussian elimination. We refer to this approach as the *inhomogeneous method* since it leads to the inhomogeneous linear equation in Equation (12.42). The inhomogeneous method is summarized in Algorithm 12.1.

• Pseudo-inverses are discussed in Toolbox Section 3.4.2.

The residual error produced by the inhomogeneous method is obtained by inserting $\bar{\mathbf{z}}$ in Equation (12.43) into the expression for ε_{ALG} :

$$\varepsilon_{\text{ALG,res}} = \|\mathbf{b} - \mathbf{A}_0 \mathbf{A}_0^+ \mathbf{b}\|^2 = \|(\mathbf{I} - \mathbf{A}_0 \mathbf{A}_0^+) \mathbf{b}\|^2 = \|\mathbf{P} \mathbf{b}\|^2. \quad (12.44)$$

• Projection operators are discussed in Toolbox Section 3.3.9.

Here, $\mathbf{P} = \mathbf{I} - \mathbf{A}_0 \mathbf{A}_0^+$ is the projection operator onto the orthogonal complement of the subspace spanned by the columns in \mathbf{A}_0 . Remember that algebraic errors, and therefore also the magnitude of $\varepsilon_{\text{ALG,res}}$, in general has no intuitive interpretation in terms of the underlying geometric problem.

The inhomogeneous method is simple enough to be used for many practical estimation problems, but there are some issues that can lead to problems when the solution is computed. Two such issues will be discussed next.

Method degeneracy

We recall that the model parameter vector, \mathbf{z} , is a representative of a projective element. Any scalar multiplication of \mathbf{z} represents the same model. However, if some element of \mathbf{z} is set to one, we must multiply the entire vector by zero to make this element equal to zero. But $\mathbf{z} = \mathbf{0}$ does not represent any projective element.

This observation means that the inhomogeneous method cannot produce an estimated model where the designated element, set to one, is expected to be zero for the data set. An example is estimation of a 2D line from a set of points. If they lie approximately on a line that passes through the origin, we expect the estimated line \mathbf{l} to be

$$\mathbf{l} \sim \begin{pmatrix} \hat{\mathbf{l}} \\ 0 \end{pmatrix}, \quad \text{where} \quad \|\hat{\mathbf{l}}\| = 1. \quad (12.45)$$

Here $\hat{\mathbf{l}}$ is a normal vector of the line. But if we also set the third element in $\mathbf{z} = \mathbf{l}$ equal to one, there is no finite value of \mathbf{z} that can be scaled to the right-hand side of Equation (12.45).

What is described here is, again, a type of degeneracy, but it is neither data degeneracy nor model degeneracy. The data can be fitted relatively well to a particular model, at least if a suitable method is used. But that model cannot be generated by the inhomogeneous method. Hence, we refer to this situation as *method degeneracy*. It implies that the estimation method has certain “blind spots” in terms of what models it can produce. When the data comes close to particular configurations, which in principle would give a small ε_{ALG} for a \mathbf{z} in these blind spots, we should expect the resulting estimate of \mathbf{z} to be bad.

Method degeneracy is illustrated in Figure 12.8. The left plot shows a set of points, approximately lying on a line that does not intersect the origin. The inhomogeneous method is applied to this data to estimate a line, also shown in the plot. The line appears reasonable for this data set. The right plot shows a similar set of points, but now approximately lying on a line that passes through the origin. The corresponding line estimate is also shown in the plot, this time with a slope that cannot be accounted for from the given data. This is a result of degeneracy related to the inhomogeneous method, when applied to the line estimation problem.

What about choosing another element in \mathbf{l} and set it = 1? This must surely enable us to represent lines that pass through the origin without problem, right? Well, yes and no. Since the other two elements in \mathbf{l} represent the

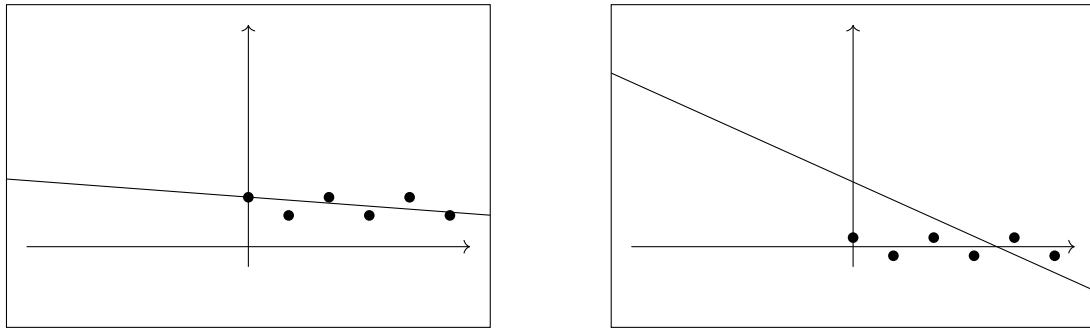


Figure 12.8: Left: A set of point that approximately lie on a line that does not pass through the origin. The figure includes also the line estimated by the inhomogeneous method applied to this data. Right: A set of point that approximately lie on a line that pass through the origin. The figure includes also the line estimated by the inhomogeneous method applied to this data.

direction of a normal vector to the line, and either element can assume a zero value for general lines, we still face the possibility of an ill-posed formulation of the problem if we set either of these two element = 1. In this case we would instead have blind spots for specific orientations of the line. In short, there is no element in \mathbf{I} that is safe to set = 1 for general lines, not even if they are restricted to proper lines.

As a rule of thumb, it makes sense have the element that is expected to be of largest absolute value in \mathbf{z} as the designated one, set equal to one. This choice has the lowest risk of producing incorrect model estimates.

We have seen method degeneracy earlier in this chapter. The line estimation problem presented in Section 12.1, using the (k, l) line parameters in combination with least squares, ended up with a simple method to estimate the line. But we also noted that such a line cannot be vertical: the estimation method is degenerate for vertical lines.

Data degeneracy

Another problematic case appears for the inhomogeneous method when the data is configured to make \mathbf{A}_0 rank deficient. For example, in the line estimation case \mathbf{A}_0 has in general rank 2. But when all points are clustered on a specific point, e.g., \mathbf{y}_0 , then the rank of \mathbf{A}_0 becomes close 1. When \mathbf{A}_0 comes sufficient close to rank deficiency, the inverse of $\mathbf{A}_0^\top \mathbf{A}_0$ is no longer reliable. In this case, there are multiple lines, all passing through \mathbf{y}_0 , which solve the estimation problem. Thus, the solutions space increases from a single solution to an infinite set of solutions.

Data degeneracy implies an increased solution space, an issue that any estimation must deal with. It becomes especially problematic for the inhomogeneous method since, unless we are able to catch the rank deficiency of \mathbf{A}_0 , it can lead to unpredictable behavior of all subsequent operations that use the estimated \mathbf{z} . A safe option is to make sure that the rank deficient case is tested and, if true, produce $\mathbf{z} = \mathbf{0}$ to flag “a unique estimate could not be found”.

Before we continue

It should be mentioned that the resulting estimate of \mathbf{z} depends on which element in \mathbf{z} is designated to be = 1. In fact, ε_{ALG} in Equation (12.41) is a different function for every different choice of the designated element. It does not even make sense to compare the residual error of one with another. After all, they are algebraic errors.

We conclude this discussion by pointing out that the inhomogeneous method in general leads to simple computations that give us the estimated model parameters. But there are also numerical issues related to this method that makes it less suitable for specific cases of data. It should only be used if we can be certain that these numerical problems will not degrade the solution.

12.13 Algebraic errors can be minimized by means of the inhomogeneous method. It is simple to implement, but it should be used with some care. The method itself is degenerate: certain models cannot be produced as estimates. Rank deficiency of \mathbf{A}_0 should be tested and dealt with.

Algorithm 12.2: The homogeneous method for minimizing algebraic errors

Input: The data matrix \mathbf{A}
Output: An estimate of \mathbf{z} , minimizing the algebraic error ε_{ALG} in Equation (12.38), where $\|\mathbf{z}\| = 1$

- 1 Determine SVD of $\mathbf{A} = \mathbf{U} \mathbf{S} \mathbf{V}^\top$
- 2 // Here, we assume that the singular values in the diagonal of \mathbf{S} are sorted in descending order
- 3 $\mathbf{z} =$ the right-most column of \mathbf{V} (= the right singular vector of \mathbf{A} with the smallest singular value)

12.5.7 The homogeneous method

To minimize the algebraic error in Equation (12.39), another option is to use $\|\mathbf{z}\| = 1$ as a normalization constraint. Formally, in this case we want to

$$\text{minimize } \varepsilon_{\text{ALG}}(\mathbf{z}) = \mathbf{z}^\top \mathbf{A}^\top \mathbf{A} \mathbf{z} \quad \text{over } \mathbf{z}, \quad \text{with the constraint} \quad c(\mathbf{z}) = \mathbf{z}^\top \mathbf{z} = 1. \quad (12.46)$$

- Lagrange's method for optimizing with additional constraint is described in Toolbox Section 4.4.1.

We apply Lagrange's method to this constrained optimization problem, and get

$$\nabla \varepsilon_{\text{ALG}} = \lambda \nabla c \Rightarrow \mathbf{A}^\top \mathbf{A} \mathbf{z} = \lambda \mathbf{z}, \quad (12.47)$$

where λ is the corresponding Lagrange multiplier. This is the same type of eigenvalue problem discussed in Section 12.2, and the solution to our estimation problem can therefore be formulated as: \mathbf{z} is a normalized eigenvector corresponding to the smallest eigenvalue of $\mathbf{A}^\top \mathbf{A}$. In Section 12.2 we also argued that there are possible numerical issues with that description, and a more stable formulation is given as: \mathbf{z} is a normalized right singular vector corresponding to the smallest singular value of the data matrix \mathbf{A} . We refer to this approach as the *homogeneous method*, summarized in Algorithm 12.2.

It is not necessary to compute a full SVD of \mathbf{A} : it is sufficient if we can determine a right singular vector corresponding to the smallest singular value. Let \mathbf{u} and \mathbf{v} denote the left and right singular vectors, corresponding to σ , the smallest singular value of \mathbf{A} , and normalized such that $\|\mathbf{u}\| = \|\mathbf{v}\| = 1$. The residual error for the homogeneous method can then be derived by setting $\mathbf{z} = \mathbf{v}$ and take into account that $\mathbf{A} \mathbf{v} = \sigma \mathbf{u}$:

$$\varepsilon_{\text{ALG,res}} = \mathbf{v}^\top \mathbf{A}^\top \mathbf{A} \mathbf{v} = \sigma^2 \mathbf{u}^\top \mathbf{u} = \sigma^2. \quad (12.48)$$

This shows that the residual error for the homogeneous method equals the square of the smallest singular value of \mathbf{A} , when the algebraic error is defined as in Equation (12.39).

In contrast to the homogeneous method, the optimal model parameters produced by the homogeneous method can be any \mathbf{z} of unit norm, and there are no degenerate cases of \mathbf{z} that should be avoided. For example, in the case of line fitting, nothing particular happens if the optimal line passes through the origin when the homogeneous method is used.

Data degeneracy

A degenerate case occurs for the line fitting problem when all points approximately coincide. If the inhomogeneous method is used in this case, we have seen that $\mathbf{A}_0^\top \mathbf{A}_0$ then becomes singular and Equation (12.42) has no unique solution. If the homogeneous method is used instead, this degeneracy implies that there is one large singular value σ_1 , and two smaller singular values⁷ $\sigma_2 \geq \sigma_3$. The magnitudes of σ_2 and σ_3 are determined mainly by how much the points deviate from being a single point. In practice, we often find that $\sigma_2 > \sigma_3$, and the right singular vector \mathbf{v}_3 , corresponding to σ_3 , is then the optimal choice for \mathbf{z} . But if σ_2 is only slightly larger than σ_3 , the residual error in Equation (12.48) is approximately the same if instead we choose \mathbf{v}_2 , the right singular vector corresponding to σ_2 , as the solution. In fact, any linear combination of \mathbf{v}_2 and \mathbf{v}_3 gives approximately the same residual error.

⁷Here we consider the case of fitting a line to a set of 2D points. In this case, the data matrix \mathbf{A} is of size $n \times 3$ and it has 3 singular values.

Consequently, the solution space is two-dimensional in this case.

12.14 The homogeneous method may provide a solution \mathbf{z} that is not unique. This happens when there are two or more singular values of \mathbf{A} having approximately the same magnitude, and all are small. In this case, the solution space has a dimension that is larger than one.

Multiple solutions is a potential problem in estimation, but if the homogeneous method is used we can at least detect this case, by investigating the singular values of \mathbf{A} . In this case, the model that is estimated by the homogeneous method is still computed based on robust computations. There are, however, additional issues related to this method, but we will defer a more detailed discussion on this topic to Section 13.2.

Since the homogeneous method does not degenerate for special cases of the solution \mathbf{z} , it is more applicable compared to the inhomogeneous method. As a consequence, the homogeneous method is the standard approach for minimization of algebraic errors. The inhomogeneous method should only be used if there are limitations in software or hardware that makes computation of SVD or EVD unpractical. And in this case, pay attention to observation 12.13 on page 211.

12.15 Algebraic errors can be minimized by means of the homogeneous method. It is more general than the inhomogeneous method and should always be used for minimizing algebraic errors unless there are practical reasons not to.

Before we continue

As a concluding remark, we observe that although both the inhomogeneous method and the homogeneous method minimize ϵ_{ALG} , they do this with different normalization constraints on \mathbf{z} . This means that the free parameters that can vary are different, and that the actual error function which is minimized is different for the two methods. Therefore, and in general, the two approaches produce different estimates of \mathbf{z} , as well as different residual errors $\|\mathbf{r}\|$, even if both methods are applied to the same data matrix \mathbf{A} .

The difference between the two methods may not have to be large, and it is not possible to give a clear statement about which of the two methods is better suited for a specific application.

12.6 Probability based estimation

The formulation of measurement errors in the form of random noise that is expressed in Equation (12.20) leads us to yet another possibility of estimating a model \mathbf{m} from observed data. For example, given a set of observed data, and some information about the statistics of the measurement noise, we can look for \mathbf{m} that is most probable to have generated the data. In this section, we will discuss this and other options for estimating models based on probabilities.

12.6.1 Back to points and a line

We return to the problem of estimating a line from a set of observed points. This time, we will formulate the problem based on probabilities instead of distances. In the end, with a suitable set of assumptions, we will see that probabilities can be translated to distances. This enables us to make a connection to the previous method of minimizing geometric errors.

We start with the simplest case and discuss the probabilities related to a single point and a line. Once this case has been sorted out, we can add one more point to the problem and see how that affects the result. Finally, we consider the general problem of fitting a line to a set of m points using a probabilistic framework.

Once these cases have been worked out, we can formulate methods to estimate the line from observed points and the probabilities.

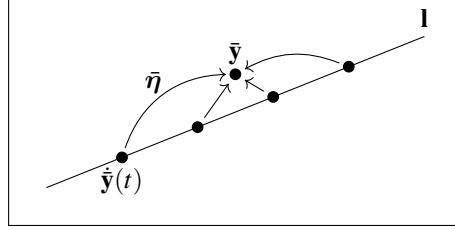


Figure 12.9: The conditional probability of observing point \bar{y} given the line I equals the probability that I generates a point $\hat{y}(t)$ on the line, which then is displaced to \bar{y} by the measurement noise $\bar{\eta}$, and finally marginalized over all possible points $\hat{y}(t)$ along the line.

A single point and a line

Let \bar{y} be the Cartesian coordinates of a point and I be dual homogeneous coordinates of a line. The basic assumption is that the point and the line fit together, which in this case means: the line is a model that has generated the point. We will now formulate this statement in a framework based on probabilities.

For example, we could try to determine $\Pr(\bar{y})$, the probability⁸ of the event “the point \bar{y} is observed”, or $\Pr(I)$, the probability of the event “the model which generates observed points is the line I ”. These are priori probabilities (or just priors) of each of the two events, without any knowledge about the other. The priors are interesting in some cases, but do not describe that the point and the line are related. To capture this relation we will instead study the conditional probability $\Pr(\bar{y} | I)$, corresponding to the event that \bar{y} is observed *given* that it has been generated by I . This is the a posteriori probability of the observation given that we know the model.

To simplify $\Pr(\bar{y} | I)$, we return to the discussion on measurement noise in Section 12.3.1. The implicit assumption is that there exists a point \hat{y} which lies somewhere on the line I , but is has been displaced by the measurement noise $\bar{\eta}$, resulting in the observed point \bar{y} . This boils down to two relations between these different entities:

$$\bar{y} = \hat{y} + \bar{\eta}, \quad \text{and} \quad \hat{y} \cdot I = 0, \quad (12.49)$$

where \hat{y} is a homogeneous representation of the ideal point coordinates. The first equation is a relation between points, which can be satisfied by any pair of two points, \bar{y} and \hat{y} , but only with a certain probability. This probability is described by $\Pr(\bar{\eta})$, the PDF of the measurement noise $\bar{\eta}$.

The condition $\hat{y} \cdot I = 0$ is guaranteed if we parameterize the point \hat{y} as:

$$\hat{y}(t) = \begin{pmatrix} \hat{u} \\ \hat{v} \end{pmatrix} = \Delta \begin{pmatrix} l_1 \\ l_2 \end{pmatrix} + t \begin{pmatrix} l_2 \\ -l_1 \end{pmatrix}, \quad \text{where} \quad I = \begin{pmatrix} l_1 \\ l_2 \\ -\Delta \end{pmatrix}, \quad \text{and} \quad l_1^2 + l_2^2 = 1. \quad (12.50)$$

The parameter t can assume any real value, corresponding to different points along the line.

With the introduction of the auxiliary variable $\hat{y}(t)$, we can study $\Pr(\bar{y} | \hat{y}(t))$, the probability that we observe \bar{y} given that it is the displacement of $\hat{y}(t)$. This probability is interesting because the corresponding event is equivalent to $\hat{y}(t)$ being perturbed by measurement noise $\bar{\eta} = \bar{y} - \hat{y}$. Assuming that $\bar{\eta}$ has an isotropic PDF, in accordance with Equation (12.21), we get

$$\Pr(\bar{y} | \hat{y}(t)) = P_{\text{noise}}(\bar{\eta}) = P_0(\|\bar{y} - \hat{y}(t)\|). \quad (12.51)$$

For a given t this event is only one of infinitely many that generates the observation \bar{y} , corresponding to different points along the line I . Any $\hat{y}(t)$ can be displaced to \bar{y} , but only with a certain probability. These events are mutually exclusive, and to obtain $\Pr(\bar{y} | I)$ we marginalize over them, assuming that all $\hat{y}(t)$ have equal probability, $\Pr(\hat{y}(t)) = \gamma$:

$$\Pr(\bar{y} | I) = \int_{-\infty}^{\infty} \Pr(\bar{y} | \hat{y}(t)) \Pr(\hat{y}(t)) dt = \gamma \int_{-\infty}^{\infty} P_0(\|\bar{y} - \hat{y}(t)\|) dt. \quad (12.52)$$

⁸Often, \bar{y} is a continuous variable, e.g., $\bar{y} \in \mathbb{R}^2$. In this case, the value of $\Pr(\bar{y})$ is the *probability density* of a random variable \hat{y} , see Toolbox Section 5.2.2. In this presentation, we will not make a formal distinction between probability and probability density. We will also not make a formal distinction between a random variable \hat{y} and the observed value of \hat{y} , here denoted as \bar{y} .

See Figure 12.9 for an illustration.

To get even further with this expression, we need to specify P_0 explicitly. A common assumption is to say that it is a zero-mean Gaussian with standard deviation σ :

$$P_0(z) = \frac{1}{(2\pi\sigma)^{p/2}} e^{-\frac{1}{2}z^2/\sigma^2}. \quad (12.53)$$

Here, p is the dimension of the observation space, in this example $p = 2$. With this P_0 plugged into Equation (12.52) and after evaluating the resulting integral⁹, we get

$$\Pr(\bar{\mathbf{y}} | \mathbf{l}) = \gamma \int_{-\infty}^{\infty} P_0(\|\bar{\mathbf{y}} - \dot{\mathbf{y}}(t)\|) dt = \frac{\gamma}{2\pi\sigma} \int_{-\infty}^{\infty} e^{-\frac{1}{2}\|\bar{\mathbf{y}} - \dot{\mathbf{y}}(t)\|^2/\sigma^2} dt = \dots = \frac{\gamma}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{d_{PD}(\mathbf{y}, \mathbf{l})}{\sigma}\right)^2}. \quad (12.54)$$

Here, $d_{PD}(\mathbf{y}, \mathbf{l})$ is the Euclidean distance from point \mathbf{y} to line \mathbf{l} , see Section 3.4.2. Consequently, $\Pr(\bar{\mathbf{y}} | \mathbf{l})$ depends in principle only on the distance between the observed point and the generating line.

Two points and a line

Next, we investigate $\Pr(\bar{\mathbf{y}}_1, \bar{\mathbf{y}}_2 | \mathbf{l})$, the conditional probability that we observe points $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$ given that they both are generated by \mathbf{l} . Using the same arguments as above, this probability can be expressed more clearly by introducing two auxiliary variables $\dot{\mathbf{y}}_1 = \dot{\mathbf{y}}(t_1)$ and $\dot{\mathbf{y}}_2 = \dot{\mathbf{y}}(t_2)$, both lying on \mathbf{l} in accordance with Equation (12.50). We then investigate $\Pr(\bar{\mathbf{y}}_1, \bar{\mathbf{y}}_2 | \dot{\mathbf{y}}(t_1), \dot{\mathbf{y}}(t_2))$, the probability that we observed $\bar{\mathbf{y}}_1$ and $\bar{\mathbf{y}}_2$ given that they are displaced from $\dot{\mathbf{y}}(t_1)$ and $\dot{\mathbf{y}}(t_2)$, respectively. Assuming that the observations are independent events, seen as observations generated from the same line \mathbf{l} , and the same properties of the measurement noise as before, we get

$$\Pr(\bar{\mathbf{y}}_1, \bar{\mathbf{y}}_2 | \dot{\mathbf{y}}(t_1), \dot{\mathbf{y}}(t_2)) = \Pr(\bar{\mathbf{y}}_1 | \dot{\mathbf{y}}(t_1)) \cdot \Pr(\bar{\mathbf{y}}_2 | \dot{\mathbf{y}}(t_2)) = P_0(\|\bar{\mathbf{y}}_1 - \dot{\mathbf{y}}(t_1)\|) \cdot P_0(\|\bar{\mathbf{y}}_2 - \dot{\mathbf{y}}(t_2)\|). \quad (12.55)$$

To obtain $\Pr(\bar{\mathbf{y}}_1, \bar{\mathbf{y}}_2 | \mathbf{l})$, we marginalize over the two auxiliary points $\dot{\mathbf{y}}(t_1)$ and $\dot{\mathbf{y}}(t_2)$. Using the fact that the corresponding double integral is separable into the single variable integral in Equation (12.54), and using the same assumptions about the probabilities of $\dot{\mathbf{y}}(t)$ and $\bar{\eta}$ as before, we get

$$\begin{aligned} \Pr(\bar{\mathbf{y}}_1, \bar{\mathbf{y}}_2 | \mathbf{l}) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} P_0(\|\bar{\mathbf{y}}_1 - \dot{\mathbf{y}}(t_1)\|) P_0(\|\bar{\mathbf{y}}_2 - \dot{\mathbf{y}}(t_2)\|) \Pr(\dot{\mathbf{y}}(t_1)) \Pr(\dot{\mathbf{y}}(t_2)) dt_1 dt_2 = \\ &= \int_{-\infty}^{\infty} P_0(\|\bar{\mathbf{y}}_1 - \dot{\mathbf{y}}(t_1)\|) \Pr(\dot{\mathbf{y}}(t_1)) dt_1 \cdot \int_{-\infty}^{\infty} P_0(\|\bar{\mathbf{y}}_2 - \dot{\mathbf{y}}(t_2)\|) \Pr(\dot{\mathbf{y}}(t_2)) dt_2 = \\ &= \frac{\gamma}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{d_{PD}(\mathbf{y}_1, \mathbf{l})}{\sigma}\right)^2} \cdot \frac{\gamma}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{d_{PD}(\mathbf{y}_2, \mathbf{l})}{\sigma}\right)^2} = \frac{\gamma^2}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{d_{PD}(\mathbf{y}_1, \mathbf{l})^2 + d_{PD}(\mathbf{y}_2, \mathbf{l})^2}{\sigma^2}}. \end{aligned} \quad (12.56)$$

Consequently, the conditional probability depends in principle only on the sum of the squared distances between the observed points and the line.

Multiple points and a line

We have now developed all the results necessary to deal with the general case. We consider $\Pr(\bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_m | \mathbf{l})$, the conditional probability of observing the points, $\bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_m$ given that they are generated by \mathbf{l} . Using the collection of argumentations and assumptions presented for the two previous cases, we get

$$\Pr(\bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_m | \mathbf{l}) = \left(\frac{\gamma}{(2\pi)^{1/2}\sigma} \right)^m e^{-\frac{1}{2}\frac{\sum_{i=1}^m d_{PD}(\mathbf{y}_i, \mathbf{l})^2}{\sigma^2}}. \quad (12.57)$$

Consequently, in the general case of m observed points the conditional probability of making these observations given that they are generated by \mathbf{l} is, in principle, only dependent on the sum of the squared distances from the observations to the line.

⁹We leave these calculations as an exercise to the reader.

12.6.2 Maximum a posteriori estimation

So far, we have examined the conditional probability of observing m points given that they are generated by the same line model \mathbf{l} . This is the a posteriori probability of the observations given a known model m . In most practical cases, however, it works the other way around: we know the observations and want to find the probability of m , this is the a posteriori probability of m given the data. One way to estimate a model would then be to use the m which maximizes this probability. Consequently, we seek m that produce the *maximum a posteriori probability (MAP)* given the observed data. This is known as *maximum a posteriori probability estimation*, or *MAP estimation*.

For the line estimation problem, the corresponding MAP can be obtained by applying Bayes' rule to the conditional probability in Equation (12.57):

$$\Pr(\mathbf{l} | \bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_m) = \frac{\Pr(\bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_m | \mathbf{l}) \Pr(\mathbf{l})}{\Pr(\bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_m)}. \quad (12.58)$$

We can simplify this MAP formulation by assuming that $\Pr(\mathbf{l})$ is a relatively simple expression, e.g., it is constant if all lines are equally probable. But the denominator in Equation (12.58) is much more problematic. The different observations can be independent for a given line, but they are not independent as priors. As a consequence, the denominator cannot be further simplified without introducing additional assumptions that would make this estimation problem much less applicable.

In summary, the MAP approach to estimating a model is intuitively appealing, it makes sense to find the model that is most probable to have generated the observed data. Unfortunately, the MAP estimate is in most cases non-trivial to determine and, therefore, it is of less practical use. This is true for estimation problems in general, not only when we estimate a line from a set of points.

12.6.3 Maximum likelihood estimation

An alternative to MAP estimation is to accept a weaker formulation of the estimation problem, based on the conditional probability already presented for the line estimation problem in Equation (12.57). To this end, we first reformulate it to make it more applicable to the problems discussed here.

Let $\bar{\mathbf{y}}_1, \dots, \bar{\mathbf{y}}_m \in \mathbb{R}^P$ be data in the form of observed points and let m denote a model that explains the data. This means that to each $\bar{\mathbf{y}}_i$ there is a set of distinct candidates $\dot{\mathbf{y}}_i(t)$. They are all consistent with m and all could potentially have generated $\bar{\mathbf{y}}_i$ in accordance with

$$\bar{\mathbf{y}}_i = \dot{\mathbf{y}}_i(t) + \boldsymbol{\eta}, \quad (12.59)$$

although with different probabilities corresponding to the measurement noise $\boldsymbol{\eta}$.

In the line estimation problem discussed in Section 12.6.1, where m is the line \mathbf{l} , the candidate set consists of all points lying on the line, parameterized by t . In the general case, the set of candidates can consist of a single point, but can also contain multiple points. In the latter case, we use t as an abstract index of the different candidates, regardless of whether this set is enumerable by an integer, or if its elements are a function of a real number, or of an element in \mathbb{R}^P . Using these candidates, we can write

$$\Pr(\bar{\mathbf{y}}_i | m) = \Pr(\bar{\mathbf{y}}_i | \dot{\mathbf{y}}_i(t)) \quad \text{marginalized over } t. \quad (12.60)$$

For the line estimation problem, this marginalization corresponds to the integrals in Equation (12.52).

In the simplest case, the model produce a single candidate $\dot{\mathbf{y}}(t) = \dot{\mathbf{y}}$ corresponding to the observed data $\bar{\mathbf{y}}$. Since the two points are related as in Equation (12.59), we can formulate the conditional probability of observing $\bar{\mathbf{y}}$ given that it has been generated by model m :

$$\Pr(\bar{\mathbf{y}} | m) = \Pr(\bar{\mathbf{y}} | \dot{\mathbf{y}}) = \Pr_{\text{noise}}(\bar{\mathbf{y}} - \dot{\mathbf{y}}) = \Pr_0(\|\bar{\mathbf{y}} - \dot{\mathbf{y}}\|). \quad (12.61)$$

Here, we are again assuming that the measurement noise is isotropic in accordance with Equation (12.21).

As a function of m , $\Pr(\bar{\mathbf{y}} | m)$ does not behave like a probability of m . In contexts where we want to explore

• Bayes' rule
is presented
in Toolbox
Section
5.2.4.2.

this expression for a known $\bar{\mathbf{y}}$ and variable \mathbf{m} , we therefore need to call it something else.

Definition 12.4: Likelihood

Given observed data \mathbf{y} , the conditional probability $\Pr(\bar{\mathbf{y}} | \mathbf{m})$ is referred to as the *likelihood* of \mathbf{m} corresponding to observation $\bar{\mathbf{y}}$.

To find the likelihood of \mathbf{m} corresponding to the entire set of m observations, we assume that the different points have independent measurement noise:

$$\begin{aligned}\Pr(\mathbf{y}_1, \dots, \bar{\mathbf{y}}_m | \mathbf{m}) &= \Pr(\mathbf{y}_1, \dots, \bar{\mathbf{y}}_m | \dot{\mathbf{y}}_1, \dots, \dot{\mathbf{y}}_m) = \Pr(\bar{\mathbf{y}}_1 | \dot{\mathbf{y}}_1) \cdot \dots \cdot \Pr(\bar{\mathbf{y}}_m | \dot{\mathbf{y}}_m) = \\ &= \Pr_0(\|\bar{\mathbf{y}}_1 - \dot{\mathbf{y}}_1\|) \cdot \dots \cdot \Pr_0(\|\bar{\mathbf{y}}_m - \dot{\mathbf{y}}_m\|).\end{aligned}\quad (12.62)$$

Finding the model \mathbf{m} that maximizes this likelihood is referred to as *maximum likelihood estimation* or *ML-estimation*.

Connection to geometric errors

By assuming also a Gaussian formulation for the measurement noise, described in Equation (12.53), the model likelihood can be written as

$$\Pr(\mathbf{y}_1, \dots, \bar{\mathbf{y}}_m | \mathbf{m}) = \left(\frac{1}{2\pi\sigma} \right)^{\frac{mp}{2}} e^{-\frac{1}{2} \frac{\sum_{i=1}^m \|\bar{\mathbf{y}}_i - \dot{\mathbf{y}}_i\|^2}{\sigma^2}}. \quad (12.63)$$

The logarithm is a strictly monotonic function, which implies that maximizing the model likelihood is equivalent to minimizing its negative logarithm:

$$-\log \Pr(\mathbf{y}_1, \dots, \bar{\mathbf{y}}_m | \mathbf{m}) = \frac{mp}{2} \log(2\pi\sigma) + \frac{1}{2} \frac{\sum_{i=1}^m \|\bar{\mathbf{y}}_i - \dot{\mathbf{y}}_i\|^2}{\sigma^2}. \quad (12.64)$$

In the end, this expression is minimized by a model that makes

$$\varepsilon_{\text{SSD}} = \sum_{i=1}^m \|\bar{\mathbf{y}}_i - \dot{\mathbf{y}}_i\|^2 \quad (12.65)$$

minimal. This result enables us to make the following observation:

12.16 Assuming that the measurement noise is isotropic, independent, and Gaussian, Maximum Likelihood estimation corresponds to minimizing a geometric error using the L_2 -norm, described in Section 12.4.1.

Chapter 13

Estimation of Transformations

Before you read this chapter, you should have a thorough understanding of the homogeneous representations of transformations and projections, described in Chapters 4 and 6 and in Chapters 7 and 8. You should also have a look at the analysis presented in Toolbox Section 8.4, on of how the eigensystem of a matrix is perturbed when the matrix itself is perturbed.

In this chapter we continue to develop the theme on estimation that began in Chapter 12, where we concluded that as long as we formulate the estimation problem in terms of algebraic errors, it can be solved relatively easily based on linear methods: the inhomogeneous and the homogeneous method. We also discussed some estimation problems based on geometric errors that can be dealt with in similar ways. In this chapter, we observe that geometric errors in general lead to non-linear minimization problems. As an alternative to geometric errors for the estimation problems discussed in this chapter, algebraic errors is an alternative that lead to simple solutions. Although they do not minimize geometric quantities, and therefore not reliable as a basis for estimation in geometry, they can be justified by providing useful initial solutions to the iterative methods that minimize geometric errors. They can also provide an initial characterization of the solution space of the problem, i.e., if we are likely to find a model that fits nicely to the data, or not, and how unique such a model would be.

13.1 Homography estimation

At this point we leave the simple example of fitting a line to a set of 2D points, discussed in Chapter 12, and instead consider the problem of fitting a homography to a set of corresponding 2D points. Walking through the different steps towards a working solution of this problem, we will make several general observations, which apply also to other, similar, estimation problems.

In this context, *corresponding points* refer to the “same” point before and after the homography transformation has been applied, as described in observation 9.1 on page 122. We can formally described this estimation problem as: we want to determine a homography \mathbf{H} such that

$$\mathbf{y}'_k \sim \mathbf{H}\mathbf{y}_k, \quad k = 1, \dots, m, \tag{13.1}$$

where the pair $(\mathbf{y}_k, \mathbf{y}'_k)$ of homogeneous coordinates of point k before and after the transformation is known. These pairs of corresponding points are the data, and the model to be fitted to the data is $\mathbf{H} \in P(\mathbb{R}^{3 \times 3})$. We want to determine \mathbf{H} that makes the left and right-hand sides of Equation (13.1) as equal as possible for $k = 1, \dots, m$. As before, the data cannot be fitted perfectly to a homography since it contains measurement errors.

To find this \mathbf{H} we first need to formulated an error function that measures the fit between the data and the model. In Chapter 12, we argue that errors should preferably have a reasonable geometric interpretation, so this is what we try first.

13.1.1 Geometric errors

A simple geometric error function for the problem of estimating a homography is

$$\varepsilon_{\text{HOM}} = \sum_{k=1}^m d_{\text{PP}}(\mathbf{y}'_k, \mathbf{H}\mathbf{y}_k)^2, \quad (13.2)$$

where d_{PP} is the Euclidean point-to-point distance function in \mathbb{E}^2 , defined in Equation (3.31). Let us now examine the error function ε_{HOM} and draw some conclusions from it.

To simplify the analysis, we express the homography \mathbf{H} in terms of its rows:

$$\mathbf{H} = \begin{pmatrix} - & \mathbf{h}_1 & - \\ - & \mathbf{h}_2 & - \\ - & \mathbf{h}_3 & - \end{pmatrix}. \quad (13.3)$$

The error function ε_{HOM} can then be rewritten as

$$\varepsilon_{\text{HOM}} = \sum_{k=1}^m \left(\frac{\mathbf{h}_1 \cdot \mathbf{y}_k}{\mathbf{h}_3 \cdot \mathbf{y}_k} - u'_k \right)^2 + \left(\frac{\mathbf{h}_2 \cdot \mathbf{y}_k}{\mathbf{h}_3 \cdot \mathbf{y}_k} - v'_k \right)^2, \quad (13.4)$$

where $\bar{\mathbf{y}}_k = (u'_k, v'_k)$ are the Cartesian coordinates corresponding to homogeneous coordinates \mathbf{y}'_k . ε_{HOM} is a function of the three vectors $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3 \in \mathbb{R}^3$, and a common scalar multiplication of all three vectors does not change ε_{HOM} . The optimal ε_{HOM} is a stationary point with respect to these variables, and is characterized by $\nabla_{\mathbf{H}}\varepsilon_{\text{HOM}} = \mathbf{0}$, where $\nabla_{\mathbf{H}}$ is the gradient with respect to all 9 elements in the three vectors. We can see $\nabla_{\mathbf{H}}\varepsilon_{\text{HOM}}$ as the concatenation of the gradients with respect to each of the three rows in \mathbf{H} :

$$\begin{aligned} \nabla_{\mathbf{h}_1}\varepsilon_{\text{HOM}} &= 2 \sum_{k=1}^m \frac{\mathbf{y}_k}{\mathbf{h}_3 \cdot \mathbf{y}_k} \left(\frac{\mathbf{h}_1 \cdot \mathbf{y}_k}{\mathbf{h}_3 \cdot \mathbf{y}_k} - u'_k \right), \\ \nabla_{\mathbf{h}_2}\varepsilon_{\text{HOM}} &= 2 \sum_{k=1}^m \frac{\mathbf{y}_k}{\mathbf{h}_3 \cdot \mathbf{y}_k} \left(\frac{\mathbf{h}_2 \cdot \mathbf{y}_k}{\mathbf{h}_3 \cdot \mathbf{y}_k} - v'_k \right), \\ \nabla_{\mathbf{h}_3}\varepsilon_{\text{HOM}} &= -2 \sum_k \left(\frac{\mathbf{y}_k(\mathbf{h}_1 \cdot \mathbf{y}_k)}{(\mathbf{h}_3 \cdot \mathbf{y}_k)^2} \left(\frac{\mathbf{h}_1 \cdot \mathbf{y}_k}{\mathbf{h}_3 \cdot \mathbf{y}_k} - u'_k \right) + \frac{\mathbf{y}_k(\mathbf{h}_2 \cdot \mathbf{y}_k)}{(\mathbf{h}_3 \cdot \mathbf{y}_k)^2} \left(\frac{\mathbf{h}_2 \cdot \mathbf{y}_k}{\mathbf{h}_3 \cdot \mathbf{y}_k} - v'_k \right) \right). \end{aligned} \quad (13.5)$$

Here is where the problems start. Setting all three gradients to the zero vector and solving for $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3$ is not trivial, since the unknown variables appear in both the numerators and the denominators of the terms in the three sums. Even worse, each denominator depends on k , implying that an expression of a sum that has a common denominator for all terms produces numerators of high order products in the unknowns. In fact, there is no known method that can determine \mathbf{H} such that $\nabla_{\mathbf{H}}\varepsilon_{\text{HOM}} = \mathbf{0}$ in a straightforward way, for general points $(\mathbf{y}_k, \mathbf{y}'_k)$.

13.1 Some estimation problems based on geometric errors, in fact most of the interesting ones, cannot be solved based on simple methods, such as solving a linear equation. Instead, these problems have to be addressed using more complicated techniques based on iterative non-linear optimization.

We will return to the problem of estimating homographies based on geometric errors in Section 14.5. Instead, we turn to algebraic errors for homography estimation, which fortunately is a simpler nut to crack.

13.1.2 Direct Linear Transformation (DLT)

How can we formulate an algebraic error function such that its minimization leads to solving a linear equation in the unknowns? A complication is that Equation (13.1) does not express an equality between the left and right-hand sides, it expresses equivalence¹. If we think of $\mathbf{y}'_k, \mathbf{y}_k$ as vectors in \mathbb{R}^3 and \mathbf{H} as a 3×3 matrix, then we are allowed to multiply either of them by a non-zero scalar without changing their interpretation as a homogeneous

¹As it is defined in Toolbox Section 7.1.1, Equation (7.1).

representation of a point or a transformation in \mathbb{E}^2 . Such an operation, however, in general changes the difference of the vector on the left and right-hand sides of Equation (13.1). Consequently, we need to way of expressing the degree of equivalence between the left and right-hand sides of Equation (13.1) that is independent of absolute magnitude of the vectors or matrices that are involved. This is what direct linear transformation is about.

One way to deal with Equation (13.1) is to write the whole expression as a standard equation based on equality between the left and right sides:

$$\gamma_k \mathbf{y}'_k = \mathbf{H} \mathbf{y}_k, \quad k = 1, \dots, m, \quad (13.6)$$

where

$$\mathbf{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}, \quad \mathbf{y}_k = \begin{pmatrix} u_k \\ v_k \\ 1 \end{pmatrix}, \quad \mathbf{y}'_k = \begin{pmatrix} u'_k \\ v'_k \\ 1 \end{pmatrix}. \quad (13.7)$$

Here, we assume that \mathbf{y}_k and \mathbf{y}'_k are vectors in \mathbb{R}^3 and that \mathbf{H} is a 3×3 matrix. The two vectors are representatives of projective elements, in this case the canonical form of the homogeneous coordinates of the points (u_k, v_k) and (u'_k, v'_k) , respectively.

One difference between Equation (13.1) and Equation (13.6) is that the latter expresses an equality between the left and right-hand sides, and since both sides are vectors in \mathbb{R}^3 there are, in fact, three equations that describe how the unknown elements of the matrix \mathbf{H} depend on the coordinates of the points. On the other hand, there are now additional unknown variables: γ_k . These *auxiliary variables* represent any possible scalar multiplication onto any of \mathbf{y}_k , \mathbf{y}'_k and \mathbf{H} that would be used to accomplish equality between the left and right-hand sides. Consequently, there must be one such independent variable, γ_k , for each $k = 1, \dots, m$. We can then solve this variable from one of the three equations and insert it into the other two. As a result, we get *two* equations that only contain the known points and the unknown elements of \mathbf{H} , and not γ_k .

This is a reasonable outcome: the homography transformation that is described in Equation (13.1) defines $\bar{\mathbf{y}}'_k = (u'_k, v'_k)$, the Cartesian coordinates of the resulting point, as an expression in $\bar{\mathbf{y}}_k$ and \mathbf{H} . Each of u'_k and v'_k is a non-linear expression in $\bar{\mathbf{y}}_k$ and \mathbf{H} and, as a result, Equation (13.1) describes two equations in the unknown \mathbf{H} for each k . To proceed we can, for example, solve γ_k from the third equation in Equation (13.6):

$$\gamma_k = h_{31}u_k + h_{32}v_k + h_{33}. \quad (13.8)$$

We insert this into the other two equations, and get:

$$\begin{aligned} 0 &= h_{11}u_k + h_{12}v_k + h_{13} - (h_{31}u_k + h_{32}v_k + h_{33})u'_k, \\ 0 &= h_{21}u_k + h_{22}v_k + h_{23} - (h_{31}u_k + h_{32}v_k + h_{33})v'_k. \end{aligned} \quad (13.9)$$

This implies that we get *two linear homogeneous equations* in the unknown elements of \mathbf{H} for each $k = 1, \dots, m$, in total $2m$ equations. This approach of rewriting Equation (13.1), which expresses equivalence between the left and right-hand sides, to the linear homogeneous equations in Equation (13.9) is referred to as a *direct linear transformation*, or *DLT* for short. It was introduced by Abdel-Aziz & Karara [1] as a means for solving problems in photogrammetry.

An alternative approach for dealing with Equation (13.1) that leads to the same result is to rewrite it as

$$\mathbf{0} = \mathbf{y}'_k \times (\mathbf{H} \mathbf{y}_k), \quad k = 1, \dots, m, \quad (13.10)$$

a cross product between \mathbf{y}'_k and $\mathbf{H} \mathbf{y}_k$. This is an equivalent expression to Equation (13.1): if \mathbf{y}'_k satisfies Equation (13.1) it must satisfy Equation (13.10), and vice versa. Yet another equivalent form is to express the cross product in terms of the cross product operator:

$$\mathbf{0} = [\mathbf{y}'_k] \times \mathbf{H} \mathbf{y}_k, \quad k = 1, \dots, m. \quad (13.11)$$

This corresponds to three linear homogeneous equations in the unknown elements of \mathbf{H} . In accordance with , however, the 3×3 matrix² $[\mathbf{y}'_k] \times$ has only rank 2, which means that at least one of its rows can be expressed as a linear combination of the other two rows. Consequently, at least one of the three equations in Equation (13.11) can be expressed as a linear combination of the other two. In general, we can therefore remove one of the three equations in Equation (13.11) without losing any constraints in \mathbf{H} . For example, if we discard the third equation in

²See Toolbox Section 3.7.3

Equation (13.11) and keep the other two, the latter are exactly the ones derived in Equation (13.9), but they come in different order and one of the equations has opposite sign.

This second approach to deriving the DLT-equations in Equation (13.9) is not only simpler, since we do not have to introduce additional variables which then are removed by solving equations, it is also more general since it does not assume any particular interpretation of the vectors \mathbf{y}'_k . Here, they are homogeneous coordinates of points in \mathbb{E}^2 , but the derivation of Equation (13.11) works also for other types of homogeneous representations, such as lines in 2D.

So far we have concluded that each pair of corresponding points $(\mathbf{y}_k, \mathbf{y}'_k)$ provides two linear homogeneous equations in the unknown elements of \mathbf{H} . We can formalize these two equations as

$$\mathbf{A}_k \mathbf{z} = \mathbf{0}, \quad (13.12)$$

where $\mathbf{A}_k \in \mathbb{R}^{2 \times 9}$ and $\mathbf{z} \in \mathbb{R}^9$ are given as

$$\mathbf{A}_k = \begin{pmatrix} u_k & 0 & -u_k u'_k & v_k & 0 & -v_k u'_k & 1 & 0 & -u'_k \\ 0 & u_k & -u_k v'_k & 0 & v_k & -v_k v'_k & 0 & 1 & -v'_k \end{pmatrix}, \quad \mathbf{z} = \begin{pmatrix} h_{11} \\ h_{21} \\ h_{31} \\ h_{12} \\ h_{22} \\ h_{32} \\ h_{13} \\ h_{23} \\ h_{33} \end{pmatrix}. \quad (13.13)$$

This means that we can summarize the $2m$ linear homogeneous equations as

$$\mathbf{A} \mathbf{z} = \mathbf{0}, \quad \text{where} \quad \mathbf{A} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{pmatrix}. \quad (13.14)$$

Here, \mathbf{A} is a $2m \times 9$ matrix with elements derived from the coefficients in the DLT-equations in Equation (13.9), consisting of products of elements of the two vectors \mathbf{y}_k and \mathbf{y}'_k , where $k = 1, \dots, m$. Furthermore, \mathbf{z} is a *vectorization* of the 3×3 matrix \mathbf{H} into a vector in \mathbb{R}^9 . Thus, we have arrived at a linear homogeneous equation, Equation (13.14), for the estimation problem. This can be solved by means of either the inhomogeneous or the homogeneous method, with the second option as the recommended. Here, we have reshaped the 3×3 matrix \mathbf{H} to the vector $\mathbf{z} \in \mathbb{R}^9$. This reordering of the elements in \mathbf{H} to a vector can be done in any other way as long as we stick to a specific reordering all the time, and we order the columns of \mathbf{A}_k accordingly. Once \mathbf{z} has been estimated we can then reshape it back to \mathbf{H} by applying the inverse reordering.

13.2 For some estimation problem we may need to reshape the homogeneous representation of the geometric object that is estimated to a vector. In this case, we estimate this vector and then reshape it back to the original homogeneous representation.

Using DLT as it is described here works fine in most cases, but there is a minor pitfall. From the outset, it may appear as if we always can solve γ_k from the third equation in Equation (13.6), or as if we can discard the third equation in Equation (13.11) and only use the other two. This is correct as long as the third element of \mathbf{y}'_k is not zero, i.e., \mathbf{y}'_k is not a point at infinity. In this case, one of the first two equations simply says: $0 = 0$. This equation is correct, but it does not provide any constraint on the unknown elements in \mathbf{H} . If we know that we are dealing with proper points, this is not an issue, but if points at infinity are involved in the homography estimation, or if there are other geometric objects than points involved, this means that one pair of points $(\mathbf{y}_k, \mathbf{y}'_k)$ only provides one constraint on \mathbf{H} instead of two. In the general case, the chance for this happening may be very small, and if we have sufficiently many points is not a big deal to lose one constraint. In applications where not losing a constraint is critical for the estimation, a safer option is to use all three constraints in Equation (13.11) at the cost of increasing the size of the resulting data matrix. In this case, \mathbf{A} then becomes a $3m \times 9$ matrix. Choose wisely.

We summarize the discussion of this section as:

13.3 Equivalence relations among homogeneous representation can be transformed, using DLT, into linear homogeneous equations.

13.1.3 The correspondence problem

For some of the simpler estimation problems in geometry, there is no correspondence problem, e.g., as in the case of fitting a line to 2D points, or it is already solved for the dataset. Most of the interesting and practically relevant estimation problems in geometry, however, require correspondences. For example, the problem of estimating the homography \mathbf{H} that we discuss here relies of known correspondences, image points before and after the transformation is applied. The related problem of estimating a camera matrix, discussed in Section 13.3, also requires a set of corresponding points, where a set of 3D points and their projections in the camera image is known.

If correspondences are not available, we need to also solve a correspondence problem. Possible strategies for solving correspondence problems are discussed in Section 17.4.

13.4 Many estimation problems rely on the data set to contain correspondences. If points exist but are not brought into correspondences, standard estimation technique will fail. Instead we must consider more advanced techniques that can solve also the correspondence problem.

13.1.4 Minimal case estimation

In Section 7.1 it is shown that a homography \mathbf{H} in 2D has 8 degrees of freedom and that it requires at least 4 points in general configuration to be determined. This is consistent with the discussion in Section 13.1.2 where each pair of corresponding points provides 2 constraints in the elements of \mathbf{H} , and from 4 points we get an 8×9 data matrix \mathbf{A} . In the case that \mathbf{A} has full row rank, there is then a well-defined projective element \mathbf{z} that exactly satisfies Equation (13.14), i.e., the algebraic error $\|\mathbf{Az}\|$ can be reduced to zero. This situation is referred to as *minimal case estimation*. Minimal case estimation implies that the model can be fitted to the data with zero error simply because the number of point in the data set is so small that it matches the number of free parameters of the model. More precisely, each point in the data set corresponds to a certain number of constraints in the model parameters, and for a certain minimum number of points the constraints are sufficient to determine the model parameters. Another characteristic property of minimal case estimation is that the determination of the model parameters often can be made by means of simplified or specialized methods that work only for the minimal case.

In the case of homography estimation, the main simplification for minimal case estimation is that the homography can be determined by means of either the inhomogeneous or the homogeneous method and they produce the same result, at least as long as the equation system of the inhomogeneous method is not ill-conditioned. Furthermore, the computations that are used in determining this homography are fewer the fewer points that are involved. Returning to the case of line estimation, we know that the line can be determined from a minimum of two points. If $m = 2$, we do not have to go through all the work of defining an error function, and minimizing it by solving some equation. As is described in Section 3.3.1, the dual homogeneous coordinates of the line can be determined simply as $\mathbf{l} = \mathbf{y}_1 \times \mathbf{y}_2$, where \mathbf{y}_1 and \mathbf{y}_2 are the homogeneous coordinates of the two points. Similar simplifications may be possible also for the minimal case of other estimation problems, but the distinct character of these approaches is that they are *specialized* for minimal case estimation and of *fixed size* in terms of computational resources such as time and memory.

Given that it is so simple to determine the line that optimally fits two points, it may appear attractive to establish a type of heuristic method for fitting a line to a larger set of points. We simply pick any two of the points, labeled \mathbf{y}_1 and \mathbf{y}_2 , and determine the line as $\mathbf{l} = \mathbf{y}_1 \times \mathbf{y}_2$. As illustrated in Figure 13.1, this may be a good idea, left, or a bad idea, right. For this particular example, this approach works better the further apart the two points are, but for a general estimation problem, it may be difficult to establish a similar criteria for how to pick a reasonable minimal set from the original dataset, from which a model can be determined with minimal case estimation. The error between the resulting model and the remaining points may be considerable. An estimation method that takes

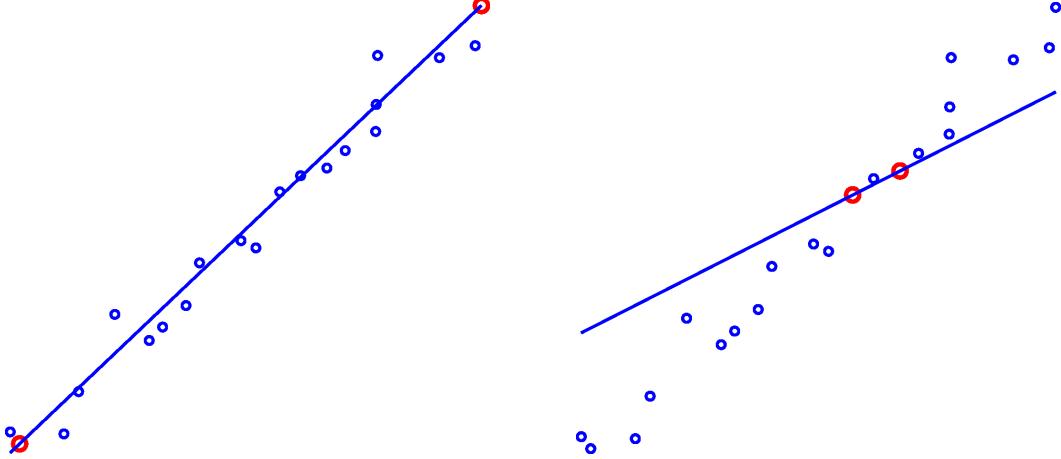


Figure 13.1: Two examples of minimal case estimation of a line from a set of points. Left: the line is estimated from two points relatively far apart. Right: the line is estimated from two points that are relatively close.

all points into account is therefore a safer option, since this tends to minimize some type of average of the error between each point and the estimated model. Despite this issue, minimal case estimation is relevant in some cases, in particular when a large number of estimations have to be made in a limited number of time.

13.5 In some situations, minimal case estimation offers computational advantages that makes it relevant, but this approach should be used with some care.

13.1.5 Degeneracies

It is mentioned both here and in Section 7.1 that the points which are used for determining a homography should be in general configuration. This means that they do not satisfy any explicit or implicit constraint of any kind: intuitively we can think of them as approximately fitting some model, but otherwise be in random positions. If this is not the case degeneracies can appear.

Returning to the line fitting example, it is obvious that if we have only a single point, or if we have multiple points but they are approximately identical, there is no single well-defined line to be estimated. If we use the inhomogeneous method, the matrix $\mathbf{A}_0^\top \mathbf{A}_0$ in Equation (12.43) is singular or is close to singular in this case. For the homogeneous method, there are instead two singular values that are equal to or are close to zero in this case. The problem is not related to the estimation method, it is rather the dataset that is ill-conditioned relative the line fitting problem. We refer to this as *data degeneracy*.

A similar situation appears for the problem of estimating a homography if the two sets of corresponding points, \mathbf{y}_k and \mathbf{y}'_k , both lie on a line, or approximately on a line. To see this, assume that we have a homography \mathbf{H} that satisfy Equation (13.1) for a set of corresponding points, but all \mathbf{y}_k lie on a line \mathbf{l} , i.e., $\mathbf{l} \cdot \mathbf{y}_k = 0$ for $k = 1, \dots, m$. This implies that Equation (13.1) also is satisfied by the homography

$$\mathbf{H}' = \mathbf{H} + \mathbf{a}\mathbf{l}^\top, \quad (13.15)$$

where $\mathbf{a} \in \mathbb{R}^3$. This follows from

$$\mathbf{H}'\mathbf{y}_k = (\mathbf{H} + \mathbf{a}\mathbf{l}^\top)\mathbf{y}_k = \mathbf{H}\mathbf{y}_k + \underbrace{\mathbf{a}(\mathbf{l} \cdot \mathbf{y}_k)}_{=0} = \mathbf{H}\mathbf{y}_k \sim \mathbf{y}'_k. \quad (13.16)$$

Consequently, \mathbf{H} is not unique in this case, there is a three-dimensional affine space of matrices \mathbf{H} that solve Equation (13.1). This means that the solution space, instead of being one-dimensional, is four-dimensional.

If we try to estimate \mathbf{H} from such a degenerate dataset, it does not matter which method we use. The method will either fail completely, or at best produce some \mathbf{H} from the solution space, but it is unlikely that it will produce the correct one. This will happen also if the points are approximately lying on a line.

13.6 For most estimation problems, there are degenerate sets of data for which there is no unique solution, so-called *data degeneracy*. Avoid such data, or make sure that your estimation technique treats this case gracefully.

As indicated earlier, data degeneracy may not have to be problem. For example, if we are using the homogeneous method for minimizing algebraic errors, data degeneracy implies that the *solution space*, i.e., the null space of \mathbf{A} in Equation (12.37), is of a dimension larger than one. For the practical case, this implies that there are several singular values of \mathbf{A} that are of similar magnitude and close to zero. As long as we pick \mathbf{z} from the space spanned by the corresponding right singular vectors, we have a model that fits the data. This is in contrast to what happens for the inhomogeneous method, where instead the linear inhomogeneous equation system Equation (12.42) that we try to solve becomes degenerate and cannot be solved by standard techniques. However, as has been discussed previously, this situation can also happen for specific cases of perfectly “normal” data for which there exists a solution to the estimation problem. For example, if we use the inhomogeneous method for estimating a line, represented by its dual homogeneous coordinates \mathbf{l} , and based on algebraic errors, this method fails to reliably estimate a line that passes through the origin if we set the last element of \mathbf{l} equal to one. Had we instead chosen to set one of the other two elements equal to one, there would still be some line that cannot be reliably estimated since the matrix $\mathbf{A}_0^\top \mathbf{A}_0$ becomes singular.

This situation is here referred to as *method degeneracy* and relates to a singularity of the estimation method rather than to the data, and it is not unique to the inhomogeneous method. Some estimation methods have certain configurations of the data from which a model cannot be estimated even if one exists. We summarize these observations as:

13.7 Some estimation methods exhibit degeneracy in that they may not be able to estimate a model for certain cases of data, even when a model can be estimated. Avoid this type of data for these methods, or avoid using such methods, or at least be aware for what data the method becomes degenerate.

Another type of data degeneracy can be said to appear when that data does not fit the model at all, for example, if we try to fit a line to points that, in fact, approximately lie on a circle, as is discussed in Section 12.3.2.

13.2 The homogeneous method revisited

We turn, again, to the problem of fitting a line to a set of 2D points, e.g., those shown in Figure 13.2, left. In accordance with Section 12.5, we can solve this problem by forming a matrix \mathbf{Y} that holds the homogeneous coordinates of the points and find the dual homogeneous coordinates of the line, \mathbf{l} , by solving Equation (12.34). Due to the errors described in Section 12.3, this equation cannot be solved exactly and, instead, we determine the \mathbf{l} that minimizes the algebraic error $\|\mathbf{Y}^\top \mathbf{l}\|$. Based on the additional constraint that \mathbf{l} has unit norm, the solution to this minimization problem is to choose \mathbf{l} as a normalized right singular vector corresponding to the smallest singular value of \mathbf{Y}^\top . The resulting line is shown together with the points in Figure 13.2, left. This is a relatively straightforward approach, but there are a few issues that need to discussed in more detail.

13.2.1 SVD profile

To the right in Figure 13.2 we see the singular values of \mathbf{Y}^\top , denoted $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$, derived from the dataset to the left. We see that there is indeed one singular value that is small, or close to zero, and the other two are significantly larger. We also see that the two larger singular values are not of the same magnitude. Since we are considering an algebraic error, there is no absolute scale that we can relate to the singular values, i.e., their absolute magnitudes do not have any obvious geometric interpretation. Instead, their relative magnitudes are more

interesting quantities. For this particular example of data, it turns out that

$$\frac{\sigma_1}{\sigma_2} \approx 12.0, \quad \text{and} \quad \frac{\sigma_2}{\sigma_3} \approx 12.7. \quad (13.17)$$

This means that the ratio between the smallest and the second smallest singular value is approximately the same as the ratio between the second smallest and the largest. On the one hand, we have a one singular value, σ_3 , which is significantly smaller than the second smallest, σ_2 . On the other hand, we also see that σ_2 is “as small” relative to σ_1 . In this case, should we say that there is one well-defined singular value that is small, or are there two small singular values?

Although the issues raised so far concerning the homogeneous method have not be resolved yet, we can at least make the observation that the SVD profile offers a useful piece of information about the data that is used in the estimation. In principle, we can divide the SVD profile into three characteristic cases:

1. There is *one* single singular value that is approximately equal to zero. This is often the ideal case that we would like to see, since it means that there is one well-defined solution to $\mathbf{A}\mathbf{z} = \mathbf{0}$.
2. There are two or more singular values that are approximately equal to zero. This case implies that the estimation problem does not have a unique solution, e.g., due to degeneracies in data or in the method.
3. There is no singular value that is approximately equal to zero. This case implies that there are no solutions to the estimation problem. This happens typically when the data does not fit the model used in the estimation.

What we are discussing here is the fact that a specific SVD profile does not necessary have to fall clearly into one of these three cases. A second illustration of this statement is provided by the two sets of 2D points in Figure 13.3, left and right, which are related by a homography transformation. The circles are the ideal points that can be fitted exactly to a homography, and the crosses are more realistic examples of observed points, with some amount of noise. The latter dataset is used to determine the data matrix \mathbf{A} in Section 13.1, and for the homogeneous method we get the singular values of \mathbf{A} that are plotted in Figure 13.4, left. In the following, we refer to this type of plot, and similarly to Figure 13.2, right, which illustrate the SVD of the data matrix, as an *SVD profile*. By examining the SVD profile we can get an understanding of the quality of the resulting estimation. In Figure 13.3, left, we see for the current problem of homography estimation from the resulting SVD profile that there are 2 relatively large singular values, 4 more that are of “medium magnitude” and, finally, three singular values that are significantly smaller than the first two groups. In this type of plot, it is even difficult to see how close to zero the smallest singular values are, and their relative differences. This analysis is simplified if we instead plot the logarithm of the singular values, as is illustrated in Figure 13.4, right.

The critical issue is that it is not obvious how to interpret this particular SVD profile. One interpretation is to say that there is a smallest singular value σ_9 , and according to theory its corresponding right singular vector represents

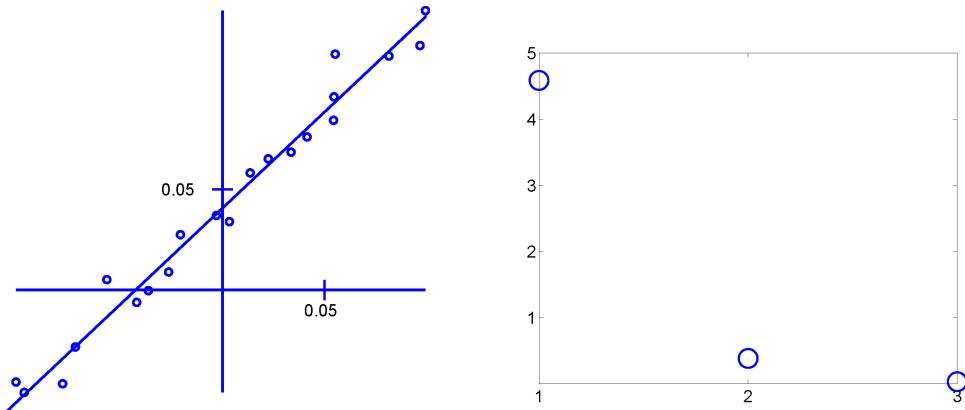


Figure 13.2: Left: a set of 2D points that approximately can be fitted to a line. This figure also shows the line that is estimated by algebraic minimization based on the homogeneous method. Right: the singular values of the corresponding data matrix.

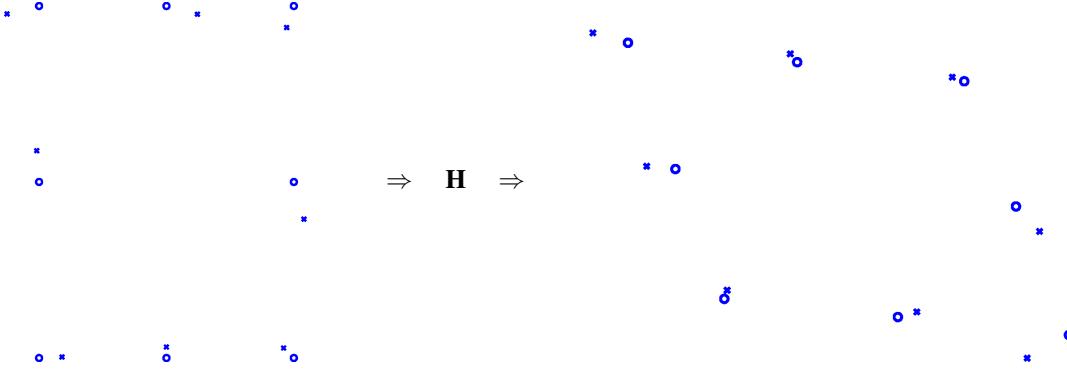


Figure 13.3: Two sets of 2D points that are related by a homography transformation \mathbf{H} . Left: before the homography transformation. Right: after the transformation. Circles: points that can be fitted exactly to a homography. Crosses: the same points perturbed by noise. The latter are what we expect to observe in practice.

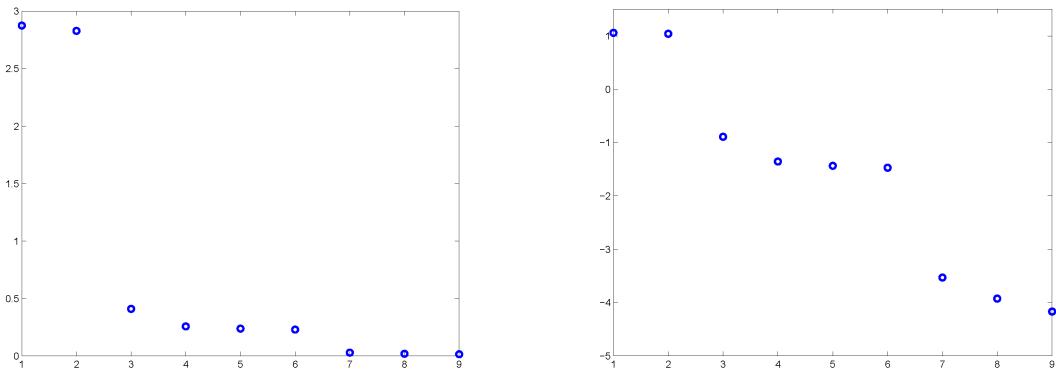


Figure 13.4: The SVD profile corresponding to the points in Figure 13.3. To the right on a logarithmic scale.

the optimal estimate of the homography, given that we minimize an algebraic error. Another interpretation is to say that since the difference between the three smallest singular values, $\sigma_7, \sigma_8, \sigma_9$, are small we have some type of degeneracy. This degeneracy implies here that the solution space for \mathbf{H} is three-dimensional, spanned by the right singular vectors corresponding to $\sigma_7, \sigma_8, \sigma_9$. For example, choosing the homography as the right singular vector corresponding to σ_8 , denoted \mathbf{H}_8 , instead of the homography related to σ_9 , denoted \mathbf{H}_9 , only makes a minor increase in the algebraic error that we minimize. Maybe this small difference is only due to noise in a degenerate dataset? This situation is illustrated in Figure 13.5. In the left plot we see the result of applying \mathbf{H}_9 to the noisy points in Figure 13.3, left. The resulting points are marked with a “+” and each is connected to its corresponding data point with a line. Given that there is some amount of noise in the data, the result appears plausible. A similar result is shown in the right plot, where we instead apply \mathbf{H}_8 . Given these plots, it is not obvious why \mathbf{H}_8 is not as good an estimate as \mathbf{H}_9 . This question is also motivated by the fact that the geometric error ε_{HOM} in Equation (13.2) happens to be almost twice as large for \mathbf{H}_9 compared to \mathbf{H}_8 .

13.8 Mind the SVD profile when the homogeneous method is used. It provides additional information about the estimate, not available from the inhomogeneous method.

The situation that is described here implies that we have some type of degeneracy; we have a data set for which there should be a solution, but when the homogeneous method is applied to the data the dataset appears as if it is degenerate. This should be classified as method degeneracy since the problem does not lie in the data. Is the homogeneous method really appropriate for this dataset? The answer is: yes, it is appropriate, but it should be

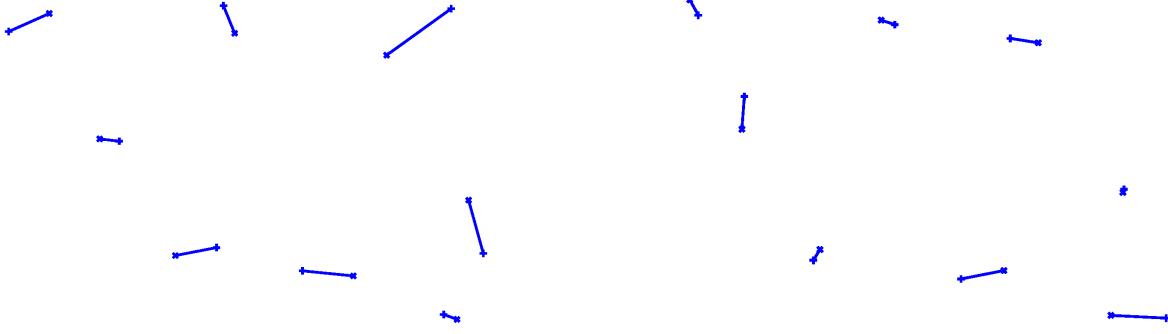


Figure 13.5: The result of applying the estimated homography to the dataset in Figure 13.3, left and right. Crosses: the positions of the points in the right set. Pluses: positions of the left set transformed by the estimated homography. Left: using the estimate \mathbf{H}_9 corresponding to σ_9 . Right: using the estimate \mathbf{H}_8 corresponding to σ_8 . Notice that the geometric error is larger for \mathbf{H}_9 than it is for \mathbf{H}_8 .

applied to normalized data.

13.2.2 Data normalization

We leave the problem of a possible degeneracy for the moment, and instead return to the original homogeneous equation in Equation (12.37), where \mathbf{A} is the data matrix and \mathbf{z} is the model parameters to be estimated. Until now we have not made any particular assumptions about the formation of \mathbf{A} , other than each of its rows, \mathbf{a}_k , represents a linear constraint in \mathbf{z} defined by some data. More precisely, each \mathbf{a}_k should be orthogonal to \mathbf{z} , and we quantify this condition by the scalar product $\mathbf{a}_k \cdot \mathbf{z}$. The orthogonality condition is invariant to a multiplication by any non-zero scalar onto \mathbf{a}_k . This makes sense since \mathbf{a}_k is derived from homogeneous representations of various geometric objects which, by definition, can have an arbitrary scalar multiplied to it. In the practical case, however, when the data is perturbed by noise, $\mathbf{a}_k \cdot \mathbf{z}$ has a numerical value that is non-zero and then the scaling of \mathbf{a}_k matters. A scaling of \mathbf{a}_k by a factor of 2 means that the corresponding algebraic error $(\mathbf{a}_k \cdot \mathbf{z})^2$ is 4 times larger before it is being added together with the other errors into the overall algebraic error ε_{ALG} in Equation (12.36). Consequently, if we scale only a certain row of \mathbf{A} by a factor of 2, the corresponding constraint in \mathbf{z} gets a higher weight in the estimation and becomes slightly smaller for the optimal \mathbf{z} than otherwise, at the expense of the other constraints. If we want to, we can control this individual weighting of the constraints in \mathbf{A} by multiplying them with a suitable scalar. For most estimation problems, however, we want the data in the dataset to be treated with equal weights, although exactly how to implement this in the general case is not straightforward. For simple geometric objects like points it makes sense to apply the P-normalization described in Section 3.1.1, or the D-normalization described in Section 3.2.1 for a line, since the corresponding scalar products then can be interpreted in terms of a Euclidean distance. For more complex geometric object the individual normalization has to be determined in some other way.

13.9 Use P-normalization on points, or D-normalization on lines, before they are applied to the calculations for estimating geometric objects.

This individual normalization is often not sufficient for obtaining reasonable estimates. For example, the potential degeneracy discussed in Section 13.2.1 emerges from data represented in P-normalized homogeneous coordinates. As is demonstrated in this section, the estimate from the homogeneous method is not independent to coordinate transformations, it matters which coordinate system is being used. In particular, it matters for the homogeneous method when applied to the non-minimal case when the data is perturbed by noise. Consider, again, the problem of fitting a line \mathbf{l} to a set of 2D points, by minimizing the algebraic error $\|\mathbf{Y}^\top \mathbf{l}\|$ where \mathbf{Y} holds the homogeneous coordinates of the points in its columns. The homogeneous method is based on an SVD of \mathbf{Y}^\top :

$$\mathbf{Y}^\top = \mathbf{U} \mathbf{S} \mathbf{V}^\top. \quad (13.18)$$

Given that the singular values in the diagonal of \mathbf{S} are sorted in descending order, the optimal estimate of \mathbf{l} is the rightmost column of \mathbf{V} .

Consider what happens if we apply some transformation \mathbf{T} on the homogeneous coordinates of the points: $\mathbf{y}'_k = \mathbf{T}\mathbf{y}_k$, where \mathbf{y}'_k are the transformed points. We interpret this as a transformation of the coordinate system, and \mathbf{T} can be a rigid, affine, or even a homography transformation. The transformed data matrix is

$$\mathbf{Y}' = \mathbf{T}\mathbf{Y}, \quad (13.19)$$

with a corresponding SVD given as

$$\mathbf{Y}'^\top = \mathbf{U}'\mathbf{S}'\mathbf{V}'^\top. \quad (13.20)$$

For this transformed dataset, the optimal estimate \mathbf{l}' is the rightmost column of \mathbf{V}' . The question is: are the two estimates \mathbf{l} and \mathbf{l}' representing the same line but in two different coordinate systems, related by \mathbf{T} ? In other words: is $\mathbf{l}' \sim \mathbf{T}^{-\top}\mathbf{l}$, where $\mathbf{T}^{-\top}$ is the dual transformation relative to \mathbf{T} ? In general, the answer is *no*.

To see this, combine Equation (13.19) with Equation (13.18) and Equation (13.20) to get

$$\mathbf{U}\mathbf{S}(\mathbf{T}\mathbf{V})^\top = \mathbf{U}'\mathbf{S}'\mathbf{V}'^\top. \quad (13.21)$$

We assume that $\mathbf{U}, \mathbf{U}' \in O(m)$, $\mathbf{V}, \mathbf{V}' \in O(3)$ and both \mathbf{S} and \mathbf{S}' are diagonal. For a general \mathbf{T} , this means that we cannot write $\mathbf{V}' = \mathbf{T}\mathbf{V}$. Instead, the application of a general \mathbf{T} on \mathbf{V} in Equation (13.21) affects all of \mathbf{U}', \mathbf{S}' and \mathbf{V}' in a non-trivial way. The singular values in \mathbf{S} , i.e., the SVD profile, as well as the corresponding right singular vectors in \mathbf{V} change in a way that cannot be formalized in a simple way. In short, the estimated geometric object is not independent to coordinate transformations of the dataset:

13.10 Transformations of the coordinate system of the data may have a significant effect of the performance of the estimation. Both the SVD profile and the estimated model can change when the data is transformed to a different coordinate system.

An example (I)

We now consider a first example that illustrates the normalization issue. To do this, we return to the problem of the degenerate SVD profiles discussed in Section 13.2.1. The issue is that instead of one small singular value that specifies a well-defined estimate, we get multiple singular values which can be classified as “small” corresponding to multiple estimates that appear as equally good. Consider the specific example of fitting a line to a set of 2D points, where the estimated line \mathbf{l} is given as the right singular vector corresponding to the smallest singular value of \mathbf{Y}^\top , where the latter holds the homogeneous coordinates of the points in its columns. In accordance with Toolbox Section 8.2, this \mathbf{l} is also an eigenvector of $\mathbf{Y}\mathbf{Y}^\top$, corresponding to the smallest eigenvalue. We get

$$\mathbf{Y}\mathbf{Y}^\top = \sum_{k=1}^m \mathbf{y}_k \mathbf{y}_k^\top. \quad (13.22)$$

Let us take a closer look at the matrix \mathbf{Y} :

$$\mathbf{Y} = \begin{pmatrix} u_1 & u_2 & \dots & u_m \\ v_1 & v_2 & \dots & v_m \\ 1 & 1 & \dots & 1 \end{pmatrix}. \quad (13.23)$$

where (u_k, v_k) , are the Cartesian coordinates of points $k = 1, \dots, m$. Inserted into Equation (13.22) this leads to

$$\mathbf{Y}\mathbf{Y}^\top = \sum_{k=1}^m \begin{pmatrix} u_k^2 & u_k v_k & u_k \\ u_k v_k & v_k^2 & v_k \\ u_k & v_k & 1 \end{pmatrix} = m \begin{pmatrix} s_{11} + s_1^2 & s_{12} + s_1 s_2 & s_1 \\ s_{12} + s_1 s_2 & s_{22} + s_2^2 & s_2 \\ s_1 & s_2 & 1 \end{pmatrix}, \quad (13.24)$$

where $\bar{s} = (s_1, s_2)$ is the centroid of the m points, and s_{11}, s_{22}, s_{12} are the variances and cross-correlation defined in Equation (12.8).

We now make the assumption that the coordinates of the points have been defined relative to a coordinate system in which the centroid is $\bar{\mathbf{s}} = (0, 0)$. This gives

$$\mathbf{Y}\mathbf{Y}^\top = m \begin{pmatrix} s_{11} & s_{12} & 0 \\ s_{12} & s_{22} & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (13.25)$$

The vector $\mathbf{l} \in \mathbb{R}^3$, holding the dual homogeneous coordinates of the line, is an eigenvector corresponding to the smallest eigenvalue of $\mathbf{Y}\mathbf{Y}^\top$, so we should try to characterize the eigensystem of this matrix. First, it has an eigenvector $\mathbf{e}_0 = (0, 0, 1)$ with eigenvalue m , representing the line at infinity. In most practical cases, this is not the correct solution to our estimation problem. Second, since $\mathbf{Y}\mathbf{Y}^\top$ is a symmetric matrix that, in general, has orthogonal eigenvectors, there are two more eigenvectors of the type

$$\mathbf{e}_1 = \begin{pmatrix} \hat{\mathbf{l}}_1 \\ 0 \end{pmatrix}, \quad \mathbf{e}_2 = \begin{pmatrix} \hat{\mathbf{l}}_2 \\ 0 \end{pmatrix}, \quad (13.26)$$

where $\hat{\mathbf{l}}_1, \hat{\mathbf{l}}_2$ is an ON-basis of \mathbb{R}^2 . More precisely, they are the two eigenvectors of the 2×2 matrix

$$\mathbf{S} = \begin{pmatrix} s_{11} & s_{12} \\ s_{12} & s_{22} \end{pmatrix}. \quad (13.27)$$

A closer look at \mathbf{S} shows that it is identical to $\mathbf{A}\mathbf{A}^\top$ that appears in Equation (12.19), where we minimized a *geometric error* between the points and the line based on total least squares. In fact, in Section 12.2 we show that the solution to that estimation problem is in the form of a vector $\hat{\mathbf{l}}$ corresponding to the smallest eigenvalue of $\mathbf{S} = \mathbf{A}\mathbf{A}^\top$. This vector is a normal to the estimated line, and its distance to the origin is given as $\Delta = \bar{\mathbf{s}} \cdot \hat{\mathbf{k}}$. In our case, $\bar{\mathbf{s}} = \mathbf{0}$, and it follows that the estimated line must pass through the origin.

Consequently, minimization of the algebraic error $\|\mathbf{Y}^\top \mathbf{l}\|$ over $\|\mathbf{l}\| = 1$ becomes equivalent to minimization of the geometric error ε_{TOT} in Equation (12.16), based on the assumption $\bar{\mathbf{s}} = \mathbf{0}$. Since geometric errors are the real deal in estimation problems in geometry, we get geometric optimization “for free” if we make sure that the points are expressed in a coordinate system that has the centroid $\bar{\mathbf{s}}$ at the origin, even if we are using a traditional algebraic approach to the estimation problem. In general, the centroid does not lie at the origin, but we can apply a translation on the coordinates that moves the centroid to the origin, estimate the line in this translated coordinate system using the homogeneous method, and then transform the line back to the original coordinate system by means of the inverse translation. The result is the same line produced by the total least squares method in Section 12.2.

Well, there is one remaining detail. The solution $\hat{\mathbf{l}}$ that appears in Section 12.2 is the eigenvector corresponding to the smallest eigenvalue of \mathbf{S} , here denoted λ . The eigenvalues of $\mathbf{Y}\mathbf{Y}^\top$ includes m and λm , where the former corresponds to \mathbf{e}_0 , the line at infinity, and the latter corresponds to the desired solution of the algebraic minimization problem. This requires $\lambda < 1$, a condition that is not automatically satisfied. If not satisfied, the eigenvector corresponding to the smallest eigenvalue instead turns out to be the line at infinity.

To see why this requirement may not be satisfied, we need to determine the magnitude of λ . We assume that each image coordinate is perturbed by noise in accordance with

$$(u_k, v_k) = (\dot{u}_k, \dot{v}_k) + (\eta_{uk}, \eta_{vk}), \quad (13.28)$$

where (η_{uk}, η_{vk}) is an independent zero mean noise of variance σ^2 , and (\dot{u}_k, \dot{v}_k) are the unperturbed coordinates that can be fitted exactly to a line. We plug these coordinates into the expressions for the variances and cross-correlation and get

$$\mathbf{S} = \frac{1}{m} \sum_{k=1}^m \begin{pmatrix} \dot{u}_k^2 - 2\dot{u}_k\eta_{uk} + \eta_{uk}^2 & \dot{u}_k\dot{v}_k - \dot{u}_k\eta_{vk} - \dot{v}_k\eta_{uk} + \eta_{uk}\eta_{vk} \\ \dot{u}_k\dot{v}_k - \dot{u}_k\eta_{vk} - \dot{v}_k\eta_{uk} + \eta_{uk}\eta_{vk} & \dot{v}_k^2 - 2\dot{v}_k\eta_{vk} + \eta_{vk}^2 \end{pmatrix}. \quad (13.29)$$

This is the expression for \mathbf{S} given a fixed set of points and a specific instance of the observed noise. Assuming the points are fixed, and using the stated properties of the noise, the expectation value of \mathbf{S} can be expressed as

$$\mathbb{E}[\mathbf{S}] = \frac{1}{m} \sum_{k=1}^m \begin{pmatrix} \dot{u}_k^2 + \eta_{uk}^2 & \dot{u}_k\dot{v}_k \\ \dot{u}_k\dot{v}_k & \dot{v}_k^2 + \eta_{vk}^2 \end{pmatrix} = \begin{pmatrix} \dot{s}_{11} & \dot{s}_{12} \\ \dot{s}_{12} & \dot{s}_{22} \end{pmatrix} + \sigma^2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \dot{\mathbf{S}} + \sigma^2 \mathbf{I} \quad (13.30)$$

where $\hat{\mathbf{S}}$ is the ideal, unperturbed, version of \mathbf{S} where one eigenvalue is zero. Adding the second term in the previous equation, however, increases the smallest eigenvalue of $E[\mathbf{S}]$ to σ^2 . This does not mean that $E[\lambda] = \sigma^2$, but it indicates that we should expect to see an increase in λ due to the noise that is in the order of σ^2 .

From this discussion we conclude that we need to assure that $\sigma^2 \ll 1$. This can be done by suitable scaling operation *after* the translation that moves the centroid to the origin has been made. If we apply a uniform scaling by s , changing (\dot{u}_k, \dot{v}_k) to $(s \dot{u}_k, s \dot{v}_k)$, then also the noise scales in the same way: (η_{uk}, η_{vk}) transforms to $(s \eta_{uk}, s \eta_{vk})$ and σ transforms to $s\sigma$. If we have an estimate of σ , we should apply a scaling by s such that $s\sigma \ll 1$. The practical implementation of this scaling operation may be done in various ways, the important point made here is that it may be necessary to apply the scaling in order to assure that we get the desired solution.

We summarize this example. A 2D line \mathbf{l} is estimated from a set of points by minimizing an algebraic error using the homogeneous method. By first applying a translation of the coordinate system to move the centroid of the points to the origin, the resulting line is equal to the line that is produced by minimizing a geometric error based on total least squares. In order to assure that the homogeneous method gives the correct solution, however, we may need to also scale the coordinate system such that the noise variance $\sigma \ll 1$. In practice this means that we should apply a translation and a uniform scaling of the coordinate system *before* the line is estimated, then estimate the line based on the homogeneous method, and finally apply the inverse coordinate transformation, or *re-normalize*, in order to describe the line relative to the original coordinate system.

We formulate the following observation:

13.11 Translating the centroid of the dataset to the origin is a good idea when the homogeneous method is used for algebraic minimization. It makes algebraic errors more similar to geometric errors. A suitable uniform scaling after the translation is also a good idea, otherwise the result may turn out to be completely wrong.

At this point we have only demonstrated these statements with one simple example. As it turns out, however, a *normalizing transformation* in terms of a translation that moves the centroid to the origin followed by a suitable scaling is of general validity. This is further illustrated by the next example.

An example (II)

We now return to the problem of estimating a homography from a set of corresponding points, based on the algebraic minimization presented in Section 13.1.2 in combination with the homogeneous method. This method uses a data matrix \mathbf{A} consisting of blocks of rows \mathbf{A}_k , for example the 2×9 blocks defined in Equation (13.13).

In practice, the image coordinates are perturbed by noise, but we start to consider the ideal case, when a homography can be fitted exactly to the data. We consider a very simple case when the homography is $\mathbf{H} = \mathbf{I}$, the identity mapping and that the two sets of points each are defined in terms of equidistant points on a circle. More precisely, we assume that the points in the two images are given as

$$\begin{pmatrix} u_k \\ v_k \end{pmatrix} = \begin{pmatrix} R \cos \alpha_k \\ R \sin \alpha_k \end{pmatrix}, \quad \begin{pmatrix} u'_k \\ v'_k \end{pmatrix} = \begin{pmatrix} R \cos \alpha_k \\ R \sin \alpha_k \end{pmatrix}, \quad \alpha_k = \frac{2\pi k}{m}, \quad (13.31)$$

where R is the radius of the circles in the two images. Notice that both circles are centered at the origin, i.e., the centroids of each point-set is located at the origin. This is not a necessary requirement for the following results, but simplifies the derivation considerably. The 2×9 matrix \mathbf{A}_k in Equation (13.13) now becomes

$$\mathbf{A}_k = \begin{pmatrix} R \cos \alpha_k & 0 & -R^2 \cos^2 \alpha_k & R \sin \alpha_k & 0 & -R^2 \cos \alpha_k & 1 & 0 & -R \cos \alpha_k \sin \alpha_k \\ 0 & R \cos \alpha_k & -R^2 \cos \alpha_k \sin \alpha_k & 0 & R \sin \alpha_k & -R^2 \sin^2 \alpha_k & 0 & 1 & -R \sin \alpha_k \end{pmatrix}. \quad (13.32)$$

The estimated homography is a right singular vector corresponding to the smallest singular value of the data matrix \mathbf{A} in Equation (13.14). In this ideal example this singular value is equal to zero. This singular vector can also be obtained as an eigenvector of the symmetric matrix

$$\mathbf{M} = \frac{1}{m} \mathbf{A}^\top \mathbf{A} = \frac{1}{m} \sum_{k=1}^m \mathbf{A}_k^\top \mathbf{A}_k. \quad (13.33)$$

Inserting the expression for \mathbf{A}_k in Equation (13.32) and computing the sum gives a matrix that can be written as the block diagonal matrix \mathbf{M}_{perm} after a suitable symmetric permutation of its rows and columns:

$$\mathbf{M}_{\text{perm}} = \begin{pmatrix} \frac{R^2}{2} & 0 & -\frac{R^2}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{R^2}{2} & -\frac{R^2}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{R^2}{2} & 0 & R^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{R^4}{2} & -\frac{R^2}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{R^2}{2} & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{R^4}{2} & -\frac{R^2}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{R^2}{2} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{R^2}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{R^2}{2} \end{pmatrix}, \quad (13.34)$$

where \mathbf{M}_{perm} and \mathbf{M} have the same eigenvalues. The block diagonal form of \mathbf{M}_{perm} makes it straightforward to determine its eigenvalues:

$$\lambda \in \left\{ 0, \underbrace{\frac{R^2}{2}}_{\times 3}, \underbrace{\frac{3R^2}{2}}_{\times 2}, \underbrace{\frac{2+R^4-\sqrt{4+R^8}}{4}}_{\times 2}, \underbrace{\frac{2+R^4+\sqrt{4+R^8}}{4}}_{\times 2} \right\}. \quad (13.35)$$

The first eigenvalue in this sequence, 0, is the smallest one and corresponds to the correct estimate. But what about the other eigenvalues? How do they vary as a function of R ?

Consider first the case of $R \ll 1$. In this case:

$$\sqrt{4+R^8} = 2\sqrt{1+\frac{1}{4}R^8} \approx 2(1+\frac{1}{8}R^8) = 2 + \frac{1}{4}R^8. \quad (13.36)$$

Using this approximation, the eigenvalues of \mathbf{M} becomes

$$\lambda \in \left\{ 0, \underbrace{\frac{R^2}{2}}_{\times 3}, \underbrace{\frac{3R^2}{2}}_{\times 2}, \underbrace{\frac{R^4}{4}-\frac{R^8}{16}}_{\times 2}, \underbrace{1+\frac{R^4}{4}+\frac{R^8}{16}}_{\times 2} \right\} \approx \left\{ 0, \underbrace{\frac{R^2}{2}}_{\times 3}, \underbrace{\frac{3R^2}{2}}_{\times 2}, \underbrace{\frac{R^4}{4}}_{\times 2}, \underbrace{\frac{1}{2}}_{\times 2} \right\}. \quad (13.37)$$

Consequently, the smallest non-zero eigenvalue is $\approx R^4/4$, which has a multiplicity of 2.

Consider instead the case of $R \gg 1$. In this case:

$$\sqrt{4+R^8} = R^4\sqrt{1+\frac{4}{R^8}} \approx R^4(1+\frac{2}{R^8}) = R^4 + \frac{2}{R^4}. \quad (13.38)$$

Using this approximation, the eigenvalues of \mathbf{M} become

$$\lambda \in \left\{ 0, \underbrace{\frac{R^2}{2}}_{\times 3}, \underbrace{\frac{3R^2}{2}}_{\times 2}, \underbrace{\frac{1}{2}-\frac{1}{2R^4}}_{\times 2}, \underbrace{\frac{1}{2}+\frac{R^4}{2}+\frac{1}{2R^4}}_{\times 2} \right\} \approx \left\{ 0, \underbrace{\frac{R^2}{2}}_{\times 3}, \underbrace{\frac{3R^2}{2}}_{\times 2}, \underbrace{\frac{1}{2}}_{\times 2}, \underbrace{\frac{R^4}{2}}_{\times 2} \right\}. \quad (13.39)$$

Consequently, the smallest non-zero eigenvalue is $\approx 1/2$, which has a multiplicity of 2.

Finally, consider the case $R = 1$, leading to:

$$\lambda \in \left\{ 0, \underbrace{\frac{1}{2}}_{\times 3}, \underbrace{\frac{3}{2}}_{\times 2}, \underbrace{\frac{3-\sqrt{5}}{4}}_{\times 2}, \underbrace{\frac{3+\sqrt{5}}{4}}_{\times 2} \right\} \approx \left\{ 0, \underbrace{0.5}_{\times 3}, 1.5, \underbrace{0.2}_{\times 2}, \underbrace{1.3}_{\times 2} \right\}. \quad (13.40)$$

Let us stop here for a moment and consider the implications of this result. First, the first two cases that the eigenvalues of \mathbf{M} come in four categories: there is a single eigenvalue equal to zero, there are four eigenvalues that depend on R^2 , two depend on R^4 , and two that assume a value in the range $0.5 - 1$. This is consistent with the final case, where one eigenvalue equals zero and the other lie in the range $0.2 - 1.5$. This means that the SVD

profile of \mathbf{A} , where the singular values of \mathbf{A} are the square roots of the eigenvalues of \mathbf{M} , come in four categories: one is zero, four are approximately linear in R , two depend on R^2 , and two are close to one. Consequently, the smallest non-zero singular value of \mathbf{A} is in the order of 1, unless $R \ll 1$ in which case it is in the order of R^2 . To see why this is an important observation, we need to understand how much \mathbf{M} is perturbed by noise in the image coordinates.

In any practical case, the image coordinates (u_k, v_k) and (u'_k, v'_k) are perturbed by noise:

$$\begin{pmatrix} u_k \\ v_k \end{pmatrix} = \begin{pmatrix} R \cos \alpha_k + \eta_{uk} \\ R \sin \alpha_k + \eta_{vk} \end{pmatrix}, \quad \begin{pmatrix} u'_k \\ v'_k \end{pmatrix} = \begin{pmatrix} R \cos \alpha_k + \eta'_{uk} \\ R \sin \alpha_k + \eta'_{vk} \end{pmatrix}, \quad \alpha_k = \frac{2\pi k}{m}, \quad (13.41)$$

where (η_{uk}, η_{vk}) and (η'_{uk}, η'_{vk}) are the noise perturbation of image coordinates (u_k, v_k) and (u'_k, v'_k) , respectively. The perturbed version of \mathbf{A}_k in Equation (13.32), denoted \mathbf{A}'_k , becomes

$$\mathbf{A}'_k = \mathbf{A}_k + \mathbf{D}_k, \quad k = 1, \dots, m, \quad (13.42)$$

where the perturbation term \mathbf{D}_k is given as

$$\mathbf{D}_k = \begin{pmatrix} \eta_{uk} & 0 & -\eta_{uk}\eta'_{uk} - u'_k\eta_{uk} - u_k\eta'_{uk} & \eta_{vk} & 0 & -\eta_{vk}\eta'_{uk} - u'_k\eta_{vk} - v_k\eta'_{uk} & 0 & 0 & -\eta'_{uk} \\ \eta_{uk} & 0 & -\eta_{uk}\eta'_{uk} - u'_k\eta_{uk} - u_k\eta'_{uk} & \eta_{vk} & 0 & -\eta_{vk}\eta'_{uk} - u'_k\eta_{vk} - v_k\eta'_{uk} & 0 & 0 & -\eta'_{uk} \end{pmatrix} \quad (13.43)$$

Notice that the noise terms appear with different scalings in this matrix. Column 1, 2, 4, 5 and 9 contain the noise with unit scale, while columns 3 and 6 hold noise that is scaled by the coordinates, and columns 7 and 8 have no noise at all. The perturbed version of \mathbf{M} can be written

$$\mathbf{M}' = \mathbf{M} + \underbrace{\frac{1}{m} \sum_{k=1}^m (\mathbf{A}_k^\top \mathbf{D}_k + \mathbf{D}_k^\top \mathbf{A}_k + \mathbf{D}_k^\top \mathbf{D}_k)}_{:= \Delta \mathbf{M}} = \mathbf{M} + \Delta \mathbf{M} \quad (13.44)$$

where $\Delta \mathbf{M}$ is the perturbation term of \mathbf{M} . The magnitudes of the elements of $\Delta \mathbf{M}$ is even more complex than for \mathbf{D}_k since the former matrix contains both \mathbf{D}_k multiplied with itself and multiplied with \mathbf{A}_k , where the latter contains both constants, coordinates, and products of coordinates.

To simplify the analysis, we set

$$\eta_{uk} = \eta_{vk} = \eta'_{uk} = \eta'_{vk} = \eta, \quad \text{for } k = 1, \dots, m, \quad (13.45)$$

where η represents a ‘‘magnitude’’ of the noise. For example, η can specify the standard deviation, or accuracy, or the noise, with the additional assumption that $\eta \ll R$. We want to know the order of magnitude of the elements of $\Delta \mathbf{M}$, which can be obtained by simply replacing the coordinates by R since this is their maximal magnitude. As a result, the elements of $\Delta \mathbf{M}$ can be described as linear combinations of the following factors:

$$\eta, \eta R, \eta R^2, \eta^2, \eta^2 R, \eta^2 R^2, \eta^3, \eta^3 R, \eta^4 \quad (13.46)$$

Let us again consider the three different cases of R , and see how they affect the magnitude of $\Delta \mathbf{M}$. If $R \ll 1$ then the factors in Equation (13.46) is dominated by η . If $R \gg 1$, it is instead ηR^2 that dominates. Finally, if $R = 1$ the dominating term is η .

At this point, we turn our attention to $\hat{\mathbf{e}}_9$, a normalized eigenvector corresponding to eigenvalue $\lambda_9 = 0$ in \mathbf{M} . Since this eigenvector is derived from an ideal and noise-free dataset it represents the correct estimate of \mathbf{H} . Due to the perturbation $\Delta \mathbf{M}$ in \mathbf{M} , however, there is a corresponding perturbation also in $\hat{\mathbf{e}}_9$, and we want to quantify it. This can be done by applying the result derived in Toolbox Section 8.4.2.2 to the current discussion, which implies that

$$\hat{\mathbf{e}}'_9 = \hat{\mathbf{e}}_9 - \underbrace{\sum_{k=1}^8 \frac{\hat{\mathbf{e}}_k^\top \Delta \mathbf{M} \hat{\mathbf{e}}_9}{\lambda_l} \hat{\mathbf{e}}_k}_{:= \Delta \hat{\mathbf{e}}_9} + O(\|\Delta \mathbf{M}\|^2). \quad (13.47)$$

where $\hat{\mathbf{e}}'_9$ is the eigenvector corresponding to the smallest eigenvalue of \mathbf{M}' . The perturbation from $\hat{\mathbf{e}}_9$ to $\hat{\mathbf{e}}'_9$ is described mainly by the second term $\Delta \hat{\mathbf{e}}_9$ in the right-hand side of Equation (13.47), and we want to know how large it can be.

Using the observations already made, we get the following result:

- If $R \ll 1$, then $\Delta\mathbf{M}$ has a magnitude in the order of η , and the smallest non-zero eigenvalue λ_l is in the order of R^4 . $\Delta\hat{\mathbf{e}}_9$ is in the order of ηR^{-4} .
- If $R \gg 1$, then $\Delta\mathbf{M}$ has a magnitude in the order of ηR^2 , and the smallest non-zero eigenvalue λ_l is in the order of 1. $\Delta\hat{\mathbf{e}}_9$ is in the order of ηR^2 .
- If $R = 1$, then $\Delta\mathbf{M}$ has a magnitude in the order of η , and the smallest non-zero eigenvalue λ_l is in the order of 1. $\Delta\hat{\mathbf{e}}_9$ is in the order of η .

Let us now look at what this result means. First, assume that $R = 1$ and that η corresponds to the inaccuracy for this size of the circle. The perturbation in the eigenvector is in the order of η . Second, apply a uniform scaling of the coordinates system that changes the radius to $R = s$. This scaling also affects the noise, it becomes $s\eta$. If s is small, $s \ll 1$, the perturbation of $\hat{\mathbf{e}}_9$ has a magnitude that is of order $\eta/s^3 \gg \eta$. On the other hand, if s is large, $s \gg 1$, the same perturbation has a magnitude in the order $\eta s^3 \gg \eta$. Consequently, regardless of whether we reduce or increase the scale, we obtain larger perturbation than if $R = 1$. This is the final result of this discussion:

13.12 It is a good idea to scale image coordinates such that they lie at approximately unit distance to the centroid. This makes the perturbation due to noise close to minimal.

Here, we have only provided a single simplified example that supports this statement, but there exist studies which show a positive influence on the quality of the resulting estimate from this type of normalizing transformation. This observation is generally true, and does not only apply for estimation of lines or homographies. In fact, normalizing transformation are sometime critical for obtaining estimates that are useful. To cite Hartley & Zisserman:

13.13 “Data normalization [...] must not be considered optional” [30].

A support for this strong statement is given by the fact that the perturbation term $\Delta\hat{\mathbf{e}}_9$ in our last example *grows* approximately as s^3 for large s and as s^{-3} for small s , where s is the radius of the circles.

The last example does not clearly explain why the centroid should be moved to the origin, other than if this is done then certain conclusions can be drawn about the scaling. It can be demonstrated that this type of translation has a similar effect as the scaling. Moving the centroid far from the origin makes the eigenvalues of \mathbf{M} to spread out such that the smallest one decreases reciprocal to the square of the distance, while keeping the noise at the same level. Consequently, a *normalizing transformation* should first translate the centroid to the origin, then scale the points to have an approximate distance to the centroid close to one. In the literature this type of transformation is sometimes also referred to as a *equilibrating transformation* or *pre-conditioning* of the data. The model is then estimated relative to this transformed coordinate system, and needs to be re-normalized to the original coordinate system in the end.

This example also leaves to explain what should be done in the case that the estimated homography is not close to \mathbf{I} , or what happens when the points in the two images are not lying equidistant on a circle, or if something else than a homography is estimated. The second issue can be addressed intuitively by observing that it should be sufficient that the points lie approximately at unit distance, or even approximately and on average at unit distance from the origin. The exact implementation of the scaling part of the normalizing transformation can be done in various ways. In the simplest case by means of a uniform scaling, but more advanced approaches apply a non-uniform scaling. A variant that is common in the literature, described by Hartley, is based on a uniform scaling.

Hartley normalization

In a seminal study of normalizing transformation for estimation of geometric objects [31], Hartley proposed a particular variant of the normalizing transformation. It has the same translation as already been discussed: move the centroid to the origin, and the scaling is uniform and chosen such that the average distance from the points to the centroid becomes $s = \sqrt{2}$. The specific factor of $\sqrt{2}$ is perhaps difficult to motivate, other than from experiments on realistic estimation problems, but at least it falls well into the general category of normalizing transformations that are outlined in this section.

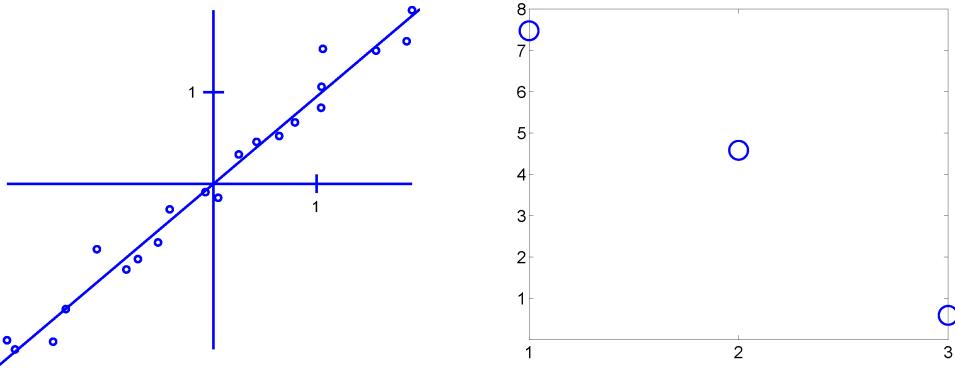


Figure 13.6: Left: the same point-set as in Figure 13.2 after a H-normalization. This figure also shows the line that is estimated by algebraic minimization based on the homogeneous method. Right: the singular values of the corresponding data matrix.

In the following presentation, we refer to this specific normalizing transformation as *Hartley normalization*, or *H-normalization* for short, and it results in H-normalized coordinates. This specific normalization for 2D points is presented in Algorithm 13.1. In the case of 3D coordinates, the same algorithm can be used, extending with a third coordinate but using the same translation and scaling.

13.2.3 SVD profile, again

At this point we return to the estimation problems that were discussed in the beginning of Section 13.2. It was said then that the estimate can be corrupted when the SVD profile shows several small singular values. The previous discussion suggest that this has something to do with the choice of coordinate system, and that a normalizing transformation should have a positive effect both on the SVD profile and on the quality of the resulting estimate. Let us see if this is correct for the two estimation problems, for example by applying H-normalization on the dataset before the estimation is made.

In Figure 13.2 we saw a set of points, left, from which we estimate a line, and their corresponding SVD profile, right. If H-normalization we get instead the point-set shown in Figure 13.6, left. Notice that the two sets are identical except for a translation and a uniform scaling. To the right we also see the SVD profile after the normalizing transformation. If we compare this profile to the original one, in Figure 13.2, right, we can make the observation that the normalized dataset has a better separation between “small” and “large” singular values. The

Algorithm 13.1: Normalization of 2D points using Hartley’s approach.

Input: A set of m 2D points with Cartesian coordinates (u_k, v_k) .

Output: The same points represented as H-normalized coordinates as $(\tilde{u}_k, \tilde{v}_k)$.

Output: The 3×3 transformation matrix \mathbf{T} that represents the normalizing transformation on homogeneous coordinates.

- 1 Compute centroid $(s_1, s_2) = \text{mean of } (u_k, v_k) \text{ over all } k = 1, \dots, m$.
- 2 Subtract the centroid from all points: $(u'_k, v'_k) = (u_k - s_1, v_k - s_2)$ for all $k = 1, \dots, m$.
- 3 Compute mean distance to centroid: $d = \text{mean of } \| (u'_k, v'_k) \| \text{ over all } k = 1, \dots, m$.
- 4 Scale to make average distance equal to $\sqrt{2}$: $(\tilde{u}_k, \tilde{v}_k) = \sqrt{2}(u'_k, v'_k)/d$, for all $k = 1, \dots, m$.
- 5 // This gives the resulting H-normalized coordinates.
- 6 Set the transformation matrix \mathbf{T} , from (u_k, v_k) to $(\tilde{u}_k, \tilde{v}_k)$, as
- 7
$$\mathbf{T} = \begin{pmatrix} \sqrt{2}/d & 0 & -s_1 \\ 0 & \sqrt{2}/d & -s_2 \\ 0 & 0 & 1 \end{pmatrix}$$

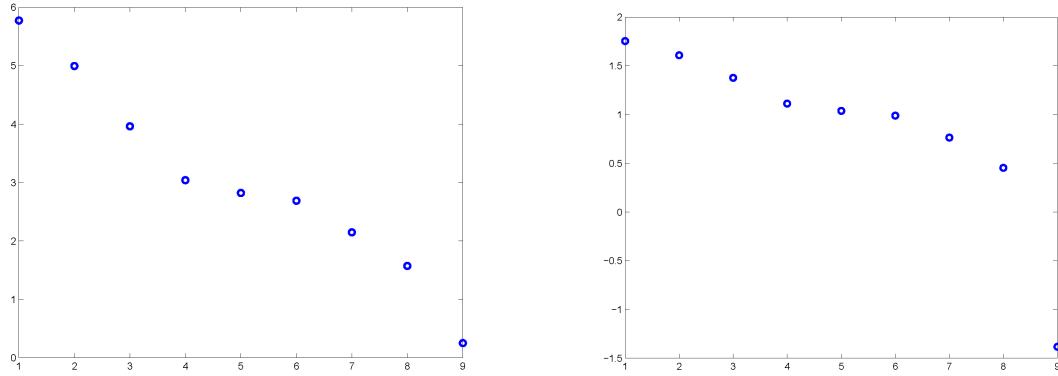


Figure 13.7: The SVD profile corresponding to the points in Figure 13.3 after H-normalization. To the right on a logarithmic scale.

ratios between the singular values of the normalized dataset is given as

$$\frac{\sigma_1}{\sigma_2} \approx 1.6, \quad \text{and} \quad \frac{\sigma_2}{\sigma_3} \approx 7.8, \quad (13.48)$$

which should be compared with the same fractions in Equation (13.17) for the original data. We also see that the algebraic error ϵ_{ALG} , represented by σ_3 , has increased after the normalizing transform. Since there is no absolute scale for algebraic errors, this increase does not mean that the estimated line is worse in this case, we are simply minimizing a different algebraic error this time. The estimated line is shown in Figure 13.6, left. Comparing this line to the one this is estimated from the original dataset in Figure 13.2, left, we see that they are almost identical, although represented in two different coordinate systems. Hence, for this particular example, the gain of applying H-normalization is mainly in a less ambiguous SVD profile than obtaining a better estimate.

This is not the situation when we apply H-normalization to the data presented in Figure 13.3, before a homography is estimated. In this case the normalization implies that each of the two subsets, before and after the homography transformation, are treated separately: each subset is H-normalized independent of the other. The normalized datasets produce a new data matrix with an SVD profile illustrated in Figure 13.7, left, and to the right on a logarithmic scale. We see here that the ambiguity of the SVD profile has decreased since there is now only one singular value that is much smaller than the other, instead of three.

The figure also shows that, when H-normalization is applied to the dataset, the homography corresponding to the smallest singular value σ_9 , denoted \mathbf{H}_9 , has approximately half the geometric error compared to \mathbf{H}_9 derived from the original data. This is illustrated in Figure 13.8 showing the same type plot as in Figure 13.5 but now for \mathbf{H}_9 derived from the normalized data. Finally, we see that for the original dataset it was the case that \mathbf{H}_8 , the homography corresponding to the second smallest singular value, in fact gave a smaller geometric error than \mathbf{H}_9 did. For the normalized data, however, \mathbf{H}_8 , has a geometric error that is much larger than for \mathbf{H}_9 , and does not represent a useful estimate for the normalized dataset.

Consequently, H-normalization or a similar normalizing transformation is a critical step when we are estimating a homography. Not only does it produce a less ambiguous SVD profile, in general it also reduces the geometric error of the singular vector corresponding to the smallest singular value compared to the other singular vectors. This observation is also relevant for many other estimation problems in geometry:

13.14 Normalizing transformations, such as the H-normalization, of the coordinate system of the data have a significant effect of the performance of the estimation. Both the SVD profile and the estimated model are in general significantly improved.

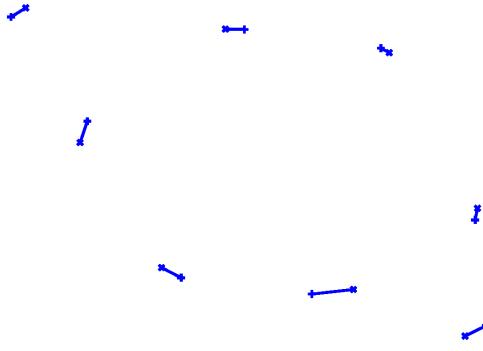


Figure 13.8: A homography is estimated from normalized data, and then applied to the same dataset. Pluses and crosses denote points in the same way as in Figure 13.5.

13.2.4 Closing remarks

The normalizing transformations that are one of the main results of this section can be motivated theoretically based on several approaches. Here, we have focused on the equilibration of the non-zero singular values of the data matrix, and the reduced perturbation of the singular vector corresponding to the smallest singular value. Another approach is to say that in the ideal case, without noise, the unperturbed data matrix \mathbf{A} is rank deficient such that it has a single right singular vector $\hat{\mathbf{z}}$ corresponding to singular value zero. The observed matrix \mathbf{A}' is perturbed by noise such that $\mathbf{A}' = \mathbf{A} + \mathbf{D}$, and from it we can estimate a model $\hat{\mathbf{z}}'$ that is given as the right singular vector corresponding to the smallest singular value of \mathbf{A}' . The question is whether or not

$$\mathbb{E}[\hat{\mathbf{z}}'] = \hat{\mathbf{z}} \quad (13.49)$$

where the expectation value in the left-hand side is taken over the noise that perturbs the data. This this case, we have an *unbiased estimate*, and means that the estimated vector $\hat{\mathbf{z}}'$ is correct, at least on average. In a study by Mühlbach & Mester [49], it is shown that normalizing transformations of the type described here represent a sufficient condition for an unbiased estimate of $\hat{\mathbf{z}}$.

Re-normalization

As a final remark, we remind that data normalization implies that the model is estimated relative to a different coordinate system than the original data set refers to. In general, we want the model to be described relative to the original coordinate system, and this can be done by a re-normalization. The exact form of the re-normalization, however, differs from one estimation problem to another.

For example, let $\{\mathbf{y}_k, k = 1, \dots, m\}$ be a set of 2D point from which we estimate a line. A coordinate transformation, such as suitable normalizing transformation, can be represented by a 3×3 transformation matrix \mathbf{T} :

$$\tilde{\mathbf{y}}_k \sim \mathbf{T} \mathbf{y}_k, \quad k = 1, \dots, m, \quad (13.50)$$

where $\tilde{\mathbf{y}}_k$ are the transformed homogeneous coordinates of the points. Based on the transformed dataset, we estimate a line with dual homogeneous coordinates $\tilde{\mathbf{l}} \in P(\mathbb{R}^3)$. To return to the original coordinate system we apply the inverse transformation given by \mathbf{T}^{-1} . But, since \mathbf{T} is defined relative homogeneous coordinates of points, and $\tilde{\mathbf{l}}$ holds the dual homogeneous coordinates of a line, we must apply the inverse of the *dual transformation*. In accordance with Section 4.5, the dual transformation of \mathbf{T} is $\mathbf{T}^{-\top}$, and its inverse is \mathbf{T}^\top . Consequently, the re-normalization should be implemented as

$$\mathbf{l} \sim \mathbf{T}^\top \tilde{\mathbf{l}}, \quad (13.51)$$

where \mathbf{l} is the estimated line described in the original coordinate system.

Algorithm 13.2: Basic algorithm for algebraic estimation of a homography

Input: Two sets of m corresponding points, $\{\mathbf{y}_k\}$ and $\{\mathbf{y}'_k\}$.
Output: An estimate of \mathbf{H} optimally satisfying $\mathbf{y}'_k \sim \mathbf{H}\mathbf{y}_k$.

- 1 Set $\mathbf{A} = \text{empty } 0 \times 9$ matrix
- 2 **foreach** $k = 1, \dots, m$ **do**
- 3 From \mathbf{y}_k and \mathbf{y}'_k compute the 2×9 matrix \mathbf{A}_k in Equation (13.13)
- 4 // Optionally: include also a third row in \mathbf{A}_k representing the third linear equation
- 5 Concatenate \mathbf{A}_k at the end of \mathbf{A} : $\mathbf{A} = [\mathbf{A}; \mathbf{A}_k]$
- 6 **end**
- 7 Find $\mathbf{z} \in \mathbb{R}^9$ that minimize the algebraic error $\|\mathbf{A}\mathbf{z}\|^2$
- 8 // Use either the homogeneous method, Algorithm 12.2 on page 212,
- 9 // or the inhomogeneous method, Algorithm 12.1 on page 210.
- 10 Reshape \mathbf{z} to 3×3 matrix \mathbf{H} using Equation (13.7) and Equation (13.13)

Algorithm 13.3: Algebraic estimation of \mathbf{H} using the homogeneous method with data normalization.

Input: Two sets of m corresponding points, $\{\mathbf{y}_k\}$ and $\{\mathbf{y}'_k\}$.
Output: An estimate of \mathbf{H} optimally satisfying $\mathbf{y}'_k \sim \mathbf{H}\mathbf{y}_k$

- 1 Compute normalized coordinates $\tilde{\mathbf{y}}_k$ and transformation \mathbf{T} from the set $\{\mathbf{y}_k\}$, using Algorithm 13.1 on page 235.
- 2 Compute normalized coordinates $\tilde{\mathbf{y}}'_k$ and transformation \mathbf{T}' from the set $\{\mathbf{y}'_k\}$, using Algorithm 13.1 on page 235.
- 3 Estimate $\tilde{\mathbf{H}}$ from $\{\tilde{\mathbf{y}}_k\}$ and $\{\tilde{\mathbf{y}}'_k\}$ using Algorithm 13.2 on page 238 with the homogeneous method.
- 4 Re-normalize: $\mathbf{H} = \mathbf{T}'^{-1} \tilde{\mathbf{H}} \mathbf{T}$

Similarly, let \mathbf{y}_k and \mathbf{y}'_k be two sets of corresponding points from which we estimate a homography. Each set is first subject to a normalizing transformation:

$$\tilde{\mathbf{y}}_k \sim \mathbf{T}\mathbf{y}_k, \quad \tilde{\mathbf{y}}'_k \sim \mathbf{T}'\mathbf{y}'_k, \quad k = 1, \dots, m, \quad (13.52)$$

where \mathbf{T} and \mathbf{T}' of the transformation matrix the first and second set, respectively. From the transformed data sets, a homography $\tilde{\mathbf{H}}$ is estimated but it refers to the transformed coordinate systems, and we want to bring it back to the original ones. In the case that the homography fits the data perfectly, we have

$$\tilde{\mathbf{y}}'_k \sim \tilde{\mathbf{H}}\tilde{\mathbf{y}}_k, \quad k = 1, \dots, m. \quad (13.53)$$

Replacing $\tilde{\mathbf{y}}_k$ and $\tilde{\mathbf{y}}'_k$ with the equivalent expressions in Equation (13.52) leads to

$$\mathbf{T}'\mathbf{y}'_k \sim \tilde{\mathbf{H}}\mathbf{T}\mathbf{y}_k, \quad \Rightarrow \quad \mathbf{y}'_k \sim \underbrace{\mathbf{T}'^{-1}\tilde{\mathbf{H}}\mathbf{T}}_{:=\mathbf{H}}\mathbf{y}_k, \quad k = 1, \dots, m. \quad (13.54)$$

This implies that the resulting estimate of the homography, denoted \mathbf{H} , connecting \mathbf{y}_k with \mathbf{y}'_k in the original coordinate systems, is given as

$$\mathbf{H} \sim \mathbf{T}'^{-1}\tilde{\mathbf{H}}\mathbf{T}. \quad (13.55)$$

This last expression describes the re-normalization from $\tilde{\mathbf{H}}$ to \mathbf{H} .

Algorithms for algebraic homography estimation

We have now reached a point where we can summarize the results derived so far for estimation of homographies based on algebraic errors. The basic approach is described in Algorithm 13.2. In the case that the homogeneous method is used, and based on the discussion in Section 13.2.2, it is strongly recommended that the estimation is done from a normalized dataset. This approach is presented in Algorithm 13.3.

13.3 Camera matrix estimation

The frameworks that is established above for estimation of a homography can be applied also to estimation of the camera matrix \mathbf{C} , relating 3D points with their projections into the camera image. For the same reasons already described in Section 13.1.1 for homographies, minimization of geometric error related to estimation of camera matrices lead to non-linear equations that only can be approached by means of iterative methods. As an alternative, we can also formulate algebraic errors for estimation of \mathbf{C} using DLT in much the same way as for \mathbf{H} .

Let $\{\mathbf{x}_k, k = 1, \dots, m\}$ be a set of m 3D points and $\{\mathbf{y}_k, k = 1, \dots, m\}$ be the projection of these points through a camera represented by the matrix \mathbf{C} :

$$\mathbf{y}_k \sim \mathbf{C} \mathbf{x}_k, \quad k = 1, \dots, m. \quad (13.56)$$

Given that we know only the two points sets³ we want to estimate \mathbf{C} , with the assumption that both sets are perturbed by noise so that Equation (13.56) is only approximately correct. Applying DLT to Equation (13.56) leads to

$$\mathbf{0} = [\mathbf{y}_k]_{\times} \mathbf{C} \mathbf{x}_k, \quad k = 1, \dots, m. \quad (13.57)$$

The camera matrix \mathbf{C} has elements given as

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \end{pmatrix}, \quad (13.58)$$

and Equation (13.57) describes 3 linear equations in the elements of \mathbf{C} . The equations can be reformulated as

$$\mathbf{A}_k \mathbf{z} = \mathbf{0}, \quad k = 1, \dots, m, \quad (13.59)$$

where \mathbf{A}_k is a 2×12 matrix given by

$$\mathbf{A}_k = \begin{pmatrix} 0 & -x_{1k} & x_{1k}y_{2k} & 0 & -x_{2k} & x_{2k}y_{2k} & 0 & -x_{3k} & x_{3k}y_{2k} & 0 & -1 & y_{2k} \\ x_{1k} & 0 & -x_{1k}y_{1k} & x_{2k} & 0 & -x_{2k}y_{1k} & x_{3k} & 0 & -x_{3k}y_{1k} & 1 & 0 & -y_{1k} \end{pmatrix}, \quad (13.60)$$

and \mathbf{z} is a *vectorization* of \mathbf{C} :

$$\mathbf{z}^T = (c_{11} \ c_{21} \ c_{31} \ c_{12} \ c_{22} \ c_{32} \ c_{13} \ c_{23} \ c_{33} \ c_{14} \ c_{24} \ c_{34}). \quad (13.61)$$

Here, we have chosen to consider only the first two of the three linear equations since they are linear dependent, as described in Section 13.1.2. The set of $2m$ equations defined in Equation (13.59) can be compiled in a compact form as

$$\mathbf{A} \mathbf{z} = \mathbf{0}, \quad \text{where} \quad \mathbf{A} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{pmatrix}. \quad (13.62)$$

Since the data is affected by measurement noise, we cannot expect to find a \mathbf{z} that solves Equation (13.62) exactly. Instead we define an algebraic error as

$$\epsilon_{ALGC} = \|\mathbf{A} \mathbf{z}\|, \quad (13.63)$$

and minimize ϵ_{ALGC} over \mathbf{z} , a problem that is discussed in Section 12.5. To do so, we need to introduce a constraint on the vector $\mathbf{z} \in \mathbb{R}^{12}$, leading either to the inhomogeneous or the homogeneous method. In the first case, the constraint is given by setting one of the elements in \mathbf{z} to 1, and leads to a linear inhomogeneous equation that needs to be solved. In the second case, the constraint is given by setting $\|\mathbf{z}\| = 1$ and leads to an SVD of the $2m \times 12$ matrix \mathbf{A} . In accordance with the discussion in Section 12.5, the homogeneous method is the preferred approach since it less prone to the possible degeneracy that appears from setting an element in \mathbf{C} equal to one that in fact should be close to zero, and also since the SVD profile of the homogeneous method provides information about possible data degeneracy. In the case of the homogeneous method, however, it is important to apply normalizing transformations to the data before the camera matrix is estimated, independently in the image and in 3D space, and to re-normalize the resulting estimate to the original coordinate systems.

³In fact, we also assume that we know the correspondences between the two sets.

Minimal case estimation

As seen in Section 8.3.3, the camera matrix \mathbf{C} has 11 degrees of freedom. Consequently, we need at minimum 11 equations to determine the free parameters of \mathbf{C} . In general, DLT provides 2 independent equations for each pair of corresponding points $(\mathbf{y}_k, \mathbf{x}_k)$. This means that we need at least 6 of these correspondences to obtain the necessary 11 equations. In fact, this gives 12 equations, which is one too many for the minimal case. In the ideal noise-free case, these 12 equations are linearly dependent and one of them can be discarded without loss of information.

Closing remarks

While estimation of homographies, described earlier in this chapter, is a relatively common practice in geometry, estimation of \mathbf{C} is not. This is a consequence of the fact that we seldom have access to a dataset in the form of corresponding 3D and 2D points, where \mathbf{C} is unknown. An exception is PnP, presented in Section 15.4. In addition, in many applications \mathbf{C} is variable over time as the camera changes its position and orientation (its external parameters) over time. In many practical applications, however, we are interested in determining the internal parameters of a camera, which often are fixed. The process of determining the internal camera parameters is referred to as *camera calibration*, and is described in more detail in Chapter 18.

Chapter 14

Non-linear Estimation

Before you read this chapter, you should have a look at [54] Chapter 9 which presents common techniques for non-linear optimization.

Chapters 12 and 13 have presented a collection of estimation problems that are relatively simple to solve, based on minimizing *algebraic errors*. These estimation problems have in common that the corresponding error function $\epsilon(\mathbf{z})$ is a quadratic function in the free parameters \mathbf{z} . Once a suitable constraint has been established for \mathbf{z} , ϵ has in general a unique minimum and the corresponding minimizer can be determined by solving a linear equation in \mathbf{z} . There are simple and effective methods for implementing the corresponding numerical computations and they produce the expected result at a limited computational cost.

On the other hand, many relevant estimation problems are instead formulated using *geometric errors*. They are often based on Euclidean distances that should be minimized for the estimated model, and can be motivated by providing a *maximum likelihood estimate* of the model parameters \mathbf{z} . If we are interested in finding minimizers of geometric errors, the minimizers of algebraic errors can work as reasonable approximations, but to obtain solutions of high numerical accuracy we need to apply methods dedicated to geometric errors. With a few exceptions, such methods have to use non-linear optimization techniques, such as those described in [54] Chapter 9.

We have already touched upon specific examples of estimation problems that cannot be solved by linear methods. In Section 13.1.1 a homography is estimated based on minimizing the geometric error ϵ_{HOM} in Equation (13.2). Since the minimum in this case is characterized as the solution of a non-linear equation, Equation (13.5), this case had to be postponed since we did not have the necessary tools for finding the minimizer of ϵ_{HOM} . In this chapter, we look at some basic approaches for dealing with this and other non-linear estimation problems in the context of geometry.

For many of these estimation problems discussed here, there exist software that can implement the necessary computations. This software can sometimes be used more or less out of the box, but more often it needs some tuning of parameters or choosing the type of optimization method to be used. Making good choices in these matters may have a significant effect on the quality of the estimated model and, therefore, has to be done with some care. These choices require a basic understanding of non-linear estimation, what options exist and how they affect the result. The purpose of this chapter is to make an introduction to this topic, focusing on the general aspects rather than providing all details that are relevant for efficient implementations.

14.1 Non-linear estimation techniques

Geometric errors can be defined in rather general ways, as long as they refer to a quantity (or quantities) that relates to the Euclidean geometry of the space (or spaces) where the estimation problem is defined. In these cases we may use general non-linear optimization techniques, which do not make any specific assumption about ϵ other than a well-defined gradient (and Hessian), for example those described in Toolbox Section 9.1.

In practice, most estimation problems in geometry can be defined in terms of Euclidean distances in such a way that the minimization of ϵ leads to a *non-linear least squares problem*, described in Toolbox Section 9.2. Such

estimation problems are characterized by formulating the error function as $\varepsilon(\mathbf{z}) = \frac{1}{2} \|\mathbf{r}(\mathbf{z})\|^2$, where \mathbf{r} is a *residual vector* that quantifies the distances. For example, \mathbf{r} could contain one distance per element but, as is discussed in Section 14.3.2, other choices may be more suitable. In these methods, the gradient \mathbf{g} and Hessian \mathbf{H} of ε , which are used in each refinement for updating the solution, are expressed in terms of the Jacobian matrix, $\mathbf{J}(\mathbf{z})$. The Jacobian, in turn, contains the derivatives of $\mathbf{r}(\mathbf{z})$ with respect to the free variables in \mathbf{z} .

In this chapter we are mainly interested in non-linear least squares problem and methods for finding their solutions, for example, using the methods described in Toolbox Section 9.2 such as the Gauss-Newton method or the Levenberg-Marquardt method. In practice, \mathbf{J} can be either an analytic expression in \mathbf{z} , or it can be estimated from \mathbf{r} as long as the latter can be evaluated for arbitrary \mathbf{z} .

14.1.1 Initial solutions

The iterative non-linear optimization methods for minimizing a geometric error ε discussed here require an initial solution \mathbf{z}_0 from where the iterations start. The selection of \mathbf{z}_0 is one of the important choices mentioned above, which must be made with care. For example, using a fixed or random \mathbf{z}_0 is not a recommended strategy. A better approach is to formulate a corresponding algebraic error ε_{ALG} , which has the same minimizer as ε in the noise-free case, and find the minimizer of ε_{ALG} using linear techniques. As already mentioned, this often leads to relatively simple computations and provides a \mathbf{z}_0 that is likely to lie relatively close to the minimizer of the geometric error. This is one motivation to why we are interested in formulating and minimizing algebraic errors: with a limited set of computations they provide useful initial solutions for minimizing geometric errors.

In Chapter 12 we have seen that algebraic errors can be formulated and solved in different ways. For one and the same estimation problem, we can use the homogeneous method or the inhomogeneous method, in practice minimizing different algebraic errors. The inhomogeneous method and the homogeneous method in general give slightly different minimizers for the corresponding algebraic error. But there is no clear analysis that indicates which one of the two approaches is the better choice, in particular for providing \mathbf{z}_0 for non-linear optimization.

There can be practical reasons for preferring one of the two approaches. As is mentioned in Section 12.5.6, the inhomogeneous method is expected to give poor performance when the element that is set = 1 is zero, or close to zero, for \mathbf{z}_0 . Furthermore, the homogeneous method requires an accurate implementation of SVD, but this may be worth the effort since it provides a characterization of the solution space, based on the SVD profile described in Section 13.2.1. In the case that the homogeneous method is used, it should be based on normalized data, as described in Section 13.2.2. In general, minimization of algebraic errors applied normalized data can produce a solution that is closer to the minimizer of a geometric error than if applied to unnormalized data. As mentioned in observation 13.10 on page 229, data normalization should always be used in these cases.

14.1.2 Parameterizations

Another important aspect of non-linear estimation is how to parameterize the model to be estimated. So far, in discussions about estimation, the estimated model has been in the form of a homogeneous representation of a geometric object, e.g., in the form of a vector or a matrix. The parameters \mathbf{z} of this model are then the elements of the vector or the matrix and the estimation method determines the parameters \mathbf{z} more or less directly by solving linear equations. Consequently, we determine the elements of the model directly as the solutions of these equations.

This identification with the model and the parameters does not work in general, and in the following we instead keep the model and the parameters as separate entities. The model is in general denoted \mathbf{m} , and may in practice refer to, e.g., the dual homogeneous coordinates of a line \mathbf{l} , or a homography \mathbf{H} . In other cases, \mathbf{m} can be a rotation matrix, $\mathbf{R} \in SO(3)$, or an algebraic representation of a more complicated type of geometric object. The model is a function of some set of parameters, which we continue to denote as \mathbf{z} : $\mathbf{m} = \mathbf{m}(\mathbf{z})$. The critical issue is that the parameterization of \mathbf{m} often can be done in several different ways, and since ε is a function of \mathbf{m} and the data, it follows that ε can be formulated as a function of \mathbf{z} in different ways. This means that even if we define ε as a function of \mathbf{m} in one specific way, e.g., as a total least squares error, this ε can be described as different functions of the parameters \mathbf{z} depending on how \mathbf{m} is parameterized. These differences in parameterization of \mathbf{m} (and of ε), in turn, can have a significant impact on the quality of the estimated model and on the cost of computing it.

This section introduces some basic concepts in relation to parameterization that are further illustrated with examples in the following sections.

Minimal parameterization

Formally, we can define the degrees¹ of freedom of a specific model as the rank of the corresponding Hessian or the Jacobian. When the model is parameterized by the same number of parameters as its degree of freedom, this is referred to as a *minimal parameterization*. Consequently, a minimal parameterization in general leads to a Hessian or Jacobian of full rank. This property may be important in some cases, e.g., if we are using Newton's method or the Gauss-Newton method for the minimization of ε , since it guarantees that the refinement step can be computed without problems².

It should be noted that a minimal parameterization is a necessary but not sufficient condition for full rank of H or J . For some parameterizations it is the case that these matrices have full rank for almost all cases of the parameters z , but there are singularities, specific values of z that make them rank deficient. If the minimizer of ε lies at or close to such a singularity, it is likely that finding this minimization may be difficult.

- 14.1** If a minimal parameterization is used in non-linear estimation, it is important to choose a minimal parameterization that does not exhibit unnecessary singularities where H or J become rank deficient. To avoid such singularities, we need to investigate whether or not $H(z)$ or $J(z)$ become rank deficient for some z .

Over-parameterization

The alternative to a minimal parameterization is to use an over-parameterization, i.e., have more parameters for the model m than its degrees of freedom. For example, for the representation of a 2D line, which has 2 degrees of freedom, we may use as parameters the three elements of the vector $I \in \mathbb{R}^3$, the dual homogeneous coordinates of the line. This is an over-representation of the line since any multiplication of the three elements by a common scalar produces a new set of parameters that represent the same line. Another example of how to represent a 2D line is to consider two distinct points, representing the unique line that intersects both points. Since there are many choices of the two points that represent the same line, we have an over-representation of the line. The parameters of this particular representation could be the coordinates of the two points, a set of 4 scalars.

There are some immediate consequences of over-parameterizing the model. Clearly, each additional parameter adds one dimension to the parameter space, which means that the gradient, the Hessian, and the Jacobian become larger. Therefore, the computational complexity related to determining the refinement step increases. And we have already mentioned that the Hessian and the Jacobian related to the optimization problem become rank deficient, i.e., singular. This means that the computation of the refinement step, both for Newton's method and for the Gauss-Newton method, become degenerate and fail to produce a well-defined result. Modified versions of these methods can still be used, but standard implementations can often not be used out of the box.

With the above observations in mind, it may not be obvious why over-parameterizations are useful, but they are. For one thing, there are optimization methods that do not require the Hessian or the Jacobian to have full rank in order to work well. One example is the Levenberg-Marquardt method, described in Toolbox Section 9.2.2. This method is robust to most types of parameterizations which is one of the reasons for its popularity.

A second reason for over-parameterization is that empirical evidence exists that ε often becomes more complicated, containing many non-global optima, when a minimal parameterization is used compared to an over-parameterization. This increases the likelihood for the optimization process to get stuck in a local minimum rather than the global one. This can be avoided if we use an over-parameterized model [30].

- 14.2** A minimal parameterization is not necessarily an optimal choice for solving model estimation in geometry. In particular if you have access to optimization methods that are insensitive to rank deficiency in the Jacobian, such as the Levenberg-Marquardt method, then an over-parameterized model may be the best choice.

As will be presented later on, complicated estimation problems can often be solved efficiently by introducing additional free variables in the formulation of the error function ε , referred to as *auxiliary variables*. Although

¹Degrees of freedom is discussed in Toolbox Section 4.5.

²The Newton refinement step is described in Toolbox Section 9.1.2, Equation (9.9), Gauss-Newton's refinement is given in Equation (9.20).

they are an integral part of the formulation of ε , the auxiliary variables are usually not interesting after ε has been minimized and are then just discarded. Auxiliary variables appear, for example, in the case of optimal triangulation, Section 16.1.3, and in the gold standard method for estimation of the fundamental matrix described in Section 16.2.4. The method for solving the structure from motion problem in Chapter 21 relies heavily on the camera positions as auxiliary variables.

14.3 Over-parameterizations can also be useful to described particular estimation problem in a simple way, using auxiliary variables which, in the end, are not part of the model to be estimated.

Consistent parameterization

In a *consistent parameterization* each value of the parameters \mathbf{z} produces a valid model $\mathbf{m}(\mathbf{z})$. The concept of consistent parameterizations make sense for models with an algebraic representation that must satisfy some internal constraint. To better understand this idea, let us look at an example: how to parameterize a 3D line that is to be estimated from some data set.

In Section 5.3.2 the Plücker coordinates of a 3D line were presented, in terms of a matrix \mathbf{L} , and we can use this \mathbf{L} as the model \mathbf{m} that is estimated from some data set. Since \mathbf{L} is 4×4 and anti-symmetric it has 6 elements that need to be specified:

$$\mathbf{L} = \begin{pmatrix} 0 & a & b & c \\ -a & 0 & d & e \\ -b & -d & 0 & f \\ -c & -e & -f & 0 \end{pmatrix}. \quad (14.1)$$

These 6 elements form an over-representation of the 3D line. Since \mathbf{L} is a projective element, multiplying the matrix \mathbf{L} by a non-zero scalar gives a new set of elements that represents the same line and, hence, the same geometric error. However, if we allow the 6 elements to take arbitrary values, we do not have a consistent representation of a 3D line. A matrix \mathbf{L} which is a proper representation of a 3D line must satisfy an internal constraint, Equation (5.29). Consequently, we could in principle use only 5 of the elements in \mathbf{L} as parameters, and get the sixth element by solving $af - be + cd = 0$. This provides a consistent parameterization of a 3D line since the internal constraint always is satisfied. Another way to form a parameterization of a 3D line is to represent it by means of two distinct 3D points, lying on the line. Again, this is an over-parameterization of the line, but it is consistent since there is a one unique line that can be associated with any choice of two distinct points. We will look closer at this parameterization in Section 14.2.

The issue of consistency may or may not appear when geometric objects are parameterized. Some objects have an algebraic representation without internal constraints. For example, any non-zero vector in \mathbb{R}^3 can be used as the homogeneous coordinates of a 2D point, or the dual homogeneous coordinates of a 2D line. Similarly, any non-singular 3×3 matrix represents a homography transformation. But there are also algebraic representations of geometric objects where internal constraints appear, and the Plücker coordinates of a 3D line is just one example. We also seen these constraints in Chapters 4 and 6 for the matrices that represent various types of geometric transformations. A prominent example is a rotation matrix \mathbf{R} , which must satisfy $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$, and $\det \mathbf{R} = +1$. These constraints are the main reason for the distinction between a model \mathbf{m} and its parameters \mathbf{z} . A model \mathbf{m} can be an algebraic representation of some type of geometric object, but if it has to satisfy internal constraints, only a consistent parameterization of \mathbf{m} makes this happen for all values of \mathbf{z} .

In general, we should use a parameterization that is consistent. This does not have to mean that the representation we use always must be consistent. For example, when algebraic errors are minimized using the linear methods described in Chapters 12 and 13 the resulting model \mathbf{z} may not satisfy the internal constraints. This can be “fixed” by applying a step commonly referred to as *constraint enforcement*, which modifies \mathbf{m} to \mathbf{m}' where the latter satisfies the internal constraints. This is normally done by making the smallest possible modification to \mathbf{m} , e.g., minimizing $\|\mathbf{m} - \mathbf{m}'\|$, where \mathbf{m}' is a consistent representation.

The idea of constraint enforcement is simple and works fine in algebraic minimization. When geometric errors are minimized, it is not useful to first spend a significant amount of computational resources to minimize ε using an inconsistent parameterization of the model, and then apply constraint enforcement. The latter step will perturb the estimated model, increasing the residual error in an uncontrolled manner. Instead we should use a consistent

parameterization of the model in the optimization, which does not have to be modified in order to satisfy internal constraints in a later step.

14.4 Always use a consistent parameterization of the model to be estimated when non-linear estimation methods are used.

Complete parameterization

A final issue related to the parameterization of the estimated model is whether it is possible to represent *any* conceivable model by appropriately selecting the model parameters. If it is, we have a *complete parameterization* of the model, otherwise an *incomplete parameterization*.

For example, a complete parameterization of (proper) 2D points is given by their Cartesian coordinates (u, v) . Similarly, a 2D line can be represented by its (k, l) parameters, the slope and the intercept, but this is an incomplete parameterization. The reason is that a vertical line cannot be described by finite values of these parameters since, in principle, we would have to use $k = \infty$ for this case. Consequently, all vertical (and very near to vertical) lines are out of reach for this parameterization. As an alternative, the parameters (α, Δ) introduced in Section 3.2, provide a complete parameterization of a 2D line.

A complete parameterization is often the preferred choice, but is not a strict requirement in non-linear estimation. Non-linear estimation in geometry methods normally use algebraic minimization to obtain an initial solution \mathbf{z}_0 , which is assumed to lie relatively close to the global minimizer of a geometric error. This means that it is sufficient that we can parameterize models that lie in a region Ω of some size around \mathbf{z}_0 , where the global minimizer lies in Ω . An incomplete but efficient parameterization only for models in Ω can then be used for the optimization.

14.2 Re-parameterization

In the course of developing a useful non-linear estimation method it is sometimes the case that we use an initial set of parameters, which we may refer to as $\tilde{\mathbf{z}}$, and then later change to another set, which is denoted \mathbf{z} , where the former is a function of the latter: $\tilde{\mathbf{z}} = \tilde{\mathbf{z}}(\mathbf{z})$. For example, the initial parameters may be the elements of the model \mathbf{m} that we want to estimate, $\tilde{\mathbf{z}} = \mathbf{m}$, and there are some internal constraints on the model. In this case, $\tilde{\mathbf{z}}$ is an inconsistent parameterization, and at some point we should change to a consistent set of parameters. Another reason may be that we want to test different parameterizations of the model \mathbf{m} , without having to re-implement a lot of computations.

The process of changing from one set of parameters to another is referred to as *re-parameterization*, and involves only a few steps that need to be implemented in order to change the computations of the non-linear estimation. We recall that the refinement step in non-linear least squares problems requires two specific entities: the residual vector \mathbf{r} and the Jacobian matrix. The latter contains the derivatives of \mathbf{r} with respect to the elements of $\tilde{\mathbf{z}}$ and, therefore, we denote it as $\tilde{\mathbf{J}}$. Both \mathbf{r} and $\tilde{\mathbf{J}}$ are functions of the parameters $\tilde{\mathbf{z}} \in \mathbb{R}^{\tilde{m}}$:

$$\mathbf{r} = \mathbf{r}(\tilde{\mathbf{z}}) \in \mathbb{R}^n, \quad \text{and} \quad \tilde{\mathbf{J}} = \tilde{\mathbf{J}}(\tilde{\mathbf{z}}) \in \mathbb{R}^{n \times \tilde{m}}, \quad (14.2)$$

and we need to implement the computations for determining \mathbf{r} and $\tilde{\mathbf{J}}$ from any value of $\tilde{\mathbf{z}}$.

Now we do a re-parameterization, from $\tilde{\mathbf{z}}$ to $\mathbf{z} \in \mathbb{R}^m$. In general, $m \neq \tilde{m}$, and it can be either that $m < \tilde{m}$ or $m > \tilde{m}$. This re-parameterization means that both the residual and the Jacobian now should be seen as functions of \mathbf{z} . But it also means that the Jacobian now should contain the derivatives of \mathbf{r} with respect to the new parameters \mathbf{z} . In principle, we can determine explicitly how \mathbf{r} depends on \mathbf{z} , and implement the corresponding computations. Similarly, we can determine the explicit expressions for the derivatives of \mathbf{r} with respect to the new parameters \mathbf{z} . A simpler approach, however, may be to implement the mapping $\tilde{\mathbf{z}}(\mathbf{z})$, and by plugging this into the computation of \mathbf{r} and $\tilde{\mathbf{J}}$ we can determine them for any value of \mathbf{z} :

$$\mathbf{r} = \mathbf{r}(\tilde{\mathbf{z}}(\mathbf{z})), \quad \text{and} \quad \tilde{\mathbf{J}} = \tilde{\mathbf{J}}(\tilde{\mathbf{z}}(\mathbf{z})). \quad (14.3)$$

What is missing is that, now, we want to define the Jacobian matrix \mathbf{J} to contain the derivatives of \mathbf{r} with respect

to the new parameters \mathbf{z} , not with respect to the old ones. This \mathbf{J} can be determined by means of the *chain rule*³:

$$[\mathbf{J}]_{ij} = \frac{\partial r_i}{\partial z_j} = \sum_{k=1}^m \frac{\partial r_i}{\partial \tilde{z}_k} \underbrace{\frac{\partial \tilde{z}_k}{\partial z_j}}_{:= [\mathbf{D}]_{kj}} = \sum_{k=1}^m [\tilde{\mathbf{J}}]_{ik} [\mathbf{D}]_{kj}, \quad (14.4)$$

leading to

$$\mathbf{J} = \tilde{\mathbf{J}} \mathbf{D}. \quad (14.5)$$

Consequently, to obtain \mathbf{J} for a specific set of new parameters \mathbf{z} we simply need to determine the “old” Jacobian $\tilde{\mathbf{J}}$ from Equation (14.3) and then multiply it with the $\tilde{n} \times n$ matrix \mathbf{D} , holding the *inner derivatives* of the mapping from $\mathbf{z} \in \mathbb{R}^m$ to $\tilde{\mathbf{z}} \in \mathbb{R}^{\tilde{n}}$.

14.2.1 An example

To see how re-parameterization works, let us look at a simple example: how to estimate a 3D line. We will not go into the actual estimation, but focus on the issue of how to parameterize the 3D line. Initially, we can use the parameters specified by the Plücker coordinates in Equation (14.1):

$$\tilde{\mathbf{z}} = (\tilde{z}_1, \tilde{z}_2, \tilde{z}_3, \tilde{z}_4, \tilde{z}_5, \tilde{z}_6) = (a, b, c, d, e, f) \in \mathbb{R}^6. \quad (14.6)$$

We also have a residual vector $\mathbf{r}(\tilde{\mathbf{z}}) \in \mathbb{R}^n$ that specifies the error function $\varepsilon(\tilde{\mathbf{z}}) = \|\mathbf{r}(\tilde{\mathbf{z}})\|$. From this \mathbf{r} we can determine the Jacobian matrix as

$$\mathbf{J}(\mathbf{z}) = \begin{pmatrix} \nabla^\top r_1(\mathbf{z}) \\ \nabla^\top r_2(\mathbf{z}) \\ \vdots \\ \nabla^\top r_n(\mathbf{z}) \end{pmatrix}, \quad (14.7)$$

an $n \times 6$ matrix. The Jacobian, in turn, determines a gradient and Hessian that can be used in the iterative computations of the refinement steps in the optimization of ε .

The parameterization defined by \mathbf{z} is not consistent, however, since not every combination of these parameters satisfies the internal constrain $af - be + cd = 0$. One way to establish a consistent parameterization is to use two 3D points, with Cartesian coordinates

$$\bar{\mathbf{x}} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad \text{and} \quad \bar{\mathbf{x}}' = \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix}, \quad (14.8)$$

and use these Cartesian coordinates as the new parameters:

$$\mathbf{z} = (z_1 \ z_2 \ z_3 \ z_4 \ z_5 \ z_6) = (x_1 \ x_2 \ x_3 \ x'_1 \ x'_2 \ x'_3) \in \mathbb{R}^6. \quad (14.9)$$

This produces a consistent parameterization if we set $\mathbf{L} = \mathbf{x} \mathbf{x}'^\top - \mathbf{x}' \mathbf{x}^\top$, where \mathbf{x} and \mathbf{x}' are the corresponding homogeneous coordinates, in canonical form:

$$\mathbf{L} = \mathbf{x} \mathbf{x}'^\top - \mathbf{x}' \mathbf{x}^\top = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} \begin{pmatrix} x'_1 & x'_2 & x'_3 & 1 \end{pmatrix} - \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \\ 1 \end{pmatrix} \begin{pmatrix} x_1 & x_2 & x_3 & 1 \end{pmatrix} = \begin{pmatrix} 0 & x_1 x'_2 - x_2 x'_1 & x_1 x'_3 - x_3 x'_1 & x_1 - x'_1 \\ \cdot & 0 & x_2 x'_3 - x_3 x'_2 & x_2 - x'_2 \\ \cdot & \cdot & 0 & x_3 - x'_3 \\ \cdot & \cdot & \cdot & 0 \end{pmatrix}. \quad (14.10)$$

This means that the old parameters $\tilde{\mathbf{z}}$ can be expressed in terms of the new ones as

$$\tilde{\mathbf{z}} = \tilde{\mathbf{z}}(\mathbf{z}) = \begin{pmatrix} \tilde{z}_1 \\ \tilde{z}_2 \\ \tilde{z}_3 \\ \tilde{z}_4 \\ \tilde{z}_5 \\ \tilde{z}_6 \end{pmatrix} = \begin{pmatrix} z_1 z_5 - z_2 z_4 \\ z_1 z_6 - z_3 z_4 \\ z_1 - z_4 \\ z_2 z_4 - z_3 z_5 \\ z_2 - z_5 \\ z_3 - z_6 \end{pmatrix}, \quad (14.11)$$

³The chain rule and inner derivatives are discussed in Toolbox Section 4.2.1.4.

and the internal constraint $0 = af - be + cd = \tilde{z}_1\tilde{z}_6 - \tilde{z}_2\tilde{z}_5 + \tilde{z}_3\tilde{z}_4$ is now satisfied for all choices of $\tilde{\mathbf{z}}$. The mapping $\tilde{\mathbf{z}}(\mathbf{z})$ in Equation (14.11) allows us to determine \mathbf{r} as a function of \mathbf{z} for all values of \mathbf{z} , assuming that we already have an implementation of the $\mathbf{r}(\tilde{\mathbf{z}})$.

Finally, we also need the new Jacobian, \mathbf{J} . In accordance with Equation (14.5), it is obtained as $\tilde{\mathbf{J}}\mathbf{D}$, where the inner derivatives \mathbf{D} in this case are given as

$$\mathbf{D} = \begin{pmatrix} \frac{\partial \tilde{z}_1}{\partial z_1} & \dots & \frac{\partial \tilde{z}_1}{\partial z_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial \tilde{z}_6}{\partial z_1} & \dots & \frac{\partial \tilde{z}_6}{\partial z_6} \end{pmatrix} = \begin{pmatrix} z_5 & -z_4 & 0 & -z_2 & z_1 & 0 \\ z_6 & 0 & -z_4 & -z_3 & 0 & z_1 \\ 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & z_4 & -z_5 & z_2 & -z_3 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{pmatrix}. \quad (14.12)$$

In this example, both $\tilde{\mathbf{z}}$ and \mathbf{z} contains 6 parameters, which means that both are over-parameterizations of the 3D line, with only 4 degrees of freedom. Furthermore, of the two, \mathbf{z} forms a consistent parameterization, while $\tilde{\mathbf{z}}$ is inconsistent.

14.3 The residual

In the context of non-linear least squares estimation, the residual vector can be an arbitrary function of the parameters: $\mathbf{r} = \mathbf{r}(\mathbf{z})$, with the implicitly assumption that a small $\|\mathbf{r}\|$ means that we are close to the optimal model \mathbf{z} . In the case of minimizing geometric errors, which is what is discussed in this presentation, we can initially think of the elements of \mathbf{r} , here denoted as r_k , as geometric distances in an image or in 3D space. For example, r_k can be the distance from point k to a line that is to be estimated from a set of points. By using the error function $\epsilon_{TOT} = \frac{1}{2}\|\mathbf{r}\|^2$ in the estimation, we then find the line that minimizes the sum of the squared distances from the line to all the points.

14.3.1 Re-mapping of the residual

In some applications, we want the error function to be more general than just adding the squares of distances and, instead, allow some function or functions to be applied to the distances, before they are squared and added together:

$$\epsilon = \sum_{k=1}^n f_k^2(r_k). \quad (14.13)$$

Here, f_k is the specific function that is applied to distance r_k , although in some case we can be interested in using a single function f on all distances. These functions perform a *re-mapping of the residual vector* \mathbf{r} , to a new residual vector $\tilde{\mathbf{r}}$:

$$\tilde{\mathbf{r}} = \begin{pmatrix} \tilde{r}_1 \\ \tilde{r}_2 \\ \vdots \\ \tilde{r}_n \end{pmatrix} = \begin{pmatrix} f_1(r_1) \\ f_2(r_2) \\ \vdots \\ f_m(r_n) \end{pmatrix}, \quad (14.14)$$

and we define the error function as $\tilde{\epsilon} = \|\tilde{\mathbf{r}}\|^2$.

In the simplest case, these re-mapping functions can be used to apply an individual weight w_k to distance r_k , allowing us to control how the distances should be taken into account in the estimation to a larger or lesser degree. This implies an error function like

$$\tilde{\epsilon} = \sum_{k=1}^n w_k r_k^2, \quad (14.15)$$

which corresponds to choosing $f_k(r) = \sqrt{w_k}r$. A second example of residual re-mapping occurs if we are interested in adding the distances taken to power p before adding them, producing a p -norm of the residual \mathbf{r} . This can be accomplished by choosing a common function f on all distances, where $f(r) = r^{p/2}$. A third alternative appears if we want to use the techniques for estimation based on robust errors, discussed in Section 17.2. In this case, we use a common function f on all distances, which reduces the influence of outliers on the result of the estimation.

• The p -norm is defined in Toolbox Section 3.1.3.

Just as in the case of re-parameterization, changing the residual from \mathbf{r} to $\tilde{\mathbf{r}}$ can be done by re-implementing all calculations and produce the new residual $\tilde{\mathbf{r}}$ from \mathbf{z} , the set of parameters that is used. As an alternative, we can reuse the existing calculations that produce \mathbf{r} and simply apply the functions f_k on its elements to get $\tilde{\mathbf{r}}$, Equation (14.13). In addition to that, we also need to modify the Jacobian to give the derivatives of $\tilde{\mathbf{r}}$ with respect to the parameters \mathbf{z} and, again, these are obtained by means of the chain rule:

$$\nabla \tilde{\mathbf{r}}_k = \nabla f_k(r_k) = \frac{\partial f_k}{\partial r_k} \nabla r_k. \quad (14.16)$$

This gives a Jacobian as

$$\tilde{\mathbf{J}} = \begin{pmatrix} \nabla^\top \tilde{r}_1 \\ \nabla^\top \tilde{r}_2 \\ \vdots \\ \nabla^\top \tilde{r}_n \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{\partial f_1}{\partial r_1} & 0 & \dots & 0 \\ 0 & \frac{\partial f_2}{\partial r_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{\partial f_n}{\partial r_n} \end{pmatrix}}_{:=\mathbf{G}} \underbrace{\begin{pmatrix} \nabla^\top r_1 \\ \nabla^\top r_2 \\ \vdots \\ \nabla^\top r_n \end{pmatrix}}_{=\mathbf{J}} = \mathbf{G} \mathbf{J}. \quad (14.17)$$

Consequently, the computations of the gradient and the Hessian need to be modified in accordance with:

$$\tilde{\mathbf{g}} = \tilde{\mathbf{J}}^\top \tilde{\mathbf{r}} = \mathbf{J}^\top \mathbf{G}^\top \tilde{\mathbf{r}}, \quad \text{and} \quad \tilde{\mathbf{H}} = \tilde{\mathbf{J}}^\top \tilde{\mathbf{J}} = \mathbf{J}^\top \mathbf{G}^\top \mathbf{G} \mathbf{J} \quad (14.18)$$

Unsuitable choices of re-mapping functions

In principle, the re-mappings functions f_k in Equation (14.13) can be arbitrary, but there are some choices that should be avoided. First, each f_k should be a monotonous function where $f(0) = 0$. If f_k is not monotonous, and instead contains local minima, these may introduce local minima also in the error function ε . These, in turn, increase the chance of making the non-linear estimation produce a non-global minimum as a solution.

Second, we are in general interested in a Jacobian that does not contain very large values. If, for some reason, one or a few of its elements grow very large in magnitude, in practice this means that the rest of its elements become irrelevant and can almost be neglected. This implies that the Jacobian is not only rank deficient, but also degenerate: dominated by only of few of its elements.

In the context of residual re-mapping, this type of degeneracy occurs if any of the derivatives $\frac{\partial f_k}{\partial r_k}$ becomes very large. Consequently, we should avoid functions f_k with unbounded derivatives. For example, the above mentioned function $f(r) = r^{1/2}$, which implements the 1-norm of the residual vector \mathbf{r} , is unsuitable since it has an unbounded derivative close to $r = 0$. Taking these issues into account, there is still a large range of re-mapping functions that can be applied to the residual, adapting it to more general types of error functions.

14.5 Re-mapping of the residual vector allows us to work with more general types of error functions.

14.3.2 Choosing residual

In many of the applications that are described here, the estimation problem is defined in terms of a set of n observed image points $\bar{\mathbf{y}}_{O,k} = (u_{O,k}, v_{O,k})$ and a corresponding set of reprojected points $\bar{\mathbf{y}}_{M,k}(\mathbf{z}) = (u_{M,k}(\mathbf{z}), v_{M,k}(\mathbf{z}))$ that depend on the model parameters in \mathbf{z} in a non-linear way. The estimation problem boils down to determining the parameters that minimize the geometric error

$$\varepsilon(\mathbf{z}) = \frac{1}{2} \sum_{k=1}^n \|\bar{\mathbf{y}}_k - \bar{\mathbf{y}}'_k(\mathbf{z})\|^2 = \frac{1}{2} \sum_{k=1}^n \underbrace{(u_{O,k} - u_{M,k}(\mathbf{z}))^2}_{:=u_k(\mathbf{z})} + \underbrace{(v_{O,k} - v_{M,k}(\mathbf{z}))^2}_{:=v_k(\mathbf{z})} = \frac{1}{2} \sum_{k=1}^n u_k(\mathbf{z})^2 + v_k(\mathbf{z})^2. \quad (14.19)$$

In term of non-linear least squares optimization, this is equivalent to formulating a residual vector $\mathbf{r} \in \mathbb{R}^{2n}$ as

$$\mathbf{r} = \mathbf{r}(\mathbf{z}) = \begin{pmatrix} u_1(\mathbf{z}) \\ v_1(\mathbf{z}) \\ u_2(\mathbf{z}) \\ v_2(\mathbf{z}) \\ \vdots \\ u_n(\mathbf{z}) \\ v_n(\mathbf{z}) \end{pmatrix} = \underbrace{\begin{pmatrix} u_{O,1} \\ v_{O,1} \\ u_{O,2} \\ v_{O,2} \\ \vdots \\ u_{O,n} \\ v_{O,n} \end{pmatrix}}_{\text{observed coordinates}} - \underbrace{\begin{pmatrix} u_{M,1}(\mathbf{z}) \\ v_{M,1}(\mathbf{z}) \\ u_{M,2}(\mathbf{z}) \\ v_{M,2}(\mathbf{z}) \\ \vdots \\ u_{M,n}(\mathbf{z}) \\ v_{M,n}(\mathbf{z}) \end{pmatrix}}_{\text{model coordinates}}, \quad (14.20)$$

and minimizing $\varepsilon(\mathbf{z}) = \frac{1}{2} \|\mathbf{r}(\mathbf{z})\|^2$. With a suitable initial solution \mathbf{z}_0 this minimization can be done using either the Gauss-Newton method or the Levenberg-Marquardt method.

Let us now consider an alternative way of defining the residual, in terms of the Euclidean distances between the observed points $\bar{\mathbf{y}}_{O,k}$ and the parameterized points of the model $\bar{\mathbf{y}}_{M,k}$:

$$\tilde{\mathbf{r}} = \begin{pmatrix} d_1(\mathbf{z}) \\ d_2(\mathbf{z}) \\ \vdots \\ d_k(\mathbf{z}) \end{pmatrix} \in \mathbb{R}^m, \quad \text{where } d_k(\mathbf{z}) = \sqrt{u_k(\mathbf{z})^2 + v_k(\mathbf{z})^2}. \quad (14.21)$$

The residual $\tilde{\mathbf{r}}$ has a corresponding error function $\tilde{\varepsilon}(\mathbf{z}) = \frac{1}{2} \|\tilde{\mathbf{r}}(\mathbf{z})\|^2$.

From this formulation of the residual $\tilde{\mathbf{r}}$ it should be clear that the two error functions are equal: $\varepsilon(\mathbf{z}) = \tilde{\varepsilon}(\mathbf{z})$. Hence, minimizing ε should be equivalent to minimizing $\tilde{\varepsilon}$, should it not? Furthermore, $\tilde{\mathbf{r}}$ has half as many dimensions as \mathbf{r} which implies that the computations involved in minimizing $\tilde{\varepsilon}$ could be approximately half as many compared to minimizing ε . Therefore, minimization based on the residual $\tilde{\mathbf{r}}$ may even be more attractive than minimization based on \mathbf{r} . The issue of whether to use \mathbf{r} or $\tilde{\mathbf{r}}$ as the residual is subtle, and in many cases the result of the estimation does not have a great dependency on the choice. As will be demonstrated, however, the two alternatives are not entirely equivalent, and the choice may have an effect on the computational cost.

Singularities

Since $\varepsilon = \tilde{\varepsilon}$ it must be the case that $\nabla \varepsilon = \nabla \tilde{\varepsilon}$. We begin this investigation by looking at how the two gradients are formed in terms of the corresponding Jacobians. The Jacobian corresponding to \mathbf{r} is given as

$$\mathbf{J} = \begin{pmatrix} \nabla^\top u_1 \\ \nabla^\top v_1 \\ \vdots \\ \nabla^\top u_n \\ \nabla^\top v_n \end{pmatrix} \in \mathbb{R}^{2n \times m}, \quad (14.22)$$

and the gradient of ε with respect to \mathbf{z} is

$$\nabla \varepsilon = \mathbf{J}^\top \mathbf{r} = u_1 \nabla u_1 + v_1 \nabla v_1 + \dots + u_n \nabla u_n + v_n \nabla v_n. \quad (14.23)$$

The Jacobian corresponding to $\tilde{\mathbf{r}}$ is instead given as

$$\tilde{\mathbf{J}} = \begin{pmatrix} \frac{1}{d_1} (u_1 \nabla^\top u_1 + v_1 \nabla^\top v_1) \\ \vdots \\ \frac{1}{d_n} (u_n \nabla^\top u_n + v_n \nabla^\top v_n) \end{pmatrix} \in \mathbb{R}^{n \times m}, \quad (14.24)$$

and the gradient of ε with respect to \mathbf{z} is

$$\nabla \tilde{\varepsilon} = \tilde{\mathbf{J}}^\top \tilde{\mathbf{r}} = u_1 \nabla u_1 + v_1 \nabla v_1 + \dots + u_n \nabla u_n + v_n \nabla v_n = \nabla \varepsilon. \quad (14.25)$$

Hence, the two gradients are indeed equal. But should also be clear that $\tilde{\mathbf{J}}$ has a possibility of becoming numerically unstable when some $d_k \approx 0$. In practice, the chance for the occurrence of such a singularity is small, but should not be neglected. As we will see next, the choice of residual may also have other effects on the optimization.

The Hessian approximation

In accordance with Toolbox Section 9.2.3, the approximation of the Hessian of $\varepsilon = \tilde{\varepsilon}$ is based on the assumption that we can neglect the Hessians of the elements of the residual vector. The term in the Hessian of ε that is neglected is given as

$$\Delta H = \sum_{k=1}^n u_k H\{u_k\} + v_k H\{v_k\}. \quad (14.26)$$

In the case of $\tilde{\varepsilon}$, we instead neglect

$$\Delta \tilde{H} = \sum_{k=1}^n d_k H\{d_k\}. \quad (14.27)$$

Not only are these two terms different, they can have different magnitudes. To see this, we expand $H\{d_k\}$:

$$H\{d_k\} = \frac{1}{d_k} \left(u_k H\{u_k\} + v_k H\{v_k\} + \nabla u_k \nabla^\top u_k + \nabla v_k \nabla^\top v_k + \right) - \frac{1}{d_k^3} (u_k \nabla u_k + v_k \nabla v_k) (u_k \nabla^\top u_k + v_k \nabla^\top v_k), \quad (14.28)$$

and insert it into Equation (14.27):

$$\begin{aligned} \Delta \tilde{H} &= \sum_{k=1}^n u_k H\{u_k\} + v_k H\{v_k\} + \nabla u_k \nabla^\top u_k + \nabla v_k \nabla^\top v_k - \frac{1}{d_k^2} (u_k \nabla u_k + v_k \nabla v_k) (u_k \nabla^\top u_k + v_k \nabla^\top v_k) = \\ &= \Delta H + \sum_{k=1}^n \frac{1}{d_k^2} (v_k \nabla u_k - u_k \nabla v_k) (v_k \nabla u_k - u_k \nabla v_k)^\top. \end{aligned} \quad (14.29)$$

Normally, we assume that ΔH can be neglected. Does this mean that $\Delta \tilde{H}$ can be neglected, too? In general, a small ΔH does not guarantee a small $\Delta \tilde{H}$ since the latter requires the gradients ∇u_k and ∇v_k to be small. This does not happen unless u_k and v_k are close to a local stationary point for all k , and in general they are not. Consequently, if we use $\tilde{\mathbf{r}}$ as the residual instead of \mathbf{r} , we should expect to get a worse approximation of the Hessian of $\varepsilon = \tilde{\varepsilon}$, which leads to a less optimal refinement step \mathbf{h} to be computed in each iteration. This does not mean that this \mathbf{h} is wrong, but rather that it is perturbed by noise related to the coarse approximation of the Hessian. With a good initial solution \mathbf{z}_0 , the iterations will still lead to the expected minimum if we use $\tilde{\mathbf{r}}$, but this would in general require more iterations than if \mathbf{r} is used.

On top of that, the computational cost for multiplying matrix \mathbf{J} , that has double the number of rows as $\tilde{\mathbf{J}}$, indeed doubles, but this is only a smaller part of the total computations involved in each refinement step, where the inversion of the Hessian is the main cost. Therefore, minimizing the number of iterations is a better option than minimizing the size of the Jacobian. We summarize these finding as:

14.6 The residual vector should be defined in terms of differences of observed and modeled coordinates, rather than in terms of distances between the corresponding points.

14.4 An example: estimation of a line

With the discussions in the previous sections as a background, we will now look at a more complete example of non-linear estimation using least squares methods. Again, we return to the problem of finding a 2D line that optimally fits a set of points, introduced in Section 12.1. We have already seen several variants of this problem and solved it by linear methods, notably the total least squares approach in Section 12.2, and now we will try to solve it by means of non-linear methods. This does not mean that non-linear methods should be applied for this particular problem, but this problem is discussed here as a simple illustration from which several observations can be made.

14.4.1 Using a homogeneous representation as parameters

With reference to the initial discussion in Section 12.2, we want to find a line \mathbf{l} that minimizes the total least square error relative to the set of m points $\{\mathbf{y}_k\}$, where the error is defined as

$$e_{TOT} = \sum_{i=1}^n d_{PD}(\mathbf{y}_i, \mathbf{l})^2 = \sum_{i=1}^n |\text{norm}_{\mathbf{P}}(\mathbf{y}_i) \cdot \text{norm}_{\mathbf{D}}(\mathbf{l})|^2. \quad (14.30)$$

In Section 12.2 this estimation problem is solved by parameterizing the line in terms of its slope k and intercept l . In Equation (14.30), we instead formulate ε_{TOT} in terms of the homogeneous representation of the line: the vector $\mathbf{l} \in \mathbb{R}^3$. This means that the parameters $\tilde{\mathbf{z}}$ of this optimization problem are chosen as the elements of $\mathbf{l} = \tilde{\mathbf{z}} = (\tilde{z}_1, \tilde{z}_2, \tilde{z}_3)$. We assume that the homogeneous coordinates of the points already are in a canonical form: $\mathbf{y}_i = (u_i, v_i, 1)$. This gives

$$\varepsilon_{\text{TOT}}(\tilde{\mathbf{z}}) = \sum_{i=1}^n \left(\frac{u_i \tilde{z}_1 + v_i \tilde{z}_2 + \tilde{z}_3}{\sqrt{\tilde{z}_1^2 + \tilde{z}_2^2}} \right)^2, \quad (14.31)$$

which is to be minimized over the parameter vector $\tilde{\mathbf{z}}$.

This formulation is compatible with the Gauss-Newton method⁴ for non-linear least squares minimization. We define the elements of the residual vector⁵ \mathbf{r} , as

$$r_k = \frac{u_k \tilde{z}_1 + v_k \tilde{z}_2 + \tilde{z}_3}{\sqrt{\tilde{z}_1^2 + \tilde{z}_2^2}} = \frac{1}{\sqrt{\tilde{z}_1^2 + \tilde{z}_2^2}} (u_k \quad v_k \quad 1) \begin{pmatrix} \tilde{z}_1 \\ \tilde{z}_2 \\ \tilde{z}_3 \end{pmatrix} = \mathbf{y}_k^\top \frac{\tilde{\mathbf{z}}}{\sqrt{\tilde{z}_1^2 + \tilde{z}_2^2}}, \quad (14.32)$$

and minimizing ε_{TOT} is then equivalent to minimizing $\frac{1}{2} \|\mathbf{r}\|^2$. The vector \mathbf{r} itself can be expressed as

$$\mathbf{r} = \mathbf{r}(\tilde{\mathbf{z}}) = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{y}_1^\top \\ \mathbf{y}_2^\top \\ \vdots \\ \mathbf{y}_m^\top \end{pmatrix}}_{:= \mathbf{Y}^\top} \frac{\mathbf{z}}{\sqrt{\tilde{z}_1^2 + \tilde{z}_2^2}} = \mathbf{Y}^\top \frac{\mathbf{z}}{\sqrt{\tilde{z}_1^2 + \tilde{z}_2^2}}. \quad (14.33)$$

To use the Gauss-Newton method we also need $\tilde{\mathbf{J}}$, the corresponding Jacobian matrix⁶. Each row of $\tilde{\mathbf{J}}$, with index k , is the gradient of r_k with respect to the free variables, i.e., the elements of $\tilde{\mathbf{z}}$:

$$\tilde{\nabla} r_k = \begin{pmatrix} \frac{\partial}{\partial \tilde{z}_1} \\ \frac{\partial}{\partial \tilde{z}_2} \\ \frac{\partial}{\partial \tilde{z}_3} \end{pmatrix} r_k = \begin{pmatrix} \frac{u_k \tilde{z}_2 - v_k \tilde{z}_1 - \tilde{z}_3}{(\tilde{z}_1^2 + \tilde{z}_2^2)^{3/2}} \\ \frac{-u_k \tilde{z}_1 \tilde{z}_2 + v_k \tilde{z}_1^2 - \tilde{z}_2 \tilde{z}_3}{(\tilde{z}_1^2 + \tilde{z}_2^2)^{3/2}} \\ \frac{1}{(\tilde{z}_1^2 + \tilde{z}_2^2)^{1/2}} \end{pmatrix} = \frac{1}{(\tilde{z}_1^2 + \tilde{z}_2^2)^{3/2}} \begin{pmatrix} u_k \tilde{z}_2^2 - v_k \tilde{z}_1 \tilde{z}_2 - \tilde{z}_1 \tilde{z}_3 \\ -u_k \tilde{z}_1 \tilde{z}_2 + v_k \tilde{z}_1^2 - \tilde{z}_2 \tilde{z}_3 \\ \tilde{z}_1^2 + \tilde{z}_2^2 \end{pmatrix}. \quad (14.34)$$

This expression can be formulated in a more compact form as

$$\tilde{\nabla} r_k = \underbrace{\frac{1}{(\tilde{z}_1^2 + \tilde{z}_2^2)^{3/2}} \begin{pmatrix} \tilde{z}_2^2 & -\tilde{z}_1 \tilde{z}_2 & -\tilde{z}_1 \tilde{z}_3 \\ -\tilde{z}_1 \tilde{z}_2 & \tilde{z}_1^2 & -\tilde{z}_2 \tilde{z}_3 \\ 0 & 0 & \tilde{z}_1^2 + \tilde{z}_2^2 \end{pmatrix}}_{:= \tilde{\mathbf{Z}}} \begin{pmatrix} u_k \\ v_k \\ 1 \end{pmatrix} = \tilde{\mathbf{Z}} \mathbf{y}_k. \quad (14.35)$$

The Jacobian $\tilde{\mathbf{J}}$ holds these gradients, transposed, in its rows:

$$\tilde{\mathbf{J}} = \begin{pmatrix} \mathbf{y}_1^\top \\ \mathbf{y}_2^\top \\ \vdots \\ \mathbf{y}_m^\top \end{pmatrix} \tilde{\mathbf{Z}}^\top = \mathbf{Y}^\top \tilde{\mathbf{Z}}^\top. \quad (14.36)$$

Based on these expressions, we can now write the gradient and Hessian of ε_{TOT} as⁷

$$\tilde{\mathbf{g}} = \tilde{\mathbf{J}}^\top \mathbf{r} = \tilde{\mathbf{Z}} \mathbf{Y} \mathbf{Y}^\top \frac{\tilde{\mathbf{z}}}{\sqrt{\tilde{z}_1^2 + \tilde{z}_2^2}}, \quad \text{and} \quad \tilde{\mathbf{H}} = \tilde{\mathbf{J}}^\top \tilde{\mathbf{J}} = \tilde{\mathbf{Z}} \mathbf{Y} \mathbf{Y}^\top \tilde{\mathbf{Z}}^\top. \quad (14.37)$$

⁴The Gauss-Newton optimization method is described in Toolbox Section 9.2.1

⁵The residual vector is defined in Toolbox Section 9.2, Equation (9.10).

⁶The Jacobian is defined in Toolbox Section 9.2, Equation (9.14).

⁷The expressions for the Jacobian and the Hessian are found in Toolbox Section 9.2.1, Equations (9.17) and (9.18). The expression for the Hessian is actually an approximation, but is used in the computations of the refinement step.

Assuming that we also have a reasonable initial solution $\tilde{\mathbf{z}}_0$, we are now in possession of all the necessary components for applying the Gauss-Newton method to the minimization of ε_{TOT} , possibly in combination with steepest descent, for example as done in the Levenberg-Marquardt method.

Before doing that, let us first look at the results at hand. Since a 2D line has 2 degrees of freedom and we are using 3 parameters in \mathbf{z} , the model is over-parameterized. In Section 14.1.2 we said that in the case of an over-parameterized model, we should expect the Jacobian to be rank deficient, possibly causing problems for some estimation methods. It can easily be verified that $\tilde{\mathbf{z}}^\top \tilde{\mathbf{Z}} = \mathbf{0}$, leading to $\tilde{\mathbf{J}} \tilde{\mathbf{z}} = \mathbf{0}$, and we conclude that $\tilde{\mathbf{J}}$ is indeed rank deficient. This means that the Gauss-Newton method cannot be used out of the box, since $\tilde{\mathbf{H}} = \tilde{\mathbf{J}}^\top \tilde{\mathbf{J}}$ cannot be inverted. This observation does not mean that we should avoid the proposed parameterization. It simply means that we should use a method that is not sensitive to rank deficient Jacobian, e.g., the Levenberg-Marquardt method.

14.4.2 Minimal parameterizations

We know that the line \mathbf{l} has 2 degrees of freedom, i.e., it is sufficient with only 2 free parameters to describe the line. Instead of using the 3 elements of $\mathbf{l} \in \mathbb{R}^3$, where we know that a common scaling of its elements leaves ε_{TOT} unchanged, we can describe the line with some suitable choice of 2 parameters. This re-parameterization can be done in several ways, and we investigate two of them here.

Using α and Δ

In Section 3.2 the 2D line is parameterized using the direction of a normal to the line, in terms of the angle α , and the distance from the line to the origin Δ . The corresponding parameter vector \mathbf{z} is given as

$$\mathbf{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \alpha \\ \Delta \end{pmatrix}. \quad (14.38)$$

We can express the old parameters $\tilde{\mathbf{z}} = \mathbf{l}$ as a function of these new parameters:

$$\tilde{\mathbf{z}} = \tilde{\mathbf{z}}(\mathbf{z}) = \begin{pmatrix} \tilde{z}_1 \\ \tilde{z}_3 \\ \tilde{z}_3 \end{pmatrix} = \begin{pmatrix} \cos \alpha \\ \sin \alpha \\ -\Delta \end{pmatrix} = \begin{pmatrix} \cos z_1 \\ \sin z_1 \\ -z_2 \end{pmatrix}. \quad (14.39)$$

Formally, we should insist that $z_2 = \Delta \geq 0$ since it represents a distance. However, we can also think of Δ as a signed distance, described in Section 3.4.2, relative to the direction given by α . This simplifies the estimation, since then we do not have to worry about the sign of z_2 during the minimization of ε_{TOT} . The re-parameterization of $\tilde{\mathbf{z}}$ described in Equation (14.39) leads automatically to a D-normalization of $\mathbf{l} = \tilde{\mathbf{z}}$, since it assures $\tilde{z}_1^2 + \tilde{z}_2^2 = 1$.

Let us now look at how this re-parameterization changes the optimization, using the approach outlined in Section 14.2, by taking the following steps:

1. The residual vector \mathbf{r} is now a function of \mathbf{z} : $\mathbf{r} = \mathbf{r}(\mathbf{z})$. Based on Equation (14.33), this leads to

$$\mathbf{r}(\mathbf{z}) = \mathbf{Y}^\top \tilde{\mathbf{z}}(\mathbf{z}). \quad (14.40)$$

Alternatively, and this is an important observation, we can also think of \mathbf{r} as a function of the previous parameters $\tilde{\mathbf{z}}$, where the latter is a function of \mathbf{z} , Equation (14.39). Consequently, we can alternatively write $\mathbf{r} = \mathbf{r}(\tilde{\mathbf{z}}(\mathbf{z}))$, with $\mathbf{r}(\tilde{\mathbf{z}})$ defined as in Equation (14.33).

2. The gradient and the Hessian are now derived from a Jacobian \mathbf{J} that is defined in terms of gradients of r_k with respect to the new parameters \mathbf{z} . These gradients, ∇r_k , are 2-dimensional vectors rather than the 3-dimensional vectors $\tilde{\nabla} r_k$ in Equation (14.34). On the other hand, since we can think of \mathbf{r} as a function of $\tilde{\mathbf{z}}$, and $\tilde{\mathbf{z}}$ is a function of \mathbf{z} , Equation (14.39), we can express ∇r_k in terms of $\tilde{\nabla} r_k$ using the chain rule:

$$\nabla r_k = \begin{pmatrix} \frac{\partial r_k}{\partial z_1} \\ \frac{\partial r_k}{\partial z_2} \end{pmatrix} = \begin{pmatrix} \frac{\partial r_k}{\partial \tilde{z}_1} \frac{\partial \tilde{z}_1}{\partial z_1} + \frac{\partial r_k}{\partial \tilde{z}_2} \frac{\partial \tilde{z}_2}{\partial z_1} + \frac{\partial r_k}{\partial \tilde{z}_3} \frac{\partial \tilde{z}_3}{\partial z_1} \\ \frac{\partial r_k}{\partial \tilde{z}_1} \frac{\partial \tilde{z}_1}{\partial z_2} + \frac{\partial r_k}{\partial \tilde{z}_2} \frac{\partial \tilde{z}_2}{\partial z_2} + \frac{\partial r_k}{\partial \tilde{z}_3} \frac{\partial \tilde{z}_3}{\partial z_2} \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{\partial \tilde{z}_1}{\partial z_1} & \frac{\partial \tilde{z}_2}{\partial z_1} & \frac{\partial \tilde{z}_3}{\partial z_1} \\ \frac{\partial \tilde{z}_1}{\partial z_2} & \frac{\partial \tilde{z}_2}{\partial z_2} & \frac{\partial \tilde{z}_3}{\partial z_2} \end{pmatrix}}_{= \mathbf{D}^\top} \underbrace{\begin{pmatrix} \frac{\partial r_1}{\partial \tilde{z}_1} \\ \frac{\partial r_1}{\partial \tilde{z}_2} \end{pmatrix}}_{= \tilde{\nabla} r_k} = \mathbf{D}^\top \tilde{\nabla} r_k. \quad (14.41)$$

3. These new gradients imply that the new Jacobian \mathbf{J} is given as

$$\mathbf{J} = \tilde{\mathbf{J}} \mathbf{D}. \quad (14.42)$$

where $\tilde{\mathbf{J}}$ is the Jacobian defined in Equation (14.36). Consequently, in order to construct \mathbf{J} from $\tilde{\mathbf{J}}$ we need \mathbf{D} , containing the *inner derivatives* which describe the mapping from \mathbf{z} to $\tilde{\mathbf{z}}$. In this example, \mathbf{D} is given from

$$\mathbf{D}^\top = \begin{pmatrix} \frac{\partial \tilde{z}_1}{\partial z_1} & \frac{\partial \tilde{z}_2}{\partial z_1} & \frac{\partial \tilde{z}_3}{\partial z_1} \\ \frac{\partial \tilde{z}_1}{\partial z_2} & \frac{\partial \tilde{z}_2}{\partial z_2} & \frac{\partial \tilde{z}_3}{\partial z_2} \end{pmatrix} = \begin{pmatrix} -\sin z_1 & \cos z_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -\tilde{z}_2 & \tilde{z}_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (14.43)$$

Inserted into Equation (14.42), and combined with Equation (14.36), this leads to

$$\mathbf{J} = \mathbf{Y}^\top \tilde{\mathbf{Z}}^\top \mathbf{D} = \mathbf{Y}^\top \mathbf{Z}^\top, \quad (14.44)$$

with

$$\mathbf{Z} = \mathbf{D}^\top \tilde{\mathbf{Z}} = \begin{pmatrix} -\tilde{z}_2 & \tilde{z}_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \tilde{z}_2^2 & -\tilde{z}_1 \tilde{z}_2 & -\tilde{z}_1 \tilde{z}_3 \\ -\tilde{z}_1 \tilde{z}_2 & \tilde{z}_1^2 & -\tilde{z}_2 \tilde{z}_3 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} -\tilde{z}_2 & \tilde{z}_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (14.45)$$

4. The corresponding gradient \mathbf{g} is given as

$$\mathbf{g} = \mathbf{J}^\top \mathbf{r} = \mathbf{D}^\top \tilde{\mathbf{g}} = \mathbf{Z} \mathbf{Y} \mathbf{Y}^\top \tilde{\mathbf{z}}(\mathbf{z}) \in \mathbb{R}^2, \quad (14.46)$$

and the Hessian is

$$\mathbf{H} = \mathbf{J}^\top \mathbf{J} = \mathbf{D}^\top \tilde{\mathbf{H}} \mathbf{D} = \mathbf{Z} \mathbf{Y} \mathbf{Y}^\top \mathbf{Z} \in \mathbb{R}^{2 \times 2}. \quad (14.47)$$

These steps imply that we can reuse the calculations already made in Section 14.4.1 to produce $\tilde{\mathbf{J}}$. The only additional information needed here is the mapping $\tilde{\mathbf{z}}(\mathbf{z})$, and \mathbf{D} , the derivatives of this mapping.

From Section 3.3.3 we recall that \mathbf{Y} has full rank ($= 3$) when the points are not exactly on a line, which in a practical estimation situation they never are. From $\tilde{z}_1 = \cos z_1$ and $\tilde{z}_2 = \sin z_1$ follows that \mathbf{Z} , too, is of full rank. Consequently, \mathbf{H} is of full rank and has a well-defined inverse and, therefore, the Gauss-Newton method works fine.

14.7 Choosing a minimal parameterization is an advantage if we want to use Gauss-Newton, since it is a necessary condition for a non-singular Hessian.

As we will see next, choosing a minimal parameterization is not a sufficient condition for a full rank Hessian.

Using k and l

The parameterization of \mathbf{l} in terms of α and Δ is not the only one possible. A line can also be described in terms of its slope k and intercept l , so we may also use the parameters $\hat{\mathbf{z}} = (\hat{z}_1, \hat{z}_2) = (k, l)$. This implies that the corresponding dual homogeneous coordinates of the line, Equation (3.15), are given as

$$\mathbf{z}(\hat{\mathbf{z}}) = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} k \\ -1 \\ l \end{pmatrix} = \begin{pmatrix} \hat{z}_1 \\ -1 \\ \hat{z}_2 \end{pmatrix}. \quad (14.48)$$

Given that we already have investigated what happens when we change the parameterization from \mathbf{z} to the parameters $\tilde{\mathbf{z}}$, this time we expect that

1. The residual error becomes $\mathbf{r} = \mathbf{r}(\hat{\mathbf{z}}) = \mathbf{Y}^\top \mathbf{z}(\hat{\mathbf{z}})$.
2. The gradients of r_k , the elements of \mathbf{r} , with respect to $\hat{\mathbf{z}}$ can be written as $\hat{\nabla} r_k = \hat{\mathbf{D}} \nabla r_k$. Here, $\hat{\mathbf{D}}$ is a 2×3 matrix that contains the inner derivatives:

$$\hat{\mathbf{D}} = \begin{pmatrix} \frac{\partial z_1}{\partial \hat{z}_1} & \frac{\partial z_2}{\partial \hat{z}_1} & \frac{\partial z_3}{\partial \hat{z}_1} \\ \frac{\partial z_1}{\partial \hat{z}_2} & \frac{\partial z_2}{\partial \hat{z}_2} & \frac{\partial z_3}{\partial \hat{z}_2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (14.49)$$

3. The corresponding Jacobian is given as $\tilde{\mathbf{J}} = \mathbf{J} \tilde{\mathbf{D}}^\top = \mathbf{Y}^\top \tilde{\mathbf{Z}}^\top$, where

$$\tilde{\mathbf{Z}} = \tilde{\mathbf{D}} \mathbf{Z} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \frac{1}{(z_1^2 + z_2^2)^{3/2}} \begin{pmatrix} z_2^2 & -z_1 z_2 & -z_1 z_3 \\ -z_1 z_2 & z_1^2 & -z_2 z_3 \\ 0 & 0 & z_1^2 + z_2^2 \end{pmatrix} = \frac{1}{(z_1^2 + z_2^2)^{3/2}} \begin{pmatrix} z_2^2 & -z_1 z_2 & -z_1 z_3 \\ 0 & 0 & z_1^2 + z_2^2 \end{pmatrix} \quad (14.50)$$

4. The gradient and Hessian of ϵ_{TOT} with respect to $\hat{\mathbf{z}}$ are given as

$$\hat{\mathbf{g}} = \hat{\mathbf{J}}^\top \mathbf{r} = \hat{\mathbf{D}} \mathbf{g} = \hat{\mathbf{Z}} \mathbf{Y} \mathbf{Y}^\top \mathbf{z}(\hat{\mathbf{z}}), \quad \text{and} \quad \hat{\mathbf{H}} = \hat{\mathbf{J}}^\top \hat{\mathbf{J}} = \hat{\mathbf{D}} \mathbf{H} \hat{\mathbf{D}}^\top = \hat{\mathbf{Z}} \mathbf{Y} \mathbf{Y}^\top \hat{\mathbf{Z}}^\top \quad (14.51)$$

These expressions for the gradient, Hessian, or the Jacobian can be plugged into the Gauss-Newton method and it works fine in most cases, but not always. We already know that the (k, l) parameterization of the line is problematic: it cannot represent a vertical line since, in that case, $k = \hat{z}_1 = \infty$. This observation affects also the non-linear estimation of the line, if the points are configured in such a way that the optimal line is vertical (or very near to vertical). In each refinement step, k will then be changed to an ever increasing and unbounded value, even if it starts from an initial solution (k_0, l_0) where k_0 is bounded. A closer look at $\hat{\mathbf{D}}$ reveals that it goes to $\mathbf{0}$ when $z_1 = k$ goes to infinity, i.e., $\hat{\mathbf{J}}$ in practice becomes so close to zero in this iterative process that, eventually, the numerical accuracy of the computations deteriorates. In this case, the problem is not that the Jacobian becomes rank deficient, it is rather the case that both sides of $\mathbf{J}^\top \mathbf{J} \mathbf{h} = -\mathbf{J}^\top \mathbf{r}$ become approximately zero, leaving the refinement step \mathbf{h} undefined. Hence, also the Levenberg-Marquardt method may fail to estimate a vertical line using the (k, l) -parameterization.

14.8 It is not sufficient to choose a minimal parameterization to avoid singularities in the Hessian. It is also necessary to choose a *good* minimal parameterization.

In this context, a good minimal parameterization means one which does not make the Hessian singular for any choice of the parameters. The (α, Δ) parameterization is good since $\tilde{\mathbf{J}}$ has full rank for all values of (α, Δ) . Compare this to the (k, l) parameterization, which makes the Jacobian rank deficient for vertical lines. It is therefore not suitable for neither the Gauss-Newton method nor the Levenberg-Marquardt method.

14.5 An example: estimation of a homography

As second and final example of the methods discussed here, let us estimate a homography \mathbf{H} by minimizing the geometric error ϵ_{HOM} in Equation (13.2) from a set of corresponding points $\{\mathbf{y}'_k \sim \mathbf{H} \mathbf{y}_k, k = 1, \dots, m\}$. Since \mathbf{H} has no internal constraints, a straightforward parameterization is to use the 9 parameters in

$$\mathbf{z} = \begin{pmatrix} z_1 & z_4 & z_7 \\ z_2 & z_5 & z_8 \\ z_3 & z_6 & z_9 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} = \begin{pmatrix} - & \mathbf{h}_1 & - \\ - & \mathbf{h}_2 & - \\ - & \mathbf{h}_3 & - \end{pmatrix} = \mathbf{H}. \quad (14.52)$$

The residual vector $\mathbf{r} \in \mathbb{R}^{2n}$ corresponding to ϵ_{HOM} has elements given as

$$\mathbf{r} = \begin{pmatrix} \vdots \\ \frac{\mathbf{h}_1 \cdot \mathbf{y}_k}{\mathbf{h}_3 \cdot \mathbf{y}_k} - u'_k \\ \frac{\mathbf{h}_2 \cdot \mathbf{y}_k}{\mathbf{h}_3 \cdot \mathbf{y}_k} - v'_k \\ \vdots \end{pmatrix}, \quad \text{where} \quad \mathbf{y}_k = \begin{pmatrix} u_k \\ v_k \\ 1 \end{pmatrix}. \quad (14.53)$$

The corresponding Jacobian is

$$\mathbf{J} = \begin{pmatrix} \vdots & \vdots \\ \frac{u_k}{\mathbf{h}_3 \cdot \mathbf{y}_k} & 0 & -\frac{u_k(\mathbf{h}_1 \cdot \mathbf{y}_k)}{(\mathbf{h}_3 \cdot \mathbf{y}_k)^2} & \frac{v_k}{\mathbf{h}_3 \cdot \mathbf{y}_k} & 0 & -\frac{v_k(\mathbf{h}_1 \cdot \mathbf{y}_k)}{(\mathbf{h}_3 \cdot \mathbf{y}_k)^2} & \frac{1}{\mathbf{h}_3 \cdot \mathbf{y}_k} & 0 & -\frac{\mathbf{h}_1 \cdot \mathbf{y}_k}{(\mathbf{h}_3 \cdot \mathbf{y}_k)^2} \\ 0 & \frac{u_k}{\mathbf{h}_3 \cdot \mathbf{y}_k} & -\frac{u_k(\mathbf{h}_2 \cdot \mathbf{y}_k)}{(\mathbf{h}_3 \cdot \mathbf{y}_k)^2} & 0 & \frac{v_k}{\mathbf{h}_3 \cdot \mathbf{y}_k} & -\frac{v_k(\mathbf{h}_2 \cdot \mathbf{y}_k)}{(\mathbf{h}_3 \cdot \mathbf{y}_k)^2} & 0 & \frac{1}{\mathbf{h}_3 \cdot \mathbf{y}_k} & -\frac{\mathbf{h}_2 \cdot \mathbf{y}_k}{(\mathbf{h}_3 \cdot \mathbf{y}_k)^2} \\ \vdots & \vdots \end{pmatrix}. \quad (14.54)$$

It is straightforward to show that $\mathbf{J}\mathbf{z} = \mathbf{0}$, i.e., \mathbf{J} is rank deficient. This is expected since \mathbf{z} is an over-parameterization of \mathbf{H} as a projective element. This observation also means that any refinement \mathbf{h} of \mathbf{z} is orthogonal to \mathbf{z} itself. Consequently, over the iterations that optimize ε_{HOM} , \mathbf{z} will not significantly change its norm but rather its direction in \mathbb{R}^9 . An initial solution \mathbf{z}_0 can be found by an algebraic estimation of \mathbf{H} , where the homogeneous approach with data normalization in Algorithm 13.3 on page 238 is recommended.

Symmetric error function

The function ε_{HOM} measures errors only in the image where the points \mathbf{y}'_k are found. As a consequence of fact, we notice that \mathbf{J} only depends on the image coordinates \mathbf{y}_k , not on \mathbf{y}'_k . Minimization of errors only in one image implies that the geometric errors also in the other image should become small, but there is no guarantee for optimality in terms of maximum likelihood. More precisely, let us assume that we instead formulate the error function

$$\varepsilon'_{\text{HOM}} = \sum_k d_{\text{PP}}(\hat{\mathbf{H}}\mathbf{y}'_k, \mathbf{y}_k)^2, \quad (14.55)$$

and find its minimizer $\hat{\mathbf{H}}$. This can be done by the method described above by just exchanging \mathbf{y}'_k and \mathbf{y}_k , and as result we expect to get $\hat{\mathbf{H}} = \mathbf{H}^{-1}$. This, however, is in general not the case. Indeed, in most practical cases we get $\hat{\mathbf{H}} \approx \mathbf{H}^{-1}$ as projective elements, but there is no guarantee for perfect symmetry based on minimizing cost functions such as ε_{HOM} or $\varepsilon'_{\text{HOM}}$.

Instead, a completely symmetric error function can be formulated as

$$\varepsilon_{\text{SYM}} = \sum_k d_{\text{PP}}(\mathbf{y}'_k, \mathbf{H}\mathbf{y}_k)^2 + d_{\text{PP}}(\mathbf{H}^{-1}\mathbf{y}'_k, \mathbf{y}_k)^2, \quad (14.56)$$

The corresponding residual \mathbf{r} and Jacobian \mathbf{J} , as functions of \mathbf{z} in Equation (14.52), can be derived, but lead to relatively complicated expressions. We leave this as an exercise to the reader, but conclude with some guidance for the derivation.

In principle, we can see ε_{SYM} as a function of two homographies \mathbf{H} and $\hat{\mathbf{H}}$ where in the end we want $\hat{\mathbf{H}} = \mathbf{H}^{-1}$. Thus, we parameterize ε_{SYM} by $\hat{\mathbf{z}} \in \mathbb{R}^{18}$ that contains the elements of these two homography matrices: $\hat{\mathbf{z}} = (\mathbf{H}, \hat{\mathbf{H}})$, in vectorized form. This is an inconsistent parameterization since it does not take into account the internal constraints $\hat{\mathbf{H}} = \mathbf{H}^{-1}$. On the other hand, with this parameterization it is straightforward to formulate a residual as

$$\mathbf{r} = \begin{pmatrix} \vdots \\ \mathbf{h}_1 \cdot \mathbf{y}_k - u'_k \\ \mathbf{h}_3 \cdot \mathbf{y}_k - v'_k \\ \mathbf{h}_2 \cdot \mathbf{y}_k - v'_k \\ \mathbf{h}_3 \cdot \mathbf{y}_k - u'_k \\ \hat{\mathbf{h}}_1 \cdot \mathbf{y}'_k - u_k \\ \hat{\mathbf{h}}_3 \cdot \mathbf{y}'_k - v_k \\ \hat{\mathbf{h}}_2 \cdot \mathbf{y}'_k - v_k \\ \hat{\mathbf{h}}_3 \cdot \mathbf{y}'_k - u_k \\ \vdots \end{pmatrix}, \quad (14.57)$$

where $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3$ are the columns of $\hat{\mathbf{H}}$. The corresponding $4n \times 18$ Jacobian $\hat{\mathbf{J}}$ is given by extending Equation (14.54) to this case. Notice that $\hat{\mathbf{J}}$ is relatively sparse, it contains n blocks of 2×9 size, consisting entirely of zeros

What remains is to arrive at a consistent parameterization. To this end we use $\mathbf{z} \in \mathbb{R}^9$ in Equation (14.52), and map it to $\hat{\mathbf{z}} \in \mathbb{R}^{18}$ by

$$\hat{\mathbf{z}}(\mathbf{z}) = (\mathbf{z}, \underbrace{\mathbf{z}^{-1}}_{\text{matrix inverse}}). \quad (14.58)$$

Finally, we also need the 18×9 matrix \mathbf{D} which contains the derivatives of this mapping. The topmost half is just an identity matrix, and the lower half contains the derivatives of every element in \mathbf{H}^{-1} with respect to every element in \mathbf{H} :

$$\mathbf{D} = \begin{pmatrix} \mathbf{I}_{9 \times 9} \\ \hat{\mathbf{D}} \end{pmatrix}, \quad \text{where } [\hat{\mathbf{D}}]_{ij} = \frac{\partial [\mathbf{H}^{-1}]_i}{\partial [\mathbf{H}]_j}. \quad (14.59)$$

Here, \mathbf{H} is treated as a vector in \mathbb{R}^9 on which the operation of 3×3 matrix inversion can be applied.

The derivatives in $\hat{\mathbf{D}}$ can be worked out by hand and then implemented in code, for example using the recursive expression for matrix inversion in Toolbox Section 3.66. The resulting expressions, however, are somewhat complicated and fall outside the scope of this presentation. In the end, we get $\mathbf{J} = \hat{\mathbf{J}}\mathbf{D}$, where \mathbf{J} is the Jacobian related to the consistent parameters in \mathbf{z} .

14.6 Practical issues

The methods described in this chapter are useful for most of the estimation problems that appear in geometry, at least when \mathbf{J} is of moderate size. Remember that \mathbf{J} is of size $m \times n$, where m is the number of elements in the residual vector \mathbf{r} and n is the number of free parameters in \mathbf{z} . The critical parameter is n , the number of parameters, since it determines the size of the gradient vector and of the Hessian matrix. This size is relevant in particular since the Hessian, in one way or another, is inverted in to find the refinement step in the Gauss-Newton method. This is the case both for the Gauss-Newton method and the Levenberg-Marquardt method. But a large n also increases the computational complexity since it determines the size of the matrix multiplications that are involved in producing the gradient and the Hessian from the Jacobian.

14.6.1 Computing the Jacobian

The Jacobian matrix must be determined in each iteration, as a function of the parameters \mathbf{z} , in order to compute the gradient and Hessian. This can be done in two different ways: either we have an analytic expression for each element in \mathbf{J} as a function of \mathbf{z} and we evaluate all these expressions for each iteration, or \mathbf{J} can be approximated using finite differences of the residual vector \mathbf{r} . In the latter case, recall that each row of \mathbf{J} contains the gradient of an element in \mathbf{r} with respect to \mathbf{z} . This implies that element (i, j) of \mathbf{J} can be approximated, in the simplest case, as a finite difference:

$$[\mathbf{J}]_{ij} = \frac{r_i(\mathbf{z} + \tau \mathbf{e}_j) - r_i(\mathbf{z})}{\tau}, \quad (14.60)$$

where \mathbf{e}_j is the basis vector corresponding to the j -th parameter in \mathbf{z} . Here, τ is a scalar parameter that must be chosen sufficiently small in order to make the approximation sufficiently accurate.

Most software tools for non-linear least squares optimization support both alternatives, and the user must decide which one to use. The approach of providing analytic expressions for \mathbf{J} requires a one-shot effort for formulating and implementing these expressions. Often this is a feasible task, and can sometimes even be automated to some extent by means of symbolic differentiation in mathematical software packages⁸, but for some residuals of interest the resulting expressions can become very complicated to implement, in some cases even impossible. Compared to the approximation in Equation (14.60), the upside of the symbolic approach is that we obtain an exact Jacobian which, in turn, means that the refinement step \mathbf{h} is better optimized relative to the cost function f . In the end, therefore, explicit expressions for \mathbf{J} in general lead to fewer iterations compared to the approach of approximating \mathbf{J} , but not necessarily to a better final solution.

A reasonable strategy is to start with numerical approximations of \mathbf{J} , if this is supported by the software. Should this way of computing \mathbf{J} turn out to be too slow for practical use, it is then reasonable to spend the additional efforts to determine the analytic expressions for \mathbf{J} and to implement them in software.

14.6.2 Sparse Jacobian

The computational cost of minimizing the error function depends to a large extent to the size of \mathbf{J} . But even when \mathbf{J} becomes large, it may still be possible to effectively minimize ε , e.g., using the Levenberg-Marquardt method. This assumes that the Jacobian is sparse, i.e., many or most of its elements are zero. A sparse Jacobian cannot be expected for general estimation problem, only specific optimization problems produce a sparse \mathbf{J} . More precisely, \mathbf{J} is sparse when the residual vector \mathbf{r} is formulated such that each element depends only on a limited subset of the parameters in \mathbf{z} . In this case, many of the elements of \mathbf{J} , derivatives \mathbf{r} with respect to \mathbf{z} , vanish. For some estimation problems in geometry this sparsity appears in a natural way. An example is the bundle adjustment problem described in Chapter 21, where each elements of \mathbf{r} measures the reprojection error of a specific 3D point in a specific image.

⁸For example, using Mathematica or Maple.

The sparsity of \mathbf{J} can have several positive effects on the computation. For one thing, the positions of the zero elements in \mathbf{J} are fixed for any specific minimization problem, and only the “non-zero” elements need to be computed. For example, the numerical approximation of \mathbf{J} described in Section 14.6.1 only need to be applied to the non-zero elements of \mathbf{J} , the others can immediately be set equal to zero. Some software packages for non-linear least squares optimization support a *Jacobian mask*, a binary matrix of the same size as \mathbf{J} , provided by the user. In this mask, a “1” means that the corresponding element in \mathbf{J} must be evaluated and a “0” means that the element in \mathbf{J} is zero.

Another positive effect is that only the “non-zero” elements need to be stored in memory. This can save lots of memory on a hardware platform where memory is a limited resource, but also adds some overhead for the administration of sparse matrices. How much to gain from this observation is therefore highly dependent on the application and how it is implemented.

A more relevant aspect of the sparsity in \mathbf{J} is that it can bring both sparsity and structure also to $\mathbf{J}^\top \mathbf{J}$, the Hessian approximation. Since that matrix is involved in the computation of refinement \mathbf{h} , the corresponding computations can be significantly simplified, and an efficient implementation of the optimization of ε can be achieved.

An example of this simplification is described Section 21.5.6. There it is used in combination with the fact that the Hessian approximation has a block structure, leading to the so-called Schur complement trick. This trick leads to massive simplifications in the computations that are necessary to determine the refinement step, and makes it possible to efficiently solve very large optimization problems. However, this trick also assumes sparsity and structure in \mathbf{J} that only certain estimation problems have.

Chapter 15

Estimation Involving Rotations

Before you read this chapter, you should have thorough understanding of the different types of representations that can be used for 3D rotations, presented in Chapter 11. It may also be helpful to read Chapter 17 and [54] Chapter 9 on general techniques for non-linear optimization.

In this chapter we look at estimation problems with a common theme that some of the parameters to be estimated represent a rotation. In principle, these estimation problems can be solved by standard non-linear techniques, e.g., as described in Chapter 17, given that a suitable parameterization of the rotation is chosen. As we will see in Sections 15.1, 15.2 and 15.4, there are certain estimation problems that can be solved with dedicated methods, without iterative approaches, based on either algebraic or geometric errors. A third class of problems must indeed be solved by iterative non-linear methods. In this case, then the parameterization of the rotation must be done with care to avoid problems. A discussion on useful parameterizations of rotations in the context of non-linear estimation is included in Section 15.3.

The class of rotations that have the broadest applications in geometry is 3D rotations: $SO(3)$. As a consequence, the methods described in this chapter are using $SO(3)$ as the target class of rotations. If needed, several of the problems discussed here, as well as their solutions, can be generalized in a straightforward way to rotations in \mathbb{R}^n : $SO(n)$.

An introductory example

Let us consider the problem of estimating a 2D-rotation from known image points. We have two sets of n vectors, $\{\bar{\mathbf{y}}_k\}$ and $\{\bar{\mathbf{y}}'_k\}$ for $k = 1, \dots, m$, representing the Cartesian coordinates of a set of m points before and after the rotation \mathbf{R} :

$$\bar{\mathbf{y}}'_k = \mathbf{R}\bar{\mathbf{y}}_k, \quad k = 1, \dots, m. \quad (15.1)$$

Here, $\mathbf{R} \in SO(2)$ is a 2×2 rotation matrix, Equation (4.3), and the vectors $\bar{\mathbf{y}}'_k$ and $\bar{\mathbf{y}}_k$ have elements given by

$$\bar{\mathbf{y}}'_k = \begin{pmatrix} u'_k \\ v'_k \end{pmatrix}, \quad \bar{\mathbf{y}}_k = \begin{pmatrix} u_k \\ v_k \end{pmatrix}, \quad (15.2)$$

holding the coordinates of the points relative to some specific coordinate system. As usual, the coordinates are perturbed by measurement noise which means that we cannot expect a perfect match between these coordinates and the rotation. Consequently, there is a difference between the left-hand and right-hand sides in Equation (15.1), and the estimation implies to minimize this difference over all k . As is described in Chapter 12, we can formulate a measure of this difference in various ways, e.g., based on algebraic or geometric errors.

One approach to the estimation of \mathbf{R} is to reformulate Equation (15.1) in terms of homogeneous coordinates of the points, combined with DLT:

$$\mathbf{0} = [\mathbf{y}'_k]_{\times} \mathbf{T} \mathbf{y}_k. \quad (15.3)$$

Here, \mathbf{T} is a 3×3 transformation matrix that represents the rotation \mathbf{R} , in accordance with Equation (4.5). This equation provides two constraints on \mathbf{T} for each pair of points, which means that from $m \geq 4$ pairs, we can formulate

a $2m \times 9$ data matrix \mathbf{A} such that a vectorization of \mathbf{T} lies in its null space. In principle, then, we can estimate \mathbf{T} using the standard linear methods described in Chapter 12 that minimize the algebraic error corresponding to Equation (15.3).

Solving Equation (15.3) based on linear methods, however, does not take into account that \mathbf{T} should have the form described in Equation (4.5), in particular that the upper left submatrix is a scaled rotation matrix. Although this can be “fixed” by making a suitable adjustment to this matrix, it would be better if we can determine a rotation directly from the known data, without making adjustments at the end. In the following sections, we will see how this and other estimation problems instead can be solved by dedicated methods, specially adapted for the case that the estimated variable is a rotation.

15.1 Estimation of 3D rotations

Let $\{\bar{\mathbf{x}}_k = (x_{1k}, x_{2k}, x_{3k})\}$ and $\{\bar{\mathbf{x}}'_k = (x'_{1k}, x'_{2k}, x'_{3k})\}$ be two sets of n corresponding 3D points, before and after a 3D rotation:

$$\bar{\mathbf{x}}'_k = \mathbf{R} \bar{\mathbf{x}}_k, \quad k = 1, \dots, m. \quad (15.4)$$

where $\mathbf{R} \in SO(3)$. Instead of formulating an algebraic error, let us formulate a geometric error and see what happens. In general, estimation based on geometric errors leads to non-linear solution techniques, but we will soon see that this is not necessary the case when rotations are involved.

A geometric error can, for example, be formulated as the L_2 -error

$$\varepsilon_{ROT} = \sum_{k=1}^m \|\bar{\mathbf{x}}'_k - \mathbf{R} \bar{\mathbf{x}}_k\|^2. \quad (15.5)$$

This formulation has the advantage that it is symmetric relative to the two sets of points:

$$\varepsilon_{ROT} = \sum_{k=1}^m \left\| \mathbf{R} \left(\mathbf{R}^\top \bar{\mathbf{x}}'_k - \bar{\mathbf{x}}_k \right) \right\|^2 = \sum_{k=1}^m \left\| \mathbf{R}^{-1} \bar{\mathbf{x}}'_k - \bar{\mathbf{x}}_k \right\|^2. \quad (15.6)$$

This formulation of the error has also the advantage of leading to efficient methods for determining $\mathbf{R} \in SO(3)$ that minimizes ε_{ROT} . In fact, there are several options available depending on how \mathbf{R} is parameterized.

The practical applications for this estimation problem is not limited to geometry in computer vision, and it has been formulated and solved by several authors, sometimes using different techniques. For example, it corresponds to the so-called *Wahba’s problem*, formulated for determining the orientation of a space craft [64]. It has also been addressed by Kabsch in the context of crystallography in physics [41, 42].

15.1.1 Using OPP

One approach to the minimization of ε_{ROT} is based on the solution of the orthogonal Procrustes problem (OPP), described in Toolbox Section 8.2.5. In fact, OPP is formulated almost exactly as the problem of minimizing ε_{ROT} . The only difference is that OPP in general produces $\mathbf{R} \in O(3)$: it guarantees that \mathbf{R} is an orthogonal transformation, but not necessary that $\det(\mathbf{R}) = 1$. Thus, the estimated \mathbf{R} may be a reflection when $\det(\mathbf{R}) = -1$, unless special precautions are made. Despite this observation, OPP works fine for most practical situations, when a moderate amount of noise affects the point coordinates, and the resulting rotation has positive determinant.

Strict version

$O(3)$ consists of two separate components¹ in the space of 3×3 matrices \mathbf{R} : one corresponding to $\det(\mathbf{R}) = +1$ and one corresponding to $\det(\mathbf{R}) = -1$. If there is no noise, and the two data sets in fact are related by a rotation, the above approach returns this rotation, as an element of the first component, $SO(3)$. Adding small amounts of noise to the data typically means that we get an estimate of \mathbf{R} that lies close to the correct rotation, but still in $SO(3)$. Only when large amounts of noise affect the data does it become necessary to include also a reflection in \mathbf{R} to align the two sets with a minimal error.

¹See Toolbox Section 3.3.4.3 for a discussion on the two components of $O(n)$.

Algorithm 15.1: Strict version of OPP.

Input: Two sets of corresponding points: $\{\bar{\mathbf{x}}_k\}$ and $\{\bar{\mathbf{x}}'_k\}$ that approximately satisfy Equation (15.4) for rotation \mathbf{R} .

Output: $\mathbf{R} \in SO(3)$, minimizing ϵ_{ROT} in Equation (15.5), alternatively no solution is given

- 1 Initialize: \mathbf{A} = matrix that holds $\bar{\mathbf{x}}_k$ in column k , \mathbf{B} = matrix that holds $\bar{\mathbf{x}}'_k$ in column k
- 2 SVD of $\mathbf{A}\mathbf{B}^\top = \mathbf{U}\mathbf{S}\mathbf{V}^\top$
- 3 Set: $\mathbf{R} = \mathbf{V}\mathbf{U}^\top$
- 4 **if** $\det(\mathbf{R}) = 1$ **then**
- 5 Return \mathbf{R}
- 6 **else**
- 7 Return no solution
- 8 **end**

This implies that OPP produces \mathbf{R} with negative determinant only when significant amount of noise perturbs the data. If we get an estimate \mathbf{R} with negative determinant, it may as well be discarded if $\mathbf{R} \in SO(3)$ is a requirement of the solution. In this case, we should be humble and say that OPP does not provide useful solutions for all data. This is in line with observation 12.7 on page 199, which encourages us to mind the residual error and pay little attention to estimates that correspond to large residual errors, since they are significantly affected by the noise. In the case of estimating a rotation, we should also pay little attention to rotations that must include reflections in order to minimize the error. A modification of OPP that takes this observation into account is described in Algorithm 15.1.

15.1.2 Using SOPP

In the case we insist that the estimation should produce $\mathbf{R} \in SO(3)$, even for significant levels of noise, it is necessary to modify OPP. The result can be described as the *special orthogonal Procrustes problem* or *SOPP*, special in the sense that it guarantees to produce $\mathbf{R} \in SO(3)$ that minimizes ϵ_{ROT} for all input data.

The derivation of SOPP is detailed in [26], and is outside the scope of this presentation. Intuitively, we can think of it as follows. In the case that the estimated \mathbf{R} has determinant = +1, everything is fine and no additional measures need to be made. In the case that the determinant is = -1, we want to make an adjustment on \mathbf{R} , to \mathbf{R}' where $\det(\mathbf{R}') = +1$, such that $\|\mathbf{R} - \mathbf{R}'\|_F$ is minimal, a *constraint enforcement*. A thorough analysis of this problem leads to a construction of \mathbf{R}' by first applying a special² SVD on $\mathbf{A}\mathbf{B}^\top$:

$$\mathbf{U}\mathbf{S}\mathbf{V}^\top = \mathbf{A}\mathbf{B}^\top, \quad (15.7)$$

where $\mathbf{U}, \mathbf{V} \in SO(3)$ rather than $\mathbf{U}, \mathbf{V} \in O(3)$. If necessary, this requirement is assured by changing the sign of the smallest singular value in \mathbf{S} and of the corresponding row or column in \mathbf{U} or \mathbf{V} , from an initial SVD. This sign change is only necessary when the determinants of these initial \mathbf{U} and \mathbf{V} have different signs, which effectively is the cause of $\det(\mathbf{R}) = -1$. Once \mathbf{U} and \mathbf{V} are determined in this way, $\mathbf{R}' = \mathbf{V}\mathbf{U}^\top$ as before. The details of SOPP are described in Algorithm 15.2, which uses a standard SVD.

SOPP should be used with some care. If applied, the constraint enforcement step which adjusts \mathbf{R} to be in $SO(3)$ also increases the residual error. When this happens, even if the resulting \mathbf{R} lies in $SO(3)$, the corresponding value of ϵ_{ROT} may be so large that \mathbf{R} in practice is useless.

15.1.3 Using quaternions

The estimation of \mathbf{R} that minimizes ϵ_{ROT} in Equation (15.5) can also be based on the quaternion representation of \mathbf{R} , described in Section 11.4.3, using a method described by Horn [36]. As a preliminary, we rewrite ϵ_{ROT} as

$$\epsilon_{ROT} = \sum_{k=1}^m (\bar{\mathbf{x}}'_k - \mathbf{R}\bar{\mathbf{x}}_k)^\top (\bar{\mathbf{x}}'_k - \mathbf{R}\bar{\mathbf{x}}_k) = \sum_{k=1}^m \bar{\mathbf{x}}'^\top_k \bar{\mathbf{x}}'_k - \bar{\mathbf{x}}_k^\top \mathbf{R}^\top \bar{\mathbf{x}}'_k - \bar{\mathbf{x}}'^\top_k \mathbf{R} \bar{\mathbf{x}}_k + \bar{\mathbf{x}}_k^\top \bar{\mathbf{x}}_k = \sum_{k=1}^m \|\bar{\mathbf{x}}'_k\|^2 + \|\bar{\mathbf{x}}_k\|^2 - 2\bar{\mathbf{x}}_k^\top \mathbf{R} \bar{\mathbf{x}}_k. \quad (15.8)$$

²SVD is described in Toolbox Section 8.2.7.

Algorithm 15.2: Solving the special orthogonal Procrustes problem (SOPP).

Input: Two sets of corresponding points: $\{\bar{\mathbf{x}}_k\}$ and $\{\bar{\mathbf{x}}'_k\}$ that approximately satisfy Equation (15.4) for rotation \mathbf{R} .

Output: $\mathbf{R} \in SO(3)$, minimizing ϵ_{ROT} in Equation (15.5)

- 1 Initialize: \mathbf{A} = matrix that holds $\bar{\mathbf{x}}_k$ in column k , \mathbf{B} = matrix that holds $\bar{\mathbf{x}}'_k$ in column k
- 2 SVD of $\mathbf{A}\mathbf{B}^\top = \mathbf{U}\Sigma\mathbf{V}^\top$
- 3 Set: $\tau = \text{sign}(\det(\mathbf{U})\det(\mathbf{V}))$, $\Sigma = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \tau \end{pmatrix}$
- 4 Return: $\mathbf{R} = \mathbf{V}\Sigma\mathbf{U}^\top$

This expression implies that minimizing ϵ_{ROT} is the same as maximizing ϵ'_{ROT} , defined as

$$\epsilon'_{ROT} = \sum_{k=1}^m \bar{\mathbf{x}}_k'^\top \mathbf{R} \bar{\mathbf{x}}_k. \quad (15.9)$$

In accordance with Equation (11.60), each element of \mathbf{R} is a quadratic expression in the elements of a unit quaternion that we represent as normalized vector $\mathbf{q} = (q_1, q_2, q_3, q_4) \in S^3$. Each term of the right-hand side in Equation (15.9) is then a quadratic expression in the elements of \mathbf{q} , an expression that also includes the elements of $\bar{\mathbf{x}}'_k$ and $\bar{\mathbf{x}}_k$, as follows:

$$\epsilon'_{ROT} = \sum_{k=1}^m \mathbf{q}^\top \mathbf{A}_k \mathbf{q} = \mathbf{q}^\top \underbrace{\left(\sum_{k=1}^m \mathbf{A}_k \right)}_{:=\mathbf{A}} \mathbf{q} = \mathbf{q}^\top \mathbf{A} \mathbf{q}. \quad (15.10)$$

Maximizing³ ϵ'_{ROT} over normalized vectors $\mathbf{q} \in S^3$ means that \mathbf{q} should be chosen as a normalized eigenvector corresponding to the largest eigenvalue of the 4×4 symmetric matrix \mathbf{A} , given as

$$\mathbf{A} = \sum_{k=1}^m \mathbf{A}_k, \quad \text{where } \mathbf{A}_k = \begin{pmatrix} x_{1k}x'_{1k} + x_{2k}x'_{2k} + x_{3k}x'_{3k} & x_{2k}x'_{3k} - x_{3k}x'_{2k} & x_{3k}x'_{1k} - x_{1k}x'_{3k} & x_{1k}x'_{2k} - x_{2k}x'_{1k} \\ x_{2k}x'_{3k} - x_{3k}x'_{2k} & x_{1k}x'_{1k} - x_{2k}x'_{2k} - x_{3k}x'_{3k} & x_{1k}x'_{2k} + x_{2k}x'_{1k} & x_{1k}x'_{3k} + x_{3k}x'_{1k} \\ x_{3k}x'_{1k} - x_{1k}x'_{3k} & x_{2k}x'_{1k} + x_{1k}x'_{2k} & x_{2k}x'_{2k} - x_{1k}x'_{1k} - x_{3k}x'_{3k} & x_{3k}x'_{2k} - x_{2k}x'_{3k} \\ x_{1k}x'_{2k} - x_{2k}x'_{1k} & x_{3k}x'_{1k} + x_{1k}x'_{3k} & x_{3k}x'_{2k} + x_{2k}x'_{3k} & x_{3k}x'_{3k} - x_{1k}x'_{1k} - x_{2k}x'_{2k} \end{pmatrix}. \quad (15.11)$$

This method for estimation of 3D rotations is described in Algorithm 15.3.

Algorithm 15.3: Estimation of 3D rotations based on quaternions.

Input: Two sets of corresponding points: $\{\bar{\mathbf{x}}_k\}$ and $\{\bar{\mathbf{x}}'_k\}$ that approximately satisfy Equation (15.4) for rotation \mathbf{R} .

Output: $\mathbf{R} \in SO(3)$, minimizing ϵ_{ROT} in Equation (15.5)

- 1 Initialize: $\mathbf{A} = \mathbf{0}$
- 2 **foreach** $k = 1, \dots, m$ **do**
- 3 Compute \mathbf{A}_k in accordance with Equation (15.11)
- 4 $\mathbf{A} = \mathbf{A} + \mathbf{A}_k$
- 5 **end**
- 6 Determine \mathbf{q} = a normalized eigenvector of \mathbf{A} corresponding to the largest eigenvalue
- 7 Form \mathbf{R} from \mathbf{q} using Equation (11.60)

³Finding the maximizer of a quadratic forms is discussed in Toolbox Section 4.4.4.

15.1.4 OK, so which algorithm do I choose?

Given the three variants of how to estimate a 3D rotation presented above, the obvious question is: which one should be chosen? All three variants find a rotation \mathbf{R} that minimizes the same ε_{ROT} , Equation (15.5), i.e., in general they produce the same \mathbf{R} for a given dataset. This means that the choice of algorithm is mainly guided by implementation aspects and to some extent also by taste. The first two algorithms are based on an SVD of general 3×3 matrices, while the last algorithm is based on an EVD of a symmetric 4×4 matrix. SVD and EVD have similar computational complexity, so the choice is mainly dependent on taste and whether or not implementations of SVD or EVD exist for the intended platform.

As for the two OPP-based variants, the first one should be sufficient for most applications, when moderate levels of noise affect the data. Occasionally, it produces an \mathbf{R} with negative determinant, but this happens typically only when there is significant amount of noise and, then, the result of the estimation may be of less use even if $\mathbf{R} \in SO(3)$ can be provided. Algorithm 15.2 should only be considered in the case that the noise is significant and $\det(\mathbf{R}) = 1$ is a strict requirement.

15.2 Estimation of rigid transformations

Let $\bar{\mathbf{a}}_k$ and $\bar{\mathbf{b}}_k$ be the Cartesian coordinates of two sets of m corresponding points in \mathbb{R}^3 , before and after a rigid transformation:

$$\bar{\mathbf{b}}_k = \mathbf{R}\bar{\mathbf{a}}_k + \bar{\mathbf{t}}, \quad k = 1, \dots, m, \quad (15.12)$$

as is illustrated in Figure 15.1. Due to noise in the coordinates, however, it may not be possible to perfectly fit a rigid transformation between the two sets of points. For a specific choice of rotation $\mathbf{R} \in SO(3)$ and translation $\bar{\mathbf{t}} \in \mathbb{R}^3$, we define an error function based on geometric distances as

$$\varepsilon_{\text{RIGID}} = \sum_{k=1}^m \|\mathbf{R}\bar{\mathbf{a}}_k + \bar{\mathbf{t}} - \bar{\mathbf{b}}_k\|^2. \quad (15.13)$$

For given sets of corresponding points, we want to determine \mathbf{R} and $\bar{\mathbf{t}}$ that minimize $\varepsilon_{\text{RIGID}}$.

This estimation problem appears in practical applications. For example, if the point denoted as $\bar{\mathbf{a}}_k$ refers to a fixed world coordinate system, and the one denoted as $\bar{\mathbf{b}}_k$ is the same point but referring to a camera centered coordinate system, then \mathbf{R} and $\bar{\mathbf{t}}$ describe the transformation from the world coordinate system to the camera centered coordinate system, i.e., they correspond to the rigid 3D transformation described in Section 8.2.1. In this case, minimizing $\varepsilon_{\text{RIGID}}$ is a way to determine the external parameters of the camera, in the literature sometimes referred to as the *exterior orientation* or the camera. In applications where a robot arm is used together with a camera, where the camera either is attached to the arm, or is stationary and observing the robot, this estimation problem is related to the *hand-eye calibration* problem where we want to determine the rigid transformation from the gripper of the robot arm to the camera. We assume that the correspondence problem between the two sets of points is solved, something that is non-trivial in the general case. The following solution to this estimation problem is based on Horn [36].

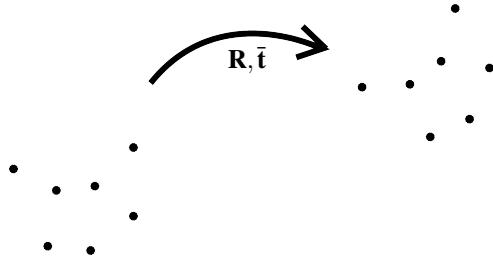


Figure 15.1: A set of points (left) is rigidly transformed to a second set (right). Given that both sets are known in terms of coordinates and correspondences, we can determine the rigid transformation.

Algorithm 15.4: Estimation of 3D rigid transformations.

- Input:** Two sets of m corresponding points: $\{\bar{\mathbf{a}}_k\}$ and $\{\bar{\mathbf{b}}_k\}$ that approximately satisfy Equation (15.12) for translation $\bar{\mathbf{t}}$ and rotation \mathbf{R} .
- Output:** $\mathbf{R} \in SO(3)$ and $\bar{\mathbf{t}} \in \mathbb{R}^3$, minimizing $\varepsilon_{\text{RIGID}}$ in Equation (15.13).
- 1 Compute the centroids of the two data sets: $\bar{\mathbf{a}}_0 = \frac{1}{m} \sum_{k=1}^m \bar{\mathbf{a}}_k$, $\bar{\mathbf{b}}_0 = \frac{1}{m} \sum_{k=1}^m \bar{\mathbf{b}}_k$
 - 2 Compute barycentric coordinates: $\bar{\mathbf{a}}'_k = \bar{\mathbf{a}}_k - \bar{\mathbf{a}}_0$, $\bar{\mathbf{b}}'_k = \bar{\mathbf{b}}_k - \bar{\mathbf{b}}_0$
 - 3 Determine \mathbf{R} from the barycentric coordinates, using any of the algorithms in Section 15.1
 - 4 Set $\bar{\mathbf{t}} = \bar{\mathbf{b}}_0 - \mathbf{R}\bar{\mathbf{a}}_0$

To solve the estimation problem, i.e., to minimize $\varepsilon_{\text{RIGID}}$ over rotations $\mathbf{R} \in SO(3)$ and translations $\bar{\mathbf{t}} \in \mathbb{R}^3$, we rewrite the error as

$$\begin{aligned} \varepsilon_{\text{RIGID}} &= \sum_{k=1}^m \| \mathbf{R}\bar{\mathbf{a}}_k + \bar{\mathbf{t}} - \bar{\mathbf{b}}_k \|^2 = \sum_{k=1}^m (\mathbf{R}\bar{\mathbf{a}}_k + \bar{\mathbf{t}} - \bar{\mathbf{b}}_k)^\top (\mathbf{R}\bar{\mathbf{a}}_k + \bar{\mathbf{t}} - \bar{\mathbf{b}}_k) = \\ &= \sum_{k=1}^m \left(\bar{\mathbf{a}}_k^\top \bar{\mathbf{a}}_k - 2(\bar{\mathbf{b}}_k - \bar{\mathbf{t}})^\top \mathbf{R}\bar{\mathbf{a}}_k + (\bar{\mathbf{b}}_k - \bar{\mathbf{t}})^\top (\bar{\mathbf{b}}_k - \bar{\mathbf{t}}) \right). \end{aligned} \quad (15.14)$$

The error $\varepsilon_{\text{RIGID}}$ assumes a minimal value when its gradient with respect to $\bar{\mathbf{t}}$ vanishes:

$$\nabla_{\bar{\mathbf{t}}} \varepsilon_{\text{RIGID}} = \sum_{k=1}^m (-2(\bar{\mathbf{b}}_k - \bar{\mathbf{t}}) + 2\mathbf{R}\bar{\mathbf{a}}_k) = 2m\bar{\mathbf{t}} + 2 \sum_{k=1}^m (\mathbf{R}\bar{\mathbf{a}}_k - \bar{\mathbf{b}}_k) = \mathbf{0}, \quad (15.15)$$

leading to

$$\bar{\mathbf{t}} = \frac{1}{m} \sum_{k=1}^m (\bar{\mathbf{b}}_k - \mathbf{R}\bar{\mathbf{a}}_k) = \bar{\mathbf{b}}_0 - \mathbf{R}\bar{\mathbf{a}}_0. \quad (15.16)$$

Here, $\bar{\mathbf{a}}_0$ and $\bar{\mathbf{b}}_0$ are the centroids for each of the two point-sets:

$$\bar{\mathbf{a}}_0 = \frac{1}{m} \sum_{k=1}^m \bar{\mathbf{a}}_k, \quad \bar{\mathbf{b}}_0 = \frac{1}{m} \sum_{k=1}^m \bar{\mathbf{b}}_k. \quad (15.17)$$

We insert this $\bar{\mathbf{t}}$ into $\varepsilon_{\text{RIGID}}$ in Equation (15.14) to get

$$\varepsilon_{\text{RIGID}} = \sum_{k=1}^m \| \bar{\mathbf{b}}_k - \bar{\mathbf{b}}_0 - \mathbf{R}(\bar{\mathbf{a}}_k - \bar{\mathbf{a}}_0) \|^2. \quad (15.18)$$

By subtracting the centroids from the respective point-set, we produce coordinates relative to coordinate systems that have their origins at the centroid (or center of gravity), so-called *barycentric coordinates*:

$$\bar{\mathbf{a}}'_k = \bar{\mathbf{a}}_k - \bar{\mathbf{a}}_0, \quad \bar{\mathbf{b}}'_k = \bar{\mathbf{b}}_k - \bar{\mathbf{b}}_0. \quad (15.19)$$

The error function can now be formulated as

$$\varepsilon_{\text{RIGID}} = \sum_{k=1}^m \| \bar{\mathbf{b}}'_k - \mathbf{R}\bar{\mathbf{a}}'_k \|^2. \quad (15.20)$$

This formulation of $\varepsilon_{\text{RIGID}}$ is equivalent to that of ε_{ROT} in Equation (15.5). Consequently, we can use any of the methods described in Section 15.1 for the estimation of the rotation \mathbf{R} . Once \mathbf{R} is determined, $\bar{\mathbf{t}}$ is obtained from Equation (15.16). We summarize this approach to estimation of rigid transformation in Algorithm 15.4.

15.2.1 Iterative Closest Point (ICP)

The method for finding a rigid transformation described above is based on the assumption that we have two sets of points, before and after the transformation is applied, with equally many points and with known correspondences.

Algorithm 15.5: The Iterative Closest Point algorithm (ICP).

```

Input: A set of  $m$  3D points:  $\{\bar{\mathbf{a}}_k\}$  in Cartesian coordinates.
Input: A set of  $m'$  3D points:  $\{\bar{\mathbf{b}}_l\}$  in Cartesian coordinates. In general:  $m \neq m'$  and no correspondences.
Input: An initial guess  $(\mathbf{R}_0, \bar{\mathbf{t}}_0)$  of the rigid transformation. Can be  $(\mathbf{R}_0, \bar{\mathbf{t}}_0) = (\mathbf{I}, \mathbf{0})$ 
Input:  $i_{max}$ , a maximum number of iterations.
Output: Correspondences  $l_k$ , such that  $\bar{\mathbf{a}}_k$  corresponds to  $\bar{\mathbf{b}}_{l_k}$ 
Output: A rigid transformation  $(\mathbf{R}, \bar{\mathbf{t}})$  that makes  $\bar{\mathbf{b}}_{l_k} \approx \mathbf{R}\bar{\mathbf{a}}_k + \bar{\mathbf{t}}$  for  $k = 1, \dots, m$ .
1 Initialize:  $i = 0$ ,  $\varepsilon_0$  = a very large value,  $\mathbf{R} = \mathbf{R}_0$ ,  $\bar{\mathbf{t}} = \bar{\mathbf{t}}_0$ 
2 repeat
3   Set  $i = i + 1$ 
4   foreach  $k = 1, \dots, m$  do
5     Transform:  $\bar{\mathbf{b}}'_k = \mathbf{R}\bar{\mathbf{a}}_k + \bar{\mathbf{t}}$ 
6     Find index  $l_k$  of the point  $\bar{\mathbf{b}}_{l_k}$  that minimizes  $\|\bar{\mathbf{b}}_{l_k} - \bar{\mathbf{b}}'_k\|$ 
7   end
8   /* The sets  $\{\bar{\mathbf{a}}_k\}$  and  $\{\bar{\mathbf{b}}_{l_k}\}$  now contain  $m$  “corresponding” points */
9   Estimate the rigid transformation  $(\mathbf{R}, \bar{\mathbf{t}})$  between the two sets  $\{\bar{\mathbf{a}}_k\}$  and  $\{\bar{\mathbf{b}}_{l_k}\}$ ,
10  together with the residual error, denoted  $\varepsilon_i$ , using Algorithm 15.4 on page 264
11 until  $\varepsilon_i \geq \varepsilon_{i-1}$  or  $i > i_{max}$ 
12 Return rigid transformation  $(\mathbf{R}, \bar{\mathbf{t}})$  and correspondences  $\{l_k\}$  from last iteration

```

In practice, the last two assumptions are not always valid. There are many applications that produce two point-sets, but there are initially no known correspondences between the points in the two sets. In general, they can also have different numbers of points, which means that there can be some points in one or both of the two sets that do not have corresponding points in the other set. For example, if the points in the two sets have been determined by two different sensors or different methods it is likely that some points in one set do not appear in the other set. If, in addition, the coordinate systems of the two sets are different, a rigid transformation must be applied to align the two sets.

In the general case, it may not even be possible to put the points into correspondence even if the correct rigid transformation is applied. The two sets could simply represent some features that lie in a partly overlapping region on a common 3D surface, or 2D curve, defined relative to two different coordinate systems. An extension of this idea is that only one set contains points while the other instead is a parametric representation of the surface that contains the points, where the two are defined relative to two different coordinate systems that can be aligned with a rigid transformation. The problem of finding this transformation is often referred to as *registration*.

The fact that there are no initial correspondences known between points in the two sets implies that the estimation problem is non-trivial in the general case, and it may have to be addressed using the robust estimation methods described in Chapter 17. In principle, this type of estimation problem implies that we need to find the correspondences during the course of the estimation, in addition to the rigid transformation. Based on the simplifying assumption that the two sets have a large group of points in common, however, the problem of finding $(\mathbf{R}, \bar{\mathbf{t}})$ together with the correspondences can be solved by the so-called *iterative closest point* algorithm, or *ICP* for short. A basic form of the algorithm, that can be applied to two point-sets, is outlined in Algorithm 15.5. It combines the method for determining a rigid transformation described previously with an iterative approach for finding the correct correspondences.

The first version of ICP was adapted to registration of range images [11], and it was adopted to more general problems and coined ICP in [6]. As for any method that tries to solve the combined problem of finding both correspondences and a model that fits the data, ICP can fail. For example, it may get stuck in a local optimum, where incorrect correspondences have been chosen between points in the two sets. It also has a lack of symmetry between the two set of points: it may not produce the inverse rigid transformation if the two point-sets are interchanged. These, and other problems, can to some extent be reduced by using various extensions of the original ICP algorithm. For an overview of ICP algorithms, see [58].

15.3 Non-linear least squares estimation involving rotations

In many geometric estimation problems, the error function cannot be solved by linear methods, or simple methods that are dedicated to rotations. Instead, the error can be defined in terms of a residual \mathbf{r} which depends one or several rotations in $SO(3)$. For this discussion, it is sufficient to consider an \mathbf{r} which is a function of a single rotation $\mathbf{R} \in SO(3)$. Initially, we can then define the free parameters of the estimation problem as $\tilde{\mathbf{z}} \in \mathbb{R}^9$, a vectorization of \mathbf{R} . This is an inconsistent parameterization, since it does not take the internal constraints related to \mathbf{R} into account. Despite this observation, we stick to $\tilde{\mathbf{z}}$ for the moment and formulate a corresponding Jacobian as

$$\tilde{\mathbf{J}} = \begin{pmatrix} \vdots & \vdots \\ \frac{\partial r_k}{\partial \tilde{z}_1} & \frac{\partial r_k}{\partial \tilde{z}_2} & \frac{\partial r_k}{\partial \tilde{z}_3} & \frac{\partial r_k}{\partial \tilde{z}_4} & \frac{\partial r_k}{\partial \tilde{z}_5} & \frac{\partial r_k}{\partial \tilde{z}_6} & \frac{\partial r_k}{\partial \tilde{z}_7} & \frac{\partial r_k}{\partial \tilde{z}_8} & \frac{\partial r_k}{\partial \tilde{z}_9} \\ \vdots & \vdots \end{pmatrix}, \quad (15.21)$$

where⁴

$$\tilde{\mathbf{z}} = \begin{pmatrix} \tilde{z}_1 & \tilde{z}_4 & \tilde{z}_7 \\ \tilde{z}_2 & \tilde{z}_5 & \tilde{z}_8 \\ \tilde{z}_3 & \tilde{z}_6 & \tilde{z}_9 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \mathbf{R}. \quad (15.22)$$

In the case that \mathbf{r} is a function of other rotations, or other parameters, there are additional columns in $\tilde{\mathbf{J}}$, but here we focus on what happens with the 12 columns in Equation (15.21).

To avoid the inconsistencies in $\tilde{\mathbf{z}}$, we need to re-parameterize \mathbf{R} to a consistent set of parameters, following the steps outlined in Section 14.2. As new parameters, we can choose one of the alternatives presented in Chapter 11. In the following, we discuss the implications of using some of these parameterizations in the context of non-linear estimation. A practical application of these results is PnP based on geometric estimation, presented in Section 15.4.2.

15.3.1 Quaternions

One common representation of 3D rotations is based on unit quaternions, described in Section 11.4. Although it is not impossible to use a unit quaternion at this point, standard software packages for non-linear optimization often do not support automatic normalization of the free parameters between each iteration. To overcome this issue, we instead use a general quaternion $\mathbf{z} = (z_1, z_2, z_3, z_4)$. Based on Equation (11.60), \mathbf{z} can be mapped to $\tilde{\mathbf{z}} = \mathbf{R} \in SO(3)$ as follows:

$$\tilde{\mathbf{z}}(\mathbf{z}) = \begin{pmatrix} \tilde{z}_1 \\ \tilde{z}_2 \\ \tilde{z}_3 \\ \tilde{z}_4 \\ \tilde{z}_5 \\ \tilde{z}_6 \\ \tilde{z}_7 \\ \tilde{z}_8 \\ \tilde{z}_9 \end{pmatrix} = \begin{pmatrix} r_{11} \\ r_{21} \\ r_{31} \\ r_{12} \\ r_{22} \\ r_{32} \\ r_{13} \\ r_{23} \\ r_{33} \end{pmatrix} = \frac{1}{\|\mathbf{z}\|^2} \begin{pmatrix} z_1^2 + z_2^2 - z_3^2 - z_4^2 \\ 2(z_2 z_3 + z_1 z_4) \\ 2(z_2 z_4 - z_1 z_3) \\ 2(z_2 z_3 - z_1 z_4) \\ z_1^2 - z_2^2 + z_3^2 - z_4^2 \\ 2(z_1 z_2 + z_3 z_4) \\ 2(z_1 z_3 + z_2 z_4) \\ 2(z_3 z_4 - z_1 z_2) \\ z_1^2 - z_2^2 - z_3^2 + z_4^2 \end{pmatrix}. \quad (15.23)$$

Hence, instead of normalizing the quaternion, the normalization is made in the mapping to \mathbf{R} . As long as $\mathbf{z} \neq \mathbf{0}$ this parameterization works fine, and it allows $\mathbf{z} \in \mathbb{R}^4$ to be treated as an unconstrained parameter, although it represents an over-parameterization of $SO(3)$.

Using the results derived in Section 14.2, the Jacobian corresponding to \mathbf{z} , denoted \mathbf{J} , is obtained by first

⁴In the following discussion, we will see $\tilde{\mathbf{z}}$ either as a vector in \mathbb{R}^9 and as a 3×3 matrix, depending on the context.

determining the derivatives of the mapping $\tilde{\mathbf{z}}(\mathbf{z})$, in Equation (14.4):

$$\mathbf{D} = \begin{pmatrix} \frac{\partial \tilde{z}_1}{\partial z_1} & \frac{\partial \tilde{z}_1}{\partial z_2} & \frac{\partial \tilde{z}_1}{\partial z_3} & \frac{\partial \tilde{z}_1}{\partial z_4} \\ \frac{\partial \tilde{z}_2}{\partial z_1} & \frac{\partial \tilde{z}_2}{\partial z_2} & \frac{\partial \tilde{z}_2}{\partial z_3} & \frac{\partial \tilde{z}_2}{\partial z_4} \\ \frac{\partial \tilde{z}_3}{\partial z_1} & \frac{\partial \tilde{z}_3}{\partial z_2} & \frac{\partial \tilde{z}_3}{\partial z_3} & \frac{\partial \tilde{z}_3}{\partial z_4} \\ \frac{\partial \tilde{z}_4}{\partial z_1} & \frac{\partial \tilde{z}_4}{\partial z_2} & \frac{\partial \tilde{z}_4}{\partial z_3} & \frac{\partial \tilde{z}_4}{\partial z_4} \\ \frac{\partial \tilde{z}_5}{\partial z_1} & \frac{\partial \tilde{z}_5}{\partial z_2} & \frac{\partial \tilde{z}_5}{\partial z_3} & \frac{\partial \tilde{z}_5}{\partial z_4} \\ \frac{\partial \tilde{z}_6}{\partial z_1} & \frac{\partial \tilde{z}_6}{\partial z_2} & \frac{\partial \tilde{z}_6}{\partial z_3} & \frac{\partial \tilde{z}_6}{\partial z_4} \\ \frac{\partial \tilde{z}_7}{\partial z_1} & \frac{\partial \tilde{z}_7}{\partial z_2} & \frac{\partial \tilde{z}_7}{\partial z_3} & \frac{\partial \tilde{z}_7}{\partial z_4} \\ \frac{\partial \tilde{z}_8}{\partial z_1} & \frac{\partial \tilde{z}_8}{\partial z_2} & \frac{\partial \tilde{z}_8}{\partial z_3} & \frac{\partial \tilde{z}_8}{\partial z_4} \\ \frac{\partial \tilde{z}_9}{\partial z_1} & \frac{\partial \tilde{z}_9}{\partial z_2} & \frac{\partial \tilde{z}_9}{\partial z_3} & \frac{\partial \tilde{z}_9}{\partial z_4} \end{pmatrix} = \frac{2}{\|\mathbf{z}\|^2} \begin{pmatrix} z_1 & z_2 & -z_3 & -z_4 \\ z_4 & z_3 & z_2 & z_1 \\ -z_3 & z_4 & -z_1 & z_2 \\ -z_4 & z_3 & z_2 & -z_1 \\ z_1 & -z_2 & z_3 & -z_4 \\ z_2 & z_1 & z_4 & z_3 \\ z_3 & z_4 & z_1 & z_2 \\ -z_2 & -z_1 & z_4 & z_3 \\ z_1 & -z_2 & -z_3 & z_4 \end{pmatrix} - 2 \frac{\tilde{\mathbf{z}} \mathbf{z}^\top}{\|\mathbf{z}\|^2}, \quad (15.24)$$

and then compute $\mathbf{J} = \tilde{\mathbf{J}} \mathbf{D}$.

This means that instead of the inconsistent parameterization $\tilde{\mathbf{z}}$, we can use the quaternion \mathbf{z} and map it to $\tilde{\mathbf{z}} = \mathbf{R} \in SO(3)$ using Equation (15.23). Furthermore, to get the corresponding Jacobian, we first calculate $\tilde{\mathbf{J}}$ in Equation (15.21), together with the determinants in \mathbf{D} , Equation (15.24), and finally compute $\mathbf{J} = \tilde{\mathbf{J}} \mathbf{D}$. \mathbf{z} is a consistent parameterization, but it is also an over-parameterization of \mathbf{R} , the resulting Jacobian is rank deficient. More precisely, we have

$$\mathbf{D}^\top \mathbf{D} = \frac{8}{\|\mathbf{z}\|^4} (\|\mathbf{z}\|^2 \mathbf{I} - \mathbf{z} \mathbf{z}^\top), \quad (15.25)$$

which means that $\mathbf{Dz} = 0$, i.e., $\mathbf{Jz} = \mathbf{0}$. This implies that any refinement step \mathbf{h} of a solution \mathbf{z} must be orthogonal to \mathbf{z} . Assuming that these steps are small compared to \mathbf{z} , the iterations of the non-linear optimization will, therefore, not change the norm of \mathbf{z} very much, mainly modifying its direction. Consequently, if the initial solution \mathbf{z}_0 has unit norm, and is close to the minimizer of the cost function, it is very unlikely that the iterations will modify \mathbf{z} close to $\mathbf{0}$ during the course of the optimization.

From Equation (15.25), it also follows that \mathbf{D} has rank 3 for any $\mathbf{z} \neq \mathbf{0}$, which means that the rank of \mathbf{J} is not dependent on \mathbf{z} , it only depends on the data in $\tilde{\mathbf{J}}$. This implies that the quaternion parameterization presented here is safe in the sense that it does not introduce any singularities in the optimization.

15.1 The quaternion representation of rotations works fine for non-linear estimation.

15.3.2 Axis-angle representation

An alternative to using quaternions is to represent $\mathbf{R} \in SO(3)$ in terms of an axis $\hat{\mathbf{n}}$ and an angle α , as described in Section 11.2. In principle, we can represent the rotation by the vector $\alpha \hat{\mathbf{n}} \in \mathbb{R}^3$, as described in Section 11.3, but since both α and $\hat{\mathbf{n}}$ here are made explicit, we refer to this as an axis-angle representation. We will arrive at the final representation in two steps; first by defining a set of parameters that explicitly refers to the axis $\hat{\mathbf{n}}$ and the angle α , and in a second step, these parameters are formed from the vector $\alpha \hat{\mathbf{n}}$.

Let $\hat{\mathbf{z}} \in \mathbb{R}^4$ be a set of parameters defined as

$$\hat{\mathbf{z}} = \begin{pmatrix} \hat{z}_1 \\ \hat{z}_2 \\ \hat{z}_3 \\ \hat{z}_4 \end{pmatrix} = \begin{pmatrix} \alpha \\ n_1 \\ n_2 \\ n_3 \end{pmatrix}, \quad (15.26)$$

where $\hat{\mathbf{n}} = (n_1, n_2, n_3)$ is the rotation axis. We use Rodrigues' formula, Equation (11.12), to express $\tilde{\mathbf{z}} = \mathbf{R}$ in terms

of the axis and angle:

$$\tilde{\mathbf{z}}(\hat{\mathbf{z}}) = \begin{pmatrix} r_{11} \\ r_{21} \\ r_{31} \\ r_{12} \\ r_{22} \\ r_{32} \\ r_{13} \\ r_{23} \\ r_{33} \end{pmatrix} = \begin{pmatrix} n_1^2 + (1 - n_1^2) \cos \alpha & \\ n_1 n_2 (1 - \cos \alpha) + n_3 \sin \alpha & \\ n_1 n_3 (1 - \cos \alpha) - n_2 \sin \alpha & \\ n_1 n_2 (1 - \cos \alpha) - n_3 \sin \alpha & \\ n_2^2 + (1 - n_2^2) \cos \alpha & \\ n_2 n_3 (1 - \cos \alpha) + n_1 \sin \alpha & \\ n_1 n_3 (1 - \cos \alpha) + n_2 \sin \alpha & \\ n_2 n_3 (1 - \cos \alpha) - n_1 \sin \alpha & \\ n_3^2 + (1 - n_3^2) \cos \alpha & \end{pmatrix} \quad (15.27)$$

Thus, Equation (15.27) describes a mapping from any $\hat{\mathbf{z}} \in \mathbb{R}^4$ to $\tilde{\mathbf{z}} = \mathbf{R}$. At this point there is no guarantee that $\hat{\mathbf{n}}$ is normalized without additional restrictions on $\hat{\mathbf{z}}$. Consequently, also $\hat{\mathbf{z}}$ is an inconsistent parameterization and it is in general not the case that $\mathbf{R} \in SO(3)$. This is guaranteed only if $\hat{z}_2^2 + \hat{z}_3^2 + \hat{z}_4^2 = 1$.

To get the Jacobian corresponding to $\hat{\mathbf{z}}$, denoted $\hat{\mathbf{J}}$, we multiply $\hat{\mathbf{J}}$ with \mathbf{D}_1 , which contains the derivatives of $\tilde{\mathbf{z}}$ with respect to the elements in $\hat{\mathbf{z}}$:

$$\mathbf{D}_1 = \begin{pmatrix} \frac{\partial \tilde{z}_1}{\partial \alpha} & \frac{\partial \tilde{z}_1}{\partial n_1} & \frac{\partial \tilde{z}_1}{\partial n_2} & \frac{\partial \tilde{z}_1}{\partial n_3} \\ \frac{\partial \tilde{z}_2}{\partial \alpha} & \frac{\partial \tilde{z}_2}{\partial n_1} & \frac{\partial \tilde{z}_2}{\partial n_2} & \frac{\partial \tilde{z}_2}{\partial n_3} \\ \frac{\partial \tilde{z}_3}{\partial \alpha} & \frac{\partial \tilde{z}_3}{\partial n_1} & \frac{\partial \tilde{z}_3}{\partial n_2} & \frac{\partial \tilde{z}_3}{\partial n_3} \\ \frac{\partial \tilde{z}_4}{\partial \alpha} & \frac{\partial \tilde{z}_4}{\partial n_1} & \frac{\partial \tilde{z}_4}{\partial n_2} & \frac{\partial \tilde{z}_4}{\partial n_3} \\ \frac{\partial \tilde{z}_5}{\partial \alpha} & \frac{\partial \tilde{z}_5}{\partial n_1} & \frac{\partial \tilde{z}_5}{\partial n_2} & \frac{\partial \tilde{z}_5}{\partial n_3} \\ \frac{\partial \tilde{z}_6}{\partial \alpha} & \frac{\partial \tilde{z}_6}{\partial n_1} & \frac{\partial \tilde{z}_6}{\partial n_2} & \frac{\partial \tilde{z}_6}{\partial n_3} \\ \frac{\partial \tilde{z}_7}{\partial \alpha} & \frac{\partial \tilde{z}_7}{\partial n_1} & \frac{\partial \tilde{z}_7}{\partial n_2} & \frac{\partial \tilde{z}_7}{\partial n_3} \\ \frac{\partial \tilde{z}_8}{\partial \alpha} & \frac{\partial \tilde{z}_8}{\partial n_1} & \frac{\partial \tilde{z}_8}{\partial n_2} & \frac{\partial \tilde{z}_8}{\partial n_3} \\ \frac{\partial \tilde{z}_9}{\partial \alpha} & \frac{\partial \tilde{z}_9}{\partial n_1} & \frac{\partial \tilde{z}_9}{\partial n_2} & \frac{\partial \tilde{z}_9}{\partial n_3} \end{pmatrix} = \begin{pmatrix} (n_1^2 - 1) \sin \alpha & 2n_1(1 - \cos \alpha) & 0 & 0 \\ n_1 n_2 \sin \alpha + n_3 \cos \alpha & n_2(1 - \cos \alpha) & n_1(1 - \cos \alpha) & \sin \alpha \\ n_1 n_3 \sin \alpha - n_2 \cos \alpha & n_3(1 - \cos \alpha) & -\sin \alpha & n_1(1 - \cos \alpha) \\ n_1 n_2 \sin \alpha - n_3 \cos \alpha & n_2(1 - \cos \alpha) & n_1(1 - \cos \alpha) & -\sin \alpha \\ (n_2^2 - 1) \sin \alpha & 0 & 2n_2(1 - \cos \alpha) & 0 \\ n_2 n_3 \sin \alpha + n_1 \cos \alpha & \sin \alpha & n_3(1 - \cos \alpha) & n_2(1 - \cos \alpha) \\ n_1 n_3 \sin \alpha + n_2 \cos \alpha & n_3(1 - \cos \alpha) & \sin \alpha & n_1(1 - \cos \alpha) \\ n_2 n_3 \sin \alpha - n_1 \cos \alpha & -\sin \alpha & n_3(1 - \cos \alpha) & n_2(1 - \cos \alpha) \\ (n_3^2 - 1) \sin \alpha & 0 & 0 & 2n_3(1 - \cos \alpha) \end{pmatrix}. \quad (15.28)$$

This results in $\hat{\mathbf{J}} = \tilde{\mathbf{J}} \mathbf{D}_1$. $\alpha = 0$ implies that all three rightmost columns in \mathbf{D}_1 vanish, i.e., \mathbf{D}_1 has rank one in this case, as long as $\hat{\mathbf{n}} \neq 0$. This property propagates to $\hat{\mathbf{J}}$, which becomes extremely rank deficient in this case. This is not the full story, however, as we will see when we continue to the final step, where a consistent set of parameters is defined.

From $\mathbf{z} = (z_1, z_2, z_3) \in \mathbb{R}^3$ we define a set of consistent parameters for $\mathbf{R} \in SO(3)$ in terms of the mapping

$$\tilde{\mathbf{z}} = \begin{pmatrix} \hat{z}_1 \\ \hat{z}_2 \\ \hat{z}_3 \\ \hat{z}_4 \end{pmatrix} = \begin{pmatrix} \|\mathbf{z}\| \\ \frac{z_1}{\|\mathbf{z}\|} \\ \frac{z_2}{\|\mathbf{z}\|} \\ \frac{z_3}{\|\mathbf{z}\|} \end{pmatrix} = \frac{1}{\|\mathbf{z}\|} \begin{pmatrix} \|\mathbf{z}\|^2 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix}. \quad (15.29)$$

This mapping corresponds to $\alpha = \|\mathbf{z}\|$ and $\hat{\mathbf{n}} = \mathbf{z}/\|\mathbf{z}\|$, i.e., we can see \mathbf{z} as the vector \mathbf{m} that is related to the $so(3)$ representation described in Section 11.3.1. However, since we take the route over an explicit representation of the axis and the angle, the $so(3)$ representation is not really used in this case.

To arrive at the Jacobian \mathbf{J} that determines the gradient and Hessian of this parameterization, we need \mathbf{D}_2 :

$$\mathbf{D}_2 = \begin{pmatrix} \frac{\partial \hat{z}_1}{\partial z_1} & \frac{\partial \hat{z}_1}{\partial z_2} & \frac{\partial \hat{z}_1}{\partial z_3} \\ \frac{\partial \hat{z}_2}{\partial z_1} & \frac{\partial \hat{z}_2}{\partial z_2} & \frac{\partial \hat{z}_2}{\partial z_3} \\ \frac{\partial \hat{z}_3}{\partial z_1} & \frac{\partial \hat{z}_3}{\partial z_2} & \frac{\partial \hat{z}_3}{\partial z_3} \\ \frac{\partial \hat{z}_4}{\partial z_1} & \frac{\partial \hat{z}_4}{\partial z_2} & \frac{\partial \hat{z}_4}{\partial z_3} \end{pmatrix} = \begin{pmatrix} \frac{z_1}{\|\mathbf{z}\|} & \frac{z_2}{\|\mathbf{z}\|} & \frac{z_3}{\|\mathbf{z}\|} \\ \frac{z_2^2 + z_3^2}{\|\mathbf{z}\|^3} & \frac{-z_1 z_2}{\|\mathbf{z}\|^3} & \frac{-z_1 z_3}{\|\mathbf{z}\|^3} \\ \frac{-z_1 z_2}{\|\mathbf{z}\|^3} & \frac{z_1^2 + z_3^2}{\|\mathbf{z}\|^3} & \frac{-z_2 z_3}{\|\mathbf{z}\|^3} \\ \frac{-z_1 z_3}{\|\mathbf{z}\|^3} & \frac{-z_2 z_3}{\|\mathbf{z}\|^3} & \frac{z_1^2 + z_2^2}{\|\mathbf{z}\|^3} \end{pmatrix}, \quad (15.30)$$

and compute the Jacobian as $\mathbf{J} = \hat{\mathbf{J}} \mathbf{D}_2 = \tilde{\mathbf{J}} \mathbf{D}_1 \mathbf{D}_2$.

The critical issue is what happens when $\mathbf{z} \rightarrow \mathbf{0}$. We have already seen that \mathbf{D}_1 in this case approaches rank 1, when its last three columns disappear. The opposite happens for \mathbf{D}_1 , where the norms of the last three rows grow without bound when $\mathbf{z} \rightarrow \mathbf{0}$. This raises the question: what happens with the product $\mathbf{D}_1\mathbf{D}_2$ in this case? A closer examination of the two matrices shows that the last three columns of \mathbf{D}_1 have magnitudes that vary linearly with $\|\mathbf{z}\|$ when \mathbf{z} is small, due to the factors $\sin \alpha = \sin \|\mathbf{z}\|$. Furthermore, the last three rows on \mathbf{D}_2 grow proportionally $\|\mathbf{z}\|^{-1}$. In theory, then, $\mathbf{D}_1\mathbf{D}_2$ can be bounded when $\mathbf{z} \rightarrow \mathbf{0}$. In fact, it can be shown that

$$\lim_{\mathbf{z} \rightarrow \mathbf{0}} \mathbf{D}_1\mathbf{D}_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \quad (15.31)$$

In addition to this observation, it should also be clear that although $\hat{\mathbf{n}}$ is undefined when $\mathbf{z} = \mathbf{0}$, this case corresponds to $\mathbf{R} = \mathbf{I}$. Consequently, the mapping from \mathbf{z} to $\tilde{\mathbf{z}} = \tilde{\mathbf{z}}(\tilde{\mathbf{z}}(\mathbf{z}))$ is in fact well-defined for all $\mathbf{z} \in \mathbb{R}^3$, and formally this applies to $\mathbf{J}(\mathbf{z})$ as well. The only glitch is that at $\mathbf{z} = \mathbf{0}$, and in practice close to this point, we must use alternative expressions for these two functions.

A final note In accordance with the above discussion, the singularity of the axis-angle representation at $\mathbf{R} = \mathbf{I}$ may not have to be an issue if the computations are implemented in a careful way, with $\mathbf{z} = \mathbf{0}$ as a special case. As an alternative, the rotation matrix \mathbf{R} can simply be “rotated away” from \mathbf{I} by expressing it as

$$\mathbf{R}(\mathbf{z}) = \mathbf{R}_2(\mathbf{z})\mathbf{R}_1, \quad (15.32)$$

where \mathbf{R}_1 is some arbitrary but fixed matrix, and $\mathbf{R}_2 = \mathbf{R}\mathbf{R}_1^\top$. If the initial solution to \mathbf{R} , e.g., provided by an algebraic estimation, lies close to \mathbf{I} this can be taken as an indication that formulation in Equation (15.32) should be applied. This means that we replace \mathbf{R} by $\mathbf{R}_2\mathbf{R}_1$ and instead optimize over \mathbf{R}_2 . By choosing \mathbf{R}_1 sufficiently far from \mathbf{I} , the rotation \mathbf{R}_2 will never come close to the singularity at \mathbf{I} during the optimization.

15.2 The axis-angle representation of rotations works fine for non-linear optimizations, although it may cause problems for rotations close to \mathbf{I} unless it has been carefully implemented, e.g., using the rewrite in Equation (15.32).

15.3.3 Euler angles

In the literature, it is common to use a representation of $SO(3)$ based on three angles for solving various problems. Although this representation may have attractive properties in other contexts, it is less suitable for non-linear optimization. We consider here the particular three-angle representation referred to as Euler angles, described in Section 11.5, but the result is general and applies to other similar representations. The derivations made for this case are less detailed than above, but they are based on the same steps.

We write \mathbf{R} as the concatenation of three rotation: $\mathbf{R} = \mathbf{R}_3(\alpha_3)\mathbf{R}_2(\alpha_2)\mathbf{R}_1(\alpha_1)$, where $\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3$ are defined in Equation (11.69). A consistent and minimal parameterization of \mathbf{R} is then given by the parameters $\mathbf{z} = (\alpha_1, \alpha_2, \alpha_3) \in \mathbb{R}^3$, the Euler angles. It is straight-forward to show that the corresponding Jacobian \mathbf{J} has the property

$$\det(\mathbf{J}^\top \mathbf{J}) = 8 \sin^2 \alpha_2. \quad (15.33)$$

As is mentioned in Section 11.5.1, the Euler representation has a singularity when $\alpha_2 = 0$, case (b), or $\alpha_2 = \pi$, case (c), since α_1, α_3 are ambiguous in these cases. This singularity is reflected in the fact that \mathbf{J} in these cases is rank deficient. More precisely, when $\sin \alpha_2 = 0$ then \mathbf{J} has a null space spanned by $(1, 0, -1)$. Consequently, close

to this singularity the refinement step \mathbf{h} cannot change α_1 and α_3 in different directions to any large degree, even if this is needed to reduce the cost function ε .

In contrast to the singularity of the axis-angle representation, which appears for one single rotation ($\mathbf{R} = \mathbf{I}$), the singularity of the Euler angles occurs on the two planes $\alpha_2 = 0$ and $\alpha_2 = \pi$, in the 3-dimensional parameter space. This makes the singularity more probable in practical applications, and makes the Euler angle representation less suitable for non-linear estimation.

15.3 Avoid using the Euler angle representation for parameterization of $SO(3)$ in the context of non-linear estimation.

15.4 The Perspective n -Point Problem (PnP)

Another geometric estimation problem that relates to rigid transformations is the so-called *perspective n-point problem* or *PnP* for short. A pinhole camera observes a set of m 3D points with known positions $\{\mathbf{x}_k, k = 1, \dots, m\}$, producing a corresponding set of image points $\{\mathbf{y}_k, k = 1, \dots, m\}$. Both point-sets are known, and we also know the correspondences between 3D points and image points. Another important assumption is that the camera is calibrated such that the image points are C-normalized. This implies that we can write the camera matrix as $\mathbf{C} \sim (\mathbf{R} | \bar{\mathbf{t}})$, where $(\mathbf{R}, \bar{\mathbf{t}})$ represent a rigid transformation relative to the world coordinate system.

The projection from 3D points to image points is expressed as

$$\mathbf{y}_k \sim \mathbf{C} \mathbf{x}_k = (\mathbf{R} | \bar{\mathbf{t}}) \begin{pmatrix} \bar{\mathbf{x}}_k \\ 1 \end{pmatrix} = \mathbf{R} \bar{\mathbf{x}}_k + \bar{\mathbf{t}}, \quad k = 1, \dots, m. \quad (15.34)$$

PnP implies determining $\mathbf{R}, \bar{\mathbf{t}}$ such that Equation (15.34) is satisfied for all $k = 1, \dots, m$. Since we have to assume that both the 3D points and the image points are affected by noise, PnP reduces to an estimation problem. To this end, we need to formulate an error to be minimized over $(\mathbf{R}, \bar{\mathbf{t}})$, either as an algebraic error or as a geometric error.

15.4.1 Algebraic estimation

In the case PnP is solved based on an algebraic error, Equation (15.34) can be reformulated using DLT as

$$\mathbf{0} = [\mathbf{y}_k]_{\times} \mathbf{C} \mathbf{x}_k, \quad k = 1, \dots, m. \quad (15.35)$$

This equation represents 2 linearly independent constraints in the unknown normalized camera matrix \mathbf{C} and, not taking the normalization of the camera into account, it can be estimated from $n \geq 6$ corresponding points using the homogeneous or inhomogeneous method, as is described in Section 13.3. In general, due to the noise that perturbs the dataset, this initial estimate of the camera matrix, denoted \mathbf{C}_0 , does not represent a normalized camera. To obtain a normalized camera, we need to apply a *constraint enforcement* step on \mathbf{C}_0 , described below.

Constraint enforcement

We have an estimate $\mathbf{C}_0 = (\mathbf{A} | \bar{\mathbf{b}})$ of a normalized camera, i.e., $\lambda \mathbf{C}_0 \approx (\mathbf{R} | \bar{\mathbf{t}})$ for $\mathbf{R} \in SO(3), \bar{\mathbf{t}} \in \mathbb{R}^3$, and some $\lambda \in \mathbb{R}$, but these last variables are not yet determined. One way to determine them is to look at the problem of minimizing

$$\varepsilon_{PnP} = \|\lambda(\mathbf{A} | \bar{\mathbf{b}}) - (\mathbf{R} | \bar{\mathbf{t}})\|_F^2 = \|\lambda \mathbf{A} - \mathbf{R}\|_F^2 + \|\lambda \bar{\mathbf{b}} - \bar{\mathbf{t}}\|_F^2, \quad (15.36)$$

for all possible choices of $\mathbf{R} \in SO(3), \bar{\mathbf{t}} \in \mathbb{R}^3$, and $\lambda \in \mathbb{R}$. The last term in ε_{PnP} can always be set = 0 by choosing $\bar{\mathbf{t}} = \lambda \bar{\mathbf{b}}$, a choice that does not affect the first term. Consequently, it remains to minimize only the first term over $\mathbf{R} \in SO(3)$, and $\lambda \in \mathbb{R}$. Based on an SVD of $\tau \mathbf{A} = \mathbf{U} \mathbf{S} \mathbf{V}^\top$, where $\tau = \text{sign}(\det(\mathbf{A}))$, this term can be rewritten as

$$\begin{aligned} \|\lambda \mathbf{A} - \mathbf{R}\|_F^2 &= \|\lambda \underbrace{\tau^2}_{=1} \mathbf{A} - \mathbf{R}\|_F^2 = \|\lambda \tau \underbrace{\tau \mathbf{A}}_{=\mathbf{U} \mathbf{S} \mathbf{V}^\top} - \mathbf{R}\|_F^2 = \|\lambda \tau \mathbf{U} \mathbf{S} \mathbf{V}^\top - \mathbf{R}\|_F^2 = \\ &= \|\lambda \tau \mathbf{S} - \underbrace{\mathbf{U}^\top \mathbf{R} \mathbf{V}}_{:=\tilde{\mathbf{R}}}\|_F^2 = \|\lambda \tau \mathbf{S} - \tilde{\mathbf{R}}\|_F^2. \end{aligned} \quad (15.37)$$

The reason for introducing the factor τ in the SVD is that it assures that determinants of \mathbf{U} and \mathbf{V} must have the same signs and, hence, it is guaranteed that $\tilde{\mathbf{R}} \in SO(3)$ and, in the end, also that $\mathbf{R} = \mathbf{U}\tilde{\mathbf{R}}\mathbf{V}^\top \in SO(3)$.

In summary, the minimization of ε_{PnP} implies finding $\tilde{\mathbf{R}} \in SO(3)$ that is as close as possible to $\lambda \tau \mathbf{S}$, where \mathbf{S} is a positive diagonal matrix, $\lambda \in \mathbb{R}$, and $\tau = \pm 1$. Both \mathbf{S} and τ have known values but λ is still undetermined. Using the same arguments as in Toolbox Section 8.2.5, the minimizer here is $\tilde{\mathbf{R}} = \mathbf{I}$, independent of λ . This implies that the optimal value for λ is

$$\lambda = \frac{3\tau}{\text{trace}(\mathbf{S})} = \frac{3\tau}{\sigma_1 + \sigma_2 + \sigma_3}. \quad (15.38)$$

The normalized camera that minimizes ε_{PnP} is then constructed from

$$\mathbf{R} = \mathbf{U}\mathbf{V}^\top, \quad \bar{\mathbf{t}} = \lambda \bar{\mathbf{b}}, \quad \mathbf{C} = (\mathbf{R} | \bar{\mathbf{t}}). \quad (15.39)$$

This method of solving PnP using algebraic errors and constraint enforcement is summarized in Algorithm 15.6.

A final note. Any constraint enforcement step, for example the one described above, modifies the resulting camera matrix. Consequently, instead of \mathbf{C}_0 , which minimizes the algebraic error, we get \mathbf{C} which in general does not minimize the algebraic error. At best, \mathbf{C} and \mathbf{c}_0 are approximately equivalent but, since we are dealing with algebraic errors, it is difficult to quantify the consequences of this modification.

Algorithm 15.6: PnP based on algebraic minimization.

```

Input: A set of  $m$  3D points:  $\{\mathbf{x}_k\}$  in homogeneous coordinates,  $m \geq 6$ 
Input: A set of  $m$  corresponding image points  $\{\mathbf{y}_k\}$ , in C-normalized homogeneous coordinates
Output:  $(\mathbf{R} | \bar{\mathbf{t}})$ , minimizing the algebraic error corresponding to Equation (15.35)

1 foreach  $k = 1, \dots, m$  do
2   foreach row*  $\mathbf{r}_l \in [\mathbf{y}_k]_\times$  do
3      $\mathbf{a}$  = vectorization of the  $3 \times 4$  matrix  $\mathbf{r}_l \mathbf{x}_k^\top$ , as a row vector
4     Append this row at the bottom of  $\mathbf{A}$ :  $\mathbf{A} = [\mathbf{A}; \mathbf{a}]$ 
5   end
6 end
7 Determine  $\mathbf{C}_0$  from data matrix  $\mathbf{A}$ :
8   Use either the homogeneous method, or the inhomogeneous method, to find  $\mathbf{c}_0$  in the null space of  $\mathbf{A}$ 
9   Reshape the vector  $\mathbf{c}_0$  to  $3 \times 4$  matrix  $\mathbf{C}_0$ 
10 Constraint enforcement of  $\mathbf{C}_0 = (\mathbf{A} | \bar{\mathbf{b}})$ :
11   Set  $\tau = \text{sign}(\det(\mathbf{A}))$ 
12   SVD:  $\tau \mathbf{A} = \mathbf{U} \mathbf{S} \mathbf{V}^\top$ 
13   Set  $\mathbf{R} = \mathbf{U} \mathbf{V}^\top$ ,  $\lambda = 3\tau/\text{trace}(\mathbf{S})$ ,  $\bar{\mathbf{t}} = \lambda \bar{\mathbf{b}}$ 
14   Return  $\mathbf{C} = (\mathbf{R} | \bar{\mathbf{t}})$ 
```

* Possibly, use only two of the rows in $[\mathbf{y}_k]_\times$ since the 3 rows are linearly dependent

15.4.2 Geometric minimization

If we want an ML-estimate of the camera pose, algebraic estimation is not sufficient and, instead, we should minimize a geometric error. For example, we can consider the error

$$\varepsilon_{PnP,GEO} = \sum_{k=1}^m d_{PP}(\mathbf{y}_k, \mathbf{y}'_k)^2, \quad \text{where } \mathbf{y}'_k = \mathbf{R}\bar{\mathbf{x}}_k + \bar{\mathbf{t}}, \quad (15.40)$$

and minimize it over all $\mathbf{R} \in SO(3)$ and $\bar{\mathbf{t}} \in \mathbb{R}^3$. This error function can be minimized by method belonging to the group of non-linear least squares problems, where a corresponding residual vector is formulated as

$$\mathbf{r} = \begin{pmatrix} \vdots \\ u_k - \frac{y'_{1k}}{y'_{3k}} \\ v_k - \frac{y'_{2k}}{y'_{3k}} \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ u_k - \frac{r_{11}x_{1k} + r_{12}x_{2k} + r_{13}x_{3k} + t_1}{r_{31}x_{1k} + r_{32}x_{2k} + r_{33}x_{3k} + t_3} \\ v_k - \frac{r_{21}x_{1k} + r_{22}x_{2k} + r_{23}x_{3k} + t_1}{r_{31}x_{1k} + r_{32}x_{2k} + r_{33}x_{3k} + t_3} \\ \vdots \end{pmatrix} \in \mathbb{R}^{2m}. \quad (15.41)$$

Here, $\bar{\mathbf{y}}_k = (u_k, v_k)$ are the Cartesian coordinates of image point k , $\bar{\mathbf{x}}_k = (x_{1k}, x_{2k}, x_{3k})$ are the Cartesian coordinates of 3D point k , and the camera pose is determined by

$$\mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}, \quad \bar{\mathbf{t}} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}. \quad (15.42)$$

This \mathbf{r} is defined in terms of the difference between observed coordinates and model coordinates, rather than in terms of distances between the two sets of points, as discussed in Section 14.3.2. Minimizing $\varepsilon_{PnP,GEO}$ is equivalent to minimizing $\frac{1}{2}\|\mathbf{r}\|^2$ which, for example, can be addressed by the Gauss-Newton method or the Levenberg-Marquardt method. A suitable initial solution can be determined by the algebraic estimation described in Section 15.4.1.

As mentioned in Section 14.1.2, the parameterization that is used for a particular problem may have a significant impact on the result. Initially, we can parameterize PnP using the 9 elements in \mathbf{R} and the 3 elements in $\bar{\mathbf{t}}$, forming a parameter vector

$$\tilde{\mathbf{z}} = (r_{11}, r_{21}, r_{31}, r_{12}, r_{22}, r_{32}, r_{31}, r_{32}, r_{33}, t_1, t_2, t_3) \in \mathbb{R}^{12}. \quad (15.43)$$

Despite the fact that this is not a consistent parameterization, we use $\tilde{\mathbf{z}}$ as the initial parameterization of PnP, and calculate a corresponding Jacobian:

$$\tilde{\mathbf{J}} = \begin{pmatrix} \vdots & \vdots \\ -\frac{x_{1k}}{y'_{3k}} & 0 & \frac{y'_{1k}x_{1k}}{y'_{3k}^2} & -\frac{x_{2k}}{y'_{3k}} & 0 & \frac{y'_{1k}x_{2k}}{y'_{3k}^2} & -\frac{x_{3k}}{y'_{3k}} & 0 & \frac{y'_{1k}x_{3k}}{y'_{3k}^2} & -\frac{1}{y'_{3k}} & 0 & \frac{y'_{1k}}{y'_{3k}^2} \\ 0 & -\frac{x_{1k}}{y'_{3k}} & \frac{y'_{2k}x_{1k}}{y'_{3k}^2} & 0 & -\frac{x_{2k}}{y'_{3k}} & \frac{y'_{2k}x_{2k}}{y'_{3k}^2} & 0 & -\frac{x_{3k}}{y'_{3k}} & \frac{y'_{2k}x_{3k}}{y'_{3k}^2} & 0 & -\frac{1}{y'_{3k}} & \frac{y'_{2k}}{y'_{3k}^2} \\ \vdots & \vdots \end{pmatrix} \in \mathbb{R}^{2n \times 12}. \quad (15.44)$$

To obtain a consistent parameterization, we need to re-parameterize \mathbf{R} , using one of the alternatives discussed in Section 14.1.2. As a result, we obtain a compact set of expression both for the residual and the Jacobian, in terms of the final parameters \mathbf{z} .

A final note. The formulation of the error in Equation (15.40) only takes into account measurement errors in the image, it does not recognize the possibility of having measurement errors also in the 3D coordinates. A cost function that measures errors both in the image and in 3D space can be formulated, but in most applications Equation (15.40) is sufficient for determining an ML-estimate of the camera pose.

15.4.3 Minimal case estimation (P3P)

In Equation (15.35), we formulate PnP as a linear homogeneous equation in the unknown camera matrix \mathbf{C} . For each pair of corresponding points in the image and in 3D space, \mathbf{y}_k and \mathbf{x}_k , this equation provides us with two constraints in \mathbf{C} . The camera is normalized and, therefore, it has 6 degrees of freedom: 3 from $\mathbf{R} \in SO(3)$ and 3 more from $\bar{\mathbf{t}} \in \mathbb{R}^3$. This implies that it should be sufficient with only 3 pairs of corresponding points to determine $\mathbf{C} = (\mathbf{R} \parallel \bar{\mathbf{t}})$.

This observation is correct, and constitutes the basis for a classical problem in geometry: how to determine the rigid transformation $(\mathbf{R}, \bar{\mathbf{t}})$ from 3 pairs of corresponding points, commonly referred to as *P3P*. The first solution

to this problem was formulated already in 1841 by Grunert [25], and over time it has been addressed by several authors. For an overview of the field, see [43] and [23]. Although the exact implementation of various methods differ in their details, they can be summarized as follows: there exist up to four solutions to P3P, each in the form of a distinct rigid transformation $(\mathbf{R}, \bar{\mathbf{t}})$, and a common approach to solving P3P is to formulate a quartic polynomial in the known data, \mathbf{y}_k and $\bar{\mathbf{x}}_k$ for $k = 1, 2, 3$, and from each root of this polynomial determine the corresponding $(\mathbf{R}, \bar{\mathbf{t}})$.

Among the different methods for P3P, we point out the recent approach by Kneip *et al.* [43], which has a reasonable computational complexity combined with accurate results. Furthermore, it parameterizes the problem directly in the rigid transformation $(\mathbf{R}, \bar{\mathbf{t}})$, which most of the other methods do not. In addition to that, software implementations of this method can be downloaded here [44].

Chapter 16

Estimation for Two-View Geometry

Before you read this chapter, you should have thorough understanding of epipolar geometry, presented in Chapter 10, and also of the linear estimation techniques described in Chapters 12 and 13. You may also want to have a look at Chapter 14 on non-linear estimation.

In this chapter we apply methods that have been established in earlier chapters to various estimation problem related to two views of a common scene. This includes epipolar geometry, for example, estimation of a 3D point that projects onto two known image points, or of the fundamental matrix \mathbf{F} that matches a set of correspondences in two views. We will look at estimation of rotational homographies.

16.1 Triangulation

Given that two image points \mathbf{y}_1 and \mathbf{y}_2 are in correspondence, i.e., they satisfy the epipolar constraint Equation (10.7) defined by the fundamental matrix \mathbf{F} , their respective projection lines intersect at a point \mathbf{x} in 3D space, see Figure 10.6. If also the camera matrices are known, it is possible to determine \mathbf{x} from $\mathbf{y}_1, \mathbf{y}_2$ and $\mathbf{C}_1, \mathbf{C}_2$. Due to the measurement noise in the image coordinates, the epipolar constraint is in practice not satisfied exactly, even if the two points are in correspondence. This means that the two projection lines come close to intersecting, but not exactly, and the determination of \mathbf{x} is then subject to an estimation procedure where we want to find the 3D point that, in one way or another, optimally fits the data in the form of $\mathbf{y}_1, \mathbf{y}_2$ and $\mathbf{C}_1, \mathbf{C}_2$. The optimality criteria can be defined in several different ways and, here, some of the more common approaches are presented.

16.1.1 The mid-point method

Let $\mathbf{y}_1, \mathbf{y}_2$ be a pair of corresponding image points, each in an image corresponding to cameras \mathbf{C}_1 and \mathbf{C}_2 , respectively. To each of the two image points there is a projection line, \mathbf{L}_1 and \mathbf{L}_2 , respectively. From \mathbf{C}_1 and \mathbf{C}_2 we get the camera centers \mathbf{n}_1 and \mathbf{n}_2 , and two additional points: $\mathbf{x}_1 \sim \mathbf{C}_1^+ \mathbf{y}_1$ and $\mathbf{x}_2 \sim \mathbf{C}_2^+ \mathbf{y}_2$. The two points \mathbf{n}_1 and \mathbf{x}_1 are distinct and lie on the projection line \mathbf{L}_1 , and the two points \mathbf{n}_2 and \mathbf{x}_2 are distinct and lie on the projection line \mathbf{L}_2 . Using Cartesian coordinates, the two projection lines are then parameterized as

$$\bar{\mathbf{x}}_1(s) = s \bar{\mathbf{x}}_1 + (1 - s) \bar{\mathbf{n}}_1, \quad \bar{\mathbf{x}}_2(t) = t \bar{\mathbf{x}}_2 + (1 - t) \bar{\mathbf{n}}_2. \quad (16.1)$$

We want to determine the 3D point $\bar{\mathbf{x}}$ that corresponds to the two image points. To do so we define an error function as

$$\varepsilon_{MP} = \|\bar{\mathbf{x}}_1(s) - \bar{\mathbf{x}}\|^2 + \|\bar{\mathbf{x}}_2(t) - \bar{\mathbf{x}}\|^2, \quad (16.2)$$

and formulate the triangulation problems as: find $\bar{\mathbf{x}}$ together with s and t that minimize ε_{MP} . A formal analysis of this problem is straightforward and can be summarized as: s and t must be chosen such that $\bar{\mathbf{x}}_1(s)$ and $\bar{\mathbf{x}}_2(t)$ are the two points on each of the two projection lines that are as close together as possible. Through these points passes the 3D line \mathbf{L}_x , perpendicular to \mathbf{L}_1 and \mathbf{L}_2 . $\bar{\mathbf{x}}$ is the *mid-point* of the segment of \mathbf{L}_x that lies between $\bar{\mathbf{x}}_1(s)$ and

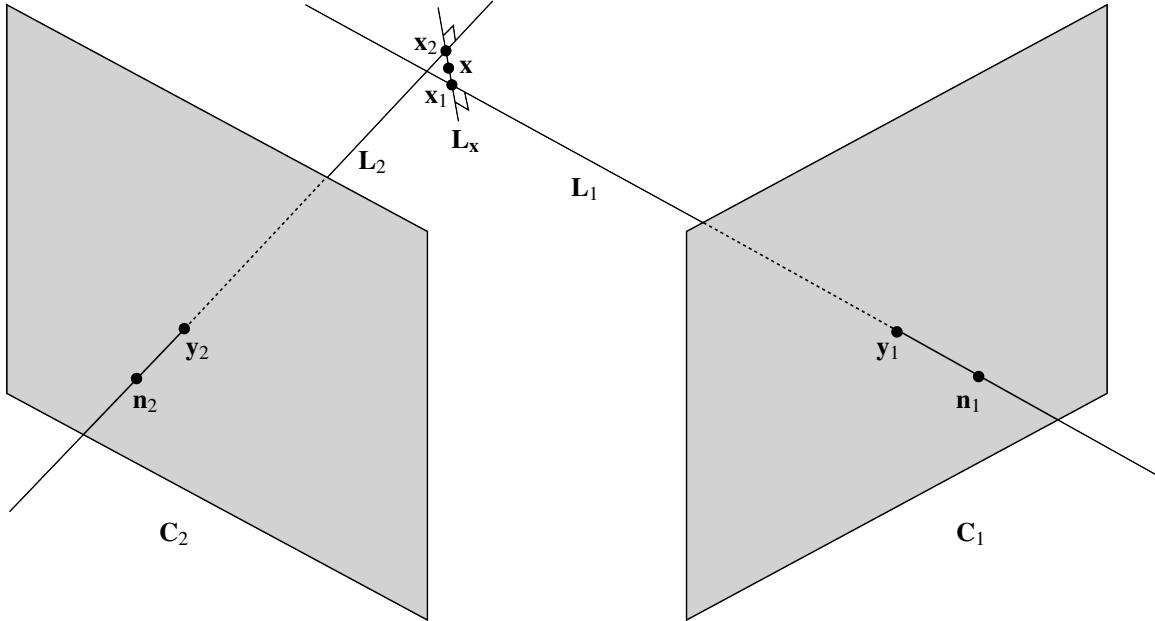


Figure 16.1: The mid-point method for 2-view triangulation.

$\bar{x}_2(t)$. This form of the triangulation problem is referred to as the *mid-point method* and its geometry is illustrated in Figure 16.1.

Thus, we seek s and t that minimize

$$\varepsilon'_{\text{MP}} = \|\bar{x}_1(s) - \bar{x}_2(t)\|^2. \quad (16.3)$$

The minimal value of ε'_{MP} is obtained by (s_0, t_0) that solve the corresponding normal equation

$$(\bar{t}_1 - \bar{t}_2)^T (\bar{t}_1 - \bar{t}_2) \begin{pmatrix} s_0 \\ t_0 \end{pmatrix} = (\bar{t}_1 - \bar{t}_2)^T (\bar{n}_1 - \bar{n}_2), \quad (16.4)$$

where $\bar{t}_1 = \bar{x}_1 - \bar{n}_1$ and $\bar{t}_2 = \bar{x}_2 - \bar{n}_2$ are tangent vectors of the respective projection line. Consequently, by solving Equation (16.4) for (s_0, t_0) , and then inserting these parameters into Equation (16.1) we get the two points $\bar{x}_1(s_0)$ and $\bar{x}_2(t_0)$ on each of the projection lines that are closest to each other. The point \bar{x} that minimizes ε_{MP} lies right in the middle of these two points:

$$\bar{x} = \frac{\bar{x}_1(s_0) + \bar{x}_2(t_0)}{2}. \quad (16.5)$$

The mid-point method for 2-view triangulation is summarized in Algorithm 16.1.

Algorithm 16.1: The mid-point method for 2-view triangulation

Input: A pair of stereo cameras C_1, C_2 .
Input: A pair of corresponding image points, y_1, y_2 , from the two cameras.

Output: A 3D point \bar{x} corresponding to y_1, y_2 .

- 1 Compute camera centers n_1 and n_2
- 2 Compute two additional 3D points: $x_1 = C_1^+ y_1$ and $x_2 = C_2^+ y_2$
- 3 Compute tangent vectors of the projection lines: $\bar{t}_1 = \bar{x}_1 - \bar{n}_1$ and $\bar{t}_2 = \bar{x}_2 - \bar{n}_2$
- 4 Solve (s_0, t_0) from Equation (16.4), and insert to get $\bar{x}_1(s_0)$ and $\bar{x}_2(t_0)$ in Equation (16.1)
- 5 Compute \bar{x} as the mid-point of $\bar{x}_1(s_0)$ and $\bar{x}_2(t_0)$, Equation (16.5)

To derive the above expressions, we have been using Cartesian coordinates rather than homogeneous coordinates all the way. The reason is that homogeneous coordinates are very useful for expressing geometric relations,

but not for expressing distances in Euclidean spaces. Since this triangulation method is based on minimizing geometric errors in 3D space, Cartesian coordinates rule.

16.1.2 Algebraic methods

Triangulation can also be done based on algebraic errors. We have the two equivalence relations:

$$\mathbf{y}_1 \sim \mathbf{C}_1 \mathbf{x}, \quad \mathbf{y}_2 \sim \mathbf{C}_2 \mathbf{x}, \quad (16.6)$$

on which we can apply DLT to get

$$\mathbf{0} = [\mathbf{y}_1] \times \mathbf{C}_1 \mathbf{x}, \quad \mathbf{0} = [\mathbf{y}_2] \times \mathbf{C}_2 \mathbf{x}, \quad (16.7)$$

A more compact form of these two relations is

$$\mathbf{0} = \underbrace{\begin{pmatrix} [\mathbf{y}_1] \times \mathbf{C}_1 \\ [\mathbf{y}_2] \times \mathbf{C}_2 \end{pmatrix}}_{:=\mathbf{A}} \mathbf{x} = \mathbf{A} \mathbf{x}. \quad (16.8)$$

This mean that \mathbf{x} is the solution of a homogeneous equation given by the 6×4 matrix \mathbf{A} . As usual when DLT is applied, there are linearly dependent rows in \mathbf{A} . In this case the first three rows and the last three rows are linearly dependent. In general, we can therefore remove an arbitrary row in each of the two sets of rows to get a 4×4 matrix \mathbf{A} . Furthermore, in practice it is the case that there is measurement noise in the coordinates of $\mathbf{y}_1, \mathbf{y}_2$, meaning that Equation (16.8) is not satisfied exactly for any $\mathbf{x} \neq \mathbf{0}$. Instead we find \mathbf{x} that minimizes the algebraic error

$$\varepsilon_{\text{ALG,TRI}} = \|\mathbf{A} \mathbf{x}\|. \quad (16.9)$$

Standard approaches for this minimization are the inhomogeneous and the homogeneous methods described in Section 12.5.6 and Section 12.5.7.

Data normalization

If the triangulation is made by minimizing the algebraic error $\varepsilon_{\text{ALG,TRI}}$ by means of the homogeneous method, the result is in general dependent on the specific coordinate system that is used for the images. As discussed in Section 13.2.2, it is important to apply normalization of the image coordinates when the homogeneous method is used, and it proposes H-normalization as a general approach to deal with this issue. This practice, or similar, should be applied also in the case if triangulation in order to obtain estimates that have small geometric errors.

Triangulation from multiple views

An obvious question in the context of triangulation of 3D points is: what if the 3D point is observed in more than two views? Can it still be reconstructed based on its projection in all these views? Answer is: in principle yes, but it may become more complicated to extend some methods than it is for others. The error function ε_{MP} in Equation (16.2) can be extended in a straightforward way for the mid-point method, but the solution becomes less trivial to formulate. for this reason, we leave this as an exercise to interested readers.

The algebraic method, however, can be extended without effort. The data matrix \mathbf{A} in Equation (16.8) contains 3 row (or 2 rows if we choose to remove linearly dependencies) for each image point \mathbf{y}_k that is observed in camera \mathbf{C}_k . If we have $n \geq 2$ views, this simply means that we can get additional constraints on \mathbf{x} , represented by the $3n \times 4$ matrix \mathbf{A} . The minimization of $\varepsilon_{\text{ALG,TRI}}$ in Equation (16.9) can still be solved by either the homogeneous or inhomogeneous methods. In consequence of this observation, we formulate the algebraic approach to triangulation in Algorithm 16.2 as a general method for $n \geq 2$ views.

16.1.3 Optimal triangulation

The mid-point method finds a 3D point that minimizes a geometric error defined in 3D space, Equation (16.2), and the algebraic methods find a 3D point that minimizes an algebraic error Equation (16.9). Neither of these methods

Algorithm 16.2: Algebraic method for triangulation of a 3D point from multiple views.

```

Input: A set of  $n \geq 2$  cameras  $\mathbf{C}_1, \dots, \mathbf{C}_n$ .
Input: A set of  $n$  corresponding image points,  $\mathbf{y}_1, \dots, \mathbf{y}_n$ , from the cameras.
Output: A 3D point  $\mathbf{x}$  corresponding to  $\mathbf{y}_1, \dots, \mathbf{y}_n$ .
1 Initialize:  $\mathbf{A} = 0 \times 4$  matrix
2 foreach image point  $\mathbf{y}_k$  do
3   Compute the  $3 \times 4$  matrix  $\mathbf{M} = [\mathbf{y}_k]_{\times} \mathbf{C}_k$ 
4   Possibly remove one of the rows in  $\mathbf{M}$  since they are linearly dependent
5   Append this matrix at the bottom of  $\mathbf{A}$ :  $\mathbf{A} = [\mathbf{A}; \mathbf{M}]$ 
6 end
7 Find  $\mathbf{x}$  that minimizes  $\|\mathbf{A} \mathbf{x}\|$  by means of the homogeneous or the inhomogeneous method

```

produce a maximum likelihood estimate of \mathbf{x} , which in some applications is a desired property. Finding an ML-estimate of \mathbf{x} is often referred to as *optimal triangulation*, which is a misnomer since the other methods mentioned here are optimal, too, relative to their specific error functions. The method that is discussed in this section is optimal as an ML-estimate, obtained by minimizing a cost function defined in terms of geometric distances in the images [33].

• Maximum likelihood estimates are discussed in Section 12.6.

We have two image points $\mathbf{y}_1, \mathbf{y}_2$ that are approximately in correspondence. We seek a 3D point \mathbf{x} which, when projected into the two images, produced a pair of auxiliary points

$$\mathbf{y}'_1 \sim \mathbf{C}_1 \mathbf{x}, \quad \text{and} \quad \mathbf{y}'_2 \sim \mathbf{C}_2 \mathbf{x}, \quad (16.10)$$

such that \mathbf{y}'_1 lies as close as possible to \mathbf{y}_1 , and \mathbf{y}'_2 lies as close as possible to \mathbf{y}_2 . We quantify the latter statement by the geometric error

$$\varepsilon_{\text{OPT}} = d_{\text{PP}}(\mathbf{y}_1, \mathbf{y}'_1)^2 + d_{\text{PP}}(\mathbf{y}_2, \mathbf{y}'_2)^2. \quad (16.11)$$

As discussed in Section 12.6, ε_{OPT} is the negative logarithm of the probability that we observe the projected points at \mathbf{y}_1 and \mathbf{y}_2 given that \mathbf{y}'_1 and \mathbf{y}'_2 have been perturbed by independent additive Gaussian noise. Consequently, minimizing ε_{OPT} over different choices of \mathbf{x} is equivalent to maximizing the probability that \mathbf{y}_1 and \mathbf{y}_2 are the observed positions given different choices of \mathbf{x} .

We want to minimize ε_{OPT} over all 3D points \mathbf{x} , an optimization problem which, in principle, has 3 degrees of freedom. This problem can be simplified if we take into account that each \mathbf{x} corresponds to an epipolar plane \mathbf{p} , as described in Section 10.1.3. The epipolar plane generates a pair of epipolar lines: \mathbf{l}_1 in image 1 and \mathbf{l}_2 in image 2, in accordance with Section 10.1.3. These lines are in correspondence according to the transfer relation described in Section 10.2.5. Furthermore, for each such a pair of corresponding epipolar lines, the optimal position of \mathbf{x} is the one that makes $\bar{\mathbf{y}}'_1$ be the orthogonal projection of $\bar{\mathbf{y}}_1$ onto the epipolar line \mathbf{l}_1 , and $\bar{\mathbf{y}}'_2$ be the orthogonal projection of $\bar{\mathbf{y}}_2$ onto the epipolar line \mathbf{l}_2 . Consequently, instead of searching over all 3D points, we can search over all epipolar planes \mathbf{p} . This set consists of all planes that intersect the two camera centers, and has only 1 degree of freedom.

In fact, we can simplify even further and use the set of epipolar lines in one of the two images as the search space. For example, we can use the epipolar line \mathbf{l}_2 in image 2 as the free parameter of this problem. By means of the transfer homography \mathbf{H}_{12} in Equation (10.18), \mathbf{l}_2 is mapped to the epipolar line $\mathbf{l}_1 \sim \mathbf{H}_{12}\mathbf{l}_2$ in image 1. Hence, each point-to-point distance in Equation (16.11) corresponds to a point-to-line distance, and we can write

$$\varepsilon_{\text{OPT}} = d_{\text{PD}}(\mathbf{y}_1, \mathbf{l}_1)^2 + d_{\text{PD}}(\mathbf{y}_2, \mathbf{l}_2)^2 = d_{\text{PD}}(\mathbf{y}_1, \mathbf{H}_{12}\mathbf{l}_2)^2 + d_{\text{PD}}(\mathbf{y}_2, \mathbf{l}_2)^2. \quad (16.12)$$

This is the same error function as in Equation (16.11), but now parameterized by the epipolar line \mathbf{l}_2 , instead of the 3F point \mathbf{x} . As shown in [33], this optimization problem can be solved analytically, in terms of finding and examining the roots of a 6th order polynomial.

The two camera matrices \mathbf{C}_1 and \mathbf{C}_2 are assumed to be known, which also applies to the fundamental matrix \mathbf{F} , Equation (10.2). In the rest of this section we derive a method that minimizes ε_{OPT} for the general case when none of the epipoles \mathbf{e}_{12} and \mathbf{e}_{21} are at infinity. In this case, there is always a pair of translations $\mathbf{T}_1, \mathbf{T}_2$, one in each image, which move the epipoles to the origins. In these translated coordinates, the fundamental matrix has

the form

$$\mathbf{F}' = \mathbf{T}_1^{-\top} \mathbf{F} \mathbf{T}_2^{-1} = \begin{pmatrix} F'_{11} & F'_{12} & 0 \\ F'_{21} & F'_{22} & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad (16.13)$$

since it must be the case that $\mathbf{e}'_{12}^\top \mathbf{F}' = \mathbf{0}$ and $\mathbf{F}' \mathbf{e}_{21} = \mathbf{0}$ for $\mathbf{e}'_{12} = \mathbf{e}'_{21} = (0, 0, 1)$. Furthermore, if we apply an SSVD¹ of the 2×2 upper left submatrix of \mathbf{F}' :

$$\mathbf{U} \mathbf{S} \mathbf{V}^\top = \begin{pmatrix} F'_{11} & F'_{12} \\ F'_{21} & F'_{22} \end{pmatrix}, \quad (16.14)$$

we can apply the rotation \mathbf{U} in image 1 and \mathbf{V} in image 2, after the translations have been made, to obtain an even sparser form of the fundamental matrix:

$$\tilde{\mathbf{F}} = \mathbf{T}_U^{-\top} \mathbf{T}_1^{-\top} \mathbf{F} \mathbf{T}_2^{-1} \mathbf{T}_V^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \text{where } 1 > \sigma. \quad (16.15)$$

Here we assume that $\tilde{\mathbf{F}}$ has been normalized such that its largest singular value equals one.

An important observation to be made here is that we have changed the coordinate system in image 1 by the transformation $\mathbf{T}_V \mathbf{T}_1$, and in image 2 by $\mathbf{T}_U \mathbf{T}_2$, and both are rigid transformations. Consequently, distances are invariant to these transformations, and we can formulate the cost function as

$$\varepsilon_{\text{OPT}} = d_{\text{PD}}(\tilde{\mathbf{y}}_1, \tilde{\mathbf{H}}_{12} \tilde{\mathbf{l}}_2)^2 + d_{\text{PD}}(\tilde{\mathbf{y}}_2, \tilde{\mathbf{l}}_2)^2, \quad (16.16)$$

where $\tilde{\mathbf{y}}_1 = \mathbf{T}_U \mathbf{T}_1$ and $\tilde{\mathbf{y}}_2 = \mathbf{T}_V \mathbf{T}_2$. Furthermore, $\tilde{\mathbf{l}}_2$ is the epipolar line in image 2, relative to the translated and rotated coordinate system there, and $\tilde{\mathbf{M}}$ is a homography that transfers $\tilde{\mathbf{l}}_2$ to the corresponding epipolar line $\tilde{\mathbf{l}}_1$ in image 1. Also $\tilde{\mathbf{M}}$ is represented relative to the translated and rotated coordinate systems in each image.

Since $\tilde{\mathbf{l}}_2$ intersects with the point $\tilde{\mathbf{e}}_{21} = (0, 0, 1)$, we can parameterize it as

$$\tilde{\mathbf{l}}_2 = \begin{pmatrix} \cos \alpha \\ \sin \alpha \\ 0 \end{pmatrix}, \quad (\text{already in a D-normalized form}). \quad (16.17)$$

In accordance with observation 10.13 on page 152 and Equation (10.44), the corresponding epipolar line in image 1 is then given as

$$\tilde{\mathbf{l}}_1 \sim \tilde{\mathbf{M}} \tilde{\mathbf{l}}_2 = \begin{pmatrix} 0 & 1 & 0 \\ -\sigma & 0 & 0 \\ 0 & 0 & \lambda \end{pmatrix} \begin{pmatrix} \cos \alpha \\ \sin \alpha \\ 0 \end{pmatrix} = \begin{pmatrix} \sin \alpha \\ -\sigma \cos \alpha \\ 0 \end{pmatrix} \sim \underbrace{\frac{1}{\sqrt{\sin^2 \alpha + \sigma^2 \cos^2 \alpha}}}_{\text{D-normalization}} \begin{pmatrix} \sin \alpha \\ -\sigma \cos \alpha \\ 0 \end{pmatrix}. \quad (16.18)$$

The transformed image coordinates, in canonical form, are given by $\tilde{\mathbf{y}}_1 = (\tilde{u}_1, \tilde{v}_1, 1)$ and $\tilde{\mathbf{y}}_2 = (\tilde{u}_2, \tilde{v}_2, 1)$. This allows us to rewrite ε_{OPT} in Equation (16.16) as

$$\varepsilon_{\text{OPT}} = (\tilde{\mathbf{y}}_1 \cdot \tilde{\mathbf{l}}_1)^2 + (\tilde{\mathbf{y}}_2 \cdot \tilde{\mathbf{l}}_2)^2 = \frac{(\tilde{u}_1 \sin \alpha - \tilde{v}_1 \sigma \cos \alpha)^2}{\sin^2 \alpha + \sigma^2 \cos^2 \alpha} + (\tilde{u}_2 \cos \alpha + \tilde{v}_2 \sin \alpha)^2. \quad (16.19)$$

Making the substitution

$$z = \tan \alpha, \quad (16.20)$$

and then setting the derivative of ε_{OPT} with respect to z equal to zero give us a polynomial equation in the unknown parameter z :

$$p = p_6 z^6 + p_5 z^5 + p_4 z^4 + p_3 z^3 + p_2 z^2 + p_1 z + p_0 = 0, \quad (16.21)$$

where

$$\begin{aligned} p_6 &= \sigma \tilde{u}_1 \tilde{v}_1 - \tilde{u}_2 \tilde{v}_2, & p_5 &= \sigma^2 (\tilde{u}_1^2 - \tilde{v}_1^2) - (\tilde{u}_2^2 - \tilde{v}_2^2), & p_4 &= (2\sigma - \sigma^3) \tilde{u}_1 \tilde{v}_1 + (1 - 2\sigma^2) \tilde{u}_2 \tilde{v}_2, \\ p_3 &= 2\sigma^2 (\tilde{u}_1^2 - \tilde{u}_2^2 - \tilde{v}_1^2 + \tilde{v}_2^2), & p_2 &= (\sigma - 2\sigma^3) \tilde{u}_1 \tilde{v}_1 + (2\sigma^2 - \sigma^4) \tilde{u}_2 \tilde{v}_2, \\ p_1 &= \sigma^2 (\tilde{u}_1^2 - \tilde{v}_1^2) - \sigma^4 (\tilde{u}_2^2 - \tilde{v}_2^2), & p_0 &= -\sigma^3 \tilde{u}_1 \tilde{v}_1 + \sigma^6 \tilde{u}_2 \tilde{v}_2. \end{aligned} \quad (16.22)$$

¹SSVD is described in Toolbox Section 8.2.7.

Algorithm 16.3: Optimal triangulation

Input: Two corresponding image points \mathbf{y}_1 and \mathbf{y}_2 , in canonical form
Input: Camera matrices \mathbf{C}_1 and \mathbf{C}_2
Output: 3D point \mathbf{x} with minimal reprojection error ε_{OPT} relative to \mathbf{y}_1 and \mathbf{y}_2

- 1 Determine fundamental matrix \mathbf{F} from \mathbf{C}_1 and \mathbf{C}_2 , Equation (10.2)
- 2 Determine epipoles \mathbf{e}_{12} and \mathbf{e}_{21} from $\mathbf{e}_{12}^\top \mathbf{F} = \mathbf{0}$ and $\mathbf{F} \mathbf{e}_{21} = \mathbf{0}$
- 3 Set: $\mathbf{T}_1, \mathbf{T}_2$ = translation matrices such that $\mathbf{T}_1 \mathbf{e}_{12} = \mathbf{T}_2 \mathbf{e}_{21} = (0, 0, 1)$
- 4 Set: $\mathbf{F}' = \mathbf{T}_1^{-1} \mathbf{F} \mathbf{T}_2^{-1}$, Equation (16.13)
- 5 SSVD: $\mathbf{U} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \mathbf{V}^\top = \begin{pmatrix} F'_{12} & F'_{12} \\ F'_{21} & F'_{22} \end{pmatrix}$, where $\mathbf{U}, \mathbf{V} \in SO(2)$
- 6 Set: $\mathbf{T}_U, \mathbf{T}_V$ = rotation matrices corresponding to 2D rotations \mathbf{U} and \mathbf{V} , and $\sigma = \sigma_2/\sigma_1$
- 7 Transform coordinates: $\tilde{\mathbf{y}}_1 = \mathbf{T}_U \mathbf{T}_1 = (\tilde{u}_1, \tilde{v}_1, 1)$ and $\tilde{\mathbf{y}}_2 = \mathbf{T}_V \mathbf{T}_2 = (\tilde{u}_2, \tilde{v}_2, 1)$
- 8 Compute polynomial coefficients p_0, \dots, p_6 , Equation (16.22)
- 9 Compute all 6 roots z of the polynomial p , Equation (16.21)
- 10 **foreach** root z **do**
- 11 Compute ε_{OPT} in Equation (16.19) for $\alpha = \tan^{-1} z$
- 12 **end**
- 13 Determine optimal α that minimizes ε_{OPT} in the previous step
- 14 Compute optimal epipolar lines: $\tilde{\mathbf{l}}_1$ in Equation (16.18) and $\tilde{\mathbf{l}}_2$ in Equation (16.17)
- 15 Transform back to original coordinate system: $\mathbf{l}_1 = \mathbf{T}_1^\top \mathbf{T}_U^\top \tilde{\mathbf{l}}_1$, $\mathbf{l}_2 = \mathbf{T}_2^\top \mathbf{T}_V^\top \tilde{\mathbf{l}}_2$
- 16 Determine projection transformations \mathbf{T}_{l_1} and \mathbf{T}_{l_2} corresponding to \mathbf{l}_1 and \mathbf{l}_2 , in accordance with Section 4.6
- 17 Determine auxiliary points $\mathbf{y}'_1 = \mathbf{T}_{l_1} \mathbf{y}_1$ and $\mathbf{y}'_2 = \mathbf{T}_{l_2} \mathbf{y}_2$
- 18 Triangulate \mathbf{x} from \mathbf{y}'_1 and \mathbf{y}'_2 using the homogeneous method, Algorithm 16.2 on page 278

Consequently, by finding the real roots of p in Equation (16.21), we get up to six possible values for z . Each such z corresponds to an α from Equation (16.20), which gives a value for ε_{OPT} in Equation (16.19). In the end, we seek the α that gives the minimal value of ε_{OPT} . This particular α maps to an epipolar line $\tilde{\mathbf{l}}_2$ in image 2, Equation (16.17), and a corresponding epipolar line $\tilde{\mathbf{l}}_1$ in image 1, Equation (16.18). By transforming the lines back to the original coordinate system², we get \mathbf{l}_1 and \mathbf{l}_2 and, finally, $\bar{\mathbf{y}}'_1$ and $\bar{\mathbf{y}}'_2$ are obtained by an orthogonal projection of $\bar{\mathbf{y}}_1$ onto the line \mathbf{l}_1 and $\bar{\mathbf{y}}_2$ onto the line \mathbf{l}_2 .

Once the auxiliary points $\bar{\mathbf{y}}'_1, \bar{\mathbf{y}}'_2$ are determined, we can triangulate \mathbf{x} using any of the previous methods. Since $\bar{\mathbf{y}}'_1, \bar{\mathbf{y}}'_2$ satisfy the epipolar constraint exactly, any reasonable triangulation method should give the same \mathbf{x} , at least in principle. The homogeneous methods based on algebraic errors is a good choice, since it does not have the type of degeneracies that appear in other methods. The computational steps in optimal triangulation are summarized in Algorithm 16.3.

Some final notes: (I) The method for optimal triangulation presented here does not work if any of the two epipoles are at infinity. On the one hand, we could argue that this case is very unlikely for two cameras in general configuration. On the other hand, as we will see in Chapter 20, a rectified stereo camera is characterized by having its epipoles at infinity. For these cases, a dedicated version of optimal triangulation must be implemented. (II) Some of the 6 roots of the polynomial p can be complex. Formally, these do not correspond to valid solutions of the minimization of ε_{OPT} , and could be discarded. In practice, however, minor numerical inaccuracies in the initial parameters to the optimization can change a real root to complex root, but with a very small imaginary part. Instead of discarding these roots, an option can be to use the real parts of the roots as the z that generates $\alpha = \tan^{-1} z$.

16.1.4 Closing remarks

Before we leave the triangulation problem, some additional remarks are made here to illustrate important qualities of the proposed methods.

²Use the dual transformations for the lines!

Method Degeneracy

The methods for triangulations that are proposed here are not completely without pitfalls. For example, in the mid-point method, we need to solve the linear equation Equation (16.4) to determine (s_0, t_0) . This step degenerates if the two tangents \bar{t}_1 and \bar{t}_2 are parallel. In this case there are infinitely many solutions to the minimization of ε'_{MP} . Therefore, there are also infinitely many solutions to the minimization of ε_{MP} . In short: the mid-point method does not deliver a unique 3D point for the case when the two image points y_1, y_2 generate parallel projection lines. This happens when x is a point at infinite distance from the two cameras. In fact, due to numerical inaccuracies in the calculations that are used to compute x , the computation of x becomes fragile already when the lines are close to parallel, i.e., when x is far away, but not infinitely far away, from the cameras.

A similar observation can be made also for the algebraic approach in Section 16.1.2, when the inhomogeneous method is applied and some specific element in the vector $x \in \mathbb{R}^4$ is set = 1 and we solve the remaining elements from a linear equation. Since x contains the homogeneous coordinates of a 3D points, it is natural to set $[x]_4 = 1$, implying a P-normalization of the vector x . On the other hand, it is a perfectly valid situation that y_1 and y_2 correspond to a 3D point at infinity. Again, this implies parallel projection lines for the two image points. Algebraically, it implies that $[x]_4 = 0$, which the inhomogeneous method cannot deliver, unless we choose to set another element in x as = 1. This problem does not appear if we apply the homogeneous method for minimization of $\varepsilon_{\text{ALG,TRI}}$ in Equation (16.9), and is a motivation why we prefer the homogeneous method for algebraic minimization in triangulation. We summarize this discussion as:

16.1 Both the algebraic approach, Algorithm 16.2, based on the inhomogeneous method, and the mid-point method in Algorithm 16.1 exhibit *method degeneracy* for points at infinity.

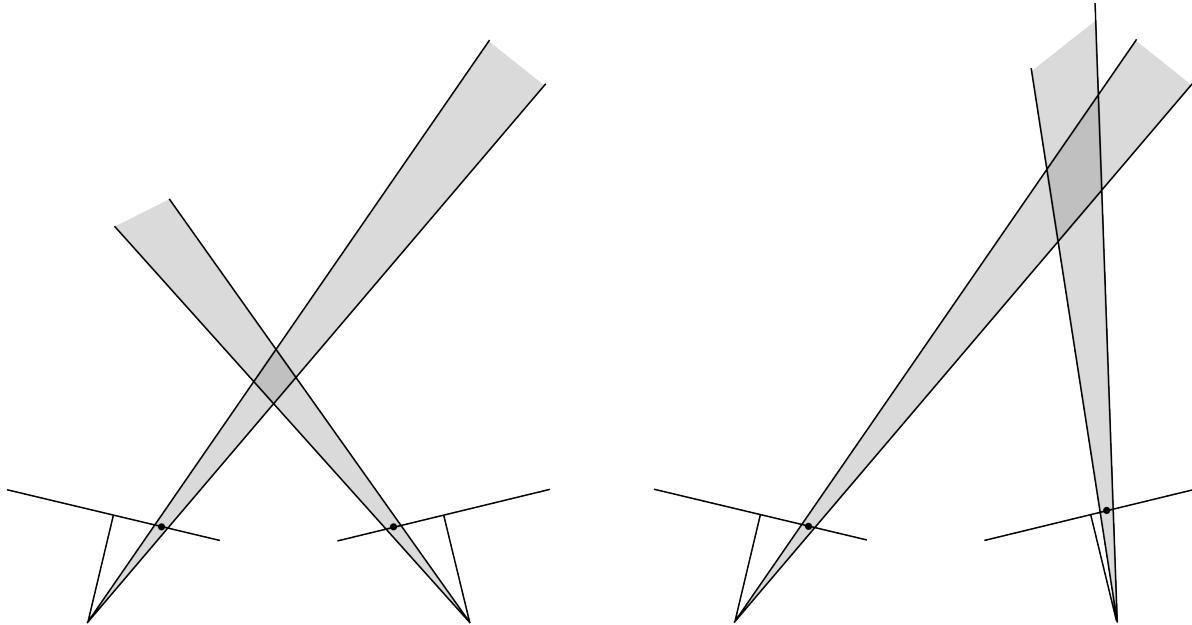


Figure 16.2: Two stylized cases of error propagation in triangulation. The bright shaded areas represent the set of projection lines corresponding to each of the two image points, caused by the measurement noise. The dark shaded areas represent the possible locations of the triangulated 3D point.

Data Degeneracy

The degeneracy described above happens when the two projection lines are parallel, and so they intersect at infinity. An even more problematic case occurs when the lines coincide. This happens for all 3D points \mathbf{x} that lie on the baseline³, which includes both camera centers. In this case, the projection into the two images is the same regardless on where along the baseline \mathbf{x} lies, which means that triangulation is not possible for any method.

16.2 All points on the baseline correspond to *data degeneracy* for stereo triangulation.

Reconstruction errors

In practice we have to expect that there is some amount of noise on the image coordinates of the two points $\mathbf{y}_1, \mathbf{y}_2$ that we use for the triangulation of the 3D point \mathbf{x} . The question is then: how do these measurement errors in the image coordinate propagate to \mathbf{x} ? A quantitative discussion about the error in \mathbf{x} can be made, but is not straightforward and, therefore, outside the scope of this presentation.

Instead we make a qualitative reasoning, based on Figure 16.2. The measurement noise on an image point implies that the corresponding projection line can be described as a set of projection lines, each with a probability density defined by the probability density function of the measurement noise η . To simplify this discussion, we can assume that η is uniformly distributed within a circle of some radius. This implies that the set of projection lines for each image point approximately form a cone. Intuitively, this means that \mathbf{x} , the 3D points that corresponds to the two image points \mathbf{y}_1 and \mathbf{y}_2 , lies in the intersection of the two cones generated by \mathbf{y}_1 and \mathbf{y}_2 , respectively. The two cones are illustrated in Figure 16.2 as bright shaded areas, and their intersections as the dark shaded areas. This, however, does not mean that \mathbf{x} has a uniform distribution within each of these intersections.

The important observation here is that the intersection of the two cones has a volume and shape that depends on where \mathbf{x} is located in the scene relative to the cameras and the relative pose of the cameras. As see in Figure 16.2, the area is smaller if \mathbf{x} is close to the cameras (left) and larger if \mathbf{x} is further away (right). Also, the shape of the region where we expect to find \mathbf{x} can be relatively uniform when the point is close (left) and is elongated in the direction “away” from the cameras when the point is further away (right). In summary: the uncertainty of where \mathbf{x} is located in 3D space grows, in particular in the “depth direction” the further away from the cameras \mathbf{x} lies.

16.2 Estimation of \mathbf{F}

Given that we initially know the camera matrices \mathbf{C}_1 and \mathbf{C}_2 , we can then can compute \mathbf{F} directly from Equation (10.2). We refer to this as *calibrated epipolar geometry*, at least if both cameras refer to a Euclidean world coordinate system. This approach is relevant in certain applications, but it is of limited use since we need to know \mathbf{C}_1 and \mathbf{C}_2 , at least to some level of detail.

An alternative is to determine \mathbf{F} entirely from known correspondences of image points. Let $\mathbf{y}_1, \mathbf{y}_2$ be a pair of corresponding image points, i.e., they satisfy the epipolar constraint Equation (10.7) relative to the fundamental matrix \mathbf{F} . If the coordinates of the points are known, and we know that they are in correspondence, the epipolar constraint can be interpreted as a constraint on \mathbf{F} . With sufficiently many pairs of corresponding points is should be possible to determine or estimate \mathbf{F} . The standard method for estimation of \mathbf{F} based on algebraic methods is the normalized 8-point algorithm.

16.2.1 The 8-point algorithm

Consider a set of n corresponding points, $\mathbf{y}_{1k}, \mathbf{y}_{2k}, k = 1, \dots, n$. This means that they satisfy the epipolar constraint defined by the fundamental matrix \mathbf{F} :

$$0 = \mathbf{y}_{1k}^\top \mathbf{F} \mathbf{y}_{2k}, \quad k = 1, \dots, n. \quad (16.23)$$

³In this context, the baseline extends all the way to infinity in both directions.

We express the homogeneous coordinates of the two points as

$$\mathbf{y}_{1k} = \begin{pmatrix} \alpha_k \\ \beta_k \\ \gamma_k \end{pmatrix}, \quad \mathbf{y}_{2k} = \begin{pmatrix} \delta_k \\ \epsilon_k \\ \zeta_k \end{pmatrix}, \quad (16.24)$$

and the fundamental matrix \mathbf{F} as

$$\mathbf{F} = \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix}, \quad (16.25)$$

which allows us to rewrite the epipolar constraint in Equation (16.23) as

$$0 = \alpha_k \delta_k f_{11} + \beta_k \delta_k f_{21} + \gamma_k \delta_k f_{31} + \alpha_k \epsilon_k f_{12} + \beta_k \epsilon_k f_{22} + \gamma_k \epsilon_k f_{32} + \alpha_k \zeta_k f_{13} + \beta_k \zeta_k f_{23} + \gamma_k \zeta_k f_{33} = \\ = \underbrace{(\alpha_k \delta_k \quad \beta_k \delta_k \quad \gamma_k \delta_k \quad \alpha_k \epsilon_k \quad \beta_k \epsilon_k \quad \gamma_k \epsilon_k \quad \alpha_k \zeta_k \quad \beta_k \zeta_k \quad \gamma_k \zeta_k)}_{:=\mathbf{a}_k} \underbrace{\begin{pmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{pmatrix}}_{:=\mathbf{z}}, \quad k = 1, \dots, n. \quad (16.26)$$

This means that the epipolar constraint is equivalent to

$$0 = \mathbf{a}_k \cdot \mathbf{z}, \quad k = 1, \dots, n. \quad (16.27)$$

In a more compact form, we can write this as

$$\mathbf{0} = \underbrace{\begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_n \end{pmatrix}}_{:=\mathbf{A}} \mathbf{z} \Leftrightarrow \mathbf{A} \mathbf{z} = \mathbf{0}. \quad (16.28)$$

To summarize: n pairs of corresponding points $\mathbf{y}_{1k}, \mathbf{y}_{2k}$ generate a linear homogeneous equation, Equation (16.28), defined by the $n \times 9$ data matrix \mathbf{A} . The unknown in this equation is \mathbf{z} , representing a vectorization of \mathbf{F} .

To find \mathbf{F} in this way, we need at least 8 pairs of corresponding points in general positions. Hence, this method is called the *8-point algorithm* for estimation of \mathbf{F} . If we have $n = 8$ point pairs in general positions, then $\text{Null}(\mathbf{A})$ is 1-dimensional, and there is a unique projective element $\mathbf{z} \in \text{Null}(\mathbf{A})$. If instead $n > 8$, it may not be possible to solve Equation (16.28) exactly due to the measurement noise in the coordinates of $\mathbf{y}_{1k}, \mathbf{y}_{2k}$. Instead, we can formulate the algebraic error

$$\varepsilon_{ALG,F} = \|\mathbf{A} \mathbf{z}\|, \quad (16.29)$$

and minimize it over \mathbf{z} by means of either the inhomogeneous or the homogeneous method in accordance with Section 12.5.6 and Section 12.5.7. This approach works also for $n = 8$, and the only difference between $n = 8$ and $n > 8$ is that, in the former case, the residual error becomes $\varepsilon_{ALG,F} = 0$, while in the latter case we typically get a residual $\varepsilon_{ALG,F} > 0$. Once $\mathbf{z} \in \mathbb{R}^9$ is determined, it is reshaped to the 3×3 matrix \mathbf{F} in accordance with Equation (16.25).

Constraint enforcement

This basic version of the 8-point algorithm has a weakness. Regardless of whether $n = 8$ or $n > 8$, the measurement noise in the image coordinates implies that the resulting \mathbf{F} may not satisfy the internal constraint Equation (10.6). This constraint is a result of the construction of \mathbf{F} according to Equation (10.2) and should be satisfied if \mathbf{F} is a

proper fundamental matrix. If we construct \mathbf{F} entirely based on the epipolar constraint relative to corresponding image points with measurement noise, there is no longer a guarantee that Equation (10.6) is valid. Is this a problem? Maybe, maybe not. The fact that the epipolar lines in each of the two images all intersect at their respective epipolar point relies on Equation (10.5), and requires $\det(\mathbf{F}) = 0$. Consequently, if the internal constraint is not satisfied, the epipolar lines do not behave as expected, and the epipolar points are not well-defined.

To deal with this situation, we can use the fundamental matrix estimated from the algebraic minimization as an initial solution to our estimation problem, denoted \mathbf{F}_0 . We then modify \mathbf{F}_0 to get the final estimate \mathbf{F} , where $\det(\mathbf{F}) = 0$, by what is known as *constraint enforcement*. A simple approach to constraint enforcement in this case is based on the Eckart-Young theorem, described in Toolbox Section 8.4.1. We have an initial estimate \mathbf{F}_0 and we want to find an approximation of this \mathbf{F}_0 , in terms of \mathbf{F} that has rank 2 and is as close as possible to \mathbf{F}_0 in Frobenius norm. This is done by following steps:

1. Compute an SVD of $\mathbf{F}_0 = \mathbf{U} \mathbf{S}_0 \mathbf{V}^\top$.
2. Set $\mathbf{S} = \mathbf{S}_0$, but then set the smallest singular value in \mathbf{S} to zero.
3. Compute $\mathbf{F} = \mathbf{U} \mathbf{S} \mathbf{V}^\top$.

The constraint enforcement that is described here implies that the resulting \mathbf{F} no longer minimizes $\varepsilon_{ALG,F}$ in the same way as \mathbf{F}_0 does. Intuitively, as long as the smallest singular value of \mathbf{S}_0 is small relative to the other two, the modification of the estimated fundamental matrix, from \mathbf{F}_0 to \mathbf{F} , is small. This means that the constraint enforcement step comes at the cost of obtaining a slightly worse estimate of \mathbf{F} relative to the algebraic error $\varepsilon_{ALG,F}$. This statement, however, cannot be given any geometric interpretation since both the estimation of \mathbf{F}_0 and the constraint enforcement is made in the algebraic domain of $P(\mathbb{R}^9)$, rather than in a geometric space.

16.3 We need to enforce the constraint $\det(\mathbf{F}) = 0$ when \mathbf{F} is estimated with the 8-point algorithm. Otherwise there is no guarantee that \mathbf{F} is a proper fundamental matrix, and it may not correctly describe the epipolar geometry for the stereo cameras that have produced the two images. For example, epipolar lines in an image will not intersect at a single point, the epipolar point, if \mathbf{F} is not a proper fundamental matrix.

The normalized 8-point algorithm

A second issue of the 8-point algorithm is that of normalization of image coordinates, as discussed in Section 13.2.2. In that section, the Hartley-normalization is proposed as a standard approach when the homogeneous method is used for minimization of the algebraic error in Equation (16.29), and is motivated from the point of view that it both gives more well-defined SVD profile of the estimation problem and also a solution that has a better correspondence to a minimization of a geometric error.

16.4 As usual when estimation is done by minimizing algebraic errors using the homogeneous method, data should be normalized, for example using Hartley-normalization.

Based on this observation, the 8-point algorithm should be applied to H-normalized image coordinates, rather than pixel coordinates. This is achieved by applying H-normalization on the n points in each of the two images separately, i.e., we determine a separate normalizing transformation on each of the two image coordinate systems. As a result, we obtain a new set of corresponding image points $\mathbf{y}'_{1k}, \mathbf{y}'_{2k}$, where

$$\mathbf{y}'_{1k} = \mathbf{T}_1 \mathbf{y}_{1k}, \quad \mathbf{y}'_{2k} = \mathbf{T}_2 \mathbf{y}_{2k}, \quad (16.30)$$

and \mathbf{T}_1 and \mathbf{T}_2 are the normalizing transformations for each of the two images. The epipolar constraint can now be written as

$$0 = (\mathbf{y}'_{1k})^\top \mathbf{F}' \mathbf{y}'_{2k}, \quad k = 1, \dots, n, \quad (16.31)$$

where \mathbf{F}' is the fundamental matrix relative to the H-normalized image coordinates. In accordance with Section 10.3.2, we can obtain \mathbf{F} as a renormalization of \mathbf{F}' :

$$\mathbf{F} \sim \mathbf{T}_1^\top \mathbf{F}' \mathbf{T}_2. \quad (16.32)$$

Consequently, we can determine \mathbf{F}' using minimization of the algebraic error in Equation (16.29), where \mathbf{A} now are computed from the H-normalized coordinates $\mathbf{y}'_{1k}, \mathbf{y}'_{2k}$. Once \mathbf{F}' is determined, we get \mathbf{F} from Equation (16.32).

A remaining issue is how to combine the two ideas of constraint enforcement and normalization. We can apply the constraint enforcement either before or after the renormalization in Equation (16.32), and it matters in which order these two steps are made. A thorough treatment of this question can be found in [31], and it concludes that we should make the constraint enforcement *before* the renormalization. The combination of the linear estimation of an initial estimate of \mathbf{F} and the constraint enforcement, and all this in H-normalized coordinates that require a renormalization to obtain the resulting \mathbf{F} , is referred to as the *normalized 8-point algorithm*. This is the recommended method for the estimation of the fundamental matrix based on algebraic minimization.

The normalized 8-point algorithm is summarized in Algorithm 16.4. The basic form of this algorithm, without the constraint enforcement, was introduced by Longuet-Higgins [45]. The normalized version was presented by Hartley [31].

Algorithm 16.4: The normalized 8-point algorithm

<p>Input: A set of n corresponding image points $\mathbf{y}_{1k}, \mathbf{y}_{2k}, k = 1, \dots, n$.</p> <p>Output: An estimate of the fundamental matrix \mathbf{F} that optimally fits Equation (16.23) for all point pairs.</p>
<ol style="list-style-type: none"> 1 Compute \mathbf{T}_1 and H-normalized image coordinates in the first image: $\mathbf{y}'_{1k} \sim \mathbf{T}_1 \mathbf{y}_{1k}$ 2 Compute \mathbf{T}_2 and H-normalized image coordinates in the second image: $\mathbf{y}'_{2k} \sim \mathbf{T}_2 \mathbf{y}_{2k}$ 3 Initialize: $\mathbf{A} = \text{empty } 0 \times 9 \text{ matrix}$ 4 foreach point pair $k = 1, \dots, n$ do 5 Compute row vector \mathbf{a}_k in accordance with Equation (16.26). Use H-normalized image coordinates $\mathbf{y}'_{1k}, \mathbf{y}'_{2k}$ here! 6 Append \mathbf{a}_k at the bottom of \mathbf{A}: $\mathbf{A} = [\mathbf{A}; \mathbf{a}_k]$ 7 end 8 Find estimate \mathbf{z} that minimizes $\ \mathbf{A} \mathbf{z}\$ using the homogeneous method 9 Reshape \mathbf{z} to \mathbf{F}'_0, Equation (16.25) 10 Compute SVD of $\mathbf{F}'_0 = \mathbf{U} \mathbf{S}_0 \mathbf{V}^\top$ 11 Set $\mathbf{S} = \mathbf{S}_0$, except the smallest singular value that is set = 0 12 Compute $\mathbf{F}' = \mathbf{U} \mathbf{S} \mathbf{V}^\top$ 13 Re-normalize: $\mathbf{F} = \mathbf{T}_1^\top \mathbf{F}' \mathbf{T}_2$

16.2.2 The 7-point algorithm

The normalized 8-point algorithm is the standard method for estimating \mathbf{F} when we are minimizing algebraic errors. It requires a minimum of 8 pairs of corresponding points in the two images. On the other hand, since \mathbf{F} has 7 degrees of freedom, and each pair of corresponding points provides one constraint on \mathbf{F} as projective element, it should be possible to determine a fundamental matrix that satisfies the internal constraint from only 7 pairs of corresponding points. Such a minimal case estimation of \mathbf{F} is possible, but it comes at a cost: there may be up to three solutions to the problem of finding the optimal estimate of \mathbf{F} . They are provided by the *7-point algorithm*.

The 7-point algorithm starts in the same way as the 8-point algorithm, by forming the data matrix \mathbf{A} as in Equation (16.28). Since we have 7 pairs of points, and we assume that they are in general configurations, \mathbf{A} is a 7×9 matrix of full rank. Consequently, the solution space to $\mathbf{A} \mathbf{z} = \mathbf{0}$ is 2-dimensional, and we cannot determine a unique \mathbf{z} that corresponds to the estimated \mathbf{F} . Let \mathbf{z}_1 and \mathbf{z}_2 be a basis of $\text{Null}(\mathbf{A})$. For example, \mathbf{z}_1 and \mathbf{z}_2 can be determined as two orthogonal right singular vectors corresponding to singular value zero. The fundamental matrix we seek is then represented by a vector $\mathbf{z} \in \mathbb{R}^9$ that is a linear combination of \mathbf{z}_1 and \mathbf{z}_2 :

$$\mathbf{z} = \gamma \mathbf{z}_1 + (1 - \gamma) \mathbf{z}_2. \quad (16.33)$$

The fundamental matrix \mathbf{F} is obtained by reshaping \mathbf{z} to a 3×3 matrix, Equation (16.25), where \mathbf{z} is given as Equation (16.26). Consequently, the previous linear combination can be reformulated as

$$\mathbf{F} = \gamma \mathbf{F}_1 + (1 - \gamma) \mathbf{F}_2, \quad (16.34)$$

where \mathbf{F}_1 and \mathbf{F}_2 are the results of reshaping \mathbf{z}_1 and \mathbf{z}_2 , respectively.

To specify \mathbf{F} further, the parameter γ needs to be determined. This is done by applying the internal constraint, Equation (16.6), to \mathbf{F} in Equation (16.34):

$$0 = \det(\mathbf{F}) = \det(\gamma\mathbf{F}_1 + (1 - \gamma)\mathbf{F}_2) = p(\gamma). \quad (16.35)$$

Given that \mathbf{F}_1 and \mathbf{F}_2 are determined, the right-hand side of Equation (16.35) specifies a third order polynomial p in γ . The γ that we seek must be a root of p : $p(\gamma) = 0$. A third order polynomial with real coefficients has one, two, or three real roots. Each distinct and real root γ produces a distinct fundamental matrix from Equation (16.34). Consequently, in the general case we may get one, two, or three solutions to the minimal case estimation of \mathbf{F} .

The polynomial p can be expanded as

$$p(\gamma) = c_0 + c_1\gamma + c_2\gamma^2 + c_3\gamma^3. \quad (16.36)$$

Using the fact that the determinant of a 3×3 matrix can be written in terms of a scalar and a cross product of its rows, we can write the coefficients of the cubic p as⁴

$$c_0 = \det(\mathbf{F}_2), \quad c_1 = a - 3\det(\mathbf{F}_2), \quad c_2 = b - 2a + 3\det(\mathbf{F}_2), \quad c_3 = \det(\mathbf{F}_1) - b + a - \det(\mathbf{F}_2), \quad (16.37)$$

where

$$a = \det(\mathbf{F}_{12}) + \det(\mathbf{F}_{22}) + \det(\mathbf{F}_{32}), \quad \text{and} \quad b = \det(\mathbf{F}_{11}) + \det(\mathbf{F}_{21}) + \det(\mathbf{F}_{31}), \quad (16.38)$$

and

$$\mathbf{F}_{k1} = \mathbf{F}_1 \text{ except row } k \text{ is taken from } \mathbf{F}_2, \quad \text{and} \quad \mathbf{F}_{k2} = \mathbf{F}_2 \text{ except row } k \text{ is taken from } \mathbf{F}_1. \quad (16.39)$$

Since the internal constraint is explicitly involved in the construction of \mathbf{F} , the 7-point algorithm always produces an estimate \mathbf{F} that satisfies $\det(\mathbf{F}) = 0$, so there is no need for the constraint enforcement step that we apply in the 8-point algorithm. Furthermore, all estimates of \mathbf{F} produced by the 7-point algorithm minimize $\varepsilon_{ALG,F}$ with zero residual. This means that the minimal case estimate is invariant to coordinate transformations, i.e., there is also no need for using H-normalized coordinates. The 7-point algorithm is summarized in Algorithm 16.5.

16.2.3 Degeneracies

In the derivation of the 8-point algorithm we assume that the 3D points that generate the image points are in general configuration, i.e., they do not satisfy any particular condition, *and* the cameras have distinct centers. If either of these two assumptions fail, we may get degenerate data from which any estimation method fails to uniquely determine the epipolar geometry.

⁴Here we are using Equation (3.156) in Toolbox Section 3.7.4.

Algorithm 16.5: The 7-point algorithm
--

Input: A set of 7 corresponding image points $\mathbf{y}_{1k}, \mathbf{y}_{2k}, k = 1, \dots, n$ Output: A set of 1, 2, or 3 fundamental matrices \mathbf{F} that fit Equation (16.23) exactly

```

1 Initialize:  $\mathbf{A}$  = empty  $0 \times 9$  matrix
2 foreach point pair  $j = 1, \dots, 7$  do
3   Compute vector  $\mathbf{a}_j$  in accordance with Equation (16.26)
4   Append  $\mathbf{a}_j$  as a row vector at the bottom of  $\mathbf{A}$ :  $\mathbf{A} = (\mathbf{A}; \mathbf{a}_j)$ 
5 end
6 Apply SVD to  $\mathbf{A} = \mathbf{U} \mathbf{S} \mathbf{V}^\top$ . There are (at least) two singular values in  $\mathbf{S}$  equal to zero
7 Set  $\mathbf{z}_1$  and  $\mathbf{z}_2$  to the corresponding right singular vectors in  $\mathbf{V}$ , and reshape  $\mathbf{z}_1$  and  $\mathbf{z}_2$  to  $\mathbf{F}_1$  and  $\mathbf{F}_2$ 
8 Determine the polynomial  $p$  in Equation (16.36) from Equation (16.37) – Equation (16.39), and find the real roots of  $p$ 
9 foreach real root  $\gamma$  do
10   Determine  $\mathbf{F}$  from Equation (16.34)
11 end

```

In Section 9.3 we saw that two cameras that share a common center produce images of 3D space that can be related by a homography. Similarly, in Section 9.2 it was proven that two images from cameras that depict 3D points lying on a plane can be related by means of a homography, even if they have distinct centers. In both these cases, corresponding points \mathbf{y}_1 and \mathbf{y}_2 are related as

$$\mathbf{y}_1 \sim \mathbf{H} \mathbf{y}_2, \quad (16.40)$$

or, equivalently, as

$$\mathbf{0} = [\mathbf{y}_1]_{\times} \mathbf{H} \mathbf{y}_2. \quad (16.41)$$

Here,

$$\mathbf{H} = \begin{pmatrix} - & \mathbf{h}_1 & - \\ - & \mathbf{h}_2 & - \\ - & \mathbf{h}_3 & - \end{pmatrix} \quad (16.42)$$

is a 3×3 matrix, with rows given by the three vectors $\mathbf{h}_k, k = 1, 2, 3$, which represents the homography that relates points in the two images. The algebraic form of Equation (16.41) is not exactly that of the epipolar constraint, but it can be reformulated as

$$0 = \mathbf{y}_1^T \underbrace{\begin{pmatrix} - & \mathbf{0} & - \\ - & \mathbf{h}_3 & - \\ - & -\mathbf{h}_2 & - \end{pmatrix}}_{:=\mathbf{F}_1} \mathbf{y}_2 = \mathbf{y}_1^T \underbrace{\begin{pmatrix} - & -\mathbf{h}_3 & - \\ - & \mathbf{0} & - \\ - & \mathbf{h}_1 & - \end{pmatrix}}_{:=\mathbf{F}_2} \mathbf{y}_2 = \mathbf{y}_1^T \underbrace{\begin{pmatrix} - & \mathbf{h}_2 & - \\ - & -\mathbf{h}_1 & - \\ - & \mathbf{0} & - \end{pmatrix}}_{:=\mathbf{F}_3} \mathbf{y}_2. \quad (16.43)$$

For the two degenerate cases, this result implies that there exist three linearly independent matrices, $\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3$, each defining an “epipolar constraint” for corresponding points. In fact, this is what we expect in this case. The epipolar constraint represented by the fundamental matrix, in the non-degenerate case, is a necessary but not sufficient condition for correspondence. The “not sufficient” character comes from the fact that each point \mathbf{y} in one image determines an epipolar line in the other image, and any point on this line can be in correspondence with \mathbf{y} . In the two degenerate cases that we refer to here, this uncertainty is removed and, instead, there is a one-to-one mapping between corresponding points, as described by Equation (16.40). Consequently, there should be additional constraints to produce both necessary *and* sufficient conditions for correspondence. It also means that all the geometric relations derived in Chapter 10, in terms of epipolar points, epipolar lines, and epipolar planes, are irrelevant in the case. The algebraic constraint $\det(\mathbf{F}) = 0$ is also not relevant in this case. To be sure, $\det(\mathbf{F}_k) = 0$ for $k = 1, 2, 3$, but any arbitrary linear combination of the three matrices, \mathbf{F} , also generates a valid constraint in this case, even if $\det(\mathbf{F}) \neq 0$.

In summary, the above observations imply that if an algebraic method is applied to degenerate data, such as the 8-point or 7-point algorithms, we should expect the solutions space to become at least 3-dimensional. In practice, this means that if the 3D points lie approximately in a plane, or if the camera centers are very close to each other, then the data matrix \mathbf{A} in Equation (16.28) will have more than one singular values that is small, or close to zero. This happens regardless of any normalization that is applied to the data, and can be detected by analyzing the singular values of \mathbf{A} . Only if there exists a well-defined unique smallest singular value of \mathbf{A} , when using a suitable data normalization, can we be confident that the estimate of \mathbf{F} is reliable and does not refer to degenerate data.

If we detect degenerate data, this implies that the epipolar geometry between the stereo images is not well-defined. The concept of a fundamental matrix may still be valid, at least if the camera centers are distinct, but the observed data does not support a unique instance of \mathbf{F} .

16.5 If the epipolar geometry is estimated from observed data in the form of corresponding image points, it is important that they correspond to 3D points that are in general configuration. In particular, the 3D points should not all be in or close to a plane.

16.2.4 Minimization of geometric errors

The 8-point algorithm is based on minimizing an algebraic error. As has been discussed in Section 12.5, algebraic errors have the advantage of often leading to simple estimation techniques that use linear methods to determine

the estimate. But they also have the disadvantage of minimizing a quantity which does not have a geometric interpretation. A better option is then to minimize geometric errors which, typically, leads to iterative approaches for determining the estimate. Such iterative methods require initial solutions, which typically can be provided by minimization of algebraic errors. In this section, we present an approach for estimating the fundamental matrix \mathbf{F} from a set of corresponding image points by minimizing geometric errors. First, we discuss a simple version of geometric minimization, and then present a more sophisticated method that is more reliable than the first one.

Based on point to epipolar line distance

Let $\{\mathbf{y}_{1k}, \mathbf{y}_{2k}, k = 1, \dots, n\}$ be a set of m corresponding points in the two images. A straightforward way to define a geometric error between pair of points $\mathbf{y}_{1k}, \mathbf{y}_{2k}$ and a specific fundamental matrix \mathbf{F} is as in Equation (10.15), using the distance between the points and the epipolar lines generated by the same point. An overall error for the entire set of n pairs of points relative to \mathbf{F} can be formulated as⁵

$$\varepsilon_{GEO,F} = \sum_{k=1}^n d_{PD}(\mathbf{y}_{2k}, \mathbf{F}^\top \mathbf{y}_{1k})^2 + d_{PD}(\mathbf{y}_{1k}, \mathbf{F} \mathbf{y}_{2k})^2. \quad (16.44)$$

For a given set of image points, this error is a function of the fundamental matrix \mathbf{F} , and we want to determine the global minimum of $\varepsilon_{GEO,F}$ over all choices of \mathbf{F} that satisfy the internal constraint $\det(\mathbf{F}) = 0$. As this error function is non-linear in any parameterization of \mathbf{F} , the optimal \mathbf{F} is determined by iterative approaches.

Although this minimization is practically feasible to compute, a comparison of this approach relative to the normalized 8-point algorithm shows that the residual geometric errors are approximately similar, although the latter method minimizes an algebraic error [66]. An explanation to this observation is related to the fact that although both $\mathbf{F}\mathbf{y}_2$ and $\mathbf{F}^\top \mathbf{y}_1$ are epipolar lines, the measurement noise on both image points implies that the two lines are not exactly *corresponding* epipolar lines. Due to the noise, the two epipolar lines are not the projection of a common epipolar plane. Consequently, the Euclidean distances that appear in Equation (16.44) are not correct relative to the epipolar geometry. For this reason, we leave this method here and instead look at a method that is slightly more complicated but, in general, offers a better result.

The gold standard method

One way to describe the estimation problem at hand is to say that we have a set of corresponding points that should be matched to a fundamental matrix. An alternative description is to say that the epipolar geometry represented by the fundamental matrix \mathbf{F} is associated to a pair of cameras $\mathbf{C}_1, \mathbf{C}_2$, and also that the set of corresponding image points can be associated with a set of 3D points. From $\mathbf{C}_1, \mathbf{C}_2$ we can compute \mathbf{F} from Equation (10.2), and from the set of 3D points we can compute their projections into the two camera images. Instead of fitting \mathbf{F} directly relative to the image points as we do in the previous method, we could try to find a pair of cameras, \mathbf{C}_1 and \mathbf{C}_2 , and a set of 3D points such that when they are projected into the two images, the resulting points end up as close as possible to the observed image points.

Let $\{\mathbf{x}_k, k = 1, \dots, n\}$ be a set of n 3D points, and $\{\mathbf{y}_{1k}, \mathbf{y}_{2k}, k = 1, \dots, n\}$ be the set of image points from which we estimate \mathbf{F} . The geometric error that should be minimized is defined as⁶

$$\varepsilon_{GS} = \sum_{k=1}^n d_{PP}(\mathbf{C}_1 \mathbf{x}_k, \mathbf{y}_{1k})^2 + d_{PP}(\mathbf{C}_2 \mathbf{x}_k, \mathbf{y}_{2k})^2. \quad (16.45)$$

With fixed image points, $\mathbf{y}_{1k}, \mathbf{y}_{2k}$, we minimize ε_{GS} over $\mathbf{C}_1, \mathbf{C}_2$ and all the n 3D points \mathbf{x}_k . This is the basic approach of the *gold standard method* for the estimation of \mathbf{F} . The implementation of the geometric error ε_{GS} , means that the gold standard method tries to determine a maximum likelihood estimate⁷ of the ideal image points in the two images, with the additional constraint that they must all be consistent with the epipolar geometry, i.e., they must all satisfy the epipolar constraint relative one and the same \mathbf{F} . To implement and use this approach successfully, however, there are some issues that must be resolved.

As a start, we need initial estimates of the two camera matrices, \mathbf{C}_1 and \mathbf{C}_2 , and of the n 3D points \mathbf{x}_k . One approach is to first obtain an initial estimate of \mathbf{F} , for example by means of the normalized 8-point algorithm. The

⁵ d_{PD} is the point-to-line distance defined in Equation (3.34).

⁶ d_{PP} is the point-to-point distance defined in Equation (3.31).

⁷ This is a maximum likelihood estimate under the assumptions described in Section 12.6.

cameras can then be determined using the approach described in Section 10.3.4, more specifically Algorithm 10.1 on page 150 or Algorithm 10.2 on page 151. This produces a pair of camera matrices that are consistent with the initial estimate of \mathbf{F} , although relative to a 3D space where an unknown 3D homography has been applied. Once the cameras are determined, we apply some triangulation method to determine estimates of the 3D points given the set of corresponding image points. Examples of such triangulation methods are presented in Section 16.1. The resulting 3D points are represented in a 3D coordinate system that has been transformed by an unknown 3D homography, but both cameras and all 3D points refer to one and the same 3D coordinate system, the specific coordinate system that is used is not important for these computations.

A second observation to be made for this case, is that the 3D points are *auxiliary variables*. They take an active role in the estimation process, since we allow ε_{GS} to be minimized over different choices of the cameras *and* the 3D points. But in the end, after the minimization is done, we typically discard the 3D points since they cannot be related to a known world coordinate system. This also means that the degrees of freedom in the optimization is no longer restricted to the 7 degrees of freedom related to \mathbf{F} , but in addition it has 3 degrees of freedom for each of the n 3D point. Therefore, in total we need to optimize over $7 + 3n$ degrees of freedom.

This means that the computational complexity, in terms of time and memory needed to perform the necessary computations to obtain the estimate, grows with n . However, a careful implementation of the optimization can make this complexity grow linearly with n , which in practice makes it possible to apply this method also on problems where n is large. Also, as for any ML-estimator, we expect the result to become better the more points that are used in the estimation. In practice, there has to be a balance between n and the computational complexity, which can be resolved only by looking at the details of the specific application where the gold standard method is used.

We also see that the parameters that are optimized over in this method are the two camera matrices and the n 3D points. Since all these geometric objects refer to a 3D space with, in general, a non-Cartesian coordinate system, it is not entirely without risk to apply P-normalization to the resulting homogeneous coordinates of the 3D points that result from the triangulation. It may be the case that some of all of the 3D points are close to being at infinity, which can cause instability in the optimization. For this reason, it may in some cases be necessary to apply a normalizing transformation in 3D space such that the 3D points lie close to the origin and are equally spread out in all directions. The corresponding inverse transformation must then be applied to the camera matrices in order to keep the projection into the images invariant.

We have seen in Section 10.3.4 that we can choose one of the two cameras as fixed, for example $\mathbf{C}_1 = (\mathbf{I} | \mathbf{0})$, and only vary the other camera, \mathbf{C}_2 , in the optimization. Based on the results in Section 10.3.3 and Section 10.3.4 we can opt for a minimal parameterization of the cameras using \mathbf{C}_2 in Equation (10.37), where $\sigma_1 = 1$ and we allow $\mathbf{U}, \mathbf{V} \in SO(3)$ and σ_2 to vary. In practice, the additional cost of allowing \mathbf{C}_2 to vary over its full 11 degrees of freedom is often not a problem, in particular when m is large. For example, once an initial estimate of \mathbf{C}_2 is determined, we can write it as in Equation (10.28), and \mathbf{F} is then given directly from Equation (10.30). The gold standard method for estimation of \mathbf{F} is summarized in Algorithm 16.6.

Algorithm 16.6: The gold standard method for estimation of \mathbf{F}
--

Input: A set of n corresponding image points $\mathbf{y}_{1k}, \mathbf{y}_{2k}, k = 1, \dots, n$.

Output: An ML-estimate of the fundamental matrix \mathbf{F} .

- 1 Compute initial estimates of \mathbf{F} using the normalized 8-point algorithm, Algorithm 16.4 on page 285
- 2 Compute initial estimates of the cameras \mathbf{C}_1 and \mathbf{C}_2 from Algorithm 10.1 on page 150 or Algorithm 10.2 on page 151
- 3 **foreach** $k = 1, \dots, n$ **do**
- 4 From \mathbf{C}_1 and \mathbf{C}_2 and $\mathbf{y}_{1k}, \mathbf{y}_{2k}$ triangulate the 3D point \mathbf{x}_k
- 5 Use any of the methods described in Section 16.1
- 6 **end**
- 7 With $\mathbf{C}_1, \mathbf{C}_2$ and $\mathbf{x}_k, k = 1, \dots, n$, as initial estimates, minimize ε_{GS} in Equation (16.45) over $\mathbf{C}_1, \mathbf{C}_2$ and the n 3D points
- 8 From the optimal values of \mathbf{C}_1 and \mathbf{C}_2 , compute optimal \mathbf{F} from Equation (10.2)

16.3 Estimation of \mathbf{E}

For the case of calibrated epipolar geometry, we can be interested in estimating the essential matrix from a set of corresponding C-normalized image coordinates. Alternatively, we have a set of corresponding points in pixel coordinates, but we also know the internal calibrations for each of the two cameras, represented by the two matrices \mathbf{K}_1 and \mathbf{K}_2 .

There are several approaches to this problem in the literature, and here we present a simple adaptation of the 8-point algorithm to the estimation of \mathbf{E} , and also discuss a minimal case estimator.

16.3.1 Algebraic estimation

Since \mathbf{E} is nothing but a special type of fundamental matrix, based on C-normalized image coordinates, it should be possible to use the normalized 8-point algorithm for the estimation of \mathbf{E} . In this case, we can either apply the normalized 8-point algorithm in Algorithm 16.4 on page 285 to the C-normalized image coordinates and obtain \mathbf{E} directly, or apply the normalized 8-point algorithm to the pixel coordinates, and then transform the resulting fundamental matrix \mathbf{F} to \mathbf{E} by means of \mathbf{K}_1 and \mathbf{K}_2 using Equation (10.50). In general, the two approaches do not produce identical results, but the difference is often very small. What can be more problematic is instead that the estimated \mathbf{E} does not satisfy the internal constraint for an essential matrix, Equation (10.58). To be sure, the 8-point algorithm guarantees that one singular value of \mathbf{E} vanishes, but not that the other two are equal.

Constraint enforcement

To deal with this issue, we can apply a stronger type of constraint enforcement than is used for the estimation of \mathbf{F} . Given an initial estimate \mathbf{E}_0 from the normalized 8-point algorithm, we apply an SVD:

$$\mathbf{E}_0 = \mathbf{U} \mathbf{S} \mathbf{V}^\top, \quad (16.46)$$

and set

$$\mathbf{E} = \mathbf{U} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{V}^\top. \quad (16.47)$$

This constraint enforcement must be made in the C-normalized coordinate systems in which the essential matrix is defined. It cannot be made in other coordinate systems and then transformed back to C-normalized coordinates, since then the constraint may no longer be valid after the transformations. Furthermore, we are here dealing with the minimization of an algebraic error which, in general, can produce useful results but is not guaranteed to minimize a reasonable geometric error. In particular this is the case since we are now manipulating the minimizer of the algebraic error even more than is the case for the estimation of \mathbf{F} .

Two alternatives of this type of adaptations of the 8-point algorithm are presented in Algorithm 16.7. In the case that both sets of image coordinates already are C-normalized, no transformation is needed and either of the two alternatives can be used. If alternative 1 is used, the SVD that is used for the constraint enforcement of \mathbf{E} can be combined with the SVD that is applied for the constraint enforcement of \mathbf{F} in the 8-point algorithm.

16.3.2 Minimal case estimation of \mathbf{E}

The essential matrix \mathbf{E} has 5 degrees of freedom, and each pair of corresponding points provides a single constraint on \mathbf{E} . Thus, it should be possible to determine \mathbf{E} that satisfies all internal constraints by means of only 5 pairs of corresponding C-normalized image points. This observation is correct and the result can be seen as an adaptation of the 7-point algorithm for estimation of \mathbf{F} to the case when the resulting matrix instead must satisfy Equation (10.58), the so-called *5-point algorithm*. The derivation of the 5-point algorithm is relatively complex and therefore outside the scope of this presentation. A complete presentation of the 5-point algorithm can be found in the original publications by Nistér [52, 53]. In principle, it has a similar approach as the 7-point algorithm for estimation of \mathbf{F} , described in Section 16.2.2. The 5-point algorithm constructs a 9×5 data matrix \mathbf{A} , with a 4-dimensional null space that contains the possible solutions. This means that any \mathbf{E} , consistent with the observed data, is a linear combination of 4 matrices that can be determined from \mathbf{A} . If we insert such a linear combination

Algorithm 16.7: An adaptation of the 8-point algorithm to estimation of \mathbf{E}

Input: A set of n corresponding image points $\mathbf{y}_{1k}, \mathbf{y}_{2k}, k = 1, \dots, n$, in pixel coordinates.

Input: Internal calibrations \mathbf{K}_1 and \mathbf{K}_2 for the two cameras.

Output: An estimate of the essential matrix \mathbf{E} .

1 Alternative 1

2 Transform image coordinates to C-normalized coordinates: $\mathbf{y}'_{1k} = \mathbf{K}_1^{-1}\mathbf{y}_{1k}$ and $\mathbf{y}'_{2k} = \mathbf{K}_2^{-1}\mathbf{y}_{2k}$.

3 Determine an initial estimate \mathbf{E}_0 using Algorithm 16.4 on page 285

4 Alternative 2

5 Estimate \mathbf{F} using Algorithm 16.4 on page 285

6 Compute $\mathbf{E}_0 = \mathbf{K}_1^\top \mathbf{F} \mathbf{K}_2$

7 Compute SVD: $\mathbf{E}_0 = \mathbf{U} \mathbf{S} \mathbf{V}^\top$

8 Set $\mathbf{E} = \mathbf{U} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{V}^\top$

into the internal constraints of \mathbf{E} , Equation (10.58), we obtain a number of non-linear constraints on the coefficients of the linear combination. It is non-trivial to find coefficients that satisfy all these non-linear constraints, but it boils down to formulating a 10-th order polynomial, where each real root corresponds to an essential matrix that satisfies the epipolar constraint in Equation (10.51) relative to all 5 pairs of corresponding points, and that also is valid relative to the internal constraints in Equation (10.58).

The fact that the 5-point algorithm is based on a 10-th order polynomial implies that we can get up to 10 valid solutions to the minimal case estimation problem for the essential matrix. It also means that we need to apply accurate computations, both to determine the coefficients of this polynomial and to determine its roots. Otherwise, the numerical reliability of the result is not useful for further computations. This issue calls for carefully crafted numerical implementations of the 5-point algorithm.

16.4 Estimation of ω from a rotational homography

In Section 9.3 we discussed the problem of determining the internal camera parameters \mathbf{K} and the rotation matrix \mathbf{R} related to a known rotational homography $\mathbf{H} = \mathbf{K} \mathbf{R} \mathbf{K}^{-1}$. On the way to solving this problem, we concluded that \mathbf{H} should be normalized such that $\det \mathbf{H} = 1$, and the image of the absolute conic (IAC), defined as $\boldsymbol{\omega} = \mathbf{K}^{-\top} \mathbf{K}$, was introduced. \mathbf{K} is then the Cholesky factor of $\boldsymbol{\omega}$, and finding \mathbf{K} and \mathbf{R} amounts to solving $\boldsymbol{\omega}$ from the equation

$$\mathbf{H}^\top \boldsymbol{\omega} \mathbf{H} = \boldsymbol{\omega}, \quad \text{where } \mathbf{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}, \quad \boldsymbol{\omega} = \begin{pmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{12} & \omega_{22} & \omega_{23} \\ \omega_{13} & \omega_{23} & \omega_{33} \end{pmatrix}. \quad (16.48)$$

In the end, we concluded that this equation has no unique solution $\boldsymbol{\omega}$.

In this section, we return to this problem and formulate a strategy for finding the IAC using *multiple* rotational homographies. As first step, we notice that Equation (9.31) is a linear and homogeneous equation in $\boldsymbol{\omega}$. By a vectorization $\tilde{\boldsymbol{\omega}}$, and taking into account that $\boldsymbol{\omega}$ is a symmetric matrix, this equation can also be formulated as

$$\tilde{\mathbf{H}} \tilde{\boldsymbol{\omega}} = \mathbf{0}, \quad (16.49)$$

where

$$\tilde{\mathbf{H}} = \begin{pmatrix} h_{11}^2 - 1 & 2h_{11}h_{21} & 2h_{11}h_{31} & h_{21}^2 & 2h_{21}h_{31} & h_{31}^2 \\ h_{11}h_{12} & h_{21}h_{12} + h_{11}h_{22} - 1 & h_{31}h_{12} + h_{11}h_{32} & h_{21}h_{22} & h_{31}h_{22} + h_{21}h_{32} & h_{31}h_{32} \\ h_{11}h_{13} & h_{21}h_{13} + h_{11}h_{23} & h_{31}h_{13} + h_{11}h_{33} - 1 & h_{21}h_{23} & h_{31}h_{23} + h_{21}h_{33} & h_{31}h_{33} \\ h_{12}^2 & 2h_{12}h_{22} & 2h_{12}h_{32} & h_{22}^2 - 1 & 2h_{22}h_{32} & h_{32}^2 \\ h_{12}h_{13} & h_{22}h_{13} + h_{12}h_{23} & h_{32}h_{13} + h_{12}h_{33} & h_{22}h_{23} & h_{32}h_{23} + h_{22}h_{33} - 1 & h_{32}h_{33} \\ h_{13}^2 & 2h_{13}h_{23} & 2h_{13}h_{33} & h_{23}^2 & 2h_{23}h_{33} & h_{33}^2 - 1 \end{pmatrix}, \quad \tilde{\boldsymbol{\omega}} = \begin{pmatrix} \omega_{11} \\ \omega_{12} \\ \omega_{13} \\ \omega_{22} \\ \omega_{23} \\ \omega_{33} \end{pmatrix}. \quad (16.50)$$

We consider now the case where three or more images are taken with a single camera that rotates about its center, and where the internal parameters in \mathbf{K} are fixed. Given that we can find correspondences between two or more image pairs, we can then estimate a rotational homography \mathbf{H}_k , $k = 1, \dots, m$ for each such image pair. This can be done with the algebraic method described in Algorithm 13.3 on page 238, or the geometric method described in Section 14.5. Since the internal parameters are fixed, ω is also fixed, and must satisfy Equation (9.31) for each \mathbf{H}_k :

$$\mathbf{H}_k^\top \boldsymbol{\omega} \mathbf{H}_k = \boldsymbol{\omega}, \quad k = 1, \dots, m. \quad (16.51)$$

Alternatively, using the vectorization just described for $\boldsymbol{\omega}$:

$$\tilde{\mathbf{H}}_k \tilde{\boldsymbol{\omega}} = \mathbf{0}, \quad k = 1, \dots, m, \quad (16.52)$$

where $\tilde{\mathbf{H}}_k$ is defined from \mathbf{H}_k in the same way as $\tilde{\mathbf{H}}$ is defined from \mathbf{H} , as specified in Equation (16.50).

The results we derived in Section 9.3 imply that the linear and homogeneous Equation (9.31) has a three-dimensional solution space of 3×3 matrices. It is a straight-forward exercise to show that one of these dimensions consists of anti-symmetric matrices. This dimension vanishes when $\boldsymbol{\omega}$ is forced to be symmetric, as stipulated by the specific vectorization that is used in Equation (16.50). Consequently, the solution space of Equation (16.52) is only two-dimensional when $m = 1$, but this still makes it impossible to uniquely determine $\boldsymbol{\omega}$ from a single rotational homography.

But already when $m \geq 2$, there is in general a unique $\boldsymbol{\omega}$ that solves Equation (16.52) for all $k = 1, \dots, m$. This observation can be used as a basis for a methods that estimates $\boldsymbol{\omega}$ based on minimizing an algebraic error, described in the next section. In this case, we should be careful not to include homographies generated by two or more rotations about the same axis, as these produce matrices $\tilde{\mathbf{H}}_k$ that are linearly dependent.

16.4.1 Algebraic estimation

A simple method for algebraic estimation of $\boldsymbol{\omega}$ from rotational homographies, with a common \mathbf{K} can be formulated as follows. For each pair of images from the rotating camera, with a set of corresponding points, a homography \mathbf{H}_k is estimated using the methods in Section 14.5. Each matrix \mathbf{H}_k should be normalized such that $\det \mathbf{H}_k = 1$. To each \mathbf{H}_k , the corresponding 6×6 matrix $\tilde{\mathbf{H}}_k$ is formed from Equation (16.50), and these matrices together they form a $6m \times 6$ matrix \mathbf{A} :

$$\mathbf{A} = \begin{pmatrix} \tilde{\mathbf{H}}_1 \\ \vdots \\ \tilde{\mathbf{H}}_m \end{pmatrix}, \quad \text{such that } \mathbf{A} \tilde{\boldsymbol{\omega}} = \mathbf{0}. \quad (16.53)$$

In the end, and as usual for algebraic estimation, we find the model estimate, here of $\tilde{\boldsymbol{\omega}}$, in the null space of the data matrix \mathbf{A} . This can be done with the inhomogeneous method or the inhomogeneous method. Finally, we obtain the IAC $\boldsymbol{\omega}$ by reshaping $\tilde{\boldsymbol{\omega}}$ back to a symmetric 3×3 matrix, Equation (16.48).

Finding also \mathbf{K}

In many practical applications, estimating $\boldsymbol{\omega}$ is just an intermediate step to determine \mathbf{K} , the internal camera parameters. As already pointed out, $\mathbf{K}^{-\top}$ is the Cholesky factor of $\boldsymbol{\omega}$, which means that \mathbf{K} is uniquely determined from $\boldsymbol{\omega}$ by a Cholesky decomposition, followed by a matrix inverse and transposition.

An alternative approach, should Cholesky decomposition not be available, is to solve for the elements in \mathbf{K} from $\boldsymbol{\omega} = \mathbf{K}^{-\top} \mathbf{K}^{-1}$. This is left an exercise, where the result can be formulated as

$$\begin{aligned} \delta &= \omega_{11}\omega_{22} - \omega_{12}^2, & k_{13} &= \frac{\omega_{12}\omega_{23} - \omega_{13}\omega_{22}}{\delta}, & k_{23} &= \frac{\omega_{12}\omega_{13} - \omega_{11}\omega_{23}}{\delta}, \\ \lambda &= k_{13}\omega_{13} + k_{23}\omega_{23} + \omega_{33}, & k_{11} &= \sqrt{\frac{\lambda}{\omega_{11}}}, & k_{22} &= \sqrt{\frac{\lambda\omega_{11}}{\delta}}, & k_{12} &= -\frac{\omega_{12}k_{11}^2 k_{22}}{\lambda}. \end{aligned} \quad (16.54)$$

These expressions satisfy $\boldsymbol{\omega} = \lambda \mathbf{K}^{-\top} \mathbf{K}^{-1}$ where \mathbf{K} is normalized such that $k_{33} = 1$. Notice that $k_{11}, k_{22} > 0$ leads to $\delta > 0$, and that λ and ω_{11} always have the same sign. Thus, the square roots are applied to positive numbers.

This approach to algebraic estimation of $\boldsymbol{\omega}$, outlined in this section, and to determine \mathbf{K} , is summarized in Algorithm 16.8. For some applications, algebraic estimation may not provide a sufficiently accurate estimate of $\boldsymbol{\omega}$, instead geometric estimation is required. For the case of rotating cameras, we will return to geometric estimation of $\boldsymbol{\omega}$ in Section 18.1.1, in the context of camera calibration.

Algorithm 16.8: Algebraic estimation of ω and \mathbf{K}

Input: Correspondences in $m \geq 2$ image pairs, generated by a rotating camera with constant internal parameters \mathbf{K} .

Output: An estimate of ω and \mathbf{K} .

- 1 Set $\mathbf{A} = []$, an empty matrix
- 2 **for** $k = 1, \dots, m$ **do**
- 3 Use correspondences from image pair k to estimate homography \mathbf{H}_k .
- 4 *Use a linear method from Section 13.1 or a non-linear method from Section 14.5.*
- 5 Form $\tilde{\mathbf{H}}_k$ from \mathbf{H}_k , Equation (16.50)
- 6 Concatenate $\tilde{\mathbf{H}}_k$ at the bottom of \mathbf{A} : $\mathbf{A} = [\mathbf{A}; \tilde{\mathbf{H}}_k]$
- 7 **end**
- 8 Estimate $\tilde{\omega} \in \mathbb{R}^6$ that solves $\mathbf{A} \tilde{\omega} = \mathbf{0}$. Use the inhomogeneous method, Algorithm 12.1, or the homogeneous method, Algorithm 12.1
- 9 Reshape $\tilde{\omega}$ back to 3×3 symmetric matrix ω , Equation (16.48)
- 10 *Optional:* determine the elements of \mathbf{K} by a Cholesky decomposition of ω , followed by matrix inverse and transpose. Alternatively, use Equation (16.54)

Chapter 17

Robust Estimation

Before you read this chapter, you should have thorough understanding of the linear estimation methods described in Chapters 12 and 13, and also of the non-linear methods described in Chapter 14.

At this point in the presentation we have seen several approaches to estimation problems in geometry. When estimating a geometric object, we can use different types of errors, and they can be solved by various techniques, ranging from solving linear equations to iterative non-linear optimization. There is also a class of dedicated solutions, that are specialized for a particular type of estimation problem, such as the methods for estimation of 3D rotations described in Section 15.1.

A general observation for all these methods is that the error of the estimate, relative to an ideal estimate from data without any noise, is expected to grow with the amount of noise. This dependency may not be linear, and it may not even be monotonic. But in the end, we expect small amount of noise to perturb the estimate only a little, while large amounts of errors should cause large perturbations and may even produce useless estimates.

That perturbations of the estimate should be proportional to perturbations of the data, however, is not entirely correct. In this chapter we will discuss reasons why it is not correct, and also look at some approaches that can deal with this issue. These approaches are collectively put under the heading *robust estimation*, they aim at producing estimates that are robust relative to the perturbations of the noise.

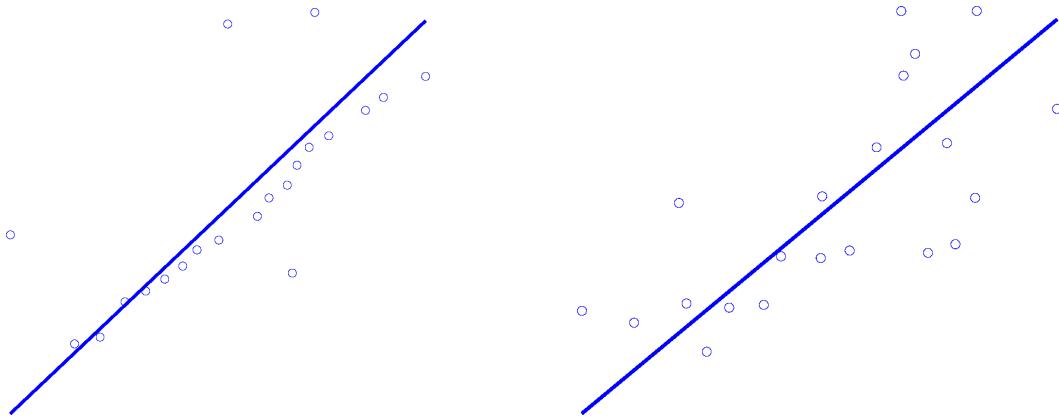


Figure 17.1: Two sets of points that give approximately the same total least squares error ε_{TOT} . Left: most points are affected by a relative small amount of noise, except for a few points that have larger errors. Right: most points are affected by a relative large amount of noise.

17.1 What causes the errors? (Part II)

In Section 12.3 we started to discuss common causes of errors that appear in estimation. Here, we continue the discussion and consider additional sources of error.

17.1.1 Inliers and outliers

In observation 12.6 on page 199 we said that it is important to choose a model that is suitable to the data that is used. Once the model is chosen, it is equally important to choose the data carefully. Consider the points in the left part of Figure 17.1. Most of them are approximately lying on a line, but there are a few of them that are very far from this line. In this context we refer to the first subset of points as *inliers*, with a reasonable error they can be fitted to whatever model that is relevant for a particular problem. The second subset, which cannot easily be fitted to the model, are the *outliers*. As is illustrated in the right part of the figure, the classification of the dataset into inliers and outliers is not always straightforward. In practice, a division of the dataset into inliers and outliers can only be made with some degree of probability or certainty.

Correspondences

A common example of outliers is related to the correspondence problem, described in Section 9.1.2. Any method for establishing correspondences between two or more sets of points occasionally fails, even if we do it manually. We have to assume that at least a small fraction of the correspondences that are produced are incorrect. If the correspondences are used for estimating, for example, a homography based on the procedure described in Section 13.1, it is likely that the resulting homography has a relatively large error, even though it is optimally fitted to the data. It does not matter if the coordinates are perturbed by insignificant amounts of noise, if we combine points from the two images in an incorrect way, the estimated homography can be affected significantly, in general in a much more severe way than if some noise is added to the coordinates of the points.

Incorrect correspondences can be seen as a sort of “high level noise”, in the sense that it does not relate to inaccuracies in the measurements of geometric features, such as coordinates or line parameters, but rather to incorrect interpretations of these measurements. These incorrect correspondences typically produce data that have a relatively large error to any model that is determined from data that is free from such points, i.e., incorrect correspondences represent outliers in the dataset.

17.1 The data which the model is fitted to should be carefully chosen. In particular, if the data includes correspondences between points, or other geometric objects, these correspondences should be reliably determined.

A common issue related to estimation in geometry is that not only is the input data perturbed by measurement noise, there may also be a significant number of outliers due to incorrect correspondences. In fact, we may have to estimate a model from data that must be in correspondence to make the estimation meaningful, but no correspondences are yet established. In other applications, there may be more outliers than inliers, and still we want to determine a model that fits the inliers. In these cases, the problem is that we do not know beforehand which points in the dataset are the inliers and which are the outliers, or which correspondences are correct and which are not. Estimation methods dedicated to this type of situation are referred to as *robust estimation*. Since solving the correspondence problem is a critical issue in many of the applications discussed in this presentation, we will often use robust estimation to refer to the problem of solving the correspondence problem in relation to estimation of some geometric object.

There are several strategies that can be employed in the case that the dataset contains outliers, and in the chapter we discuss two of them. The first strategy can be characterized as somewhat milder, and is used when there is only a small fraction of outliers that potentially can influence the estimated model, and this influence is moderate. It is based on so-called robust errors, which limit the influence of the outliers on the estimated model without explicitly classifying each data point as inlier or outlier. The second strategy is RANSAC. It is more aggressive, and tries to more or less eliminate the outliers from the dataset, thereby providing an explicit partitioning of the dataset into inliers and outliers. This comes at a cost: it produces a useful result with a probability that only can be close, but not equal, to 100%, and at a computational complexity that increases with this probability.

17.2 Robust errors

If we return to the simple example of estimating a line from a set of 2D point, we have formulated geometric errors based on a summation of distances, measured in one way or another, between each of the points and the model line. Often these distances are squared before added to an overall error to be minimized. In general, this means that if one or a few of the points deviate quite a lot from the model, the line, this point or points can have a large effect on the estimated line, even if the rest of the points can be fitted to the model. For example, see Figure 17.1, left, showing a point-set and the result of fitting a line by minimizing ϵ_{TOT} in Equation (12.16). In this example, the resulting line has a relatively large error to many of the points, because a few of the points influence the estimation although they do not adhere to the line model. The fact that most of the points can be fitted quite nicely to a line is not really taken into account here. The right part of the same figure shows another set of points and the resulting estimated line. It has approximately the same residual error as the line to the left relative to its dataset, but in this case the error is more evenly distributed over the dataset. It is not a straightforward to state which specific points that are consistent with the line model and which are not. The type of errors considered so far do not distinguish between these two cases.

To deal with this issue, to not allow very large errors from one or a few individual points to have a large influence on the estimation, we can use so-called *robust errors*. They are typically implemented by applying some type of normalizing function f on the individual point errors, e.g., ϵ_i'' in Equation (12.15), before they are added to an overall error:

$$\epsilon_{\text{ROBUST}} = \sum_{i=1}^m f(\epsilon_i''). \quad (17.1)$$

The model is then estimated by minimizing the robust error measure ϵ_{ROBUST} . Two examples of the function f are illustrated in Figure 17.2. In the left figure we see a function of some variable x in the form of a combination of x^2 , close to the origin, and a linear function outside this region. Since there is no squaring for large x , large errors are not given an extra emphasis in the same way as if a square is used. On the other hand, the square-like function close to the origin implies that the error function is continuously differentiable and that the gradient of the overall error function is well-defined everywhere. In the right figure we see an error function that is truncated in a smooth way at some suitable level ϵ_t . No individual error between a point and the model line can get a larger value than ϵ_t , and therefore not influence the optimization too much. In a similar way as in the left figure, this error function is also continuously differentiable everywhere, and the gradient of the overall error is well-defined everywhere. In both cases, however, the optimization of the overall error has to be done by means of iterative methods.

In principle we can also use a robust overall error defined in terms of the median of all individual errors:

$$\epsilon'_{\text{ROBUST}} = \text{median}_i |\epsilon_i''|. \quad (17.2)$$

This error disregards points with very large errors as long as there are only a few such points. This function, however, has partial derivatives with respect to the model parameters that are zero almost everywhere, an observation that also is true for its gradient, and for this reason it is not useful in practice.

We summarize this section by making the following observation:

17.2 In the case that non-linear methods are used for estimation, it may be an advantage to also include robust errors in the formulations of the error functions. This reduces the influence of points that are not consistent with the model.

Although robust errors can be useful in many applications there is a limit to how much they can help. They can be optimized only by non-linear iterative methods, which require initial estimates of the solution, and are therefore dependent on good initial solutions. These can be obtained by solving simpler estimation problems, typically based on algebraic errors, and they are not robust to outliers. Consequently, the outliers may degrade the initial solutions sufficiently much to render the minimization of robust errors ineffective. This implies that robust errors only can be expected to lead to an effective reduction of outlier influence when the outliers have a moderate influence on non-robust estimation methods. In most practical cases, the correspondence problem cannot be solved by means of robust errors. In this case, RANSAC may be used instead.

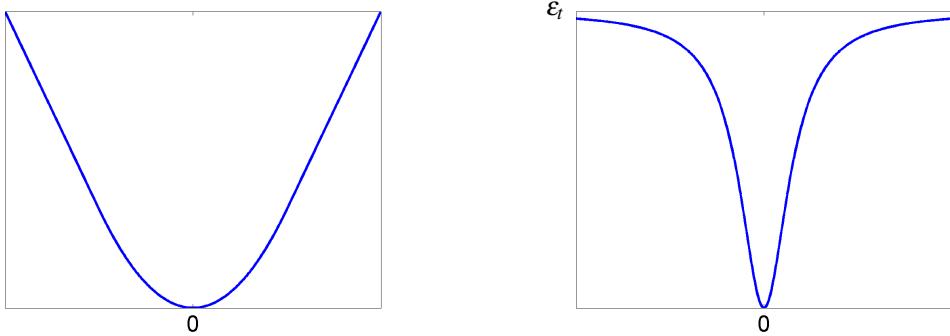


Figure 17.2: Two examples of robust error functions (vertical axis) relative to some preliminary error x (horizontal axis). Left: A combination of a linear for larger x , and a quadratic function for small x . Right: A continuous function that is truncated at ε_t .

17.3 RANSAC

In many practical estimation problems, the dataset includes outliers that have a significant influence on the estimate, unless they somehow can be removed. If not, the estimation process is fragile as it suffices with a single really bad outlier to make the estimated model really bad, even though the rest of the dataset has a low amount of noise. This is illustrated in Figure 17.3, where a line is estimated from a set of points that contains a single outlier. In some cases, we may use an “estimate, classify, and re-estimate approach”. It estimates an initial model from the entire dataset and uses this model to determine which data points are inliers and which are outliers. The underlying assumption is that the estimated model in general should be closer to the inliers than to the outliers. Once the inliers have been determined in this way, we can then re-estimate the model from only the inliers.

Although this approach may work fine in some cases, and similar to robust errors, it is of limited application. It may work if the number of outliers is relatively small compared to the inliers, or if the outliers are distributed in an even way around the inliers so that their influence on the estimated model parameters cancels. In practice, these ideal cases are not common and we need alternative strategies. For example, the proposed approach will

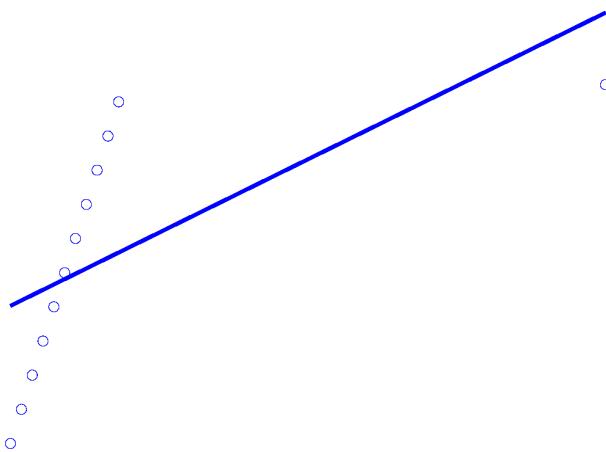


Figure 17.3: A line is estimated from a set of points using the total least squares error. The data points do not have any noise, but there is a single outlier that totally ruins the estimation.

probably not improve the result illustrated in Figure 17.3, since most of the inliers lie further away from the initially estimated line than the outlier does. This is what robust estimation is about: to determine the model parameters from a dataset with a significant number of outliers. The term “significant” means here that the outliers significantly perturb the model estimation, even if their sheer number is small.

17.3.1 An indeterministic approach

As an alternative strategy, we can consider indeterministic estimation methods. Before we do that, let us review the steps of the estimate, classify, and re-estimate approach discussed above. It produces an initial estimate of the model from the entire dataset and then tries to determine which data points are consistent with the estimated model, the inliers, and which are not consistent, the outliers. Although this approach has weaknesses, let us be more precise on what these shortcomings are. If the dataset is known to contain outliers, it is completely certain that the model has been estimated from a set of points that includes outliers. What if we divide the dataset into two halves? Since we do not know which points are inliers and which are outliers, any such dissection of the dataset leaves us with two subsets that each can contain both inliers and outliers. However, while the entire dataset contains a certain fixed number of outliers relative to inliers, the relation of outliers to inliers in the two halves follows a probability distribution if we select the halves at random. It is possible to dissect the dataset into halves where, for each of them, this ratio is either smaller or larger than it is for the entire dataset.

Based on this observation, we can make an initial attempt toward a robust estimation method by repeatedly, and each time randomly, dissecting the dataset into halves and estimating a model from each of the two halves. In this process, and with some probability, we will then dissect the dataset into a half that contains a minimal amount of outliers. When this happens, we expect that the estimated model is a better one than what we obtained when it is estimated from the entire dataset. To make this into a useful approach, however, we need to detect when this optimal dissection of the dataset occurs. We leave this problem aside for the moment, and instead make the obvious observation that if we dissect the dataset into halves a finite number of times, the “minimal outlier ratio” case may have occurred, or not, when we stop. Furthermore, the chance that this event happens depends on how many trials for the dissection we make. This is a characterizing feature of an *indeterministic estimation method*, it gives a useful result only with a certain probability that depends on how many trials are used. In practice, this number may be very large in order to assure a sufficiently high probability of success, but it may still be manageable by a modern computer.

This initial attempt at an indeterministic approach is problematic from at least two points of view. First, if the inliers constitute 50% or less of the data, dissecting the dataset into halves will never produce a half that contains only inliers and the estimation is then always perturbed by the outliers. However, the main results of this discussion are valid also if, at each trial, we select a smaller subset T from the dataset, a *trial set*. In fact, the smaller T is, the higher is the probability that a random selection of data points in T contains only inliers. This implies that T should contain as few data points as possible, preferably not more than is absolutely necessary to determine a model using minimal case estimation. This maximizes the probability that we produce a model estimated from only inliers, given that a specific T is randomly chosen from the dataset. As before, we repeatedly make random selections of the trial set T from the dataset, and do this sufficiently many times to assure a useful probability of selecting a T of only inliers.

Second, we need some way of classifying a specific choice of T as containing only inliers or as containing at least one outlier. This cannot be done with certainty, but we can apply a heuristic approach as follows. We assume that T is a relatively small subset of the dataset, and from T we can estimate a model. If T contains only inliers, the estimated model ought to fit also the remaining inliers in the dataset, those not already in T . Thus, from T we form a *consensus set* C , consisting of all data points that satisfy the model estimated from T . This implies that we need to determine an error between an individual data point and the estimated model. If this error is below some threshold, we assign that data point to C , otherwise not. If the estimated model works for the inliers of the dataset, the resulting consensus set C should be of the same size as the number of expected inliers in the dataset. Consequently, we need to know, or have an estimate of, the ratio of inliers relative to the total dataset. If C contains approximately this amount of points, it is very probable that C contains only inliers. It may not contain all of the inliers, and it may contain also a few outliers that by chance can be fitted to the estimated model but, in general, C should contain a much higher fraction of inliers to outliers than the original dataset does.

In practice, the ratio of inliers to outliers may not be known. What we can do in this case, is to simply keep the largest consensus set over the trials that we make. As long as we make sufficiently many trials to assure a

high probability of selecting a T with only inliers, it highly likely that this has happened for the largest C over all trials. This basic approach to indeterministic robust estimation is called *Random Sample Consensus* or *RANSAC* for short, and was introduced by Fischler and Bolles [20]. It can be used for solving a variety of estimation problems where outliers occur, in particular the correspondence problem. In the literature it is possible to find a variety of extensions and modifications of the basic RANSAC algorithm, some of which are useful only for particular estimation problems.

17.3.2 Robust estimation of a 2D line

To illustrate the RANSAC algorithm, let us apply it to estimation of a line from a set of 2D points that contain outliers. The smallest set T from which we can determine a line has 2 data points. Consequently, we iteratively select two random points from the data set, forming T , and determine the intersecting line \mathbf{l} of the two points in T . For each such \mathbf{l} we then determine, for example, the Euclidean distance from \mathbf{l} to all points in the dataset. If this distance is smaller than some threshold t for a specific point, we assume that \mathbf{l} has a good fit to the point, and the point then belongs to the consensus set C for this T . If this C is larger than any other C produced in previous iterations, we assume that \mathbf{l} and C produced for this particular T represent the “best” estimate of the line and of the consensus set made so far, and keep both. If not, we simply discard \mathbf{l} and C . In both cases we continue with the trials until the probability of having chosen a T consisting of only inliers, is sufficiently large.

During the trials, when we pick two random points from the dataset, there are several possible outcomes, illustrated in Figure 17.4:

1. Both points are inliers and the intersecting line has a good fit to the remaining inliers: C is relatively large.
2. Both points are inliers but the intersecting line does not have a good fit to the remaining inliers: C is small.
3. At least one point is an outlier but the intersecting line has a good fit to at least some of the points, inliers or outliers. C may become relatively large.
4. At least one point is an outlier and the intersecting line does not have a good fit to more than a few other points in the dataset. C is small.

What we have to assure, by making sufficiently many trials, is that case 1) occurs with sufficiently high certainty. Cases 2) or 3) are not problematic as long as *also* case 1) occurs *and* the corresponding consensus set is larger than for the other cases. In general, we should expect to see many trials of type 4).

An important observation from this discussion is that the points in the dataset cannot be labeled as “inliers” and “outliers” in an objective way. Instead, the outcome of a robust model estimation is dependent on the threshold t and, consequently, which points becomes labeled as outliers depends on t , too. Furthermore, RANSAC is an indeterministic method, so even for a fixed t two different RANSAC runs on the same dataset may produce two different labelings of the dataset, although they are most likely very similar. Referring again to Figure 17.4, we may implicitly assume that the topmost point is an outlier. This may or may not be true depending on t and, in particular, outer points may also become outliers for specific choices of the threshold.

The threshold t

The basic RANSAC algorithm presented here uses a threshold, t , to determine whether or not a data point belongs to the consensus set C of the model M . This is done, for each data point, by computing an error ε and only if $\varepsilon \leq t$ does the data point belong to C . How do we determine t ?

Depending on how ε is defined, this may or may not be a straightforward issue. In the case that ε refers to a geometric error, however, it is natural to relate t to the standard deviation σ of the noise that perturbs the image coordinates: t should be of the same magnitude as σ . Setting $t = \sigma$ may be a reasonable approach, but then we do not take into account that M itself has been estimated from noisy data, and has, too, an inaccuracy in its parameters. Furthermore, σ is often a parameter that has to be estimated as well, with some uncertainty. In fact, the uncertainty in the slope of the line increases when the points from which it is estimated decreases. From this point of view, setting t in the range 2σ to 5σ , where σ is an *estimate* of the coordinate noise is often a more realistic choice, even though the choice of t for a specific application may depend also on additional considerations that are specific to that application.

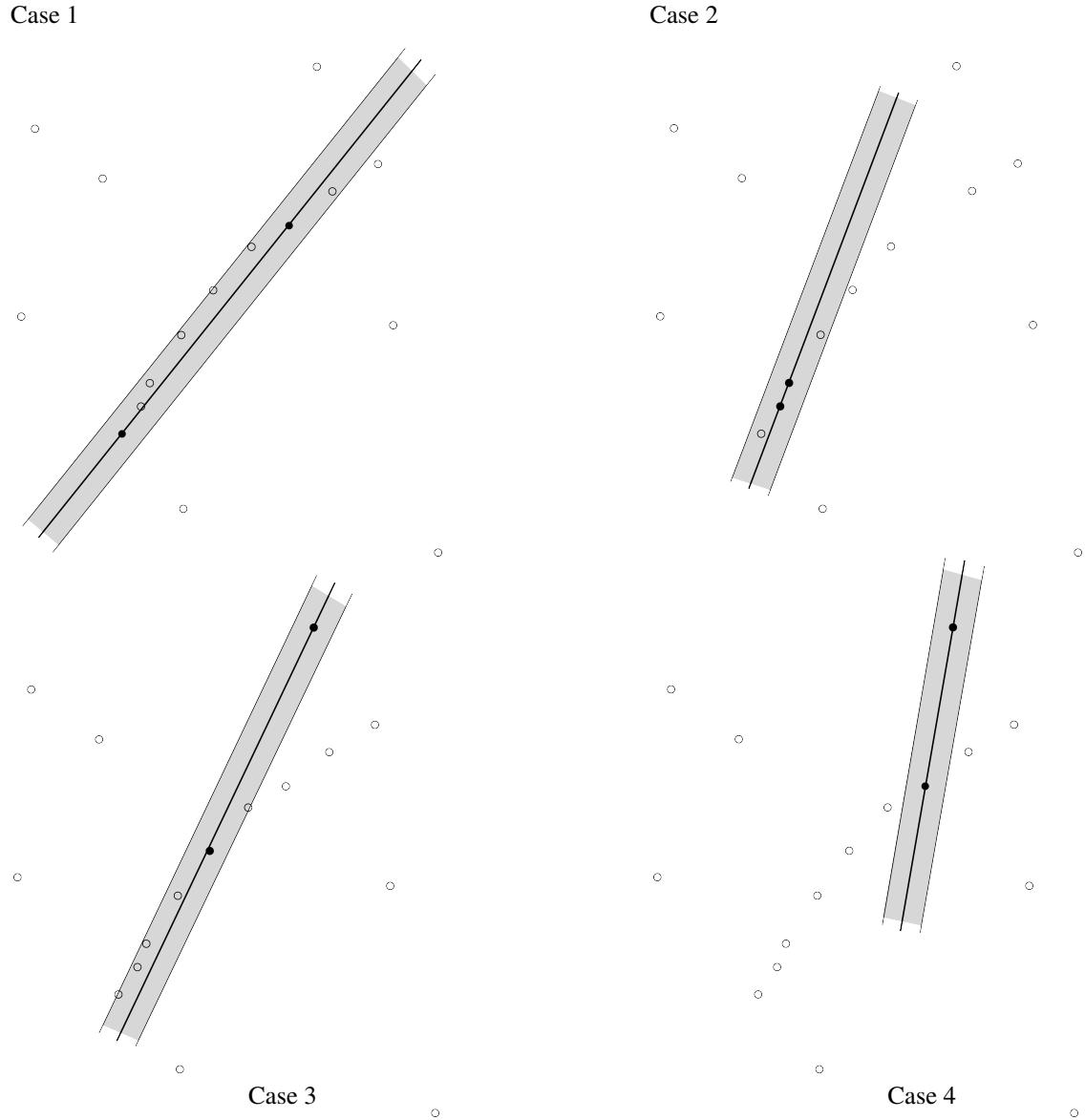


Figure 17.4: Four possible outcomes of using RANSAC for robust estimation of a line from a set of points that includes an outlier. The shaded stripe represents the threshold t which determines if a point belongs to the consensus set C or not. Case 1: the line is estimated from two inliers and generates a relatively large C . Case 2: the line is estimated from two inliers but generates a relatively small C . Case 3: the line is estimated from at least one outlier and generates a relatively large C . Case 4: the line is estimated from at least one outlier and generates a small C .

17.3.3 Probabilities and number of trials

The probabilistic framework that embeds RANSAC has so far been described in terms of some underlying probability of successfully estimating a model. It appears that this probability is related to the number of trials that are necessary to produce a useful result. In this section, we derive an explicit expression for this probability that connects it to the number of trials.

First, we assume that the dataset has n points and that a fraction w of these are inliers, i.e., the number of inliers is wn . Second, we assume that n is relatively large and, in particular, it is very large compared to m , the number of distinct data points in T that are used to determine a model. The trial set T is picked at random from the dataset. If $n \ll m$, we can assume that each point in T has a probability w of being an inlier. Based on this approximation, the probability of having only inliers in T is w^n , and the probability of T containing at least one outlier is $1 - w^n$. We perform r trials, and the probability of only choosing a T with at least one outlier in every trial is $(1 - w^n)^r$. We want the opposite to happen: at least once among the r trials we pick a T consisting of only inliers. The probability for that is

$$p = 1 - (1 - w^n)^r. \quad (17.3)$$

We want the probability p to be high, and we can determine how high it becomes by choosing a sufficiently large number of trials, r :

$$r = \frac{\log(1 - p)}{\log(1 - w^n)}. \quad (17.4)$$

Clearly, we cannot achieve $p = 1$ with a finite number of trials, but we can come as close to 1 as we want by choosing a sufficiently large r . For example, in the case of estimating a line ($n = 2$) from a dataset where $w = 0.5$, i.e., half of the points are inliers and the rest are outliers, we obtain $p \approx 0.94$ already for $r = 10$ and $p \approx 0.99$ for $r = 20$.

The derivation of Equation (17.4) assumes that only cases 1) and 4) occur in the RANSAC trials or, at least, cases 2) or 3) only happens with insignificant probability. This may or may not be the case for any particular dataset and it is also often the case that w only can be approximated. In practice, this means that Equation (17.4) can only be seen as a guideline for how many trials are needed *at minimum* to achieve a certain probability p .

17.3.4 Variants of the basic RANSAC algorithm

Numerous variants and extensions of the basic method have been presented since the original RANSAC algorithm was published in 1981. A main motivation for these approaches is to reduce the number trials, r , expressed in Equation (17.4) for fixed p and w , but also to improve the accuracy of the resulting model. To make a full overview of all variants and extensions of RANSAC is outside the scope of this presentation and, instead, we only discuss a few simple approaches that commonly appear in RANSAC based methods.

Break if C is sufficiently good

The basic RANSAC method uses a fixed number of trials, r , to produce a result that is useful with probability of p . This means that with a probability p , among all r trials there is at least one trial that provides a useful result. If we known w , we can estimate the size of the corresponding consensus set C as: $\text{size}(C) = w \text{size}(D)$. Consequently, if a trial produces a consensus set of approximately this size, there is little gain in performing the remaining trials. In this case, we instead break the RANSAC loop and use the current consensus set and corresponding model as the resulting estimate. In the ideal case, this approach can save 50% of the average number of trials, although the exact figure depends on how well we can estimate the different parameters of the problem.

Selection of points

In Section 17.3.2, four possible results of the RANSAC trial were presented. Case 2 is interesting since there are ways to reduce it. When a line is estimated from a set of points, which are perturbed by noise, it is clear that this case is more likely to occur if the two points are close, than if they are far apart. There are several studies which conclude that this is a general observation also when other geometric objects are estimated. For example, a study by Hedborg *et al.* concludes that when the points selected inside a RANSAC loop, used for estimating the essential

matrix, are subject to a distance constraint, this reduces the number of iterations required to solve their estimation problem by half [38].

17.3 The points selected inside a RANSAC loop for the trial set should be randomly selected, but should also not be too close. If they are, and even if they are all inliers, the estimated model is likely to be of poor accuracy and does not generate a large consensus set.

The specific implementation of such a distance constraint can vary between different applications.

Re-estimate model from C

In the end, RANSAC produces a model M that generates a consensus set, C , which is the largest one ever detected over all the r trials. In each iteration, the model M has been estimated from a minimal set of only n points. This means that, even if M fits all points in C , its accuracy may not be very high. This model can be improved by re-estimating it from *all* points in C . Also, to reduce computational complexity the initial model is typically estimated by minimizing an algebraic error. If we re-estimate the resulting model based on a robustly estimated consensus set, we may as well minimize a more expensive geometric error since this is not iterated over r trials.

An obvious question arises: does the re-estimated model generate the same consensus set as M does? The answer is: in general, it does not. This implies that we may re-generate the consensus set from the re-estimated model. These two steps: re-estimate the model from the consensus set, and re-generate the consensus set from the new model, can then be iterated. Either we iterate a fixed number of times, or until the consensus set converges. As a result, we obtain a robustly estimated model of higher accuracy than the original model M produced by RANSAC.

A generic version of the RANSAC algorithm for estimation of some model from a dataset is presented in Algorithm 17.1. The two issues discussed above are indicated as optional steps. Optional means: you may choose to implement either of these steps or not, but the choice may have a significant impact on the computation time and accuracy of the result.

In addition to these issues, we may also worry about how to determine w , in particular if we do not have an estimate of this parameter. This issue can be dealt with using an adaptive approach.

Adaptive number of trials

If w is known we can use Equation (17.4) to compute the number of trials, r , required for reaching a certain probability p of producing a useful model estimate. For some datasets, w is unknown and then there is no way of determining a reasonable r , other than making a worst-case guess. On the other hand, each RANSAC trial produces a consensus set C , assumed to contain mostly inliers, which means that each trial produces an estimate of w in terms of the number of points in C relative to D . In particular, each trial when we obtain a larger C than ever before in a RANSAC loop, we can update our estimate of w to be $\text{size}(C)/\text{size}(D)$, and then recalculate r from Equation (17.4). Initially, we can set $r = \infty$, alternatively to some very large number, and each time that r is updated in this way, it will be reduced.

The loop implemented in Algorithm 17.1 needs to be modified if we use this adaptive approach. The probability w should be updated when necessary, and the for-loop, which has a fixed number of iterations, r , should be a while-loop where r is a variable.

17.4 Revisiting the correspondence problem

A practical example of robust estimation is when we want to determine correspondence between two sets of points from which some model is to be estimated. For example, this applies to the case when we estimate a homography, Section 13.1, a camera matrix, Section 13.3, or a fundamental matrix, Section 16.2. In these cases, how can we establish the necessary correspondences? Without additional information, we can in principle allow any point in one set to correspond to any other point in the other set, and each such hypothetical correspondence may be correct or not. Those which are correct are the inliers and those which are incorrect are the outliers.

Algorithm 17.1: The generic RANSAC algorithm

```

Input: A dataset  $D$ , which may contain outliers, from which a model is to be estimated
Input:  $n$  = the number of points in each trial set  $T$ 
Input:  $r$  = the number of trials, see Equation (17.4)
Input:  $t$  = a threshold that determines membership of a data point to a consensus set
Output: A model  $M_{\text{est}}$  that has been estimated from only inliers with probability  $p$ 
Output: The corresponding consensus set  $C_{\text{est}}$ , containing only inliers with probability  $p$ 

1 Initialize:  $M_{\text{est}} = \emptyset, C_{\text{est}} = \emptyset$ 
2 while iteration := 1 ...  $r$  do
3   Pick a random subset  $T$  from  $D$ , with  $n$  data points. Optional: use a distance constraint
4   Determine a model  $M$  from  $T$ . Typically, use minimal case estimation
5   Initialize:  $C = \emptyset$ 
6   foreach point  $q$  in  $D$  do
7     Compute a measure  $\varepsilon$  of how well  $q$  fits  $M$ , typically some type of error
8     if  $\varepsilon < t$  then
9       Add  $q$  to  $C$ 
10    end
11  end
12  if size of  $C$  is larger than size of  $C_{\text{est}}$  then
13    Set  $C_{\text{est}} = C$ 
14    set  $M_{\text{est}} = M$ 
15  end
16  Optional:
17  if size( $C$ )  $\approx w$  size( $D$ ) then
18    break the RANSAC loop
19  end
20 end
21 Optional:
22 Repeat a fixed number of times, or until  $C_{\text{est}}$  converges
23 Re-estimate  $M_{\text{est}}$  from  $C_{\text{est}}$ . If possible, use a geometric error
24 Re-generate  $C_{\text{est}}$  from the new  $M_{\text{est}}$ 
```

By further analyzing the set of hypothetical correspondences it should be possible to somehow determine which are the correct correspondences, the inliers. Even for moderate number of points in the two sets, however, there is a combinatorial explosion that produces a very large set of hypothetical correspondences and most of them are incorrect. If we can solve the correspondences problem without an exhaustive search over the entire set of these hypothetical correspondences, the situation is improved. There exist several strategies for dealing with this type of robust estimation, and here we present one of the most common methods, based on RANSAC. In this section we present a general strategy for RANSAC based solutions of the correspondence problem, and in Section 17.5 we apply it to specific estimation problems in geometry.

17.4.1 RANSAC and the correspondence problem

Let D_1 and D_2 be two sets of data points. The Cartesian product of the two sets, denoted $D_1 \times D_2$, is the set of all ordered pairs where the first element in the pair comes from D_1 and the second element from D_2 . This means that $D = D_1 \times D_2$ includes all possible correspondences that can be formed between points in the first set with points in the other set. Only a few of all the hypothetical correspondences in D are correct correspondences, representing the inliers, and we can regard the remaining ones as outliers.

Using RANSAC, it should then be possible to determine both the correct correspondences in D and the model that fits these corresponding points. Compared to the line estimation case, however, we are now in a much worse situation. For example, consider estimation of a homography ($n = 4$) from two sets, each of 10 image points, where we do not know the correspondences. This means that D contains 100 hypothetical correspondences, but

only 10 of these are correct, i.e., $w = 0.1$. In the ideal case, when the number of trials can be computed as in Equation (17.4), we need approximately $r = 30,000$ iterations to achieve $p = 0.95$. Unless additional techniques are used, this means that the correspondence problem can require a very large number of trials in order to produce useful results. One such technique is preprocessing of the data.

17.4.2 Preprocessing of data

One technique that can be used to reduce the number of RANSAC trials is to preprocess the dataset D . More specifically, in the case that D contains hypothetical corresponding image points, it is often possible to make a comparison of their *visual appearance* in the images, and from that make a heuristic judgment whether or not they are in correspondence, i.e., they refer to the same 3D point. For example, if the images are taken with the same camera, it is very likely that the image intensities in a small neighborhood around each of the two points are similar if they are corresponding points. Even better: if we are dealing with color images, the neighboring regions of the two points should have a similar distribution of the colors. It is possible to extract also other types of local image features from the neighboring regions of two points that are in hypothetical correspondence. By comparing them we can assess the validity of the correspondence.

Furthermore, if we know that the camera poses of the two images only differ by a relatively small rigid transformation, we can safely assume that corresponding points do not have a large difference in image positions. Alternatively, if we know something about the rigid transformation, for example: it is mainly a horizontal translation, we can infer that large vertical translations in the images is not likely to occur for corresponding points.

In summary: for each pair of putative correspondence in D , we can classify it either as “unlikely” or “still hypothetical”. The first category of correspondences is discarded and the second one we keep and apply RANSAC to find the correct correspondences. A slightly more sophisticated approach is to partition D into the three disjoint subsets:

- D_{low} = the subset of D with very low probability of being correct correspondences.
- D_{medium} = the subset of D with medium probability of being correct correspondences.
- D_{high} = the subset of D with very high probability of being correct correspondences.

where $D = D_{\text{low}} \cup D_{\text{medium}} \cup D_{\text{high}}$. As before, we discard D_{low} since it contains only unlikely correspondences, i.e., mainly outliers. The remaining two subsets are used in RANSAC such that in each trial, T is selected only from D_{high} . In this way we boost w from a relatively low fraction in $D_{\text{medium}} \cup D_{\text{high}}$ to a relatively high fraction in only D_{high} . By choosing T only from D_{high} , we now need fewer trials r to achieve the same probability of success, p . In the formation of the consensus set C , however, we use also D_{medium} since we expect to find additional correct correspondences (inliers) here, not present in D_{high} . A modified version of RANSAC adapted for solving the correspondence problem is presented in Algorithm 17.2. It is based on an initial preprocessing of $D = D_1 \times D_2$ into the three categories presented above. Exactly what type of processing is applied is application dependent, and is therefore not described here in detail.

By suitably adapting Algorithm 17.2 it can be made useful for estimation of a homography, a camera matrix, or the epipolar geometry from two sets of points, for which we do not have initial correspondences. In the following section, the particular case of robust estimation of epipolar geometry is discussed more in detail.

17.4.3 Closing remarks

A practical issue when we use RANSAC, for example for the case of solving the correspondence problem, is to apply it on a dataset D that contains sufficiently many inliers, i.e., a sufficiently large w . From a theoretical point of view, if w is small this simply means that the number of trials, r , becomes large for a fixed probability p . Assuming that we have a sufficiently fast hardware that executes the computations, a large r may not have to be a problem.

Let us consider a simple example where we want to solve the correspondence problem using the 8-point algorithm ($n = 8$) on a dataset where $w = 0.1$ with $p = 0.99$. In this case, Equation (17.4) gives $r \approx 460,000,000$ iterations. A reasonable implementation of the 8-point algorithm takes in the order of $1\mu\text{s}$ per trial, and in total we get an execution time in the order of 7-8 min. The same figure for $w = 0.5$ is an execution time of 1 ms, which even may be suitable for real-time applications on video data. Clearly, there is a huge gain in execution time just by going from $w = 0.5$ to $w = 0.1$. This is the motivation for the pre-processing step described in Section 17.4.2.

Algorithm 17.2: The RANSAC algorithm adapted to the correspondence problem

```

Input: A dataset  $D_{\text{medium}}$ , classified as having a medium level of correct correspondences (inliers)
Input: A dataset  $D_{\text{high}}$ , classified as having a high level of correct correspondences
Input:  $n$  = the number of points in each trial set  $T$ 
Input:  $r$  = the number of trials, see Equation (17.4)
Input:  $t$  = a threshold that determines membership of a data point to a consensus set
Output: A model  $M_{\text{est}}$  that has been estimated from only inliers with probability  $p$ .
Output: The corresponding consensus set  $C_{\text{est}}$ , containing only inliers with probability  $p$ 

1 Initialize:  $M_{\text{est}} = \emptyset, C_{\text{est}} = \emptyset$ 
2 while iteration := 1 ...  $r$  do
3   Pick a random subset  $T$  from  $D_{\text{high}}$ , with  $n$  data points
4   Determine a model  $M$  from  $T$ . Typically, use minimal case estimation
5   Initialize:  $C = \emptyset$ 
6   foreach point  $q$  in  $D_{\text{medium}} \cup D_{\text{high}}$  do
7     Compute a measure  $\varepsilon$  of how well  $q$  fits  $M$ , typically some type of error
8     if  $\varepsilon < t$  then
9       Add  $q$  to  $C$ 
10    end
11  end
12  if size of  $C$  is larger than size of  $C_{\text{est}}$  then
13    Set  $C_{\text{est}} = C$ 
14    set  $M_{\text{est}} = M$ 
15  end
16 end
17 Optionally: use the final re-estimation step in Algorithm 17.1 on page 304

```

The gain depends to a large extent on n , the number of points that are selected at each trial. The larger n is, the larger is the gain when w increases. Another way of thinking about this observation is that going from $w = 0.1$ to $w = 0.5$, for a fixed number of trials, r , increases the probability p in Equation (17.3). For example, the same r that gives $p = 0.99$ for $w = 0.1$, gives a p that in practice is equal to 1 for $w = 0.5$.

In addition to the effect on the execution time, a small w also implies that there is a large number of outliers in relation to inliers. This, in turn, increases the probability of finding outliers that, by chance, happen to produce a model that generates a large consensus set, perhaps even larger than the largest consensus set produced by only inliers. As a consequence, a small w implies that RANSAC can result in a model that is estimated from (mainly) outliers.

In summary, we should spend any reasonable effort to increase w as much as possible, both to reduce the executing time and to increase the expected quality of the resulting model.

17.5 Robust estimation in practice

This section presents a number of practical estimation problems that rely on a dataset consisting of two sets, D_1 and D_2 , where we want to establish correspondences between points in the two sets. To do this automatically, robust estimation based on Algorithm 17.2 on page 306 can be used. Hence, the following algorithms are variants of Algorithm 17.2 on page 306 that have been adapted to estimate a particular model. All of them have in common that the input is in the form of D_{medium} and D_{high} , both subsets of $D = D_1 \times D_2$. They are formed in a preprocessing step, described in the previous section, which removes as many outliers as possible already from differences in visual appearance, and divides the remaining dataset into D_{medium} and D_{high} , containing correspondences of medium and high level of confidence, respectively. The algorithms also need r , the number of trials that are necessary to produce a useful estimate with probability p . This number can be determined in accordance with Section 17.3.3, for example using Equation (17.4). Finally, they need a threshold t that should be of the same order as the measurement noise of the point coordinates.

17.5.1 Robust estimation of \mathbf{H}

For the case of robust estimation of homographies, we can apply RANSAC to two sets of interest points, D_1 and D_2 , typically from two images that are related by a homography. This allows us to determine *both* correspondences between the point-sets *and* the homography transformation in terms of the 3×3 matrix \mathbf{H} . In this algorithm, the model M that is estimated in each trial is the matrix \mathbf{H} . This can be done by means of Algorithm 13.3 on page 238, which requires a minimum of $n = 4$ correspondences. We also need a way to quantify the fit between the data and the model, preferably using a geometric error. More specifically, we need to be able to determine for a specific hypothetical correspondence $\{\mathbf{y}_1 \in D_1, \mathbf{y}_2 \in D_2\}$ if it is consistent with a trial homography \mathbf{H} , and therefore should be put into the consensus set. A simple and useful approach is to consider the point-to-point distance

$$\varepsilon = d_{PP}(\mathbf{y}_2, \mathbf{H}\mathbf{y}_1), \quad (17.5)$$

and determine if it is above or below some threshold t . Other geometric errors that are more symmetric relative to the two points can be used instead, but will seldom produce better results from the overall robust estimation. A summary of this RANSAC based approach to robust homography estimation is presented in Algorithm 17.3.

17.5.2 Robust estimation of \mathbf{F}

We can also apply the same approach to robust estimation of epipolar geometry, where D_1 and D_2 are interest points from a pair of stereo images. This produces *both* correspondences between the points *and* the epipolar geometry in terms of the fundamental matrix. In this algorithm, the model M that is estimated in each trial is the fundamental matrix \mathbf{F} . This can be done by means of the 8-point algorithm, Algorithm 16.4 on page 285, which requires a minimum of $n = 8$ points. Alternatively, we can use the 7-point algorithm, Algorithm 16.5 on page 286, that determines \mathbf{F} from only $n = 7$ points. The 7-point algorithm produces up to 3 possible \mathbf{F} that exactly fit a set of 7 corresponding points. As a consequence, we need to form up to 3 consensus sets C inside each RANSAC loop, and evaluate each of them relative to the best consensus set found so far. This implies that the average computational cost for a loop based on the 7-point algorithm is larger than it is for the 8-point algorithm.

Algorithm 17.3: RANSAC algorithm for estimation of a homography

Input: A dataset D_{medium} , of medium level of correct correspondences between image points.
Input: A dataset D_{high} , of high level of correct correspondences between image points.
Input: r = the number of trials, see Section 17.3.3.
Input: t = a threshold that determines membership of a pair of points in the consensus set.
Output: A homography \mathbf{H}_{est} , estimated from only correct correspondences with probability p .
Output: The corresponding consensus set C_{est} , containing only correct correspondences with probability p

```

1 Initialize:  $\mathbf{H}_{\text{est}} = \emptyset, C_{\text{est}} = \emptyset$ 
2 while iteration := 1 ...  $r$  do
3   Pick a random subset  $T$  from  $D_{\text{high}}$ , with 4 pairs of corresponding points
4   Determine  $\mathbf{H}$  from  $T$  using Algorithm 13.3 on page 238. It produces 1 solutions for  $\mathbf{H}$ 
5   Initialize:  $C = \emptyset$ 
6   foreach point pair  $\{\mathbf{y}_1, \mathbf{y}_2\}$  in  $D_{\text{medium}} \cup D_{\text{high}}$  do
7     Compute  $\varepsilon$  in Equation (17.5), measuring how well the pair  $\{\mathbf{y}_1, \mathbf{y}_2\}$  fits  $\mathbf{H}$ 
8     if  $\varepsilon < t$  then
9       Add the pair  $\{\mathbf{y}_1, \mathbf{y}_2\}$  to  $C$ 
10    end
11  end
12  if size of  $C$  is larger than size of  $C_{\text{est}}$  then
13    Set  $C_{\text{est}} = C, \mathbf{H}_{\text{est}} = \mathbf{H}$ 
14  end
15 end
16 Optionally: use the final re-estimation step in Algorithm 17.1 on page 304

```

Algorithm 17.4: RANSAC algorithm for estimation of epipolar geometry

Input: A dataset D_{medium} , of medium level of correct correspondences between image points.
Input: A dataset D_{high} , of high level of correct correspondences between image points.
Input: r = the number of trials, see Section 17.3.3.
Input: t = a threshold that determines membership of a pair of points in the consensus set.
Output: A fundamental matrix \mathbf{F}_{est} , estimated from only correct correspondences with probability p .
Output: The corresponding consensus set C_{est} , containing only correct correspondences with probability p

```

1 Initialize:  $\mathbf{F}_{\text{est}} = \emptyset, C_{\text{est}} = \emptyset,$ 
2 while iteration := 1 ...  $r$  do
3   Pick a random subset  $T$  from  $D_{\text{high}}$ , with 7 pairs of corresponding points
4   Determine  $\mathbf{F}$  from  $T$  using Algorithm 16.5 on page 286. Produces between 1 and 3 real solutions for  $\mathbf{F}$ 
5   foreach estimated  $\mathbf{F}$  do
6     Initialize:  $C = \emptyset$ 
7     foreach point pair  $\{\mathbf{y}_1, \mathbf{y}_2\}$  in  $D_{\text{medium}} \cup D_{\text{high}}$  do
8       Compute  $\varepsilon$  in Equation (10.15), measuring how well the pair  $\{\mathbf{y}_1, \mathbf{y}_2\}$  fits  $\mathbf{F}$ 
9       if  $\varepsilon < t$  then
10         Add the pair  $\{\mathbf{y}_1, \mathbf{y}_2\}$  to  $C$ 
11       end
12     end
13     if size of  $C$  is larger than size of  $C_{\text{est}}$  then
14       Set  $C_{\text{est}} = C, \mathbf{F}_{\text{est}} = \mathbf{F}$ 
15     end
16   end
17 end
18 Optionally: use the final re-estimation step in Algorithm 17.1 on page 304

```

The additional computational complexity of using the 7-point algorithm can be balanced by the fact that for fixed probabilities w and p , the number of trials r in Equation (17.4) becomes smaller for $n = 7$ compared to $n = 8$. How much smaller r becomes depends on w : if w is small the reduction in r is larger, if w is larger the reduction is smaller. For example, if $w = 0.1$ the gain is a factor 10, but for $w = 0.5$ the gain is only a factor 2. This implies that in order to choose a suitable algorithm for the estimation of \mathbf{F} in each RANSAC loop, we need to have a good estimate of w , and also of the average number of (real) solutions that are provided by the 7-point algorithm for the dataset that is to be processed.

Another issue when RANSAC is applied to robust estimation of epipolar geometry is how to measure ε . A useful approach must be based on a geometric error, which allows us to determine a threshold t in a reasonable way, as described in Section 17.3.2. The simplest type of geometric error for the case of epipolar geometry, between a fundamental matrix \mathbf{F} and a pair of putative correspondences, $\mathbf{y}_1, \mathbf{y}_2$, is the one described in Equation (10.15). It represents the sum of the squared distances between each of the two image points and the epipolar line generated by the other point.

A RANSAC based approach to robust estimation of the epipolar geometry is summarized in Algorithm 17.4, where the 7-point algorithm is used to determine \mathbf{F} in each trial. This can be replaced by the 8-point algorithm, in which case the loop over the estimated fundamental matrices can be reduced to a single pass.

17.5.3 Robust estimation of \mathbf{E}

The estimation problem discussed in the previous section, for robust estimation of epipolar geometry, can be adapted also to the calibrated case. In this case, we use Algorithm 17.5 to estimate the essential matrix from two sets of C-normalized image points, for which the correspondences are unknown. In comparison to Algorithm 17.4 on page 308, a major difference is that for the calibrated case, it is possible to determine the epipolar geometry from a minimum of 5 pairs of corresponding points, instead of 7 for the uncalibrated case. This implies a significant reduction in the number of trials, r , for the same probabilities w and p , as described in Equation (17.4). This

Algorithm 17.5: RANSAC algorithm for estimation of essential matrix based on the 5-point algorithm.

Input: A dataset D_{medium} , with medium level correct correspondences of C-normalized image points.
Input: A dataset D_{high} , with high level correct correspondences of C-normalized image points.
Input: r = the number of trials, see Section 17.3.3.
Input: t = a threshold that determines membership of a pair of points in the consensus set.
Output: An estimated essential matrix \mathbf{E}_{est} .
Output: A consensus set C_{est} , containing only correct correspondences with probability p

- 1 Initialize: $\mathbf{E}_{\text{est}} = \emptyset, C_{\text{est}} = \emptyset$
- 2 **while** iteration := 1 ... r **do**
- 3 Pick a random subset T from D_{high} , with 5 pairs of corresponding points
- 4 Determine \mathbf{E} from T using the 5-point algorithm. Produces up to 10 solutions for \mathbf{E}
- 5 **foreach** estimated \mathbf{E} **do**
- 6 Initialize: $C = \emptyset$
- 7 **foreach** point pair $\{\mathbf{y}_1, \mathbf{y}_2\}$ in $D_{\text{medium}} \cup D_{\text{high}}$ **do**
- 8 Compute ε in Equation (10.15), measuring how well the pair $\{\mathbf{y}_1, \mathbf{y}_2\}$ fits \mathbf{F}
- 9 **if** $\varepsilon < t$ **then**
- 10 Add the pair $\{\mathbf{y}_1, \mathbf{y}_2\}$ to C
- 11 **end**
- 12 **end**
- 13 **if** size of C is larger than size of C_{est} **then**
- 14 Set $C_{\text{est}} = C, \mathbf{E}_{\text{est}} = \mathbf{E}$
- 15 **end**
- 16 **end**
- 17 **end**
- 18 Optionally: use the final re-estimation step in Algorithm 17.1 on page 304

Algorithm 17.6: RANSAC algorithm for estimation of essential matrix based on the 5+1-point algorithm.

Input: A dataset D_{medium} , with medium level correct correspondences of C-normalized image points.
Input: A dataset D_{high} , with high level correct correspondences of C-normalized image points.
Input: r = the number of trials, see Section 17.3.3.
Input: t = a threshold that determines membership of a pair of points in the consensus set.
Output: An estimated essential matrix \mathbf{E}_{est} .
Output: The corresponding consensus set C_{est} , containing only correct correspondences with probability p

- 1 Initialize: $\mathbf{E}_{\text{est}} = \emptyset, C_{\text{est}} = \emptyset$
- 2 **while** iteration := 1 ... r **do**
- 3 Pick a random subset T from D_{high} , with 6 pairs of corresponding points
- 4 Determine \mathbf{E} from T using the 5-point algorithm from 5 pairs in T . Produces up to 10 solutions for \mathbf{E}
- 5 Determine the \mathbf{E} that minimizes the geometric error in Equation (10.15) for the 6-th pair in T
- 6 Initialize: $C = \emptyset$
- 7 **foreach** point pair $\{\mathbf{y}_1, \mathbf{y}_2\}$ in $D_{\text{medium}} \cup D_{\text{high}}$ **do**
- 8 Compute ε in Equation (10.15), measuring how well the pair $\{\mathbf{y}_1, \mathbf{y}_2\}$ fits \mathbf{E}
- 9 **if** $\varepsilon < t$ **then**
- 10 Add the pair $\{\mathbf{y}_1, \mathbf{y}_2\}$ to C
- 11 **end**
- 12 **end**
- 13 **if** size of C is larger than size of C_{est} **then**
- 14 Set $C_{\text{est}} = C, \mathbf{E}_{\text{est}} = \mathbf{E}$
- 15 **end**
- 16 **end**
- 17 Optionally: use the final re-estimation step in Algorithm 17.1 on page 304

reduction comes at a cost: we get up to 10 possible solutions for \mathbf{E} , and each of them needs to be evaluated relative to a consensus set. This may be a reasonable approach, but an alternative is offered by extending the original 5-point algorithm to the 5+1-point algorithm.

The 5+1-point algorithm

Since the 5-point algorithm produces up to 10 valid solutions, it may be relevant to reduce the number of solutions before they are further processed. One simple approach is offered by the *5+1-point algorithm* where, from an initial set of 6 pairs of corresponding points, we use 5 pairs to determine a set of solutions for \mathbf{E} by means of the 5-point algorithm. All solutions that are produced in this way satisfy the epipolar constraint exactly for the first 5 correspondences, but not necessarily for the remaining 6-th pair. We can use this to remove solutions, for example, by determining the \mathbf{E} that minimizes the geometric error in Equation (10.15). This \mathbf{E} is the estimate of the 5 + 1-point algorithm. This algorithm is not symmetric in the set of correspondences: the estimated \mathbf{E} depends on in which order the correspondences come, i.e., which pair it denoted as the 6-th correspondence.

An algorithm for robust estimation of the essential matrix based on the 5+1-point algorithm is presented in Algorithm 17.6. Notice that since $n = 5$ in Algorithm 17.5 and $n = 6$ for in Algorithm 17.6, they differ in the number of trials that are required for fixed probabilities p and w . This means that the 5+1-point approach requires more trials, but on average each trial is less computationally demanding since only one consensus set is formed. Which approach is the faster, depends on p and w .

17.5.4 Robust PnP

Robust estimation can also be used for determining both correspondences and a camera pose, based on PnP. In this case, D_1 consists of 3D points rather than image points. If the 3D points have no a priori relation to some image points, the formation of D_{medium} and D_{high} may not be done and instead we must use the entire D as input. In certain applications, however, the 3D points have been triangulated from image points, and then ranking of correspondence confidence can be made, based on visual appearance by considering the image points from which a 3D point has been triangulated. An algorithm for robust PnP is presented in Algorithm 17.7. In each trial, $n = 3$ correspondences are needed to determine a trial pose $(\mathbf{R}, \bar{\mathbf{t}})$ using P3P which, in fact, produces up to 4 possible poses. The fit between the pose $(\mathbf{R}, \bar{\mathbf{t}})$ and correspondence (\mathbf{x}, \mathbf{y}) , is quantified by the geometric error $\varepsilon = d_{\text{PP}}(\mathbf{y}, (\mathbf{R} | \bar{\mathbf{t}})\mathbf{x})$. This was the original application of RANSAC presented in the article by Fischler and Bolles [20].

Algorithm 17.7: RANSAC algorithm for robust estimation of camera pose.

Input: D_{medium} : medium level correct correspondences between C-normalized image points and 3D points.
Input: D_{high} : high level correct correspondences between C-normalized image points and 3D points.
Input: r = the number of trials, see Section 17.3.3.
Input: t = a threshold that determines membership of a pair of points in the consensus set.
Output: An estimated camera pose $(\mathbf{R}_{\text{est}}, \bar{\mathbf{t}}_{\text{est}})$.
Output: The corresponding consensus set C_{est} , containing only correct correspondences with probability p

- 1 Initialize: $\mathbf{R}_{\text{est}} = \emptyset, \bar{\mathbf{t}}_{\text{est}} = \emptyset, C_{\text{est}} = \emptyset$
- 2 **while** iteration := 1 ... r **do**
- 3 Pick a random subset T from D_{high} , with 3 pairs of corresponding points
- 4 Determine (\mathbf{R}, \mathbf{t}) from T by means of P3P. Produces up to 4 poses
- 5 **foreach** estimated pose $(\mathbf{R}, \bar{\mathbf{t}})$ **do**
- 6 Initialize: $C = \emptyset$
- 7 **foreach** point pair $\{\mathbf{y}_k, \bar{\mathbf{x}}_k\}$ in $D_{\text{medium}} \cup D_{\text{high}}$ **do**
- 8 Compute ε in Equation (15.40), measuring how well the pair fits $(\mathbf{R}, \bar{\mathbf{t}})$
- 9 **if** $\varepsilon < t$ **then**
- 10 Add the pair $\{\mathbf{y}_k, \bar{\mathbf{x}}_k\}$ to C
- 11 **end**
- 12 **end**
- 13 **if** size of C is larger than size of C_{est} **then**
- 14 Set $C_{\text{est}} = C, \mathbf{R}_{\text{est}} = \mathbf{R}, \bar{\mathbf{t}}_{\text{est}} = \bar{\mathbf{t}}$
- 15 **end**
- 16 **end**
- 17 **end**
- 18 Optionally: use the final re-estimation step in Algorithm 17.1 on page 304

Part III

Applications

Chapter 18

Camera Calibration

Before you read this chapter, you should have thorough understanding of the pinhole camera, described in Chapter 8. You also need linear estimation methods, described in Chapters 12 and 13, as well as the non-linear methods described in Chapter 17.

Based the pinhole camera model, a 3D point \mathbf{x} is projected to an image point \mathbf{y} in accordance with

$$\mathbf{y} \sim \mathbf{C}\mathbf{x}, \quad (18.1)$$

where \mathbf{C} is the 3×4 camera matrix. For one and the same camera, the numerical values of the elements in \mathbf{C} depend on which world coordinate system is used to define the homogeneous coordinates of the 3D point, \mathbf{x} , and which coordinate system is used for to define the homogeneous coordinates of the image point, \mathbf{y} . Changing either of these coordinate systems changes the matrix \mathbf{C} even if the camera is completely fixed.

From Section 8.3.3 we also know that \mathbf{C} can be decomposed into internal and external parameters:

$$\mathbf{y} \sim \mathbf{K}(\mathbf{R}|\bar{\mathbf{t}}). \quad (18.2)$$

Here, \mathbf{K} is a 3×3 matrix that holds the internal camera parameters, transforming C-normalized image coordinates to pixel coordinates, and $\mathbf{R} \in SO(3)$ and $\bar{\mathbf{t}} \in \mathbb{R}^3$ represent a rigid transformation from the world coordinate system to the camera centered coordinate system. This transformation represents the external camera parameters, corresponding to the position and orientation of the camera centered coordinate system relative to the world coordinate system. The decomposition of \mathbf{C} into internal and external camera parameters is not unique, but if we assume \mathbf{K} to be upper triangular in accordance with Section 8.3.3:

$$\mathbf{K} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ 0 & k_{22} & k_{23} \\ 0 & 0 & 1 \end{pmatrix}, \quad (18.3)$$

then this decomposition is unique, as is described in Section 8.3.4.

In some practical applications, the camera is completely fixed in 3D space, i.e., the camera matrix \mathbf{C} , and therefore also \mathbf{K} , \mathbf{R} , and $\bar{\mathbf{t}}$, are fixed. In other applications, we allow the camera to move around in 3D space, i.e., \mathbf{R} and $\bar{\mathbf{t}}$ are functions of time. In principle, also \mathbf{K} may be a function of time, but in order to keep things simple we assume here \mathbf{K} to be fixed, e.g., by keeping the zoom of the camera fixed, or turning off auto-focus.

Camera calibration is the process of determining the camera parameters and it applies to a specific camera relative to a specific set of coordinates systems. In principle, it also refers to a specific point in time if the parameters are variable. A camera calibration may determine both the internal and external parameters, or only one of the two sets of parameters. It can refer to first finding \mathbf{C} , and then resectioning it into internal and external parameters, or to first finding \mathbf{K} and/or $\mathbf{R}, \bar{\mathbf{t}}$, from which \mathbf{C} can be inferred. Even if only the external parameters \mathbf{R} and $\bar{\mathbf{t}}$ are known, we can still determine the normalized camera matrix $(\mathbf{R}|\bar{\mathbf{t}})$. And if only the internal parameters in \mathbf{K} are known, we can still determine the camera matrix $\mathbf{K}(\mathbf{I}|\mathbf{0})$, defined relative to a camera centered coordinate system in 3D space.

In practice, camera calibration is an estimation problem: given measured data in the form of image points, possibly in combination with corresponding 3D points, we want to determine a model. This model is either the camera matrix \mathbf{C} or the internal and/or external camera parameters. This should be done such that Equation (18.1) is valid in the sense that some type of error that quantifies the match between data and model is minimized. Depending on which type of error is used, and depending on whether we want to determine the full matrix \mathbf{C} (as a projective element) or if it suffices only with the internal or external parameters, and depending on which type of data that is available, we arrive at different types of estimation problems that need different types of solutions. In this chapter, we present a brief overview of some topics that are relevant for camera calibration. We will also look more closely on one specific method for determining the internal parameters in \mathbf{K} .

18.1 Automatic camera calibration

The approach for camera calibration described in Section 13.3 allows us to estimate a full camera matrix \mathbf{C} . In many cases this method is not practical, or not even necessary. For example, image points and corresponding 3D points measured with high accuracy is not readily available in general situations. Also, in the case of a moving camera, where \mathbf{C} is time dependent, the full calibration of \mathbf{C} implies that we may need to perform the calibration at each time instance where we have observations of the image points. Although this is possible to do, limitation of the corresponding computational cost often makes such an approach impractical for real-time applications.

A simpler problem is to assume that the internal parameters in \mathbf{K} are constant. We can usually change the “zoom” of the camera and typically it also has auto-focus. Both these effects imply that the focal length of the camera changes, see Section 8.3.3. They also introduce minor changes in the optical pathway of the camera, leading to minor changes in the position of the principal point. By keeping the zoom locked, and turning off any auto-focus function, we can assume \mathbf{K} to be constant. The calibration problem can then be simplified to that of estimating a constant \mathbf{K} , as an upper triangular 3×3 matrix.

An interesting aspect of this estimation problem is that it can be solved (in principle) without any 3D points, it suffices to have the image coordinates of a set of 3D points in multiple images taken with different camera poses, and with known correspondences. It can be solved in a variety of ways, typically ending up in a non-linear minimization problem, and here we only give an outline of the basic principles behind these approaches. They are referred to as *automatic camera calibration*, since all of the required processing can be done by a computer, including finding appropriate image points and determine correspondence over multiple views. This is in contrast to the full calibration of \mathbf{C} , described in Section 13.3, which would depend on manually determined 3D coordinates and correspondences.

Practical methods for automatic camera calibration normally amount to relative complex optimization problems, which can be solved only by means of iterative methods. The mathematical foundation of such methods, however, can be illustrated by a relative simple method, described in the following section.

18.1.1 Mathematical foundation

As an example of how to do automatic camera calibration, consider what happens if you take two images with a rotating camera, as described in Section 9.3. Corresponding points in the two images are then related by a rotational homography $\mathbf{H} = \mathbf{K}^{-1} \mathbf{R} \mathbf{K}$, where \mathbf{R} is represents how much 3D rotation the camera makes about its center between the two images. Given that \mathbf{H} is estimated, e.g., using the techniques described in Section 14.5, we tried to solve \mathbf{K} and \mathbf{R} from \mathbf{H} in Section 9.3.3. In the process, the image of the absolute conic (IAC), was defined as $\boldsymbol{\omega} = \mathbf{K}^{-\top} \mathbf{K}^{-1}$, and we concluded that $\mathbf{K}^{-\top}$ is a Cholesky factor of $\boldsymbol{\omega}$. However, the result was that the matrices that we want cannot be uniquely determined from a single \mathbf{H} .

Instead, in Section 16.4 it was shown how to determine $\boldsymbol{\omega}$ from two or more instances of rotational homographies, assuming that \mathbf{K} is fixed. Once $\boldsymbol{\omega}$ is determined, we can then find $\mathbf{K}^{-\top}$ as the Cholesky factor of $\boldsymbol{\omega}$, and after inversion and translation, we get \mathbf{K} . In principle, this is a working approach to find the internal parameters \mathbf{K} of our camera, but it also has limitations.

As a start, the approach of using a rotating camera requires that the rotation is about the camera center. This can be done if the camera is place in a gimbal that can rotate about multiple axes, which all must intersect the camera center. This is feasible, but requires mechanical precision and calibration. An alternative is to assure that the points in the scene are at a large distance compared to any translation of the camera between the images. This

approach may work, but it may also give less control over the scene, therefore also over the images. Since we want them to contain many interest points, this may be a limiting factor.

Another issue relates to the fact that the method for estimating \mathbf{K} , that is outlined above, operates in two steps. It first estimate two or more homographies from sets of corresponding points, and from these homographies, ω is then estimated, and finally \mathbf{K} is determined from ω . Even if each of the homographies have been estimated by minimizing geometric errors, it is not sure they any of them are a proper rotational homography, as described in Section 9.3.2. An even they are, the ω that is estimated in Section 16.4 is not necessary optimal relative to all geometric errors in all images.

Geometric errors

What is needed to solve the issue mentioned in the last paragraph is a refinement of ω , which minimizes a geometric error that combines all measurements of point coordinates with a single hypothesis about the calibration. Section 14.5 describes a symmetric error function that is used for homography estimation, and it can be extended also to this case as

$$\epsilon_{\text{GEOM}} = \sum_{i=1}^P \sum_{j=1}^{Q_i} d_{\text{PP}}(\mathbf{y}'_{ij}, \underbrace{\mathbf{KR}_i \mathbf{K}^{-1}}_{\mathbf{H}_i} \mathbf{y}_{ij})^2 + d_{\text{PP}}(\underbrace{\mathbf{KR}_i^\top \mathbf{K}^{-1}}_{\mathbf{H}_i^{-1}} \mathbf{y}'_{ij}, \mathbf{y}_{ij})^2. \quad (18.4)$$

Here, P is the number of image pairs, and $\{\mathbf{y}'_{ij}, \mathbf{y}_{ij}, j = 1, \dots, Q_i\}$ are the corresponding points in pair i . ϵ_{GEOM} is a function of some parameters \mathbf{z} , consisting of \mathbf{K} in Equation (8.35) with 5 free parameters, and set of auxiliary parameters for the P rotation, for example using quaternions, as described in Section 15.3.1. In this case, ϵ_{GEOM} is a function of $\mathbf{z} \in \mathbb{R}^{5+4 \times P}$ that cannot be minimized by solving linear equations. Instead the non-linear estimation techniques described in Chapter 17 must be used.

Non-linear estimation methods need initial solutions for all unknown parameters in \mathbf{z} . The previous part of this section outlines a method that can provide an initial estimate of \mathbf{K} , given a set of initial estimates of the homographies \mathbf{H}_i . Given this \mathbf{K} , each \mathbf{H}_i corresponds¹ to a rotation $\mathbf{R}_i = \mathbf{K}^{-1} \mathbf{H}_i \mathbf{K}$. Together these \mathbf{R}_i form initial estimates of the camera rotations.

Notice that the free parameters that appear in the two terms of the symmetric error function in Equation (18.4) are almost identical, they differ only in that \mathbf{R} appears in one and \mathbf{R}^\top in the other. This means that the use of a symmetric error does not lead to significantly more complicated computations compared to a non-symmetric error. Furthermore, the rotations \mathbf{R}_i are in this case *auxiliary variables*, they are needed for the estimation, but are often not interesting as a result. We want the optimal \mathbf{K} , but not the rotations.

Before we continue

As we will see in the next section, the optical system of real cameras introduce so-called lens distortion. This can be seen as a deviation from the standard pinhole camera model, and it means that the mapping from 3D space to image coordinates is slightly more complicated than the usual expression $\mathbf{y} \sim \mathbf{Cx}$. As we will see, however, lens distortion can be included also in the estimation procedure described for \mathbf{K} . Once lens-distortion is included in the camera model, this method for estimation of \mathbf{K} , based on a rotating camera, can produce high-quality estimates. But we have also mentioned that the assumption of a rotating camera has practical limitations. Therefore, alternative methods, which instead are based on cameras that observe a plane, have become popular. We will return to such a method in Section 18.3 after looking into the details on lens distortion.

18.2 Lens distortion

In practice, a pinhole camera is implemented by a lens based camera. The main motivation is that the lens enables the camera to have a relatively large opening, the *aperture*, which allows a sufficient amount of light to enter the camera and illuminate the image plane. As a consequence, the pinhole camera model presented in Chapter 8 can only serve as a first order approximation of the mapping from 3D space to the image plane. One source of deviation from the pinhole camera model comes from the fact that a lens only can produce a sharp image of points in a certain region of the scene in front of the camera, the *depth-of-field*. This is a volume of a certain extension in

¹At least if \mathbf{H}_i satisfy the condition described for a rotational homography in Section 9.3.2.



Figure 18.1: Left: an image with no apparent lens distortion. Middle and right: the same image with two different distortion functions g .

3D space that is determined by the lens system and the spatial resolution of the sensor chip that measures the light energy at each pixel and converts it into an intensity value.

A second deviation refers to the *lens distortion*, *lens effect*, or *geometric distortion*. Even if the points in the depth-of-field are projected as sharp points in the image plane, they do not appear at the expected positions as predicted by the pinhole camera model. An apparent example is a so-called *fish-eye lens* that can project light within an extremely wide angle onto the image plane, which produces an image that appears deformed in particular near the borders. Lens distortion does not appear only for fish-eye lenses, any type of lens or lens system exhibits this effect to a larger or lesser degree. Very expensive and elaborate lens systems can reduce the lens distortion to a level below the resolution of the image sensor chip, but normally we have to expect that the camera lens contributes with a significant geometric distortion relative to the resolution of the sensor.

An apparent consequence of the lens effect is that the geometry in the image is deformed. An ideal pinhole camera projects 3D lines to 2D lines, see Section 8.5.2. One effect of the lens distortion is that a line instead appears more or less bent or curved, as is illustrated in Figure 18.1. In mathematical notation, an image point that should have Cartesian coordinates $\bar{\mathbf{y}}$ if the pinhole camera model was valid, instead gets a distorted coordinate $\bar{\mathbf{y}}'$ where

$$\bar{\mathbf{y}}' = g(\bar{\mathbf{y}}). \quad (18.5)$$

Here, $g : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is the *lens distortion function*, a function that depends on the lens system and also on which coordinate system is used in the image plane. It introduces smaller or larger displacements of the image, which ideally is projected in accordance with the pinhole camera model, to produce the distorted image that we actually observe. Given that the pinhole camera model serves as a first order approximation of the lens based camera, a first order approximation of g is $g(\bar{\mathbf{y}}) \approx \bar{\mathbf{y}}$. This implies that g is not an arbitrary function, but it can still be of a rather general nature. It reflects the geometrical imperfection of a lens system that in practice cannot be manufactured with arbitrary accuracy. Therefore, being able to determine g is as important as knowing the other parameters that determine the camera mapping of 3D space to the image plane, in particular when g deviates significantly from $g(\bar{\mathbf{y}}) = \bar{\mathbf{y}}$.

The optical path of the camera depends on the pose of the lens relative to the image plane. For a variable zoom camera, the lens pose depends on the zoom of the camera and, as a consequence, also g depends on the camera zoom in this case. To simplify our description of the lens distortion function g , we assume that the camera has a fixed zoom which means that g is a constant function that only depends on the image coordinate $\bar{\mathbf{y}}$. As mentioned above, a practical realization of g depends on the specific coordinate system used in the image plane. In this presentation we use the C-normalized image coordinate system as the coordinate system of g . A lens distortion function, however, may in general refer to any suitable coordinate system. Be sure that you know which coordinate system a specific g refers to.

18.1 A specific lens distortion function g refers to a specific coordinate system in the image plane. To understand how g distorts the image, the corresponding coordinate system must be known.

Geometric errors

The distortion function g is also important when we are dealing with estimation based on geometric errors. If the interest points are detected in the original and distorted image, and then mapped to undistorted coordinates by g^{-1} , we should take into account that the measurement error $\bar{\eta}$ appears in the distorted domain, not in the corrected image. This means that two image points, \bar{y}_1 and \bar{y}_2 , defined relative to the undistorted image, have a geometric error given by²

$$\bar{d}(g(\bar{y}_1), g(\bar{y}_2)), \quad (18.6)$$

rather than by $\bar{d}(\bar{y}_1, \bar{y}_2)$. Minimizing geometric errors that are derived from Equation (18.6) can only be done if g is well-defined.

Parameterization of g

The two above examples motivate why g is an important factor when dealing with cameras in practice, in particular if we want to understand how point are mapped from 3D space to the image. Each individual camera is characterized by a specific distortion function g , in addition to the camera matrix \mathbf{C} that represents the internal and external parameters of the camera. In addition to determining the internal and/or the external camera parameters in \mathbf{C} , camera calibration sometimes includes the estimation of g . A practical approach to the estimation of g is to assume that it belongs to a class of functions that can be parameterized by a set of parameters. The calibration of g then implies to determine the parameters of the g that optimally describes the geometric distortion.

As mentioned above, the distortion function g is not a general function: a suitable class of functions has some properties that should be satisfied to make them useful:

1. The first order approximation: $g(\bar{y}) \rightarrow \bar{y}$ for small \bar{y} .
2. g is a differentiable function $\mathbb{R}^2 \rightarrow \mathbb{R}^2$.
3. g is parameterized by a reasonably small number of parameters.
4. It should be easy to determine g^{-1} from the parameters of g .

Property 1 implies that the image is left undistorted by g at the center of the image. This does not necessary mean that the image cannot be distorted at the image center, i.e., at the principal point, the mapping to pixel coordinates represented by \mathbf{K} can still introduce non-uniform scaling and skewness. In mathematical terms, property 1 can be formulated as

$$g(\mathbf{0}) = \mathbf{0}, \quad \text{and} \quad \nabla g(\mathbf{0}) = \mathbf{I}_{2 \times 2}, \quad (18.7)$$

where ∇g is a 2×2 matrix that represents the derivatives of g relative to the elements of its vector argument. Property 2 is just a prerequisite for a mathematical analysis of the geometric distortion, e.g., it implies that ∇g is well-defined. The estimation of g amounts in practice to an optimization problem, and the fewer parameters that are involved in this estimation problem the more robust is the determination of its solution, hence item 3. Since both g and g^{-1} often are involved in the practical management of the geometric distortion, it is advantageous to realize both g and g^{-1} from the parameter or parameters that determine the distortion function. Otherwise implicit inversion must be carried out. In the literature, there are several proposals for how to parameterize g , and here we present some of the most common proposals.

18.2.1 Radial lens distortion

Besides properties 1 – 4, discussed above, is also common to assume that g is circular symmetric relative to the principal point. This assumption can be motivated by the circular symmetry of the camera lens, but since the manufacturing quality of the lens always is of limited accuracy, the circular symmetry of g should be seen as a simplification of the distortion effect that sometimes it reasonable. It implies that Equation (18.5) can be reformulated as

$$\bar{y}' = g(\bar{y}) = \bar{y} \cdot h(\|\bar{y}\|). \quad (18.8)$$

²The distance function \bar{d} is defined in Equation (2.4).

Here, $h : \mathbb{R} \rightarrow \mathbb{R}$ is a *radial distortion function* that describes how much each image point is displaced from its ideal position, but only in the radial direction relative to the principal point.

For $\bar{\mathbf{y}} \neq \mathbf{0}$, we can define normalized vectors $\hat{\mathbf{y}}$ and $\hat{\mathbf{y}}'$:

$$\hat{\mathbf{y}} = \frac{\bar{\mathbf{y}}}{\|\bar{\mathbf{y}}\|}, \quad \hat{\mathbf{y}}' = \frac{\bar{\mathbf{y}}'}{\|\bar{\mathbf{y}}'\|}, \quad (18.9)$$

and conclude that $\hat{\mathbf{y}} = \hat{\mathbf{y}}'$. Equation (18.8) may then be formulated alternatively as

$$\bar{\mathbf{y}}' = \hat{\mathbf{y}} \|\bar{\mathbf{y}}\| h(\|\bar{\mathbf{y}}\|). \quad (18.10)$$

Furthermore, from Equation (18.7) it follows that a general requirement on h is that

$$h(0) = 1. \quad (18.11)$$

Two typical cases can be described for the radial distortion model. If $h(r) < 1$ for $r > 0$ we refer to g as *barrel distortion*. This case implies that a square in front of the camera is projected and distorted such that the corners have been moved toward the principal point to a greater extent than the edges of the square, giving it a bulging appearance like a barrel. If $h(r) > 1$ for $r > 0$ we refer to g as *pin-cushion distortion*. In this case, the corners of the square instead have been moved away from the principal point, giving it a pin-cushion like appearance. In general, h can be a mix of both these cases, depending on the actual range of r .

Some of the more common proposals for how to parameterize a radial distortion function h are presented below.

Polynomial distortion

From a mathematical point of view, it makes sense to approximate an arbitrary and continuously differentiable function h in terms of a Maclaurin series:

$$h(r) = a_0 + a_1 r + a_2 r^2 + a_3 r^3 + \dots = / \text{Equation (18.11)} / = 1 + a_1 r + a_2 r^2 + a_3 r^3 + \dots \quad (18.12)$$

By truncating the series after the k -th order term, we arrive at k coefficients a_1, \dots, a_k that determine h as a k -th order polynomial. In practice, h is not an arbitrary polynomial. Property 1 of g , stated above, implies that $h(r) \approx 1$, i.e., the k coefficients in the polynomial must be relatively small. In the end, this parameterization leaves us with k parameters to tune in the process of estimating h . Although this parameterization is very general, since it can represent in principle any continuously differentiable function with arbitrary accuracy, it suffers from requiring a relatively large number of parameters (high polynomial order of h) in order to realize the distortion functions that appear in practice. Also, as soon as the polynomial order of h goes beyond 3, it is in practice not possible to determine g^{-1} as a simple functional expression in the parameters, and instead it has to be computed by some iterative method, e.g., Newton-Raphson.

Based on optical theory, h should be a power series of only even orders [37]. On the one hand, it thus makes sense to set $0 = a_1 = a_3 = \dots$, and instead truncate the series after the $2k$ -th order term to get k free parameters. This approach is used in most practical implementations of the distortion function g based on a radial effect. On the other hand, since g tries to compensate for effects not predicted by theory, it may be useful to also include the odd order terms, but then expect the corresponding coefficients to be small.

Physical models

For some types of lenses, it is possible to determine a mathematical model of what h looks like based on optics, at least on an approximate scale. An example that can model the large distortion effects in a fish-eye lens, is the so-called *fish-eye transform (FET)* proposed in [4]:

$$h(r) = \frac{\log(1 + \lambda r)}{\lambda r} \Rightarrow g(\mathbf{y}) = \hat{\mathbf{y}} \log(1 + \lambda \|\bar{\mathbf{y}}\|)/\lambda, \quad \Leftrightarrow \quad g^{-1}(\bar{\mathbf{y}}') = \hat{\mathbf{y}}' \left(e^{\lambda \|\bar{\mathbf{y}}'\|} - 1 \right) / \lambda, \quad (18.13)$$

This radial distortion function satisfies Equation (18.11) as a limit case. The parameter λ should be chosen such that $r\lambda \ll 1$ for all r within the actual range used for a particular camera. This model produces barrel distortion when $\lambda > 0$ and pin-cushion distortion when $\lambda < 0$, while $\lambda = 0$ corresponds to no radial distortion, see Figure 18.2.

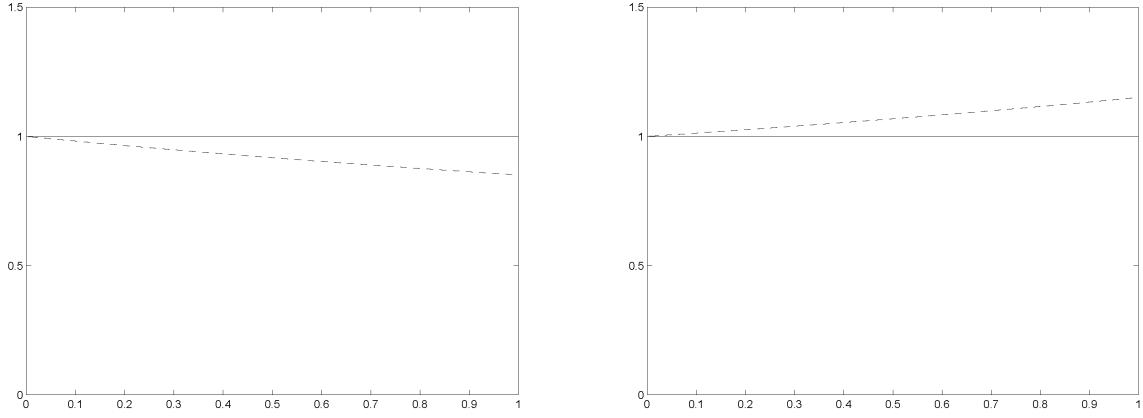


Figure 18.2: Two examples of the radial distortion function $h(r)$ of the FET-model defined in Equation (18.13), dotted curves. Left: $\lambda = -0.25$ which gives pin-cushion distortion. Right: $\lambda = 0.37$ which gives barrel distortion. Both give approximately 15% distortion at $r = 1$. Notice that both curves are relatively linear.

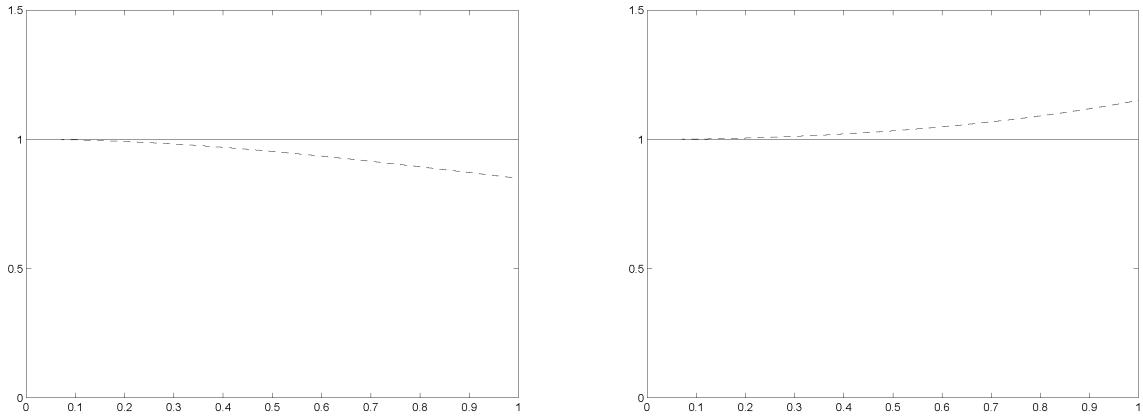


Figure 18.3: Two examples of the radial distortion function $h(r)$ of the FOV-model defined in Equation (18.14), dotted curves. Left: h for $\omega = 0.62$. Right: h_{INV} for $\omega = 0.78$. Both give approximately 15% distortion at $r = 1$. In contrast to h in Figure 18.2, initially this distortion function decreases rather slowly, then it reduces more distinctly.

An alternative function that better can model smaller amounts of geometric distortion is the *field of view model* (FOV), presented in [16]:

$$h(r) = \frac{1}{\omega r} \arctan(\omega r) \quad \Rightarrow \quad g(\bar{\mathbf{y}}) = \frac{\hat{\mathbf{y}}}{\omega} \arctan\left(2 \|\bar{\mathbf{y}}\| \tan \frac{\omega}{2}\right), \quad \Leftrightarrow \quad g^{-1}(\bar{\mathbf{y}}') = \frac{\hat{\mathbf{y}}' \tan(\|\bar{\mathbf{y}}'\| \omega)}{2 \tan \frac{\omega}{2}}. \quad (18.14)$$

This radial distortion function satisfies Equation (18.11) as a limit case. It is symmetric in the parameter ω which means that we can restrict it to $\omega > 0$ which corresponds to the pin-cushion case. This implies that h in Equation (18.14) only can be used for the case of pin-cushion distortion. If barrel distortion is wanted, the inverse mapping h_{INV} corresponding to g^{-1} can be used instead. The limit case $\omega = 0$ corresponds to no radial distortion. The FOV-model is illustrated in Figure 18.3.

Both the FET-model and the FOV-model have a single parameter, and are therefore simple to tune. They can also easily be inverted in order to obtain g^{-1} , which is an advantage in many cases. For more demanding applications, however, where high accuracy is required, they may be too simple. One approach that is suggested in [16] is to combine the polynomial expression in Equation (18.12) with one of the single-parameter models described here. Rather than to go for high order polynomial, it may be better to apply a non-linear function to a lower order polynomial.

Fraction of polynomials

Instead of approximating a more complicated function by means of a power series, as in Equation (18.12), an alternative is to model h as a fraction of two power series:

$$h(r) = \frac{1 + a_1 r + a_2 r^2 + \dots}{1 + b_1 r + b_2 r^2 + \dots}. \quad (18.15)$$

Here, the coefficients are chosen to comply with Equation (18.11). In practice the two power series are truncated to some suitable number of terms, and not necessarily of the same orders. As for the single polynomial model in Equation (18.12), practical implementations of the fractional model often use only the even order terms in both the numerator and the denominator.

18.2.2 General distortion models

One way to formally describe a general distortion function g in Equation (18.5) is as follows:

$$\bar{\mathbf{y}}' = \begin{pmatrix} u' \\ v' \end{pmatrix} = g(\bar{\mathbf{y}}) = \underbrace{\begin{pmatrix} g_r(\bar{\mathbf{y}}) & -g_t(\bar{\mathbf{y}}) \\ g_t(\bar{\mathbf{y}}) & g_r(\bar{\mathbf{y}}) \end{pmatrix}}_{:=\mathbf{G}(\bar{\mathbf{y}})} \begin{pmatrix} u \\ v \end{pmatrix} = \mathbf{G}(\bar{\mathbf{y}}) \bar{\mathbf{y}}, \quad (18.16)$$

where g_r describes the radial component, and g_t describes the tangential component of the distortion. To comply with property 1 of g , stated above, we require that $g_r(\mathbf{0}) = 1$ and $g_t(\mathbf{0}) = 0$. The functions g_r and g_t are general functions of $\bar{\mathbf{y}}$, for example parameterized as polynomials or fractions of polynomials in the elements of $\bar{\mathbf{y}} = (u, v)$. The case discussed in Section 18.2.1, where g is described in terms of only radial distortion, corresponds to $g_t = 0$ and $g_r(\bar{\mathbf{y}}) = h(\|\bar{\mathbf{y}}\|)$. We consider now the case that $g_t \neq 0$, and initially set $g_r(\bar{\mathbf{y}}) = h(\|\bar{\mathbf{y}}\|)$.

Decentering distortion

In particular for a lens system consisting of multiple lenses, the approximation that g is described only by a radial component, i.e., $g_t = 0$, is not valid. It is difficult to arrange the optical centers of the lenses exactly on a common axis, and as a consequence we get so-called *decentering distortion* which is characterized by having both a radial and a tangential component. A thorough analysis of this type of distortion [8, 13] shows that it can be described as:

$$\Delta_r = 3t(\bar{\mathbf{y}}) \sin(\phi - \phi_0), \quad \Delta_t = t(\bar{\mathbf{y}}) \cos(\phi - \phi_0), \quad (18.17)$$

where Δ_r and Δ_t are the radial and tangential components of the decentering distortion, ϕ is the angular coordinate of $\bar{\mathbf{y}} = \|\bar{\mathbf{y}}\| (\cos \phi, \sin \phi)$, ϕ_0 is the angle that gives a maximum distortion, and t is a function which describes the tangential distortion at different positions $\bar{\mathbf{y}}$. An exact formula for t is given in [8], based on ray-tracing through the lenses, which shows that t is mainly a function of the radial distance $\|\bar{\mathbf{y}}\|$, although it also has a weak dependency on ϕ . Notice that Equation (18.17) only describes distortion components that can be referred to the decentering of lenses, there may be additional radial distortion components that relate to other imperfections of the optical system. Furthermore, the radial component cannot be taken care of by the radial distortion function h , since Δ_r also has a dependency on ϕ .

Assuming that t has only a radial dependency that is linear: $t = t(r) = \alpha r$, the decentering distortion can be expressed in terms of a displacement (Δ_u, Δ_v) , in Cartesian coordinates, as

$$\begin{aligned} \Delta_u &= 2p_1uv + (r^2 + p_2)2u^2, & p_1 &= \alpha \cos \phi_0, & \bar{\mathbf{y}} &= (u, v), \\ \Delta_v &= p_1(r^2 + 2u^2) + 2p_2uv, & p_2 &= -\alpha \sin \phi_0, & r &= \|\bar{\mathbf{y}}\| = \sqrt{u^2 + v^2}. \end{aligned} \quad (18.18)$$

This formulation of the decentering distortion is sometimes referred to as *Brown-Conrady distortion*, *thin-prism distortion*, or *plumb bob distortion*. If we combine this with the radial model described in Section 18.2.1, the distortion function g can be formulated as

$$\bar{\mathbf{y}}' = g(\bar{\mathbf{y}}) = \begin{pmatrix} u' \\ v' \end{pmatrix} = h(\|\bar{\mathbf{y}}\|) \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} 2p_1uv + p_2(r^2 + 2u^2) \\ p_1(r^2 + 2u^2) + 2p_2uv \end{pmatrix}, \quad (18.19)$$

where h is chosen in some suitable way. Thus, we have two parameters, p_1, p_2 that need to be tuned in addition to h when g is determined.

18.2.3 General coordinate systems

The distortion function g is discussed here in the context of being applied to the C-normalized image coordinates $\bar{\mathbf{y}}$, which in a later step are mapped to pixel coordinates by means of the internal parameters in \mathbf{K} . This latter transformation includes scaling of the two coordinate axes, uniformly or non-uniformly, and skewing of the two axes. Hence, neither scaling nor skewing need to be included in g if it is applied on C-normalized coordinates, since these are absorbed into \mathbf{K} .

In practice, we may want to apply g directly onto the pixel coordinates, in particular if no information about the camera calibration is available. In this case, we need to take into account that the transformation to pixel coordinates implemented by \mathbf{K} typically also moves the origin of the coordinate system from the principal point to the upper left corner. Consequently, the calibration of g must in this case also include the position of the principal point $\bar{\mathbf{y}}_0 = (u_0, v_0)$, now referring to the pixel based coordinate system. Since all radial and tangential distortions discussed above are described relative to $\bar{\mathbf{y}}_0$, Equation (18.5) should in this case be replaced by

$$\bar{\mathbf{y}}' = \bar{\mathbf{y}}_0 + g(\bar{\mathbf{y}} - \bar{\mathbf{y}}_0). \quad (18.20)$$

The calibration of g must then include also $\bar{\mathbf{y}}_0$ are parameters.

Principal point versus distortion center

A subtle detail in these matters relates to the interpretation of the principal point in a real camera. The pinhole camera model is nothing more than a mathematical model, and although it is reasonably correct as a first order approximation, the introduction of a finite sized aperture and a lens or lenses have some effects on the geometry of the mapping from 3D space to the image plane. The principal point $\bar{\mathbf{y}}_0$ is the point in the image plane where this plane is intersected by the optical axis, which in turn includes the camera center, see Figure 8.3. In the ideal situation, the lens or the lens system has its optical center on the optical axis, which implies that the distortion is applied to coordinates defined from coordinate system with its origin at $\bar{\mathbf{y}}_0$.

In practice, there can be a small deviation between the *distortion center*, which serves as the origin for the distortion models discussed there, and the principal point [21]. For very accurate calibration of g , it is therefore necessary to take this deviation into account and reformulate Equation (18.5) as

$$\bar{\mathbf{y}}' = \bar{\mathbf{y}}_{DC} + g(\bar{\mathbf{y}} - \bar{\mathbf{y}}_{DC}), \quad (18.21)$$

where $\bar{\mathbf{y}}_{DC}$ the coordinates of the distortion center in the C-normalized coordinate system. This make $\bar{\mathbf{y}}_0$ yet another variable in the calibration of g , even if it is applied to C-normalization image coordinates.

18.2.4 OK, so which distortion model do I choose?

It is possible to apply many of the estimation problems presented here on image data without taking lens distortion into account, in particular if the camera has a high-quality lens. The lens distortion simply means that the more distortion that perturbs the data, the less accurate is the estimated model. At what point the lens distortion makes the result break down completely or makes it useless is difficult to predict, but a reasonable approach is to start with not taking lens effects into account at all and see what happens. If the estimated models appear to work fine, there is no need to address this issue.

Should the result be of less quality than expected, and the problem cannot be traced to other parts of the estimation, it is appropriate to adjust data in accordance with some lens distortion model. A first step is to consider only the radial component, where a simple model is used, with only a few parameters. If this does not help, a more complicated radial distortion function may be the way to go, but do not consider a radial model with many parameters until also the tangential distortion is included as well. Some software packages for camera calibration support also calibration of lens distortion. In this case, it is common that only one of the specific models described here can be used.

One issue in this discussion relates to the computational complexity of g and its inverse g^{-1} . The calibration of g is typically done as a single-shot process and from that perspective a complex distortion function, with many parameters, is acceptable even if it may be a high computational cost to determine the optimal parameters of g . On the other hand, if g needs to be applied to each pixel of every image, e.g., in a real-time application, a complex g may be a disadvantage, in particular if it cannot be inverted in a simple way. One strategy may be to

use relatively many parameters, determine an optimal g , and then investigate which of the parameters are very small. In a polynomial model, an insignificant parameter suggests that the corresponding term can be omitted in the model. If this is done in a careful way, it is then possible to reduce the complexity of g , by re-optimizing g after the insignificant terms are removed and with a smaller set of parameters.

18.2.5 Calibration of lens distortion

The calibration of the lens distortion of the camera, i.e., the estimation of the free parameters in the lens distortion function g , can be implemented in different ways. A simple approach is to do it separately from the rest of the calibration that determines, e.g., the internal camera parameters. Another approach is to include it into the calibration of all free parameters of the camera. The latter option is slightly more elaborate, but usually leads to better result.

Separate calibration of lens distortion

As is already mentioned, lens distortion makes lines that should be straight appear as more or less curved in the distorted image, as shown in Figure 18.1. In practice, the lines can be produced either by showing a special calibration pattern to the camera, containing a collection of straight lines, or viewing a scene that naturally contains straight lines, e.g., corners between walls, the floor, and the ceiling, or the edges of tables. Such a “line” in the image can then be detected and represented as a collection of points $\bar{\mathbf{y}}'_k$. Due to the lens distortion this “line” is slightly curved, and we want to find the distortion function g that makes the undistorted points $\bar{\mathbf{y}}_k = g^{-1}(\bar{\mathbf{y}}'_k)$ lie on an as straight curve as possible. A simple approach is to estimate the line from the points $\bar{\mathbf{y}}_k = g^{-1}(\bar{\mathbf{y}}')$, using any of the methods described in Chapter 12, and minimize the residual error over g [16]. If there are several lines in the image, each line is represented by a separate collection of points, and there is a separate residual error for each line relative to its points. In the end, we minimize some type of combined residual error over g . In this case, we can also use multiple images taken with the same camera where the distortion parameters are fixed.

This approach can typically be applied directly onto pixel coordinates, which implies that also the principal point need to be included into the lens distortion parameters.

Combined calibration

Using a separate step of the calibration of the lens distortion may not be not a good idea if in the next step we also want to do calibration of the internal camera parameters. At any case, the lens distortion *should not be taken care of after* the calibration of the internal parameters, since the lens distortion may have a significant impact on these parameters. The best approach is to combine the calibration of the lens distortion parameters and the internal camera parameters. In particular, this is a good idea since the transformation from C-normalized coordinates to pixel coordinates can introduce both non-uniform scaling and skewing of the coordinates.

A mathematical model of the mapping from 3D space to pixel coordinates is

$$\mathbf{y} \sim \mathbf{K} g_p((\mathbf{R} | \mathbf{t}) \mathbf{x}), \quad \text{where } g_p(\mathbf{y}) = \text{hom}(g(\text{cart}(\mathbf{y}))). \quad (18.22)$$

Here \mathbf{x} and \mathbf{y} are the homogeneous coordinates of a 3D point and its projection into the image, respectively, and cart and hom are the mappings between homogeneous coordinates to Cartesian coordinates, defined in Equations (3.9) and (3.10).

With this model of the 3D to 2D mapping of the camera, it appears natural that any camera calibration should include the lens distortion in the calibration process, and also that it should be determined together with all the other parameters. Section 18.3 presents a method for the calibration of \mathbf{K} , which can be extended to include also the calibration of the lens distortion function g .

18.2.6 Compensating for the lens distortion

Assuming that the lens distortion function g has been determined, what to next? There are at least two options: re-map the image or images, or re-map the coordinates of interest points.

The first option means that from the distorted image, I' , we create a new image, I , in terms of an array of pixels, as it would appear without the lens distortion. This implies that in I , projections of 3D lines are straight

rather than slightly curved. This re-mapping can be accomplished by defining a regular grid of Cartesian image in the undistorted coordinate system, $\bar{\mathbf{y}}$. To find the pixel values at each such grid point, we apply the distortion function g to get $\bar{\mathbf{y}}' = g(\bar{\mathbf{y}})$, an image coordinate in I' . Consequently, we obtain a coordinate in the original and distorted image, although not necessary at an integer pixel position. Using some type of interpolation scheme, for example linear interpolation of pixel values from four or more neighboring pixels, a pixel value at $\bar{\mathbf{y}}' = g(\bar{\mathbf{y}})$ can be determined for each $\bar{\mathbf{y}}$ in I' . In this approach, we only use g ; the inverse function g^{-1} is not needed. Also notice that this approach requires g to be defined relative to the pixel coordinate system, not the C-normalized image coordinates.

The resulting image, I , can then be processed in order to find interest points, and we can determine their coordinates in the undistorted coordinate system. As a result of the re-mapping, it must either be the case that there are some points, $\bar{\mathbf{y}}$, close to the edge in I , which end up at $\bar{\mathbf{y}}' = g(\bar{\mathbf{y}})$ outside the image I' . Thus, there may be some undefined pixel values in I . Alternatively, the size of I is chosen such that all $\bar{\mathbf{y}}' = g(\bar{\mathbf{y}})$ fall inside I' . In this case, all pixels of I have well-defined values, but there are some smaller regions close to the edge of I' that are not visible in I .

The second option for compensating for the lens distortion is to detect interest points in the original and distorted image I' , and bring their coordinates back to the undistorted coordinate system, without trying to reconstruct what that image looks like. Depending on the coordinate system that is used for defining g , this has to done in different ways.

If g refers to a pixel-based coordinate system, then coordinate $\bar{\mathbf{y}}'$ in I' is re-mapped to $\bar{\mathbf{y}} = g^{-1}(\bar{\mathbf{y}}')$ in I . As described in Section 18.2.3, this requires that g also includes the principal point as a parameter and that the application of g implicitly means that the distortion is made relative to this point. If g refers to the principal point, as in Equation (18.22), then

$$\mathbf{y} \sim \mathbf{K} g_p^{-1} (\mathbf{K}^{-1} \mathbf{y}). \quad (18.23)$$

Here, \mathbf{K} is the internal camera parameters, and g_p^{-1} is the inverse of g , defined on homogeneous coordinates and producing homogeneous coordinates, and \mathbf{y} and \mathbf{y}' are the homogeneous coordinates of the undistorted and distorted points, respectively, both in pixel coordinates. This approach assumes that the value of the inverse of the distortion function, g^{-1} , can be determined for arbitrary $\bar{\mathbf{y}}'$ in I' .

18.3 Zhang's calibration method

The idea of automatic camera calibration discussed in Section 18.1 may be appealing just because it is automatic, but in practice it amounts to relatively complex estimation problems. An alternative is to use a semi-automatic approach, where the image points that are used for the calibration are generated from some type of special calibration pattern that is depicted by the camera. One such popular approach is described by Zhang [68, 67], and here we make brief presentation of the basic theory and practical implementation of this method. Initially, we ignore the effect of lens distortion discussed in the previous section, and return to this issue in Section 18.3.2.

The method is based on a *calibration device*, in this case a planar surface that contains a special calibration pattern in terms of a set of m points in a regular pattern. The points are typically either small “dots” or the corners of a checker board pattern, which easily can be detected in the image. The pattern is also designed to allow the detected interest points in the image to be brought into correspondence with the points in the calibration device. In practice, this is done by arranging the points in a regular array, where each point has a well-defined position given by its row and column of the array. Positions of the points in the calibration pattern are known, e.g., in terms of the spacing between the columns and rows of the array. A typical implementation of the calibration device is to print the calibration pattern on a sheet of paper, which is then firmly attached to a flat surface, such as a piece of cardboard or, even better, made of plywood, see Figure 18.4. The distance between the points in the grid is measured as accurately as possible, e.g., in units of millimeter. This means that we can assign a 2D Cartesian coordinate $\bar{\mathbf{y}}_{0j}$ to each point $j = 1, \dots, m$ in the pattern. As it turns out, the exact location of the origin is not important as long as it is fixed within the pattern. It is, however, important that the rows and columns of the array are perpendicular which, in turn, depend on the quality of the printer.

The calibration device is shown to the camera at n positions and orientations, and for each view $i = 1, \dots, n$, the interest points from the calibration device in the image are automatically detected and brought into correspondence with the corresponding points in the pattern. Assuming that the pinhole camera model is valid, the transformation between the 2D coordinates in the pattern, with homogeneous coordinates \mathbf{y}_{0j} , and the image coordinates in view i ,

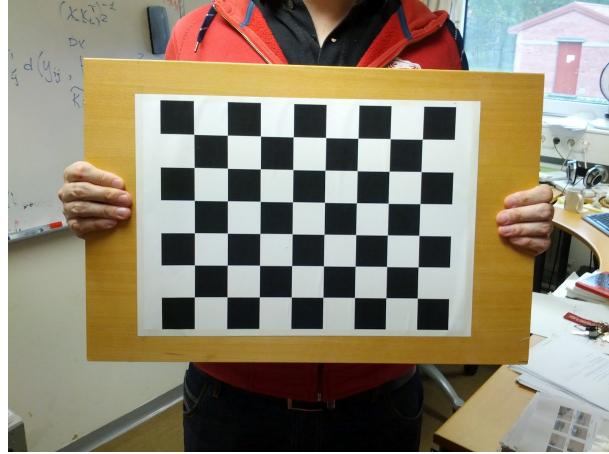


Figure 18.4: A calibration device used in Zhang's calibration method.

\mathbf{y}_{ij} , is represented by a homography, in accordance with Section 9.2:

$$\mathbf{y}_{ij} \sim \mathbf{H}_i \mathbf{y}_{0j}. \quad (18.24)$$

The image of the pattern is determined only by the internal parameters of the camera, \mathbf{K} , and of the pose of the camera *relative to the pattern*. Consequently, we can define the world coordinate system as attached to the calibration device, with its origin at the origin of the 2D coordinate system of the pattern, the first two axes aligned with the rows and the columns, and the third axis a normal to the plane. The 3D coordinates of point j in the pattern is given as

$$\mathbf{x}_j \sim \mathbf{P} \mathbf{y}_{0j}, \quad j = 1, \dots, m, \quad (18.25)$$

where

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (18.26)$$

This means that the 2D coordinates (x_1, x_2) in the pattern are mapped to 3D world coordinates $(x_1, x_2, 0)$. These 3D points are then projected to image points in accordance with

$$\mathbf{y}_{ij} \sim \mathbf{C}_i \mathbf{x}_j \sim \mathbf{K}(\mathbf{R}_i | \bar{\mathbf{t}}_i) \mathbf{x}_j \sim \mathbf{K}(\mathbf{R}_i | \bar{\mathbf{t}}_i) \mathbf{P} \mathbf{y}_{0j}. \quad (18.27)$$

Here, we assume that each camera matrix \mathbf{C}_i represents a fixed set of internal parameters \mathbf{K} and variable pose in terms of $(\mathbf{R}_i | \bar{\mathbf{t}}_i)$. The rotation matrix \mathbf{R}_i consists of three columns:

$$\mathbf{R}_i = (\bar{\mathbf{r}}_{1i} \quad \bar{\mathbf{r}}_{2i} \quad \bar{\mathbf{r}}_{3i}) \quad (18.28)$$

that form an ON-basis:

$$\bar{\mathbf{r}}_{ki} \cdot \bar{\mathbf{r}}_{li} = \delta_{kl} = \begin{cases} 1 & k = l, \\ 0 & k \neq l, \end{cases} \quad i = 1, \dots, n. \quad (18.29)$$

The combination of Equation (18.26), Equation (18.27), and Equation (18.28) gives

$$\mathbf{y}_{ij} \sim \mathbf{K}(\bar{\mathbf{r}}_{1i} \quad \bar{\mathbf{r}}_{2i} \quad \bar{\mathbf{t}}_i) \mathbf{y}_{0j}. \quad (18.30)$$

A comparison with Equation (18.24) allows us to identify the homography transformation \mathbf{H}_i as

$$\mathbf{H}_i \sim \mathbf{K}(\bar{\mathbf{r}}_{1i} \quad \bar{\mathbf{r}}_{2i} \quad \bar{\mathbf{t}}_i), \quad i = 1, \dots, n. \quad (18.31)$$

The homography matrix \mathbf{H}_i can be expressed in terms of its three columns:

$$\mathbf{H}_i = (\mathbf{h}_{1i} \quad \mathbf{h}_{2i} \quad \mathbf{h}_{3i}), \quad (18.32)$$

leading to

$$\bar{\mathbf{r}}_{1i} \sim \mathbf{K}^{-1}\mathbf{h}_{1i}, \quad \text{and} \quad \bar{\mathbf{r}}_{2i} \sim \mathbf{K}^{-1}\mathbf{h}_{2i}. \quad (18.33)$$

Finally, we insert these expressions for $\bar{\mathbf{r}}_{1i}$ and $\bar{\mathbf{r}}_{2i}$ into Equation (18.29) and get:

$$\begin{aligned} 0 &= \bar{\mathbf{r}}_{1i}^\top \bar{\mathbf{r}}_{2i} = \mathbf{h}_{1i}^\top \mathbf{K}^{-\top} \mathbf{K}^{-1} \mathbf{h}_{2i}, \\ 0 &= \bar{\mathbf{r}}_{1i}^\top \bar{\mathbf{r}}_{1i} - \bar{\mathbf{r}}_{2i}^\top \bar{\mathbf{r}}_{2i} = \mathbf{h}_{1i}^\top \mathbf{K}^{-\top} \mathbf{K}^{-1} \mathbf{h}_{1i} - \mathbf{h}_{2i}^\top \mathbf{K}^{-\top} \mathbf{K}^{-1} \mathbf{h}_{2i}. \end{aligned} \quad (18.34)$$

Here, all of a sudden, appears $\boldsymbol{\omega} = \mathbf{K}^{-\top} \mathbf{K}^{-1}$, the image of the absolute conic. Therefore, then can be rewritten as

$$\begin{aligned} 0 &= \mathbf{h}_{1i}^\top \boldsymbol{\omega} \mathbf{h}_{2i}, \\ 0 &= \mathbf{h}_{1i}^\top \boldsymbol{\omega} \mathbf{h}_{1i} - \mathbf{h}_{2i}^\top \boldsymbol{\omega} \mathbf{h}_{2i}, \quad i = 1, \dots, n. \end{aligned} \quad (18.35)$$

We stop here and summarize the results so far. For each observation of the calibration device, we can determine a homography \mathbf{H}_i that relates the 2D coordinates of the points in the calibration pattern \mathbf{y}_{0j} , with their corresponding image points \mathbf{y}_{ij} , Equation (18.24). We know \mathbf{y}_{0j} from the initial preparation of the calibration device, and can automatically find the image coordinates \mathbf{y}_{ij} . Using any of the techniques for estimation of homographies that are described in Sections 13.1 and 14.5, we can then estimate \mathbf{H}_i . This homography provides two linear homogeneous constraints in the IAC $\boldsymbol{\omega}$, specified in Equation (18.35). Since \mathbf{K} , and therefore also $\boldsymbol{\omega}$, have 5 degrees of freedom, we need at least 3 homographies, from 3 observations of the calibration device in general positions and orientations relative to the camera, in order to determine $\boldsymbol{\omega}$. In practice, it is advisable to use much more than the minimal number of observations.

A more explicit representation of the two constraints in Equation (18.35) can be obtained by writing

$$\boldsymbol{\omega} = \begin{pmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{12} & \omega_{22} & \omega_{23} \\ \omega_{13} & \omega_{23} & \omega_{33} \end{pmatrix} \quad \text{and} \quad \mathbf{h}_{1i} = \begin{pmatrix} h_{11i} \\ h_{21i} \\ h_{31i} \end{pmatrix}, \quad \mathbf{h}_{2i} = \begin{pmatrix} h_{12i} \\ h_{22i} \\ h_{32i} \end{pmatrix}. \quad (18.36)$$

Inserted into Equation (18.35), the two constraints become

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} = \underbrace{\begin{pmatrix} h_{11i}h_{12i} & h_{12i}h_{21i} + h_{11i}h_{22i} & h_{12i}h_{31i} + h_{11i}h_{32i} & h_{21i}h_{22i} & h_{22i}h_{31i} + h_{21i}h_{32i} & h_{31i}h_{32i} \\ h_{11i}^2 - h_{12i}^2 & 2h_{11i}h_{21i} - 2h_{12i}h_{22i} & 2h_{11i}h_{31i} - 2h_{12i}h_{32i} & h_{21i}^2 - h_{22i}^2 & 2h_{21i}h_{31i} - 2h_{22i}h_{32i} & h_{31i}^2 - h_{32i}^2 \end{pmatrix}}_{:=\mathbf{A}_i} \underbrace{\begin{pmatrix} \omega_{11} \\ \omega_{12} \\ \omega_{13} \\ \omega_{22} \\ \omega_{23} \\ \omega_{33} \end{pmatrix}}_{:=\mathbf{z}}. \quad (18.37)$$

This corresponds to the homogeneous equation $\mathbf{A}_i \mathbf{z} = \mathbf{0}$ for each $i = 1, \dots, n$, represented by the 2×6 matrix \mathbf{A}_i . We can combine all these $2n$ equations into

$$\underbrace{\begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_n \end{pmatrix}}_{:=\mathbf{A}} \mathbf{z} = \mathbf{0}. \quad (18.38)$$

This means that \mathbf{z} must satisfy the linear homogeneous equation

$$\mathbf{A} \mathbf{z} = \mathbf{0}, \quad (18.39)$$

where \mathbf{A} is $2n \times 6$.

For real data, Equation (18.39) cannot be solved exactly and instead we can use either the homogeneous or the inhomogeneous method, described in Chapter 12 to minimize $\|\mathbf{A} \mathbf{z}\|$. Once $\mathbf{z} \in \mathbb{R}^6$ is determined, we reshape it to the symmetric 3×3 matrix $\boldsymbol{\omega}$ in Equation (18.36), determine its Cholesky factor $\mathbf{K}^{-\top}$ and apply a matrix transpose and inverse. The result is \mathbf{K} . If Cholesky decomposition is not available, use the approach to determine \mathbf{K} described in Equation (16.54).

18.3.1 Geometric errors

To get a reliable estimate of ω , each homography \mathbf{H}_i should be estimated by minimizing a geometric error, and we should use more than the minimum 3 observations of the calibration device, preferably as many as is practically possible. The minimization of $\|\mathbf{A} \mathbf{z}\|$ by means of the homogeneous or the inhomogeneous method implies the minimization of an algebraic error, and to produce a more accurate estimate of \mathbf{K} it is better to minimize a geometric error. This is done based on the non-linear estimation methods described in Chapter 14, and they require initial solutions for the unknown parameters.

A suitable formulation of an error for geometric estimation of \mathbf{K} is in this case given as

$$\varepsilon_{GEOM,C} = \sum_{i=1}^n \sum_{j=1}^m d_{PP}(\mathbf{y}_{ij}, \mathbf{H}_i \mathbf{y}_{0j})^2 = \sum_{i=1}^n \sum_{j=1}^m d_{PP}(\mathbf{y}_{ij}, \mathbf{K}(\bar{\mathbf{r}}_{1i} \bar{\mathbf{r}}_{2i} \bar{\mathbf{t}}_i) \mathbf{y}_{0j})^2. \quad (18.40)$$

Here, d_{PP} is the distance function defined in Section 3.4.1. The final estimate of \mathbf{K} is obtained by minimizing $\varepsilon_{GEOM,C}$ over \mathbf{K} as well as over all \mathbf{R}_i and $\bar{\mathbf{t}}_i$, using the techniques presented in Chapter 17. In the end, the external camera parameters are not really interesting, but must be included in the optimization since the observed data, \mathbf{y}_{ij} , depend on them. As such they are *auxiliary variables* in this optimization problem.

One way to get initial solutions is to estimate the internal parameters \mathbf{K}_0 in accordance with the algebraic procedure described above. For each homography \mathbf{H}_i , we can then determine an initial estimate also of the external parameters from Equation (18.31):

$$(\bar{\mathbf{r}}_{1i} \quad \bar{\mathbf{r}}_{2i} \quad \bar{\mathbf{t}}_i) \sim \mathbf{K}_0^{-1} \mathbf{H}_i, \quad i = 1, \dots, n. \quad (18.41)$$

The right-hand side of this expression is based on an estimate of \mathbf{H}_i , as such it does not need to satisfy any specific constraints. On the other hand, the two leftmost columns of the matrix in the left-hand side should satisfy Equation (18.29). To deal with this situation we can normalize the matrix such that the norm of the leftmost column is = 1, which gives us $\bar{\mathbf{r}}_{1i}$ and $\bar{\mathbf{t}}_i$. A Gram-Schmidt orthogonalization is then applied on of the second column relative the first, which gives $\bar{\mathbf{r}}_{2i}$. The third column $\bar{\mathbf{r}}_{3i}$ of \mathbf{R}_i is then given as $\bar{\mathbf{r}}_{3i} = \bar{\mathbf{r}}_{1i} \times \bar{\mathbf{r}}_{2i}$. In the end, this produce an initial estimate $\mathbf{R}_{i0}, \bar{\mathbf{t}}_{i0}$ of the external camera parameters related to the camera for observation i .

18.3.2 Lens distortion

Minimizing the geometric error $\varepsilon_{GEOM,C}$ in Equation (18.40) requires some type of iterative optimization method, where initial solutions for the external and internal parameters can be determined by means of algebraic methods, see Chapter 17. Once we have decided to go all the way to implement the minimization of a geometric error using non-linear methods, there is not much additional work to also take calibration of the lens distortion into account. Using the same argumentation is in Section 18.2.5, Equation (18.22), we can formulate the geometric error as

$$\varepsilon_{GEOM,C+L} = \sum_{i=1}^n \sum_{j=1}^m d_{PP}(\mathbf{y}_{ij}, \mathbf{K} g_p((\bar{\mathbf{r}}_{1i} \bar{\mathbf{r}}_{2i} \bar{\mathbf{t}}_i) \mathbf{y}_{0j}))^2, \quad (18.42)$$

where g_p is the distortion function adapted to homogeneous coordinates described in Equation (18.22). In the simplest case it represents only radial distortion, discussed in Section 18.2.1. This adds a few more parameters to the optimization problem, but also produces a more accurate estimate of both \mathbf{K} and g , compared to first applying a separate lens distortion calibration, followed by an estimation of only \mathbf{K} based on coordinates that have been compensated for the lens effect.

18.3.3 Summary

The basic version of Zhang's method for calibration of \mathbf{K} , based on algebraic errors, is described in Algorithm 18.1. The extension based on minimization of geometric errors, which also can take the lens distortion into account, is described in Algorithm 18.2. In the latter case, which also is the recommended approach, it is necessary to parameterize the rotations \mathbf{R}_i in one way or another, as discussed in Chapter 15, and also to parameterize the lens distortion g as discussed in Section 18.2.

A comparison to the camera calibration method described in Section 18.1.1, based on rotational homographies, shows that Zhang's method has a few advantages. As a start, while a rotating camera requires either a calibrated

Algorithm 18.1: Zhang's calibration method based on algebraic minimization

```
Input: A calibration device with known 2D positions for points  $\mathbf{y}_{0j}, j = 1, \dots, m$ 
Input:  $n \geq 3$  = how many times the calibration device is shown to the camera
Output: The internal calibration  $\mathbf{K}$ 
1 Initialize:  $\mathbf{A}$  = empty  $0 \times 6$  matrix
2 foreach camera view  $i = 1, \dots, n$  do
3   Show the calibration device to the camera at random position and orientation
4   Measure the image coordinates  $\mathbf{y}_{ij}$  of the interest points generated by the calibration device
5   Estimate homography  $\mathbf{H}_i$  satisfying  $\mathbf{y}_{ij} \sim \mathbf{H}_i \mathbf{y}_{0j}$ 
6   Form  $2 \times 6$  matrix  $\mathbf{A}_i$  from  $\mathbf{H}_i$ , equation Equation (18.37)
7   Append the rows at the bottom of  $\mathbf{A}_i$  to  $\mathbf{A}$ :  $\mathbf{A} = (\mathbf{A}; \mathbf{A}_i)$ 
8 end
9 Find  $\mathbf{z}$  that solves  $\mathbf{A} \mathbf{z} = \mathbf{0}$ 
10 Reshape  $\mathbf{z}$  to  $\boldsymbol{\omega}$ , equation Equation (18.36)
11 Find  $\mathbf{K}^{-\top}$  as the Cholesky factor of  $\boldsymbol{\omega}$ , and apply matrix inverse and transpose to get  $\mathbf{K}$ 
12 Alternatively, use Equation (16.54)
```

mechanical gimbal, or that the scene is distant, the proposed method allows that the camera to be hand-held and also that it can be moved around in 3D space rather free. Zhang's method is also relatively independent of exactly which calibration pattern is used, as long as it is clearly visible in the images.

The manual interaction of this method is limited to producing the calibration device and measuring the positions of the points in the pattern, e.g., by an as accurate at possible measurement of the distance between the rows and between the columns in the pattern. Also, you need to capture $n \geq 3$ images of the calibration device in random position and orientations. The rest of the computations can be done automatically.

Concluding remarks

The journal article by Zhang [68] contains a presentation of the algebraic version of the method, while the longer technical report [67] contains are more complete presentation of the method, including using geometric minimization and managing lens distortion. The geometric minimization is the recommended approach since, in general, it produces more useful results and also easily can be combined with the calibration of the geometric distortion.

Algorithm 18.2: Zhang's calibration method based on geometric minimization

```

Input: A calibration device with known 2D positions for points  $\mathbf{y}_{0j}, j = 1, \dots, m$ 
Input:  $n \geq 3$  = how many times the calibration device is shown to the camera
Output: The internal calibration  $\mathbf{K}$ 
Output: Optional: The lens distortion  $g$ 

1 Determine an initial estimate  $\mathbf{K}_0$  using Algorithm 18.1
2 Algorithm 18.1 also provides estimates of homographies  $\mathbf{H}_j$ 
3 foreach camera view  $i = 1, \dots, n$  do
4   /* Determine initial estimates of camera pose  $\mathbf{R}_i, \bar{\mathbf{t}}_i$  */
5   Compute:  $(\bar{\mathbf{r}}_{1i} \bar{\mathbf{r}}_{2i} \bar{\mathbf{t}}_i) = \mathbf{K}_0^{-1} \mathbf{H}_i$ 
6   Normalize:  $(\bar{\mathbf{r}}_{1i} \bar{\mathbf{t}}_i) := (\bar{\mathbf{r}}_{1i} \bar{\mathbf{t}}_i) / \|\bar{\mathbf{r}}_{1i}\|$ 
7   Gram-Schmidt:  $\bar{\mathbf{r}}_{2i} = \bar{\mathbf{r}}_{2i} - \bar{\mathbf{r}}_{1i}(\bar{\mathbf{r}}_{1i} \cdot \bar{\mathbf{r}}_{2i})$ 
8   Normalize:  $\bar{\mathbf{r}}_{2i} = \bar{\mathbf{r}}_{2i} / \|\bar{\mathbf{r}}_{2i}\|$ 
9   Compute:  $\bar{\mathbf{r}}_{3i} = \bar{\mathbf{r}}_{1i} \times \bar{\mathbf{r}}_{2i}$ 
10  Set  $\mathbf{R}_i = (\bar{\mathbf{r}}_{1i} \bar{\mathbf{r}}_{2i} \bar{\mathbf{r}}_{3i})$ 
11 end
12 Optional: find initial estimate of lens distortion parameters
13 With  $\mathbf{K}_0$  and  $(\mathbf{R}_i, \bar{\mathbf{t}}_i)$  as initial estimates:
14   Minimize  $\varepsilon_{GEOM,C}$  in Equation (18.40) or, optionally, minimize  $\varepsilon_{GEOM,C+L}$  in Equation (18.42):
15     Minimize over  $\mathbf{K}$  and with  $(\mathbf{R}_i, \bar{\mathbf{t}}_i)$  as auxiliary variables, optionally also over parameters for  $g$ ;

```

Chapter 19

Image Mosaics

Before you read this chapter, you should have thorough understanding of the pinhole camera, described in Chapter 8, and of epipolar geometry, Chapter 10. In this chapter, we will also use OPP, described in Chapter 15.

In Chapter 9 we have seen that two or more images taken by a pinhole camera in certain cases can be related by a homography transformation. In section Section 9.3 it was shown that images taken by rotating cameras, i.e., cameras that have a common center, are related by a homography. Section 9.2 demonstrated that two cameras observing a plane also are related by a homography. In practice, transforming points in one image to the corresponding points in the other image using a homography is possible also if the stipulated requirements are approximately satisfied. If the cameras have approximately a common center, or if the points in the scene are approximately on a plane, it is possible to find a homography that approximately relates the coordinates of corresponding points in two images.

In Chapter 10 we have also seen that the general case when two cameras that are depicting the same scene is more complicated. There is no fixed transformation that relates corresponding image points in this case, instead they are related by the epipolar constraint. Rotating cameras and cameras observing a plane, are often described as degenerate cases of epipolar geometry in the sense that epipolar constraints and fundamental matrices are not well-defined. But these two cases also offer the possibility of relating two or more images of the same scene in a much simpler way than is possible in the general. This observation is the basis for this chapter, where it is used for the purpose of building image mosaics or panorama images. This is done by first determining the homographies that relate corresponding points in two or more images, then use the homographies to transform points to a common coordinate system and paste the images onto each other, thereby forming one large image of the scene.

19.1 Introduction

To get a better understanding of the practical aspects of image mosaics, let us first look at how we can acquire image data for the purpose of creating such mosaics.

Rotating cameras

The case of rotating cameras appears in practice if you put your camera onto a tripod and take several images of the same scene while rotating the camera on the tripod between the images, keeping the tripod stationary. The rotation center of your tripod may not coincide exactly with the camera center, but if the scene is at a sufficient distance from the camera, this deviation can be neglected. You can even take the images with a hand-held camera, pointing the camera in different directions and not moving yourself around too much between the images. If the scene is sufficiently far away, the camera centers are approximately coinciding and the cameras are approximately equivalent, and the images can approximately be related by a homography transformation.

If you have multiple cameras, another possibility to make them equivalent is to mount them to a common mechanical support, as close together as possible and pointing in different directions. As long as they depict a

scene that is sufficiently far away from the cameras, their camera centers are approximately coinciding and they can be seen as approximately equivalent. By a suitable arrangement of the cameras, they can be made to cover a larger field of view, and it is possible to produce an image mosaic from the entire field. Figure 9.7 on page 136 shows an example of this type of camera system. Another possibility along these lines is to use a single camera in combination with lenses and/or curved mirrors to extend the field of view, a so-called *omni-directional camera*. This type of cameras, however, is outside the scope of this presentation.

Cameras observing a plane

A practical example of cameras observing a plane is an aerial vehicle looking down at the ground from different viewpoints and from a high altitude, or a satellite taking pictures of the Earth. As long as the differences in altitude of the points on the ground are small compared to the altitude of the camera, and the area covered by the images is small enough to make the Earth's curvature negligible, corresponding points in different images are approximately related by a homography transformation. Other examples are if you take pictures from different viewpoints of the facade or a wall of a building, or of the floor or a wall of a room. As long as the points in the scene can be said to be approximately on a plane, i.e., the points in the scene have a distance to this plane that is small relative to the distance between the plane and the camera center, then corresponding points in two of the images are approximately related by a homography.

Image mosaics and panorama images

An observation that we already made and is relevant in this context is that images in practice are of finite extension, they are typically truncated to be defined only within a rectangular boundary, the *bounding box*. This means that it only makes sense to transform the points inside the bounding box or an image. It also means that if two cameras are equivalent, or are depicting a plane, and the image of one camera is transformed to the coordinate system of the other camera, the bounding boxes of the two images will in general not coincide. But if each camera has a field of view that is carefully chosen, there can be an overlap between the two bounding boxes. This observation can be used to "paste" images together from two equivalent cameras, or two cameras that depict a plane.

If there are images from multiple cameras, we can transform all the images to a common coordinate system, e.g., as defined by one of the cameras, and merge them to one single image. In general, for both rotating cameras and cameras observing a plane, the resulting combined image is referred to as an *image mosaic*, and the process of forming the mosaic is called *image mosaicking* or *image stitching*. In the specific case of rotating cameras, where the camera center is fixed, the terms *panorama image* and *panorama stitching* are also used.

In this chapter we will go through the process of forming image mosaics. We start in Section 19.2 with the basic approach that uses homographies for transforming corresponding points in two images. Since homographies are very general transformations, they can severely distort an image in the case of panorama images, which is the reason for considering another type of transformation, based on spherical coordinates in Section 19.3. Transforming coordinates in one way or another only gives a geometrical representation of how corresponding points in two images are related, in practice we also need to worry about the intensity values of two corresponding points. In the ideal case, two corresponding points should have the same intensity values. In practice there may be a small difference that can be an issue if two images are to be merged into a single image after the transformation. This issue is discussed in Section 19.2.2.

Lens distortion

In Chapter 18 we discussed and modeled the geometric distortion caused by the lens or lenses of a camera. This distortion can be seen as a correction to the pinhole camera model that we need to apply in order to accurately describe how 3D points are mapped to image points for a real camera. Since we use real cameras for producing the images described here, the lens effect may have to be taken into account. Either it is sufficiently small to be neglected, or can be seen as part of the measurement noise that inevitably affects the coordinates of the image points. Alternatively, it must be taken care of by choosing a suitable distortion function g , Equation (18.5), and estimate its parameters. Once g is known, we can compensate for the lens distortion and produce (approximately) undistorted coordinates, as described in Section 18.2.6. These are the coordinates that are used in this chapter, i.e., we assume that all image coordinates are not affected to any significant extent by lens effects.

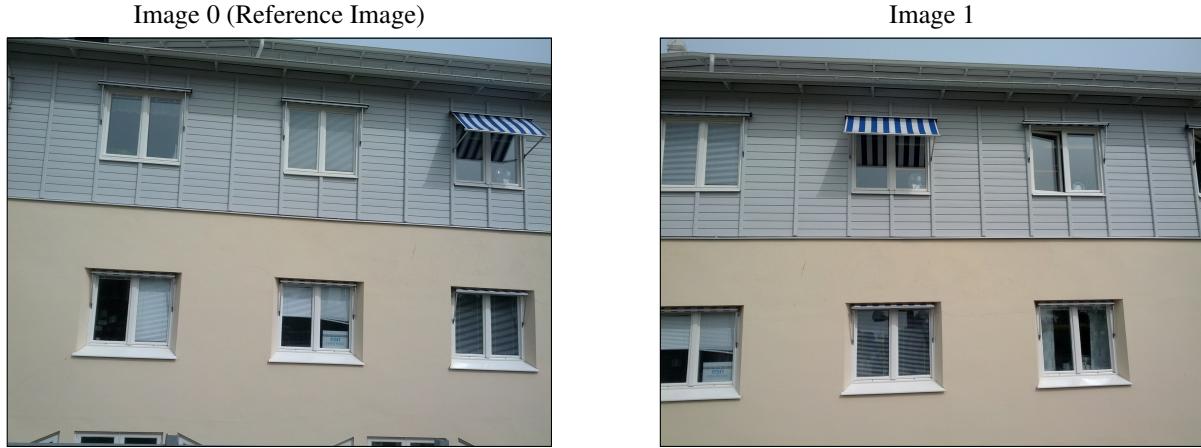


Figure 19.1: Two images of a building facade, taken from two distinct viewpoints. In the text, the left image is labeled 0, the right image is labeled 1. Image 0 is the reference image, defining the coordinate system to which image 1 is transformed.

19.2 Using homographies

In this section we will develop an approach for creating an image mosaic from a set of images that are related by homography transformations. We will consider the case of cameras observing a plane as an example, but the following results are valid also for the case of rotating cameras. We start with the simplest case where the mosaic is built from only two images, and then consider the general problem of producing a mosaic from multiple images.

19.2.1 A mosaic from two images

As an example, we use the two images in Figure 19.1 that depict the facade of a building, approximately a plane, taken from two distinct viewpoints. In the following discussion, the left and right images are referred to as image 0 and image 1, respectively.

The reference image

In the following discussion we also need to designate one of the two images as the *reference image* of the mosaic. The reference image is the image that defines the coordinate system of the resulting image mosaic, and it always has index 0. Hence, in our example it corresponds to the left image in Figure 19.1. All other images will be transformed to the coordinate system of the reference image, and are then pasted onto the reference image. This operation normally expands the reference image, and it may sometimes also require the coordinate system of the reference image to be translated. As a consequence, the image mosaic of a set of images corresponds to a sequence of reference images, all with index 0, where each one is augmented by pasting a new image from the set to create the next reference image. The final reference image in this sequence is the resulting image mosaic. With image 0 and image 1 defined from Figure 19.1, the current objective is to create a new reference image by pasting image 1 onto this initial reference image, producing a new reference image. The pasting procedure includes several steps:

1. find the homography transformation that relates image 1 to the coordinate system of the reference image,
2. find the bounding box of the new reference image,
3. transfer the old reference image and image 1 into the new reference image.

Next, we will look at these steps more in detail.

Finding the homography

To determine the homography transformation, denoted \mathbf{H} , between the two images there must be some amount of overlap in their field of view. In our example, Figure 19.1, we see that the rightmost windows in image 0 are the same as the leftmost windows in image 1. In this overlap we can find some corresponding image points, and since the corresponding 3D points lie approximately in a plane, corresponding image points are (approximately) related by a homography. These corresponding image points can be determined either manually, or automatically.

In the manual case, \mathbf{H} can be estimated directly from the correspondences using Algorithm 13.3 on page 238. In this case, the quality of \mathbf{H} is critically dependent on not picking correspondence that happen to be off the common plane that all 3D points are assumed to lie in. For example, clouds in the background, or people in the foreground of the facade should not be used.

In the case of automatic homography estimation, \mathbf{H} can be determined from a set of interest points that have been detected in each image. This can be done by applying Algorithm 17.3 on page 307 that robustly finds both a set of correspondences and \mathbf{H} , based on RANSAC. Correspondences that are off the plane should be avoided also in this case, but they are not as critical this time, since they are likely to be removed by RANSAC from the consensus set of the resulting estimate.

In both the manual and the automatic case, we want to assure that \mathbf{H} aligns the two images as well as possible. This can be done by refining an algebraic estimate of the homography by a geometric estimation, using the algebraic estimate as an initial solution, see Section 14.5. As a result of this operation we obtain the homography \mathbf{H} that can transform a point (u_1, v_1) in image 1, represented by homogeneous coordinates \mathbf{y}_1 , to the corresponding point (u_0, v_0) in image 0 (the reference image), represented by homogeneous coordinates \mathbf{y}_0 :

$$\mathbf{y}_0 \sim \mathbf{H}\mathbf{y}_1, \quad \Leftrightarrow \quad (u_0, v_0) = \text{cart}(\mathbf{H}\text{hom}(u_1, v_1)). \quad (19.1)$$

Here, hom and cart are the mappings between Cartesian and homogeneous coordinates defined in Section 3.1.1. Since a homography is invertible, \mathbf{H}^{-1} transforms \mathbf{y}_0 in image 0 back to \mathbf{y}_1 in image 1.

Once \mathbf{H} is established, it can transform *any* point in image 1 to its corresponding position in the reference image, not just the interest points from which it has been estimated. This implies that we can transform the entire pixel grid of image 1 to the coordinate system of the reference image. Before we look at this problem, however, there are some practical issues that need to be sorted out first.

Finding the bounding box of the new reference image

We assume that image coordinates (u, v) refer to the pixel grid of an image, as described in Section 8.4. In this presentation, we will assume that this coordinate system is of type (right, down) and use the zero-first enumeration convention, i.e., the origin of the coordinate system is centered on the pixel at the upper left corner of the image, and the first axis points right and the second axis points down.

Based on this assumption, the bounding box of an image can be defined in terms of the Cartesian coordinates of the lower right corner, i.e., of the lower rightmost pixel, (u_{LR}, v_{LR}) . When the image has c columns and r rows, then $\bar{\mathbf{y}}_{LR} = (u_{LR}, v_{LR}) = (c - 1, r - 1)$ when the zero-first enumeration convention is used. The Cartesian coordinates of all corners of the bounding box can then be expressed as:

$$\begin{aligned} \bar{\mathbf{y}}_{UL} &= (0, 0), & \bar{\mathbf{y}}_{UR} &= (c - 1, 0), \\ \bar{\mathbf{y}}_{LL} &= (0, r - 1), & \bar{\mathbf{y}}_{LR} &= (c - 1, r - 1). \end{aligned} \quad (19.2)$$

Let Ω_0 denote the bounding box of the $c_0 \times r_0$ pixel reference image, and let Ω_1 denote the bounding box of the $c_1 \times r_1$ pixel image 1. When image 1 is transformed by \mathbf{H} , it produces a new bounding box Ω'_1 relative to the coordinate system of the reference image. Ω'_1 can be determined by finding the Cartesian coordinates of the transformed corner points, using Equation (19.1). An important observation is that there is a possibility that Ω'_1 falls outside Ω_0 , at least partly. In fact, this is the very idea of image mosaics, we want to extend one image by pasting one or more images onto it, to produce a larger image. A practical procedure is to first determine the bounding box of the *union* of Ω_0 and Ω'_1 . This new bounding box is precisely large enough to include the pixel grid of both images, where image 1 first is transformed by \mathbf{H} .

Let the bounding boxes Ω_0 and Ω_1 be defined by their four corner points, see Figure 19.2. Using Equation (19.1), the corner points related to Ω'_1 can be expressed as:

$$\begin{aligned}\bar{\mathbf{y}}'_{1,UL} &= (u'_{1,UL}, v'_{1,UL}) = \text{cart}\left(\mathbf{H}\text{hom}(\bar{\mathbf{y}}_{1,UL})\right), & \bar{\mathbf{y}}'_{1,UR} &= (u'_{1,UR}, v'_{1,UR}) = \text{cart}\left(\mathbf{H}\text{hom}(\bar{\mathbf{y}}_{1,UR})\right), \\ \bar{\mathbf{y}}'_{1,LL} &= (u'_{1,LL}, v'_{1,LL}) = \text{cart}\left(\mathbf{H}\text{hom}(\bar{\mathbf{y}}_{1,LL})\right), & \bar{\mathbf{y}}'_{1,LR} &= (u'_{1,LR}, v'_{1,LR}) = \text{cart}\left(\mathbf{H}\text{hom}(\bar{\mathbf{y}}_{1,LR})\right).\end{aligned}\quad (19.3)$$

The bounding boxes Ω_0 and Ω'_1 now refer to the same coordinate system, and the two boxes are illustrated in Figure 19.3 (dashed) for the images related to our example in Figure 19.1. Notice that Ω'_1 is not a rectangle, but rather a general quadrilateral. Figure 19.3 also shows Ω'_0 , the bounding box of the union of the two boxes (solid). Ω'_0 is the smallest rectangle into which $\Omega_0 \cup \Omega'_1$ can be fitted, and will be the bounding box of the new reference image, after image 1 has been pasted onto the old reference image.

Assuming the prescribed coordinate system for image coordinates (u, v) , the coordinates of the upper left corner and of the lower right corner of Ω'_0 can be computed as:

$$\begin{aligned}\bar{\mathbf{y}}'_{0,UL} &= (u'_{0,UL}, v'_{0,UL}) = \text{round}\left(\min(0, u'_{1,UL}, u'_{1,UR}, u'_{1,LL}, u'_{1,LR}), \min(0, v'_{1,UL}, v'_{1,UR}, v'_{1,LL}, v'_{1,LR})\right), \\ \bar{\mathbf{y}}'_{0,LR} &= (u'_{0,LR}, v'_{0,LR}) = \text{round}\left(\max(c_0 - 1, u'_{1,LR}, u'_{1,UR}, u'_{1,LL}, u'_{1,UL}), \max(r_0 - 1, v'_{1,LR}, v'_{1,UR}, v'_{1,LL}, v'_{1,UL})\right).\end{aligned}\quad (19.4)$$

Some observations can be made based on these formulas. As a start, the new reference image has a pixel grid with $c'_0 = u'_{0,LR} - u'_{0,UL} + 1$ columns and $r'_0 = v'_{0,LR} - v'_{0,UL} + 1$ rows. In the normal case, Ω'_0 is a larger box than Ω_0 . We also see that even if the origin of the original reference image is located at the upper left corner of its bounding box, it may be located somewhere inside the new bounding box, see Figure 19.3. This means that there is an offset between the old and the new reference images, given by $\bar{\mathbf{y}}'_{0,UL}$. Notice that we use all four corners of Ω'_1 in the above formulas. This is necessary in order to deal with the possibility that the two images can be rotated relative to each other.

As final observation, since the coordinates are rounded to the nearest integer values, these formulas imply that the corners of the resulting bounding box always end up at integer values: the corresponding pixel grid is aligned with the old referenced image, except for an integer offset. This means that the expansion of the reference image requires the position of a corner of Ω'_1 to be at least half a pixel unit outside Ω_0 to achieve expansion. This strategy is simple to implement, and has a medium computational complexity, but also means that there is a possibility of losing a few pixels at the border of image 1 when it is being transformed. By changing the rounding function to a truncation upward or downward, it is possible to implement an alternative strategy that either entirely avoids the loss of border pixels, at a higher computational cost, or risks losing even more border pixels, at a lower cost. Choose wisely.

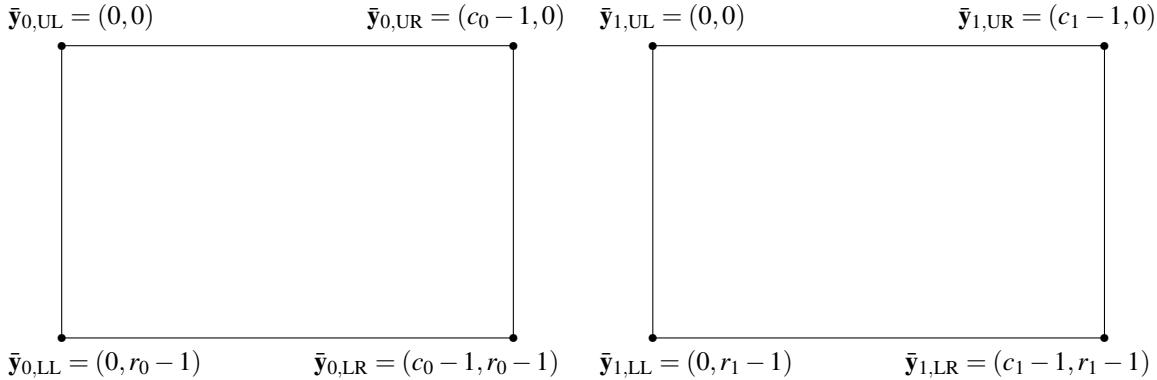


Figure 19.2: The bounding boxes of the $c_0 \times r_0$ reference image and of the $c_1 \times r_1$ image 1 (to be transformed), and the Cartesian coordinates of the corresponding corner points.

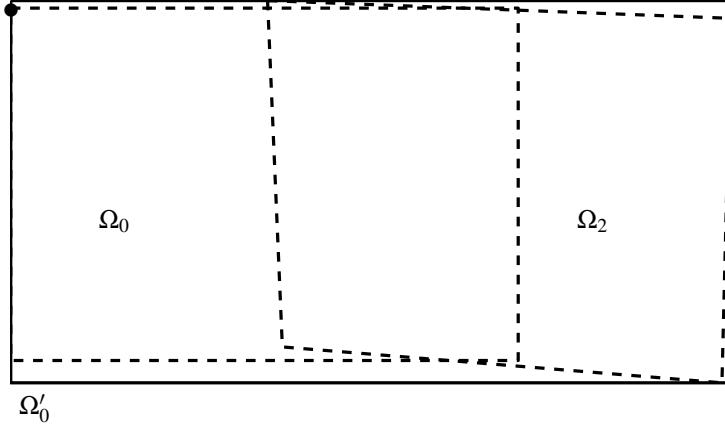


Figure 19.3: Dashed: the bounding box of the references image, Ω_0 and the transformed bounding box of image 1, Ω'_1 . Solid: the bounding box of the union of the two boxes, Ω'_0 , which forms the new reference image. The black dot indicates the origin of the coordinate system in the old reference image. In this example the old origin is slightly offset relative to the origin of the new reference image.

Four disjoint segments

Initially, we can think of the new bounding box Ω'_0 as an empty $c'_0 \times r'_0$ pixel grid, empty in the sense that the pixels are still undefined. In practice, an undefined pixel value can often be implemented by setting the pixel value to zero. Assuming that all pixels of the new pixel grid have an undefined value, the next step is to insert both the old reference image and the transformed version of image 1 into this new image. The result is an image mosaic composed of image 1 and the old reference image. This resulting reference image can be partitioned into four disjoint segments:

1. pixels that lie outside both Ω_0 and Ω'_1 ,
2. pixels that lie inside Ω_0 but outside Ω'_1 ,
3. pixels that lie inside Ω'_1 but outside Ω_0 , and
4. pixels that lie inside both Ω_0 and Ω'_1 .

See Figure 19.4 for an illustration. The segmentation of the new reference image can be done in a simple way by determining for each pixel position \bar{y}'_0 if it lies inside or outside Ω_0 or Ω'_1 , e.g., using Algorithm 3.1 on page 45. Each of the four segments is then treated in a unique way when it comes to assigning a value to $I'_0(\bar{y}'_0)$, the new reference image I'_0 at pixel position \bar{y}'_0 .

The first segment simply keeps its undefined values, and segments (2) and (3) should be assigned from the old reference image, and from image 1, respectively. The question is how segment (4) should be treated, since it can be assigned from either the old reference image, from image 1, or by combining the two. Initially, we use a simple and efficient strategy: assign segment (4) only from the old reference image. As a consequence, segments (2) and (4) are treated in the same way; their pixels are assigned only from the old reference image. In Section 19.2.2 we will consider an alternative method for segment (4), which also simplifies the management of the four segments for an efficient implementation of the computations discussed here.

Transfer of the old reference image and image 1

The union of segment (2) and segment (4) is precisely the old reference image Ω_0 , but now it lies somewhere inside the new reference image Ω'_0 , displaced by the integer shift $\bar{y}'_{0,UL}$. This makes the transfer of the old reference image into the new one straightforward: we just copy all pixel values, from the old reference image at position $\bar{y}'_0 - \bar{y}'_{0,UL}$, and insert them into new reference image, at position \bar{y}'_0 :

$$I'_0(\bar{y}'_0) = I_0(\bar{y}'_0 - \bar{y}'_{0,UL}), \quad \text{when } \bar{y}'_0 \text{ is in segment (2) or (4).} \quad (19.5)$$

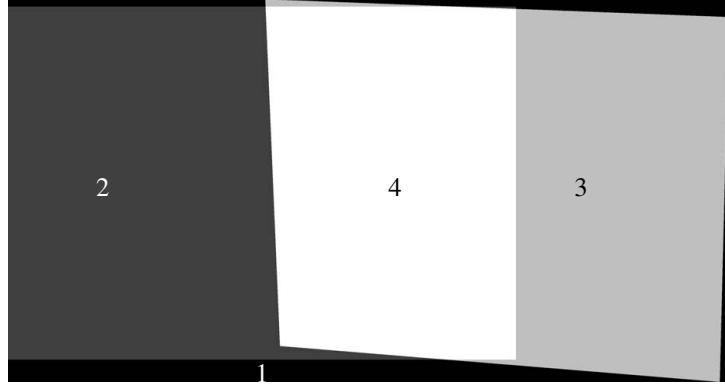


Figure 19.4: The four segments related to the example in the text. Black: segment (1) (neither Ω_0 nor Ω_2). Dark grey: segment (2) (only Ω_0). Bright grey: segment (3) (only Ω_2). White: segment (4) (intersection of Ω_0 and Ω_2).

It remains to describe what happens with segment (3), covered only by Ω'_1 . Taking the new position of the origin in the reference image into account, the mapping from image 1 to the new reference image is expressed as

$$\bar{y}'_0 = \text{cart}(\mathbf{H} \text{hom}(\bar{y}_1)) + \bar{y}'_{0,\text{UL}}. \quad (19.6)$$

One approach to transfer image 1 to the new reference image would be to scan over all pixels in the source image (image 1), and for each pixel transform its coordinates \bar{y}_1 to a point \bar{y}'_0 in the new reference image using Equation (19.6), and check if \bar{y}'_0 falls inside segment (3). If it does, the pixel value at \bar{y}_1 in image 1 should somehow affect the pixel value at \bar{y}'_0 in the new reference image. If \bar{y}'_0 is outside segment (3), we continue with the next pixel in image 1. This approach is often referred to as *forward interpolation*, and although it is conceptually simple, it has some issues. First, even if \bar{y}_1 is integer valued, \bar{y}'_0 is not in the general case. This implies that we need to figure out what it means to set the pixel at \bar{y}'_0 to some value even if this position is not at an integer valued coordinate. Second, if the solution to the first issue is solved in a too simplistic way, e.g., by rounding \bar{y}'_0 to the nearest integer valued coordinate, then there is a possibility that some pixels inside segment (3) are not assigned any value at all and they stay undefined.

As an alternative to forward interpolation, a better approach is instead to use *backward interpolation*. This implies that we scan all pixels of the grid in the target box Ω'_0 , and check if the pixel coordinate \bar{y}'_0 falls inside segment (3). If not, the pixel has already been covered by one of the other three types or segments. If \bar{y}'_0 is inside segment (3) we can compute the corresponding point \bar{y}_1 as:

$$\bar{y}_1 = \text{cart}\left(\mathbf{H}^{-1} \text{hom}(\bar{y}'_0 - \bar{y}'_{0,\text{UL}})\right). \quad (19.7)$$

For each pixel in segment (3), with Cartesian coordinate \bar{y}'_0 relative to the origin of the new reference image, Equation (19.7) maps \bar{y}'_0 to its corresponding position \bar{y}_1 in image 1. Since \bar{y}'_0 is inside segment (3) it is guaranteed that \bar{y}_1 is inside the bounding box of image 1 but, in general, \bar{y}_1 is not integer valued. This means that the pixel value in image 1 at position \bar{y}_1 somehow must be interpolated from the pixels that are nearby \bar{y}_1 . This interpolation can be done in different ways, with more or with less complex computations. A simple approach is offered by so-called *nearest neighbor interpolation*, where the pixel value at \bar{y}_1 is represented by the pixel value at the integer position $\text{round}(\bar{y}_1)$. This is a very efficient interpolation method, but can also produce artifacts in the resulting image. Other interpolation schemes can reduce such artifacts, but at a higher computational cost. Choose wisely.

Backward interpolation has an advantage over forward interpolation in that each pixel in the grid of the target box Ω'_0 will be addressed, i.e., none of its pixels stay undefined unless it falls into segment (1). Furthermore, by first computing \bar{y}_1 from \bar{y}'_0 , Equation (19.7), we can check if \bar{y}'_0 is inside Ω'_1 by checking if \bar{y}_1 is inside Ω_1 . The latter check is simple since Ω_1 is a rectangle aligned with the two coordinate axes. Backward interpolation is the recommended approach, and implies that the pixel values of the new reference image are assigned as

$$I'_0(\bar{y}'_0) = I_1\left(\text{round}\left(\text{cart}\left(\mathbf{H}^{-1} \text{hom}(\bar{y}'_0 - \bar{y}'_{0,\text{UL}})\right)\right)\right), \quad \text{when } \bar{y}'_0 \text{ is in segment (3)}. \quad (19.8)$$

Here, nearest neighbor interpolation is used.



Figure 19.5: A mosaic made of image 0 and image 1 in Figure 19.1, using the computational steps outlined in this section.

Summary

We have now presented all the steps of a simple method that pastes image 1 onto image 0, i.e., onto the reference image, and produce a new reference image as a result. Originally, the reference image is of size $c_0 \times r_0$, and after image 1 is pasted onto it, it has size $c'_0 \times r'_0$. In Section 19.2.3, we will generalize this procedure and think of image 1 just as any new image that we want to paste onto the reference image, thereby extending it to a new reference image. If we apply these steps to image 0 and image 1 in Figure 19.1, with image 0 as the reference image, the result is illustrated in Figure 19.5. In this image, black pixels at the border corresponds to undefined pixel values.

19.2.2 Blending

The result in Figure 19.5 is reasonable from a geometrical point of view. If you zoom in on the picture, there are no noticeable displacements of lines at the border of the left image (image 0) and right image (image 1) except at the roof. The latter displacements can be explained by the fact that the roof extends a bit from the wall of the building, so these lines do not lie in the plane that serves as a model for the homography transformation between the two images.

Even if the geometry is right, we also see that there is an edge in the intensity of the resulting image at the border between the two images. There are several factors behind this edge, but the main explanation is the fact that the two images are exposed independently of each other. In fact, they are taken with automatic settings of the exposure, which means that the exposure of each image is affected by the overall amount of light in the field of view at the image of exposure. Since this amount of light can be different for different camera poses, in particular for an outdoor scene a bright sunny day, it follows that the exposure can be different for the two images.

One way to reduce the intensity edge is to use *blending*. Instead of setting the pixel values in segment (4), which is covered by both images, by transferring them from only one of the two images, image 0 in our case, blending implies that each value is given as a weighted average of the pixel values from both images. Thus, in addition to the pixel value, each pixel in the two images must have a corresponding weight $w(\bar{y})$, given as a function of the pixel coordinate \bar{y} . The weight reflects how much the pixel will influence the resulting image, in relative terms. The weighting function can be defined in many different ways, but the basic idea is to let it have a large value (≈ 1) at the center of image, and a small value (≈ 0) at the border. An example of a simple weighting function, adapted to the 0-first enumeration scheme of an $c \times r$ image, is

$$w(u, v) = \begin{cases} \min\left(1 - \left|\frac{2u-c+1}{c}\right|, 1 - \left|\frac{2v-r+1}{r}\right|\right), & \text{when } (u, v) \text{ inside the bounding box,} \\ 0 & \text{otherwise.} \end{cases} \quad (19.9)$$

This function has the value $1/c$ or $1/r$ at the border pixels, instead of 0, which means that the values of these pixels will be used rather than discarded by the weighting. See Figure 19.6 for an illustration.

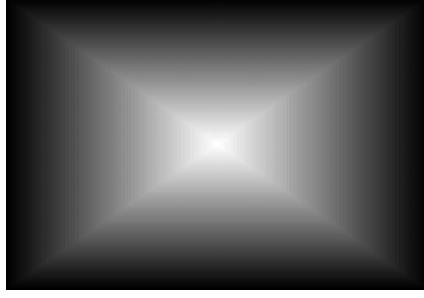


Figure 19.6: An example of a weighting function w that can be used for blending in image mosaics.



Figure 19.7: A mosaic made of image 0 and image 1 in Figure 19.1, using blending. Compare with Figure 19.5.

In our case, the two images can have different sizes, so there are two different weighting functions, one for each of the two images, w_0 and w_1 . Both weighting functions need to be transformed to the coordinate system of the new reference image I'_0 in order to be used for the weighting of the two images. Since they are well-defined for any $(u, v) \in \mathbb{R}^2$, however, they do not have to be interpolated. Pixels in segment (4) are now set as a weighted average of I_0 and I_1 :

$$I'_0(\bar{\mathbf{y}}'_0) = \frac{w_0(\bar{\mathbf{y}}_0) I_0(\bar{\mathbf{y}}_0) + w_1(\bar{\mathbf{y}}_1) I_1(\bar{\mathbf{y}}_1)}{w_0(\bar{\mathbf{y}}_0) + w_1(\bar{\mathbf{y}}_1)}, \quad \text{where} \quad (19.10)$$

$$\bar{\mathbf{y}}_0 = \bar{\mathbf{y}}'_0 - \bar{\mathbf{y}}'_{0,\text{UL}}, \quad \text{and} \quad \bar{\mathbf{y}}_1 = \text{cart}\left(\mathbf{H}^{-1} \text{hom}\left(\bar{\mathbf{y}}'_0 - \bar{\mathbf{y}}'_{0,\text{UL}}\right)\right). \quad (19.11)$$

The result of applying this simple blending is illustrated in Figure 19.7, which should be compared to Figure 19.5. Notice that not only has the intensity edge been removed, the displacement of the lines at the roof appears to have been removed as well. In fact, this displacement is still there, but since the weighting makes a smooth transition between the two images it is not as apparent as before.

Implementation

The introduction weighting functions w_0 and w_1 facilitates a simple implementation of the segments. Since w_0 and w_1 vanish outside the corresponding bounding box, Equation (19.11) produces the expected result not only in segment (3), but also in segments (2), where $w_0 = 0$, and in segment (4), where $w_1 = 0$. In fact, assuming that undefined pixels are assigned a zero value, it works also for segment (1) since the numerator vanishes there. In order for the fraction in Equation (19.11) to be well-defined in segment (1), however, we must replace the denominator with an expression that is non-zero (e.g., = 1) in segment (1) but otherwise is the same as before. In the case of color images, each pixel contains three intensity measurements, typically referred to as “red”, “green”,

and “blue”. The weighting that is used to implement the blending should then be applied to each of the three components independently.

The weighting that has been described so far includes only two images. In practice there can be some regions of the resulting mosaic that are covered by three or more of the images from which it is built. A consistent approach is to assign each pixel in the resulting mosaic a weighted average of all transformed images, where the weight is zero when the pixel is outside the corresponding transformed bounding box. This implies that both the numerator and the denominator in Equation (19.11) should be replaced by a sum over *all* images. When new images are pasted onto the reference image, the corresponding terms are then added, separately, to the numerator and the denominator. When all images have been processed in this way, as is described in Section 19.2.3, the fraction between the final numerator and denominator is computed at each pixel. In this setting, the reference image acts as an accumulator that collects weighted contributions from each image, corresponding to the numerator in Equation (19.11). In addition, we also need an image S of the same size as the reference image I_0 that accumulates the denominator in Equation (19.11).

The computational steps that have been discussed here are summarized in Algorithm 19.1. It uses backward nearest-neighbor-interpolation of image 1 when it is transferred to the new reference image, and blending. This implementation checks if $\bar{\mathbf{y}}_1$ is inside Ω_1 instead of checking if $\bar{\mathbf{y}}'_0$ is inside Ω'_1 , i.e., Algorithm 3.1 is not needed here.

19.2.3 A mosaic from several images

Consider a set of n images $I_k, k = 1, \dots, n$, where image I_k has size $c_k \times r_k$, and all images are related by homographies. Based on the results discussed in the previous section, we can now describe an incremental approach for pasting these images together to produce a mosaic that is built from all these images.

First, we need to select one of the images as the original reference image, defining the coordinate system of the resulting mosaic. In principle, this image can be chosen arbitrarily from the n images, but since all other images are to be transformed to its coordinate system, the choice should be made with some care to avoid large geometric distortions caused by the homography transformations. In the case of cameras observing a plane, it is recommended to use an image where the image plane is as parallel as possible to the 3D plane depicted in the images. In the case of rotating cameras, it is difficult to avoid large geometric distortions, in particular if the images together cover a field of view that is larger than 180° . If the field of view is less than this, choose an image with an optical axis near the center of the total field of view. This distortion issue will be addressed later in Section 19.3.

Let the original reference image be denoted as I_0 . Next, we choose an image from the set and paste it onto I_0 , using the procedure discussed in Section 19.2.1. The image pasted onto I_0 can be chosen arbitrarily, but it must have sufficiently large overlap with I_0 to allow the estimation of the homography \mathbf{H} . When this is done, the result is a new reference image I'_0 of size $c'_0 \times r'_0$. By setting $I_0 = I'_0$ and choosing a new image in the set to paste onto the new I_0 , we can iteratively build a larger and larger reference image. When all images have been pasted together in this way, the final reference image constitutes the resulting mosaic.

Based on this discussion, we assume that the images have been enumerated such that I_1 can be chosen as the original reference image, I_2 can be pasted onto I_1 and, in general, image $I_k, k = 3, \dots, n$, can be pasted to the reference image that results from pasting image I_{k-1} to the current reference image. With these notations, this iterative approach to building a mosaic is presented in Algorithm 19.2.

19.3 Using spherical coordinates for panoramas

If we want to build a mosaic from images of rotating cameras, i.e., from a camera that is rotating about its center between two images, the previous approach based on homography transformations can produce a mosaic, but it may look very distorted. This happens already if the total field of view of the images is larger than approximately 90° and the distortion becomes more and more distracting as the field of view approaches 180° . Beyond that, there will even have to be singularities where some image points are mapped to points at infinity in the resulting mosaic. To avoid the practical management of such cases, as well as reducing the geometric distortion caused by the homography transformations in general, an alternative approach must be used.

One alternative approach is offered by mapping the images to spherical coordinates. This approach is well suited for the case of equivalent cameras, where the result often is referred to as a panorama, but cannot be used for images of planes. There are also other possibilities for the panorama case, such as using cylindrical coordinates,

Algorithm 19.1: Extending a reference image with a single image.

```

Input: An accumulated reference image  $I_0$  of size  $c_0 \times r_0$ .
Input: A  $c_0 \times r_0$  image  $S$  that accumulates the sum of the weights.
Input: An image  $I_1$  of size  $c_1 \times r_1$ , to be pasted onto  $I_0$ .
Output: A new accumulated reference image  $I'_0$  of size  $c'_0 \times r'_0$ .
Output: A new  $c'_0 \times r'_0$  image  $S'$  that contains the accumulated sum of the weights.

1 /* Estimate the homography  $\mathbf{H}$  */
2 In the manual case: use Algorithm 13.3 on page 238. In the automatic case: use Algorithm 17.3 on page
   307
3 /* Find  $\Omega'_1$  using Equations (19.2) and (19.3) */
4 Set  $(u'_{1,\text{UL}}, v'_{1,\text{UL}}) = \text{cart}(\mathbf{H}\text{hom}(0,0))$  and  $(u'_{1,\text{UR}}, v'_{1,\text{UR}}) = \text{cart}(\mathbf{H}\text{hom}(c_1-1,0))$ 
5 Set  $(u'_{1,\text{LL}}, v'_{1,\text{LL}}) = \text{cart}(\mathbf{H}\text{hom}(0,r_1-1))$  and  $(u'_{1,\text{LR}}, v'_{1,\text{LR}}) = \text{cart}(\mathbf{H}\text{hom}(c_1-1,r_1-1))$ 
6 /* Find  $\Omega'_0$  using Equation (19.4) and allocate new reference image */
7  $(u'_{0,\text{UL}}, v'_{0,\text{UL}}) = \text{round}(\min(0, u'_{1,\text{UL}}, u'_{1,\text{UR}}, u'_{1,\text{LL}}, u'_{1,\text{LR}}), \min(0, v'_{1,\text{UL}}, v'_{1,\text{UR}}, v'_{1,\text{LL}}, v'_{1,\text{LR}}))$ 
8  $(u'_{0,\text{LR}}, v'_{0,\text{LR}}) = \text{round}(\max(c_0-1, u'_{1,\text{LR}}, u'_{1,\text{LL}}, u'_{1,\text{UR}}, u'_{1,\text{UR}}), \max(r_0-1, v'_{1,\text{LR}}, v'_{1,\text{LL}}, v'_{1,\text{UR}}, v'_{1,\text{UR}}))$ 
9 Set  $c'_0 = u'_{0,\text{LR}} - u'_{0,\text{UL}} + 1$  and  $r'_0 = v'_{0,\text{LR}} - v'_{0,\text{UL}} + 1$ 
10 Allocate a new  $c'_0 \times r'_0$  reference image  $I'_0$  and a new  $c'_0 \times r'_0$  image  $S'$ 
11 /* Accumulate contributions from  $I_1$  to  $I'_0$  and  $S'$  */
12 foreach  $u'_0 = 0 \dots c'_0 - 1$  do
13   foreach  $v'_0 = 0 \dots r'_0 - 1$  do
14     Set  $\bar{\mathbf{y}}'_0 = (u'_0, v'_0)$ ,  $\bar{\mathbf{y}}_0 = (u_0, v_0) = \bar{\mathbf{y}}'_0 - \bar{\mathbf{y}}'_{0,\text{UL}}$ ,  $\bar{\mathbf{y}}_1 = (u_1, v_1) = \text{cart}(\mathbf{H}^{-1}\text{hom}(\bar{\mathbf{y}}'_0 - \bar{\mathbf{y}}'_{0,\text{UL}}))$ 
15      $w = \min(1 - |2u_1 - c_1 + 1|/c_1, 1 - |2v_1 - r_1 + 1|/r_1)$  // Equation (19.9)
16     if  $0 \leq u_0 < c_0$  and  $0 \leq v_0 < r_0$  then
17       if  $0 \leq u_1 < c_1$  and  $0 \leq v_1 < r_1$  then
18         Set  $I'_0(\bar{\mathbf{y}}'_0) = I_0(\bar{\mathbf{y}}_0) + w I_1(\text{round}(\bar{\mathbf{y}}_1))$  and  $S'(\bar{\mathbf{y}}'_0) = S(\bar{\mathbf{y}}_0) + w$  // Segment (4)
19       else
20         Set  $I'_0(\bar{\mathbf{y}}'_0) = I_0(\bar{\mathbf{y}}_0)$  and  $S'(\bar{\mathbf{y}}'_0) = S(\bar{\mathbf{y}}_0)$  // Segment (2)
21       end
22     else
23       if  $0 \leq u_1 < c_1$  and  $0 \leq v_1 < r_1$  then
24         Set  $I'_0(\bar{\mathbf{y}}'_0) = w I_1(\text{round}(\bar{\mathbf{y}}_1))$  and  $S'(\bar{\mathbf{y}}'_0) = w$  // Segment (3)
25       else
26         Set  $I'_0(\bar{\mathbf{y}}'_0) = 0$  and  $S'(\bar{\mathbf{y}}'_0) = 0$  // Segment (1)
27       end
28     end
29   end
30 end

```

Algorithm 19.2: Building a mosaic from a set of images that are related by homographies.

Input: A set of n images $\{I_k\}$, related by homographies. I_1 becomes first reference image, and I_k can be pasted onto the reference image that is produced when I_{k-1} has been pasted.

Output: A mosaic I_0 of the n images.

- 1 Set $I_0(\bar{\mathbf{y}}) = w(\bar{\mathbf{y}}) \cdot I_1(\bar{\mathbf{y}})$ and $S(\bar{\mathbf{y}}) = w(\bar{\mathbf{y}})$ for all pixels $\bar{\mathbf{y}}$ in I_1 // Initial values for I_0 and S
- 2 **foreach** $k = 2, \dots, n$ **do**
- 3 Paste image I_k onto reference image I_0 using Algorithm 19.1 on page 341
- 4 // This produces a new reference image I'_0 and sum of weights S'
- 5 Set $I_0 = I'_0$ and $S = S'$
- 6 **end**
- 7 Set $I_0(\bar{\mathbf{y}}) = I_0(\bar{\mathbf{y}})/S(\bar{\mathbf{y}})$ for all pixels $\bar{\mathbf{y}}$ in I_0 where $S(\bar{\mathbf{y}}) > 0$

but these are not discussed here. As will be apparent shortly, the mapping to spherical coordinates requires the internal parameters in \mathbf{K} to be known.

The basic idea of using spherical coordinates is to map each pixel, a point in the image plane, to a point on the unit sphere centered on the camera center. Since all cameras in this case have a common center \mathbf{n} , this mapping is the same for all images from a set of rotating cameras. Each point in the image plane, with pixel coordinates \mathbf{y} , corresponds to a projection line \mathbf{L}_y , which passes through the camera center and intersects the image plane at \mathbf{y} , see Section 8.5.2. The line \mathbf{L}_y intersects also the unit sphere at some 3D point $\hat{\mathbf{x}}$. For simplicity, we can use the virtual image plane, in front of the camera center, for this discussion. See Figure 19.8 for an illustration.

To use this representation, we must first describe how an image point, with pixel coordinates \mathbf{y}_p , is mapped to the 3D point $\hat{\mathbf{x}}$. The 3D point \mathbf{y}_p in the image plane corresponds to a 3D point that has coordinates

$$\bar{\mathbf{x}} = \begin{pmatrix} u_n \\ v_n \\ 1 \end{pmatrix} \sim \mathbf{y}_n \quad (19.12)$$

relative to the camera centered coordinate system. Here, (u_n, v_n) are the C-normalized image coordinates corresponding to pixel coordinates $\bar{\mathbf{y}}_p = (u_p, v_p)$. The transformation from pixel coordinates to C-normalized coordinates is given from Equations (8.34) and (8.36) as $\mathbf{y}_n \sim \mathbf{K}^{-1}\mathbf{y}_p$, where \mathbf{K} represents the internal parameters of the camera, as defined in Section 8.3.3. Hence, Equation (19.12) leads to

$$\bar{\mathbf{x}} = \begin{pmatrix} u_n \\ v_n \\ 1 \end{pmatrix} \sim \mathbf{K}^{-1}\mathbf{y}_p. \quad (19.13)$$

Finally, $\hat{\mathbf{x}}$ is given as by normalizing $\bar{\mathbf{x}}$ to a unit vector. This normalization implies that any scale factor between the two sides of the previous expression for $\bar{\mathbf{x}}$ cancels, and we can write:

$$\hat{\mathbf{x}} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \frac{\mathbf{K}^{-1}\mathbf{y}_p}{\|\mathbf{K}^{-1}\mathbf{y}_p\|}. \quad (19.14)$$

Equation (19.14) defines a mapping from pixel coordinates $\bar{\mathbf{y}} = (u_p, v_p)$ to a 3D point $\hat{\mathbf{x}}$ on the unit sphere, a mapping that is dependent on the internal parameters in \mathbf{K} . By assuring a fixed focal length for the different images, \mathbf{K} can in practice be assumed as fixed for all images/cameras.

Mapping pixel coordinates to spherical coordinates

The next step is to map $\hat{\mathbf{x}}$ to angular spherical coordinates, as step which assumes a specified 3D coordinate system. This is done by designating one of the images as a reference image, and use the corresponding camera centered coordinate system for the mapping to spherical coordinates. We recall from Section 8.2 that the camera centered coordinate system is right-handed, see Figure 8.3, where the axes for coordinates x_1 and x_3 define a horizontal plane, and the axis for x_2 points down. A point $\hat{\mathbf{x}}$ on the unit sphere then has spherical coordinates in the form

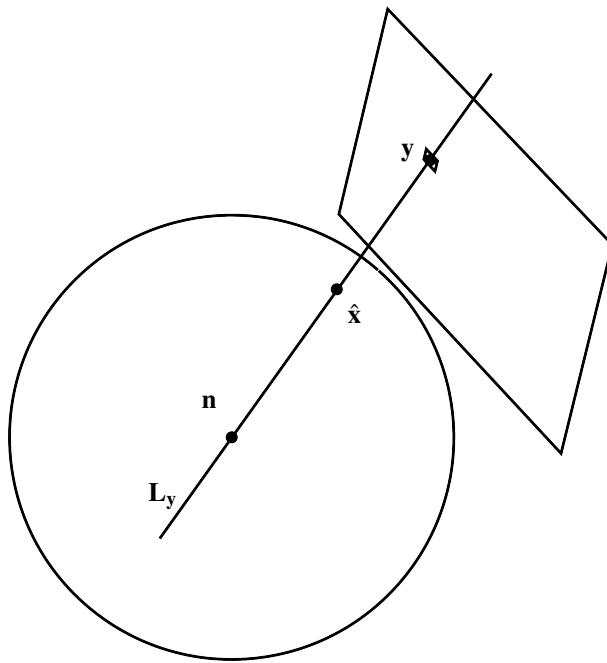


Figure 19.8: Image point \mathbf{y} is mapped to a 3D point $\hat{\mathbf{x}}$ on the unit sphere using the projection line \mathbf{L}_y . The sphere is centered on the camera center \mathbf{n} .

of an azimuthal (horizontal) angle $\theta \in [-\pi, \pi]$ and an elevation (vertical) angle $\phi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ in accordance with Figure 19.9. The relation between Cartesian 3D coordinates of $\hat{\mathbf{x}}$ and its spherical coordinates is expressed in terms

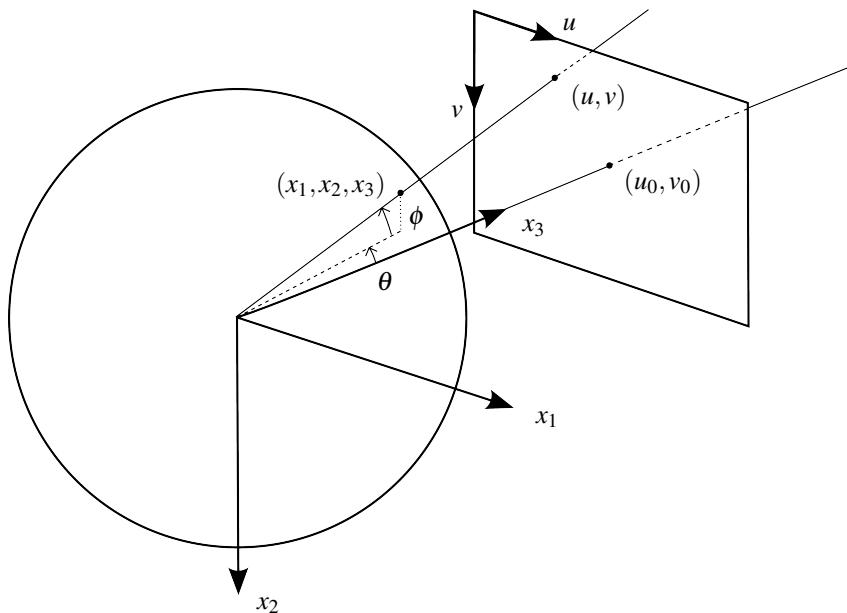


Figure 19.9: Spherical coordinates (θ, ϕ) defined relative to the camera centered coordinates (x_1, x_2, x_3) of unit vector $\hat{\mathbf{x}}$.

of the function $f: \mathbb{R}^2 \rightarrow \mathbb{R}^3$, defined as

$$\hat{\mathbf{x}} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = f(\theta, \phi) = \begin{pmatrix} \cos \phi \sin \theta \\ -\sin \phi \\ \cos \phi \cos \theta \end{pmatrix}. \quad (19.15)$$

This allows us to express the following relations between 3D coordinates and spherical coordinates:

$$\tan \theta = \frac{x_1}{x_3}, \quad \tan \phi = \frac{-x_2}{\sqrt{x_1^2 + x_3^2}}. \quad (19.16)$$

Based on this result, the inverse of f , a function that maps the unit vector $\hat{\mathbf{x}} = (x_1, x_2, x_3)$ back to spherical coordinates (θ, ϕ) , can be formulated as:

$$\begin{pmatrix} \theta \\ \phi \end{pmatrix} = f^{-1}(\hat{\mathbf{x}}) = f^{-1}(x_1, x_2, x_3) = \begin{pmatrix} \text{atan2}(x_1, x_3) \\ -\text{atan2}(x_2, \sqrt{x_1^2 + x_3^2}) \end{pmatrix}. \quad (19.17)$$

Spherical coordinates have a singularity for $\phi = \pm\frac{\pi}{2}$, when θ is undefined. These singularities correspond to $(x_1, x_3) = (0, 0)$, and happen if we need to represent pixels that lie in a direction straight up or down from the center. Although it cannot happen for the pixels in the image plane discussed here, it can happen for pixels in other image that are transformed to the spherical coordinates (θ, ϕ) . Thus, f^{-1} is well-defined only if $\hat{\mathbf{x}} \neq \pm(1, 0, 0)$.

In summary, we can express a mapping from (θ, ϕ) to pixel coordinates (u_p, v_p) as:

$$\begin{pmatrix} u_p \\ v_p \end{pmatrix} = \text{cart}(\mathbf{K}\hat{\mathbf{x}}) = \text{cart}(\mathbf{K}f(\theta, \phi)), \quad (19.18)$$

where f is defined in Equation (19.15). The inverse mapping, from pixel coordinates (u_p, v_p) to spherical coordinates (θ, ϕ) , is given as

$$\begin{pmatrix} \theta \\ \phi \end{pmatrix} = f^{-1}(\mathbf{K}^{-1}\mathbf{y}_p), \quad (19.19)$$

where f^{-1} is defined in Equation (19.17). Notice that even if $\mathbf{K}^{-1}\mathbf{y}_p$ in general is not a unit vector, the formulation of f^{-1} takes care of the scaling.

The advantage of using the mapping f for panoramas is that any pixel in any image plane of a set of rotating cameras corresponds to a spherical coordinate (ϕ, θ) within a bounding box, Ω_0 , of finite size. Figure 19.10 shows two images taken by a camera that has been rotated about its center. In Figure 19.11, left, the left image in Figure 19.10 has been transformed to a spherical coordinate system relative to the corresponding camera centered coordinate system. The bounding box Ω_0 of the resulting image is defined by $[-\pi, \pi]$ for the azimuth angle θ and $[-\frac{\pi}{2}, \frac{\pi}{2}]$ for the elevation angle, i.e., the origin of the spherical coordinates is at the center of the bounding box.



Figure 19.10: Two images, produced by rotating a camera about its center.

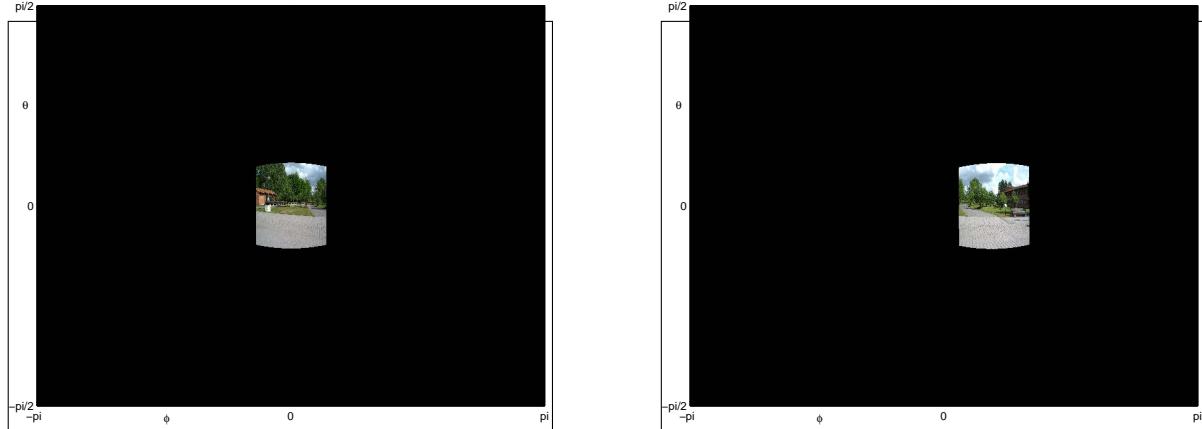


Figure 19.11: The two images in Figure 19.10 transformed to the spherical coordinates (θ, ϕ) .

19.3.1 A panorama from two images

We will now consider the problem of producing a panorama from two images using the representation based on spherical coordinates. The idea is similar to the approach based on homographies, described in Section 19.2; one image is designated as the reference image, but instead of transforming all other images to the coordinate system of the reference image, they are transformed to the spherical coordinates defined by the reference image. Another difference is that, since Ω_0 is of finite size, it is not necessary to incrementally increase Ω_0 . It can simply be chosen from the maximal ranges of θ and ϕ . In practice, this may be too large for some panoramas but it can easily be cropped to a suitable size after the panorama has been built.

A third difference to the previous approach is the resolution of the final panorama image. In Section 19.2, the pixel resolution of the final mosaic is identical to the one in the first reference image; there can only be a translation between the coordinate systems of the first and the last reference images. In the case of panoramas based on spherical coordinates, the pixel resolution of the resulting panorama is determined by how densely we want to sample the two angular coordinates. The angular resolution should, roughly, reflect the pixel resolution of the images from which the panorama is built. If it is too high, the computational cost of building the panorama increases without providing more details in the panorama image, and if it is too low, the panorama will loose details. A reasonable approach is to first build a low-resolution panorama, at a low computational cost, just to get an idea of what the result looks like, and in particular to see what ranges to use for θ and ϕ . Then, a final high-resolution panorama is built with customized ranges of the angular coordinates.

Transformation between the images

Given that one image, I_0 , has been designated as the reference image, Equations (19.19) and (19.18) express the mappings between pixel coordinates and spherical coordinates for this image. What about the other images, how are they mapped to the same spherical coordinates? Let I_1 be an image that has an overlap with I_0 . Since I_0 and I_1 are produced by rotating cameras, corresponding image coordinates are related by a homography transformation. For this application, however, the interesting relation between the two images is rather that their camera centered coordinate systems are related by a rotation \mathbf{R} . A pixel (u_p, v_p) in image I_1 is mapped by Equation (19.14) to 3D coordinates of the unit vector $\hat{\mathbf{x}}'$, relative to the camera centered coordinate system of camera 1. To express $\hat{\mathbf{x}}'$ relative to the camera centered coordinate system of camera 0, the coordinates of $\hat{\mathbf{x}}$ must be rotated by \mathbf{R} :

$$\hat{\mathbf{x}} = \mathbf{R} \hat{\mathbf{x}}' \quad (19.20)$$

Here, $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ are the coordinates of the same 3D point, expressed in the coordinate system of (the reference) camera 0 and of camera 1, respectively.

As a consequence of this observation, we can reformulate the mappings between pixel coordinates and spherical

coordinates, in Equations (19.18) and (19.19), as

$$\begin{pmatrix} u_p \\ v_p \end{pmatrix} = \text{cart}(\mathbf{K}\mathbf{R}\hat{\mathbf{x}}) = \text{cart}(\mathbf{K}\mathbf{R}f(\theta, \phi)), \quad \begin{pmatrix} \theta \\ \phi \end{pmatrix} = f^{-1}(\mathbf{R}^\top \mathbf{K}^{-1} \mathbf{y}_p), \quad (19.21)$$

where \mathbf{R} is the rotation between the camera centered coordinate systems related to I_0 and I_1 , respectively. $\mathbf{R} = \mathbf{I}$ for the reference image.

Finding \mathbf{R}

The rotation \mathbf{R} that relates one camera to the reference camera, can, in principle, be found by first determining the homography \mathbf{H} between the two images and then solve \mathbf{R} from Equation (9.25) when the internal parameters are known. Since the homography has to be estimated from noisy data, however, a simple solution of this type typically leads to a matrix \mathbf{R} that, at best, only is an approximation of a rotation. A better option is to determine \mathbf{R} directly from a set of corresponding points in the two images.

A consequence of Equations (19.14) and (19.20), is that a set of m corresponding image points, with pixel coordinates $\bar{\mathbf{y}}_{0,i}$ and $\bar{\mathbf{y}}_{1,i}$ in image 0 and image 1, respectively, are related by

$$\frac{\mathbf{K}^{-1} \mathbf{y}_{0,i}}{\|\mathbf{K}^{-1} \mathbf{y}_{0,i}\|} = \mathbf{R} \frac{\mathbf{K}^{-1} \mathbf{y}_{1,i}}{\|\mathbf{K}^{-1} \mathbf{y}_{1,i}\|}, \quad i = 1, \dots, m. \quad (19.22)$$

Hence, \mathbf{R} can be estimated directly from this set of correspondences using any of the methods described in Section 15.1, e.g., OPP. Once \mathbf{R} is determined for image I_1 , the image can be transformed to the spherical coordinates (θ, ϕ) using Equation (19.21). If we apply this strategy to the right image in Figure 19.10, the result is shown in Figure 19.11, right. After the transformation, the two images are merged in a similar way as for the homography case in Section 19.2, i.e., by segmenting the resulting image into 4 segments. Each segment is then assigned in different ways depending on which of the two transformed images that covers the segment.

Blending

One of the four segments, segment (4), is covered by both of the transformed images, and Section 19.2.2 describes a simple technique for blending the two images in order to reduce edge artifacts in the resulting mosaic. These artifacts appear also for panoramas, and it is therefore reasonable to use blending also in this case. The two images in Figure 19.10 have been blended in this way, after mapping to the spherical coordinates defined by the camera centered coordinate system of the left image, and the result is presented in Figure 19.12. The figure has also been cropped relative to the maximal bounding box Ω_0 in Figure 19.11.

Summary

The different computational steps that are described here for merging two images to a panorama based on spherical coordinates are summarized in Algorithms 19.3 and 19.4. They are simplified by assuming that all images are taken by same camera, with a fixed set of internal parameters, represented by \mathbf{K} .

19.4 Further reading

There is an extensive body of literature on image mosaics and panorama formation. A popular approach is found in the paper by Brown and Lowe [9].



Figure 19.12: A panorama mosaic of the two images in Figure 19.10. It has been cropped relative to the maximal range of the two spherical coordinates (θ, ϕ) , and it uses the blending technique described in Section 19.2.2.

Algorithm 19.3: Adding an image to the reference image. Based on spherical coordinates.

<p>Input: A reference image I_0 of size $c \times r$. Input: A range for $\theta = -\theta_1 \dots \theta_2$, with angular step $\Delta\theta$. Input: A range for $\phi = \phi_1 \dots \phi_2$, with angular step $\Delta\phi$. Input: An image I_1 of size $c_1 \times r_1$, to be pasted onto I_0. Input: Internal camera parameters \mathbf{K}. Output: An updated reference image I_0 of size $c \times r$. Output: An updated image S of size $c \times r$.</p>
--

```

1 /* Estimate the rotation  $\mathbf{R}$  */  

2 Use (for example) Algorithm 15.1 on page 261, based on Equation (19.22)  

3 /* Accumulate contributions from  $I_1$  to  $I'_0$  and  $S$  */  

4 foreach  $u = 0, \dots, (c-1)$  do  

5   foreach  $v = 0 \dots (r-1)$  do  

6     Set  $(\theta, \phi) = (\theta_1 + u \Delta\theta, \phi_1 + v \Delta\phi)$   

7     Set  $(u_1, v_1) = \text{cart}(\mathbf{K} \mathbf{R} f(\theta, \phi))$  // Equation (19.21)  

8     if  $0 \leq u_1 < c_1$  and  $0 \leq v_1 < r_1$  then  

9        $w = \min(1 - |2u_1 - c_1 + 1| / c_1, 1 - |2v_1 - r_1 + 1| / r_1)$  // Equation (19.9)  

10       $I'_0(u, v) = I_0(u, v) + w I_1(\text{round}(u_1, v_1))$  // nearest neighbor interpolation of  $I_1$   

11       $S(u, v) = S(u, v) + w$   

12    end  

13  end  

14 end
```

Algorithm 19.4: Building a panorama from a set of images. Based on spherical coordinates.

Input: A set of n images $\{I_k\}$, related by homographies. It is the camera centered coordinate system of I_1 that defines the spherical coordinates, and I_k can be pasted onto the reference image that is produced when I_{k-1} has been pasted.

Input: A range for $\theta = -\theta_1 \dots \theta_2$, with angular step $\Delta\theta$.

Input: A range for $\phi = \phi_1 \dots \phi_2$, with angular step $\Delta\phi$.

Input: Internal camera parameters \mathbf{K} .

Output: A mosaic I_0 of the n images.

- 1 Set I_0 as a $((\theta_2 - \theta_1)/\Delta\theta + 1) \times ((\phi_2 - \phi_1)/\Delta\phi + 1)$ image with zero valued pixels // Initial values for I_0
- 2 Set S as an image of the same size and values as I_0 // Initial values for S
- 3 **foreach** $k = 1, \dots, n$ **do**
- 4 Paste image I_k onto reference image I_0 using Algorithm 19.3 on page 347
- 5 // This produces an updated reference image I_0 and updated sum of weights S
- 6 **end**
- 7 Set $I_0(\bar{\mathbf{y}}) = I_0(\bar{\mathbf{y}})/S(\bar{\mathbf{y}})$ for all pixels $\bar{\mathbf{y}}$ in I_0 where $S(\bar{\mathbf{y}}) > 0$

Chapter 20

Rectified Stereo

Before you read this chapter, you need to have a good understanding of the material on epipolar geometry, which is presented in Chapter 10. You will also need the stuff on rotational homographies presented in Section 9.3.

In Chapter 10, the problem of triangulation was introduced and several solution strategies were presented in Section 16.1. Triangulation can be done for general configurations of the two cameras, but in this chapter, we will see that it becomes particularly simple when the camera poses make both optical axes parallel and perpendicular to the baseline. In this case, the epipolar lines in both images are parallel. If the two cameras are positioned side by side, all epipolar lines become horizontal. If the two cameras also have identical internal calibration, corresponding points always lie on the same vertical position in the two images. In many applications these conditions imply a significant simplification for the correspondence problem: corresponding points lie on the same row in both images, which means that the search for corresponding points can be implemented very efficiently.

The epipolar geometry corresponding to this case is called *rectified stereo*, and in this chapter we will discuss geometrical and practical consequences of rectified stereo in Section 20.1. With a few additional assumptions, we will see that each 3D point is projected into the two images such that its vertical coordinate is the same in both images. Furthermore, the horizontal coordinate is the same except for an offset, or displacement, called the *disparity*. Furthermore, for each 3D point, its disparity is inversely proportional to its *depth*, its coordinate along the optical axes of the two cameras. Consequently, if we can determine the disparity of an image point, its horizontal offset between the two images, it is trivial to determine the position of the corresponding 3D point. Thus, the triangulation problem becomes simpler than for the case of general epipolar geometry.

Stereo cameras that are arranged in a way that enables rectified stereo are referred to as *rectified stereo cameras*, and such a construction can be built in practice as a *rectified stereo rig*. But doing so requires a high degree of precision. Although this mechanical calibration can be accomplished to a reasonable degree of accuracy, it is costly, it takes time, and it may not be stable over longer time. Fortunately, there is an alternative.

The mechanical calibration described above boils down to rotating the two cameras about their centers to make them rectified. In Section 9.3, we observed that rotation of a camera about its center corresponds to a homography transformation of the image. This observation leads to a simpler strategy for rectified stereo. Instead of mechanical calibration, we transform each of the two images from general stereo cameras by a pair of suitable homographies, one in each image, such that the resulting images appear as if the stereo rig was rectified. These *rectifying homographies* are not unique, but can be determined as soon as the epipolar geometry of the stereo rig is known, in terms of the fundamental matrix. This approach is referred to as *synthetic stereo rectification*, and is discussed in Section 20.3.

20.1 Rectified stereo cameras

Let us first define what we mean by rectified stereo cameras. The concept is based on several assumptions about the relative pose of the two cameras, as well as the image coordinate systems. We will start by defining a weaker

form of rectification that is based only on the relative pose of the stereo cameras. Already this form of rectification leads to several important results and observations. Once this case has been investigated, we will move on to the case of full rectification that also includes conditions on the image coordinate systems.

20.1.1 Preliminary results

In Section 10.1, stereo cameras are defined as a general concept that produces two images of the same scene but from two distinct view-points. Stereo cameras can be implemented as a single unit that consists of two fixed cameras, a stereo rig. Alternatively, we can use a single camera that takes two images, moving between the two view-points, hence referred to as motion stereo. The poses of the stereo cameras can be general, except that the camera centers must be distinct.

Rectified stereo cameras are a special case of stereo cameras, specified in a weaker form as follows:

Definition 20.1: Weakly rectified stereo cameras

A pair of stereo cameras are *weakly rectified* if both optical axes are parallel and also perpendicular to the baseline, i.e., to the line segment between the camera centers.

In most practical applications, the displacement between the cameras is made in the horizontal direction. Since this horizontal displacement of the camera centers is a very common implementation of stereo cameras, it makes sense to label the two cameras as left (L) and right (R). See Figure 20.1 for an illustration. This configuration also resembles the stereo vision used by our own eyes.

To simplify, we also assume that the world coordinate system of the 3D space equals the camera centered coordinate system of the left camera. The two camera matrices are then given as:

$$\mathbf{C}_L = \mathbf{K}_L (\mathbf{I} | \mathbf{0}), \quad \text{where } \mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \bar{\mathbf{t}} = \begin{pmatrix} t_1 \\ t_2 \\ 0 \end{pmatrix}. \quad (20.1)$$

Note that \mathbf{R} is a rotation about the optical axis of the right camera. Further, the third element of $\bar{\mathbf{t}}$ is zero, since the

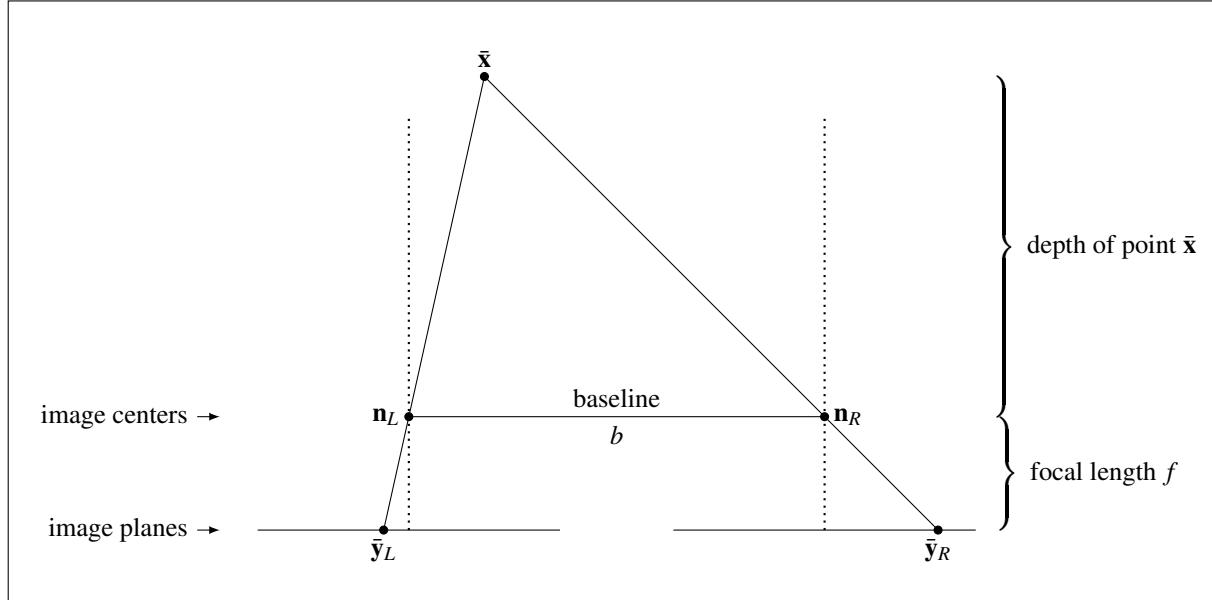


Figure 20.1: Rectified stereo cameras seen from above. The two cameras are placed side by side, and therefore designated as left (L) and right (R). The two optical axes (dotted lines) are parallel and perpendicular to the baseline. The 3D point \bar{x} is projected to \bar{y}_L in the left image, and to \bar{y}_R in the right image.

displacement of the camera centers is perpendicular to the optical axes, i.e., to the third dimension of the coordinate system. The centers of these cameras are:

$$\mathbf{n}_L = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}, \quad \mathbf{n}_R = \begin{pmatrix} -\mathbf{R}^\top \bar{\mathbf{t}} \\ 1 \end{pmatrix}. \quad (20.2)$$

Using the fact that both \mathbf{K}_L and \mathbf{K}_R are upper triangular, and that the third element of $\bar{\mathbf{t}}$ is zero, the corresponding epipoles must lie at infinity:

$$\begin{aligned} \mathbf{e}_{LR} &= \mathbf{C}_L \mathbf{n}_R = \mathbf{K}_L (\mathbf{I} | \mathbf{0}) \begin{pmatrix} -\mathbf{R}^\top \bar{\mathbf{t}} \\ 1 \end{pmatrix} = -\mathbf{K}_L \mathbf{R} \bar{\mathbf{t}} = \begin{pmatrix} u_{LR} \\ v_{LR} \\ 0 \end{pmatrix}, \\ \mathbf{e}_{RL} &= \mathbf{C}_R \mathbf{n}_L = \mathbf{K}_R (\mathbf{R} | \bar{\mathbf{t}}) \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} = \mathbf{K}_R \bar{\mathbf{t}} = \begin{pmatrix} u_{RL} \\ v_{RL} \\ 0 \end{pmatrix}. \end{aligned} \quad (20.3)$$

In fact, we can use this result as an alternative definition of what weakly rectified stereo cameras mean.

20.1 The epipoles of a weakly rectified stereo camera pair both lie at infinity.

Let \mathbf{x} be a 3D point, and project it into each of the two stereo images:

$$\mathbf{y}_L = \mathbf{C}_L \mathbf{x}, \quad \mathbf{y}_R = \mathbf{C}_R \mathbf{x}. \quad (20.4)$$

In Chapter 10 we saw that if \mathbf{y}_L is known, the search for the corresponding point \mathbf{y}_R in the right image can be restricted to an epipolar line \mathbf{l}_R in that image. And vice versa, if we instead know \mathbf{y}_R in the right image, the corresponding point \mathbf{y}_L lies somewhere along an epipolar line \mathbf{l}_L in the left image. We also learned that all epipolar lines in a specific image must intersect at the corresponding epipole. Since both epipoles are at infinity, a third way to define what weakly rectified stereo cameras means is given as:

20.2 The epipolar lines of weakly rectified stereo cameras are parallel in each image. Searching for corresponding point along parallel lines, which have the same slope, is in practice often simpler than searching along lines that have different slopes¹.

The fact that these last two observations are equivalent with definition 20.1 is left as an exercise to the reader.

Definition 20.1 has also algebraic consequences. Consider a 3D point $\bar{\mathbf{x}}$ and determine the homogeneous coordinates of its projection onto the left and right images:

$$\mathbf{y}_L \sim \mathbf{C}_L \begin{pmatrix} \bar{\mathbf{x}} \\ 1 \end{pmatrix} = \mathbf{K}_L \bar{\mathbf{x}}, \quad \mathbf{y}_R \sim \mathbf{C}_R \begin{pmatrix} \bar{\mathbf{x}} \\ 1 \end{pmatrix} = \mathbf{K}_R (\mathbf{R} \bar{\mathbf{x}} + \bar{\mathbf{t}}). \quad (20.5)$$

Next, we will see that these expressions can be further simplified by a pair of additional requirements, which leads to fully rectified stereo.

20.1.2 Fully rectified stereo

Although the requirements already introduced for weakly rectified stereo lead to some interesting results, they specify conditions only on the external parameters of the two cameras. To make rectified cameras more useful, two additional requirements will be introduced, including one on the internal parameters. This stronger form of

¹This statement applies to the case where the computational hardware is based on a general purpose CPU. Various dedicated hardware, including GPUs, can implement the search in an efficient way both for rectified and unrectified stereo images.

rectification, fully rectified stereo, is the one that has most practical applications. Consequently, we will also refer to it simply as rectified stereo.

Definition 20.2: Rectified Stereo Cameras

Rectified stereo cameras satisfy the conditions specified in definition 20.1 for weakly rectified stereo cameras. Additionally, the first coordinate axes of both cameras are co-linear and parallel to the baseline, and the internal camera parameters are identical for both cameras.

Next, we will see that these requirements lead to considerable simplifications of the epipolar geometry.

The first of these new requirements imply that there is no relative pose rotation between the cameras, i.e., $\mathbf{R} = \mathbf{I}$. The second requirement means that we can set $\mathbf{K}_L = \mathbf{K}_R = \mathbf{K}$. In combination, these results make Equation (20.1) become

$$\begin{aligned}\mathbf{C}_L &= \mathbf{K}(\mathbf{I}|\mathbf{0}), \\ \mathbf{C}_L &= \mathbf{K}(\mathbf{I}|\bar{\mathbf{t}}),\end{aligned}\quad \text{where } \bar{\mathbf{t}} = \begin{pmatrix} b \\ 0 \\ 0 \end{pmatrix}. \quad (20.6)$$

Here, b is the baseline of the stereo cameras, i.e., the distance between their camera centers. Taking the upper triangular structure of \mathbf{K} in Equation (8.35) into account, the projections on the 3D point $\bar{\mathbf{x}} = (x_1, x_2, x_3)$ in Equation (20.5) becomes:

$$\begin{aligned}\mathbf{y}_L &\sim \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ 0 & k_{22} & k_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} k_{11}x_1 + k_{12}x_2 + k_{13}x_3 \\ k_{22}x_2 + k_{23}x_3 \\ x_3 \end{pmatrix}, \\ \mathbf{y}_R &\sim \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ 0 & k_{22} & k_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 + b \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} k_{11}(x_1 + b) + k_{12}x_2 + k_{13}x_3 \\ k_{22}x_2 + k_{23}x_3 \\ x_3 \end{pmatrix}.\end{aligned} \quad (20.7)$$

The corresponding Cartesian image coordinates are

$$\bar{\mathbf{y}}_L = \begin{pmatrix} u_L \\ v_L \end{pmatrix} = \frac{1}{x_3} \begin{pmatrix} k_{11}x_1 + k_{12}x_2 + k_{13}x_3 \\ k_{22}x_2 + k_{23}x_3 \end{pmatrix}, \quad \bar{\mathbf{y}}_R = \begin{pmatrix} u_R \\ v_R \end{pmatrix} = \frac{1}{x_3} \begin{pmatrix} k_{11}(x_1 + b) + k_{12}x_2 + k_{13}x_3 \\ k_{22}x_2 + k_{23}x_3 \end{pmatrix}. \quad (20.8)$$

From these results we can make two observations. As a start, we see that $v_L = v_R$.

20.3 Corresponding points in rectified stereo cameras have the same vertical (row) coordinate, i.e., $v_L = v_R$. This means that corresponding points lie on the same row in the two images.

Although the horizontal coordinates of corresponding points differ, they do so in such a way that their difference is proportional to the inverse depth of the 3D point:

$$\Delta u = u_R - u_L = \frac{k_{11}b}{x_3} = / \text{Equation (8.36)} / = \frac{fsb}{x_3} = \frac{c}{x_3}. \quad (20.9)$$

Here, f is the focal length of the two cameras, and s is the pixel density, i.e., the number of pixels per length unit.

This result is sufficiently interesting to give the difference of horizontal coordinates its own name:

Definition 20.3: Disparity

In rectified stereo, the difference of the horizontal (column) coordinate of corresponding points is called *disparity*. It is here denoted as Δu .

Notice that $\Delta u \geq 0$, where $\Delta u = 0$ implies that the 3D point lies at infinite depth. We summarize these results as

- 20.4** The disparity of corresponding points in rectified stereo cameras is inversely proportional to the depth of the 3D point. The proportionality constant c is the product of the focal length f , the pixel density s , and the baseline b . See Equation (20.9).

In fact, rectified stereo can be summarized in terms of the coordinates of corresponding points:

$$\bar{\mathbf{y}}_L = \begin{pmatrix} u \\ v \end{pmatrix}, \quad \bar{\mathbf{y}}_R = \begin{pmatrix} u + \Delta u \\ v \end{pmatrix}, \quad (20.10)$$

where Δu is the disparity given by Equation (20.9).

In the sections that follow, we will investigate the consequences of this very simple relation between the coordinates of corresponding points in rectified stereo.

Disparity Estimation

Before going into the consequences of Equation (20.10), a short discussion on how to determine disparity may be appropriate. There are, in fact, two modes of operations for disparity estimation. In one mode, we have a set of interest points in the two images, as described in Section 9.1, and they brought into correspondence. In the rectified case, however, this problem is much simpler than for general stereo images. If we know that the images are rectified, the epipolar geometry is already known, and implies that corresponding points are always found on the same row in the image. We also know that disparity is never negative, so need only to search in one direction. This mode is called *sparse disparity estimation*.

A consequence of this simplification is that it becomes practically feasible to estimate disparity for the entire set of image points, so-called *dense disparity estimation*. Since all points are considered in this case, there is no need for an initial interest point detection. Instead, one of the two images is used as a reference, and for each point in that image, at position (u, imv) , we search for the corresponding point at position $(u + \Delta u, v)$ for $\Delta u \geq 0$ in the other image. This search is then based on the visual appearance of the region around the points in both images.

There are several approaches to both sparse and dense disparity estimation, and the specific details are outside the scope of this presentation. Dense disparity estimation, in particular, has been studied over a long period of time, and several methods have been presented. For an overview of disparity estimation methods, or general motion estimation methods, see the books by Jähne [39], Radke [56], Sonka, Hlavac and Boyle [60], or Szeliski [62].

It should, however, be emphasized that all method that produce disparity give estimates of the disparity, rather than exact measurements. Disparity estimation suffers from the same problems as the more general correspondence problem, described in Section 9.1. This includes occlusion and degeneracy for points without structure in their close neighborhood. In addition to that, most disparity estimation methods produce reliable estimates only when the disparity lies within a limited range, and when the image points are not close to the image border.

20.2 Rectified epipolar geometry

In the previous section, we derived relatively simple expressions for the image coordinates of corresponding points, Equation (20.10), given (fully) rectified stereo cameras. In this section, we will see how these results affect the epipolar geometry of rectified stereo cameras, for example in terms of the fundamental matrix, and how triangulation can be done.

20.2.1 Fundamental matrix

To derive the fundamental matrix for rectified stereo, we start by finding the two epipoles. We have Equation (20.3), which tells us that both epipoles are at infinity in the weaker case, but for the fully rectified case, we can also use $\mathbf{R} = \mathbf{I}$ and the expression for $\bar{\mathbf{t}}$ in Equation (20.6), in combination with the upper triangular structure of $\mathbf{K} = \mathbf{K}_L = \mathbf{K}_R$. The result is

$$\mathbf{e}_L \sim \mathbf{e}_R \sim \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \hat{\mathbf{b}}_1. \quad (20.11)$$

From observation 20.2, we already know that the epipolar lines in both images are parallel in the weaker case. But Equation (20.11) implies that they also are parallel with the first coordinate axes in the images, i.e., they are horizontal, in the fully rectified case. This is consistent with observation 20.3.

To determine the fundamental matrix, it turns out that Equation (10.13) provides the simplest expression to use for this case. We need the pseudo-inverse of $\mathbf{C}_1 = \mathbf{C}_L$:

$$\mathbf{C}_L^+ = \mathbf{C}_L^\top (\mathbf{C}_L \mathbf{C}_L^\top)^{-1} = \begin{pmatrix} \mathbf{K}^\top \\ \mathbf{0} \end{pmatrix} (\mathbf{K} \mathbf{K}^\top)^{-1} = \begin{pmatrix} \mathbf{K}^\top \\ \mathbf{0} \end{pmatrix} \mathbf{K}^{-\top} \mathbf{K}^{-1} = \begin{pmatrix} \mathbf{K}^{-1} \\ \mathbf{0} \end{pmatrix}. \quad (20.12)$$

With $\mathbf{C}_2 = \mathbf{C}_R$ and epipole $\mathbf{e}_{21} \sim \hat{\mathbf{b}}_1$, we get the fundamental matrix for rectified stereo:

$$\mathbf{F}_{\text{rect}} \sim \mathbf{C}_L^{+T} \mathbf{C}_R^\top [\hat{\mathbf{b}}_1]_\times = (\mathbf{K}^{-\top} \quad \mathbf{0}) \begin{pmatrix} \mathbf{I} \\ \bar{\mathbf{t}}^\top \end{pmatrix} \mathbf{K}^\top [\hat{\mathbf{b}}_1]_\times = \mathbf{K}^{-\top} \mathbf{K}^\top [\hat{\mathbf{b}}_1]_\times = [\hat{\mathbf{b}}_1]_\times = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}. \quad (20.13)$$

We see that for the case of rectified stereo, the expression for the fundamental matrix is a constant matrix, independent of all other parameters. In fact, we can even use this result as an alternative definition of rectified stereo.

20.5 The fundamental matrix for the case of rectified stereo, \mathbf{F}_{rect} , is given by Equation (20.13).

The equivalence of this observation and definition 20.2 is an exercise for the reader. It is also a simple exercise to show that this formulation of the fundamental matrix is consistent with the Cartesian coordinates of corresponding image points in Equation (20.11).

Yet another observation we can make here is that \mathbf{F}_{rect} is consistent with the formulation of the essential matrix, Equation (10.53), where $\mathbf{R} = \mathbf{I}$. Thus, for the case of rectified stereo, the epipolar geometry is specified by an essential matrix, even if the image points are specified in pixel coordinates, rather than camera normalized coordinates. This observation, however, is of little practical use. In fact, even \mathbf{F}_{rect} is of little practical use. If stereo cameras are known to be rectified, there is no need to use the epipolar constraint to check point correspondences. Instead, the epipolar constraint is replaced by the often much simpler procedure of checking if two points have the same vertical coordinate in the image.

However, the derivation of \mathbf{F}_{rect} is not in vain, it will come to use shortly.

20.2.2 Triangulation

When stereo cameras are rectified, the methods described in Section 16.1 for triangulation are still valid and can be used. But we have also seen that the rectified case of stereo leads to all sorts of simplifications and expect this to apply also to triangulation. This is what we investigate in this section.

From Section 16.1 we know that triangulation is really an estimation problem. In practice, corresponding points do not satisfy the epipolar constraint exactly, since their coordinates are perturbed by measurement noise. Later, we will analyze what this means also for the method to be developed here, but we start with the ideal case. This means that there is no measurement noise and the epipolar constraint is satisfied exactly for corresponding points, i.e., we assume that Equation (20.10) correctly represents the coordinates of corresponding points.

The relations presented in Equation (20.5) are not equalities, they are equivalence relation between projective elements. Consequently, we can write

$$\bar{\mathbf{x}} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \lambda \mathbf{K}^{-1} \mathbf{y}_L = \lambda \mathbf{y}_{Ln} = \lambda \begin{pmatrix} y_{L1} \\ y_{L2} \\ y_{L3} \end{pmatrix}. \quad (20.14)$$

Here, \mathbf{y}_{Ln} is the homogeneous coordinates of the left image point, using a C-normalized coordinate system. To properly get $\bar{\mathbf{x}}$, we must now determine λ . To do this, we use the last row of Equation (20.14) that gives an expression for x_3 , the depth of the 3D point. But we already have an expression for the depth in Equation (20.9), although that expression does not assume C-normalized coordinates. This assumption means that $f_s = 1$, and that we replace Δu by Δu_n , the displacement in C-normalized coordinates:

$$x_3 = \frac{b}{\Delta u_n}, \quad \text{where} \quad \Delta u_n = u_{Rn} - u_{Ln} = \frac{y_{R1}}{y_{R3}} - \frac{y_{L1}}{y_{L3}}. \quad (20.15)$$

This expression enables us to solve for λ :

$$\lambda = \frac{x_3}{y_{Ln3}} = \frac{b}{\Delta u_n y_{L3}}. \quad (20.16)$$

Finally, we get the Cartesian coordinates $\bar{\mathbf{x}}$ as

$$\bar{\mathbf{x}} = \frac{b}{\Delta u_n y_{L3}} \mathbf{y}_{Ln}. \quad (20.17)$$

An alternative approach is to determine the homogeneous coordinates of $\bar{\mathbf{x}}$:

$$\mathbf{x} \sim \begin{pmatrix} \bar{\mathbf{x}} \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{b}{\Delta u_n y_{L3}} \mathbf{y}_{Ln} \\ 1 \end{pmatrix} \sim \begin{pmatrix} b y_{Rn3} \mathbf{y}_{Ln} \\ \Delta u_n y_{L3} y_{Rn3} \end{pmatrix} = / \text{Equation (20.15)} / = \begin{pmatrix} b y_{Rn3} y_{L1} \\ b y_{Rn3} y_{L2} \\ b y_{Rn3} y_{L3} \\ y_{L3} y_{Rn1} - y_{L1} y_{Rn3} \end{pmatrix}. \quad (20.18)$$

This means that each element of \mathbf{x} can be expressed as a product between a specific matrix \mathbf{M}_{ni} and the two vectors \mathbf{y}_{Ln} and \mathbf{y}_{Rn} :

$$[\mathbf{x}]_i = \mathbf{y}_{Ln}^\top \mathbf{M}_{ni} \mathbf{y}_{Rn}. \quad (20.19)$$

The four matrices \mathbf{M}_{ni} are

$$\mathbf{M}_{n1} = \begin{pmatrix} 0 & 0 & b \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{M}_{n2} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & b \\ 0 & 0 & 0 \end{pmatrix}, \quad \mathbf{M}_{n3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & b \end{pmatrix}, \quad \mathbf{M}_{n4} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}. \quad (20.20)$$

The derivation of Equation (20.19) has been based on C-normalized coordinates in both images. With $\mathbf{y}_{Ln} = \mathbf{K}^{-1} \mathbf{y}_L$ and $\mathbf{y}_{Rn} = \mathbf{K}^{-1} \mathbf{y}_R$, this expression can now be reformulated as

$$[\mathbf{x}]_i = \mathbf{y}_L^\top \mathbf{K}^{-\top} \mathbf{M}_{ni} \mathbf{K}^{-1} \mathbf{y}_R = \mathbf{y}_L^\top \mathbf{M}_i \mathbf{y}_R, \quad \text{where} \quad \mathbf{M}_i = \mathbf{K}^{-\top} \mathbf{M}_{ni} \mathbf{K}^{-1}. \quad (20.21)$$

Thus, the formulation of the elements in \mathbf{x} as products between the homogeneous coordinates in the two images in combination with specific matrices applies also when the coordinates are transformed by \mathbf{K} to pixels². Taking the upper triangular structure of \mathbf{K} into account, Equation (8.35) in combination with the fact that the matrices \mathbf{M}_{ni} are very sparse, makes it possible to simplify the matrices \mathbf{M}_i as

$$\begin{aligned} \mathbf{M}_1 &= b \begin{pmatrix} 0 & 0 & k_{22} \\ 0 & 0 & -k_{12} \\ 0 & 0 & k_{12}k_{23} - k_{13}k_{22} \end{pmatrix}, & \mathbf{M}_2 &= b \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & k_{11} \\ 0 & 0 & -k_{11}k_{23} \end{pmatrix}, \\ \mathbf{M}_3 &= b \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & k_{11}k_{22} \end{pmatrix}, & \mathbf{M}_4 &= \begin{pmatrix} 0 & 0 & -k_{22} \\ 0 & 0 & k_{12} \\ k_{22} & -k_{12} & 0 \end{pmatrix}. \end{aligned} \quad (20.22)$$

20.6 In rectified stereo, triangulation can be implemented as products between the homogeneous coordinates in the left and right image, in combination with a set of matrices that only depend on the baseline and the internal camera parameters. See Equations (20.21) and (20.22).

²or whatever coordinate system \mathbf{K} transforms to

Blind plane

Observation 16.2 on page 282 tells us that it is not possible to triangulate points on the baseline, and we have seen that certain triangulation methods can exhibit degeneracy for additional points, observation 16.1 on page 281. This applies also to the method described here.

From the results that we have derived here for the triangulation of \mathbf{x} , it should be apparent that \mathbf{x} cannot have zero depth. In this case, $y_{L3} = y_{Rn3} = 0$, which renders $\bar{\mathbf{x}}$ in Equation (20.17) undefined. Alternatively, in Equation (20.18) we get $\mathbf{x} = \mathbf{0}$, which has no interpretation as a 3D point. Consequently, points that have zero depth, i.e., lie in the principal plane of the two cameras cannot be triangulated by this method. Since this set forms a plane, we make the following definition:

Definition 20.4: Blind plane

The points for which a specific triangulation method is degenerate is called the *blind plane*.

The blind plane can lie at infinity, it may be the principal plane, or some other plane depending on the triangulation method.

The blind plane that applies to the method described in this section is the principal plane of both cameras. From the outset, this may seem like a minor problem since these 3D points project to image points that lie at infinity, in both images. These image points will therefore never appear in the bounded image described in Section 8.4, and will never have to be triangulated. But as we will see in Section 20.3, rectified stereo can be the result of a synthetic rectification process, which is applied to the images rather than the cameras. If this is the case, the blind plane is no longer related to a common principal plane of the two cameras, it can be a more general plane. Even a plane that contains 3D points that are visible in both images.

Non-ideal case

In this section, we initially made the assumption that corresponding image points in the two images to satisfy Equation (20.10). We will now look at the rationale behind this assumption and look at what can be done if it does not hold.

As discussed in Section 20.2.2, there are two different modes of disparity estimation. In the case of dense disparity estimation, a correct rectification of the images is often an integral part of the underlying assumptions, i.e., corresponding points should be found on the same image row. If they are not, that will simply affect the accuracy of the estimated disparity in a negative way.

In the case of sparse disparity estimation, and when each image holds a set of interest points that should be brought into correspondence, the process can be more flexible. Corresponding points are still assumed to lie on the same row, but since the coordinates of detected interest points are affected by inaccuracy it may not be safe to implement this assumption as it is stated. Instead, the search for corresponding points should be done within a strip centered on a specific row. The height of the strip is then specified by the measurement noise.

Assuming that two corresponding image points have been found, $\mathbf{y}_L \sim (u_L, v_L, 1)$ in the left image, and $\mathbf{y}_R \sim (u_R, v_R, 1)$ in the right image, how do we triangulate the 3D point if they do not satisfy Equation (20.10) exactly? A simple approach is to first enforce the rectifying constraint in Equation (20.10), and then use either Equation (20.17) to determine $\bar{\mathbf{x}}$, or Equation (20.21) to determine \mathbf{x} . As a common value for the vertical coordinate we may simply choose the average:

$$v = \frac{v_L + v_R}{2}. \quad (20.23)$$

The price we pay for this constraint enforcement is that additional noise is introduced on the image coordinates. Thus, the accuracy in \mathbf{x} is reduced.

As alternative, we could use the methods already presented in Section 16.1 for stereo triangulation. However, if the optimal method is preferred, there are a few issues. As a start, the method presented in Section 16.1.3 assumes that neither epipole is at infinity, an invalid assumption for the case of rectified stereo. This means that Algorithm 16.3 on page 280 must be modified to handle the rectified case.

A second issue is that optimal triangulation minimizes geometric errors. It assumes that the measurement noise is added to the image coordinates in some coordinate system, e.g., the pixel-based system. As we will see in the next section, however, rectified stereo is often the result of *synthetic stereo rectification*, a process that transforms

general stereo images to rectified ones. Since minimization of geometric distances is not invariant to homography transformations, optimal triangulation will not give a correct ML-estimate of \mathbf{x} if applied to synthetically rectified stereo images. Consequently, optimal triangulation is often not relevant for rectified stereo.

20.3 Synthetic rectification

As noted in the introduction to this chapter, rectified stereo can be accomplished by placing the cameras in a mechanically stable rig and carefully adjusting the optical axes, while also aligning the image coordinate axes. Although this can be done with high accuracy, mechanical rectification is both tedious and costly to achieve, as well as to maintain over time.

The required alignment of axes for rectification can be done by rotating the cameras about their centers. Therefore, an alternative to a rectified stereo rig is offered by the results in Section 9.3, where it is shown that a rotation of the optical axis of a camera corresponds to the application of a homography transformation on the image coordinates. Based on this observation, it is sufficient to arrange the cameras in the rig as approximately rectified, and instead of rotating two optical axes to make the cameras rectified, we apply *rectifying homographies* on the two images. The rectifying homographies transform the stereo images to make them appear as if they had been produced by a rectified stereo rig. This process is here referred to as *synthetic stereo rectification*.

What makes this approach interesting is that to do so, we do not need to determine the two 3D rotations, we need only the fundamental matrix \mathbf{F} for the stereo cameras. To see this, let $\mathbf{H}_L, \mathbf{H}_R$ be a pair of homographies, one for each of the two images, in accordance with Equation (9.25). Let $\mathbf{y}_L, \mathbf{y}_R$ be the homogeneous coordinates of a pair of corresponding points in the pixel-based coordinate system of each image *before* they are transformed by the homographies. This means that \mathbf{y}_L and \mathbf{y}_R must satisfy the epipolar constraint, Equation (10.7), relative the fundamental matrix \mathbf{F} . The homogeneous coordinates of the two points *after* they have been transformed are:

$$\mathbf{y}'_L = \mathbf{H}_L \mathbf{y}_L, \quad \mathbf{y}'_R = \mathbf{H}_R \mathbf{y}_R. \quad (20.24)$$

The goal here is to choose the two homographies such that when applied to its corresponding image, the resulting images appear as rectified. This implies that the fundamental matrix after these transformations is \mathbf{F}_{rect} , Equation (20.13). Since the fundamental matrix before the transformations is \mathbf{F} , we can use the results in Section 10.3.2 to describe a relation between the un-transformed and transformed fundamental matrix:

$$\mathbf{F} \sim \mathbf{H}_L^\top \mathbf{F}_{\text{rect}} \mathbf{H}_R = \mathbf{H}_L^\top [\hat{\mathbf{b}}_1]_{\times} \mathbf{H}_R. \quad (20.25)$$

To summarize: given stereo images with a corresponding fundamental matrix \mathbf{F} , we want to determine \mathbf{H}_L and \mathbf{H}_R that satisfy Equation (20.25), and they can be used to produce a pair of rectified stereo images. They are referred to as *synthetically rectified stereo images*; they are originally not rectified, but become rectified when the two homographies are applied. In practice, rectification of stereo images is in most cases done synthetically, and therefore stereo rectification is often synonymous with synthetically rectified stereo.

Definition 20.5: Rectifying homographies

Given a fundamental matrix \mathbf{F} , any pair of homographies \mathbf{H}_L and \mathbf{H}_R that satisfy Equation (20.25) are referred to as *rectifying homographies*.

Notice that this definition no longer makes an explicit assumption that \mathbf{H}_L and \mathbf{H}_R are rotational homographies!

This definition does not specify how to find rectifying homographies. In fact, it does not even provide any evidence neither that they exist, nor that they are unique. If they exist, however, they are not unique. Let $\mathbf{H}_L, \mathbf{H}_R$ be a pair of rectifying homographies, satisfying Equation (20.25). Then, another pair of rectifying homographies, which also satisfy Equation (20.25), is given by

$$\mathbf{H}'_L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{-\top} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \mathbf{H}_L + \hat{\mathbf{b}}_1 \mathbf{u}^\top, \quad \mathbf{H}'_R = \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{pmatrix} \mathbf{H}_R + \hat{\mathbf{b}}_1 \mathbf{v}^\top. \quad (20.26)$$

Here, \mathbf{u}, \mathbf{v} are arbitrary³ vectors in \mathbb{R}^3 , and \mathbf{A} is an arbitrary full rank 2×2 matrix.

20.7 Rectifying homographies are not unique, there are infinitely many pairs of rectifying homographies for a given fundamental matrix \mathbf{F} .

It remains to establish that rectifying homographies for every choice of a proper fundamental matrix \mathbf{F} exist. As we will see in the next section, rectifying homographies can always be found for any fundamental matrix \mathbf{F} , but observation 20.7 implies that they are not unique. There are several methods in the literature that can produce rectifying homographies for a given \mathbf{F} . They differ mainly in which pair, out of the infinitely many pairs in Equation (20.26), a specific method produces.

20.3.1 Preliminary results

Let $\mathbf{e}_L \sim (u_L, v_L, 1)$ and $\mathbf{e}_R \sim (u_R, v_R, 1)$ be the epipoles in the unrectified images, neither of them at infinity. \mathbf{F} is the corresponding fundamental matrix. The previous results imply that the rectifying homographies must place both epipoles at infinity, to an ideal point represented by the homogeneous coordinates $\hat{\mathbf{e}}_1$:

$$\hat{\mathbf{b}}_1 \sim \mathbf{H}_L \mathbf{e}_L, \quad \hat{\mathbf{b}}_1 \sim \mathbf{H}_R \mathbf{e}_R. \quad (20.27)$$

There are infinitely many homographies that satisfy these relations, for example:

$$\mathbf{H}'_L \sim \begin{pmatrix} u_L & v_L & 1 \\ v_L & -u_L & 0 \\ u_L & v_L & -(u_L^2 + v_L^2) \end{pmatrix}, \quad \mathbf{H}'_R \sim \begin{pmatrix} u_R & v_R & 1 \\ v_R & -u_R & 0 \\ u_R & v_R & -(u_R^2 + v_R^2) \end{pmatrix}. \quad (20.28)$$

This choice of homographies guarantees that

$$\mathbf{H}'_L^{-\top} \mathbf{F} \mathbf{H}'_R^{-1} \sim \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{pmatrix}, \quad (20.29)$$

where \mathbf{A} is a full rank 2×2 matrix. Since the right-hand side of this relation in general is not equivalent to \mathbf{F}_{rect} , this means that Equation (20.27) is not sufficient to make rectifying homographies out of \mathbf{H}'_L and \mathbf{H}'_R . But, from this relation, we get

$$\underbrace{\mathbf{H}'_L^{-\top} \mathbf{F} \mathbf{H}'_R^{-1} \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{-1} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}}_{\mathbf{H}_R^{-1}} \sim \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \sim \mathbf{F}_{\text{rect}}. \quad (20.30)$$

This means that if we keep the left homography, $\mathbf{H}_L = \mathbf{H}'_L$, and set

$$\mathbf{H}_R^{-1} \sim \mathbf{H}'_R^{-1} \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{-1} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \Rightarrow \quad \mathbf{H}_R \sim \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{pmatrix} \mathbf{H}'_R, \quad (20.31)$$

then \mathbf{H}_L and \mathbf{H}_R form a pair of rectifying homographies. From this we conclude:

20.8 For every fundamental matrix \mathbf{F} , we can always find a pair of rectifying homographies \mathbf{H}_L and \mathbf{H}_R , which satisfy definition 20.5. By observation 20.7, these homographies are not unique.

³We should avoid \mathbf{u} and \mathbf{v} that make the corresponding homography singular.

Before we continue

Based on the derivation of the rectifying homographies in this section, we can make a few observations. First of all, we get the resulting transformations \mathbf{H}_L and \mathbf{H}_R in two steps. The first step is to find the transformations \mathbf{H}'_L and \mathbf{H}'_R that map the corresponding epipole to infinity, more precisely to the ideal point represented by $\hat{\mathbf{b}}_1$. The second step applies additional transformations either on \mathbf{H}'_L or \mathbf{H}'_R , to also make Equation (20.25) satisfied.

One way to think about these additional transformations is that in the end, the rectifying homographies map corresponding epipolar lines in the original images to *identical lines* in the transformed images. Thus, if \mathbf{l}_L and \mathbf{l}_R are two corresponding epipolar lines in the left and right image, in the original images, then

$$\mathbf{H}_L^{-\top} \mathbf{l}_L \sim \mathbf{H}_R^{-\top} \mathbf{l}_R. \quad (20.32)$$

Here, the lines are mapped by the dual transformations, defined in Section 4.5.

Definition 20.6: Matched pair of transformations

A pair of *matched transformations*, \mathbf{H}_L and \mathbf{H}_R , map any pair of corresponding epipolar lines to one and the same line in both images. The mapping of lines is defined in terms of the dual transformations, according to Equation (20.32).

Consequently, an alternative to Equation (20.25) is to require that rectifying homographies must satisfy two conditions: they map the epipoles to infinity in accordance with Equation (20.27), and they form a matching pair.

The two steps above take care of these two requirements, one at the time. Other rectification algorithms are often divided into similar steps. But, since both steps can be implemented in various ways, different rectification methods that make different choices for the two steps.

A second observation is that from the initial pair of homographies, \mathbf{H}'_L and \mathbf{H}'_R , we keep \mathbf{H}'_L as it is and apply an additional set of transformation only on \mathbf{H}'_R to get a matching pair. As an alternative, we can keep \mathbf{H}'_R and apply a set of transformations on \mathbf{H}'_L . In general, the two alternatives do not produce the same pair of rectifying homographies, as indicated by observation 20.7.

Finally, since there are many choices for the homographies that can rectify a pair of stereo images, it is to be expected that some of them can deform the images in a significant way. This issue was never discussed for the method introduced in this section. In fact, unless deformation is not a problem, that particular way of choosing the homographies *is not recommended*. Without additional steps that take care of the deformation, it will not be of much practical use.

The following section presents a standard technique for stereo rectification. It addresses this last issue, by including measures that reduce deformation.

20.3.2 Hartley's rectification method

In this section, we present a practical method that determines a pair of rectifying homographies, \mathbf{H}_L and \mathbf{H}_R , given a fundamental matrix \mathbf{F} . The method, described by Hartley [29], produces a pair of rectifying homographies which are unique in the sense that each step of the computations is specified to reduce deformation. However, deformation is here defined in a specific way, and there are alternative approaches which have a similar approach but where deformation is defined differently.

Hartley's method has a similar structure as the one described in the previous section, but it is also different. In the first step, only one of the two transformations are specified to map an epipole to infinity. In the second step, we will see that the other transformation can be chosen directly to both map its epipole to infinity, and to make a matching pair of the two transformations. In both steps, the necessary transformations are chosen to reduce deformation of the image. Here, \mathbf{H}_L is determined in the first step, while the second step produce \mathbf{H}_R , but an alternative is to do it the other way around.

As input, the method uses a set of m corresponding points, $\{\mathbf{y}_{Lk}, \mathbf{y}_{Rk}\}$. A fundamental matrix \mathbf{F} can be estimated from this set, using any of the techniques described in Section 16.2. From \mathbf{F} , we also get the two epipoles \mathbf{e}_L and \mathbf{e}_R , which satisfy $\mathbf{e}_L^\top \mathbf{F} = \mathbf{0}$ and $\mathbf{F} \mathbf{e}_R = \mathbf{0}$. Let (u_L, v_L) be the Cartesian coordinates of epipole \mathbf{e}_L .

Mapping the epipole to infinity

In this first step of the method, we will find a transformation of the left image that maps the corresponding epipole to $\hat{\mathbf{b}}_1$, a point at infinity. This transformation is constructed as a sequence of simpler transformations. For this purpose, we need a point $\bar{\mathbf{y}}_0 = (u_0, v_0)$ in the left image where we want the deformation to be minimal. Typically, $\bar{\mathbf{y}}_0$ is the center of the image, i.e., the principal point, but the exact position is not critical. It could be other point that we have reason to choose for this purpose. As we will see shortly, \mathbf{y}_0 must not coincide with the epipole \mathbf{e}_L . The first transformation in the sequence is a translation that places the origin of our coordinate system at $\bar{\mathbf{y}}_0$:

$$\mathbf{T}_{\text{trans}} \sim \begin{pmatrix} 1 & 0 & -u_0 \\ 0 & 1 & -v_0 \\ 0 & 0 & 1 \end{pmatrix}. \quad (20.33)$$

After this translation, the Cartesian coordinates of the left epipole \mathbf{e}'_L become

$$(u'_L, v'_L) = (u_L - u_0, v_L - v_0). \quad (20.34)$$

After the translation, in general, the epipole does lie on the horizontal u -axis, but can be placed on this axis by a rotation. For example, by specifying a rotation transformation as

$$\mathbf{T}_{\text{rot}} = \begin{pmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \text{where } \alpha = \text{atan2}(v'_L, u'_L), \quad (20.35)$$

the left epipole ends up at

$$\hat{\mathbf{e}}_L = \mathbf{T}_{\text{rot}} \mathbf{T}_{\text{trans}} \mathbf{e}_L \sim \begin{pmatrix} \hat{u}_L \\ 0 \\ 1 \end{pmatrix}. \quad (20.36)$$

Note that the left epipole now lie on the (horizontal) u -axis. Also note that this has been accomplished by a rigid transformation of the left image, which do not introduce any deformation of the image.

The final transformation in the first step maps the epipole to a point at infinity, intuitively by moving it along the horizontal direction. This is done by the following homography:

$$\mathbf{H}_\infty = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/\hat{u}_L & 0 & 1 \end{pmatrix}. \quad (20.37)$$

Applied to its corresponding epipole, it gives

$$\hat{\mathbf{e}}'_L = \mathbf{H}_\infty \hat{\mathbf{e}}_L \sim \begin{pmatrix} \hat{u}_L \\ 0 \\ 0 \end{pmatrix} \sim \hat{\mathbf{b}}_1. \quad (20.38)$$

Note that the left epipole now is at infinity, representing a horizontal orientation. The last transformation requires that $\hat{\mathbf{e}}_L$ not lies at the origin, i.e., that $\hat{u}_L \neq 0$. This is assured by choosing \mathbf{y}_0 distinct from \mathbf{e}_L .

The particular choice of \mathbf{H}_∞ in Equation (20.37) is motivated by the minimal deformation it has at the origin, i.e., at the point $\bar{\mathbf{y}}_0$. To see this, we express how it transforms the Cartesian coordinates of some point (u, v) :

$$\mathbf{H}_\infty \{(u, v)\} = \begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} \frac{u}{1-u/\hat{u}_L} \\ \frac{v}{1-u/\hat{u}_L} \end{pmatrix} \approx \begin{pmatrix} u(1+u/\hat{u}_L) \\ v(1+u/\hat{u}_L) \end{pmatrix}. \quad (20.39)$$

The approximation is valid when $|u| \ll |\hat{u}_L|$. The Jacobian of this transformation is

$$\frac{\partial(u', v')}{\partial(u, v)} = \begin{pmatrix} 1+2u/\hat{u}_L & 0 \\ v/\hat{u}_L & 1+u/\hat{u}_L \end{pmatrix}. \quad (20.40)$$

Close the origin, when $(u, v) \approx (0, 0)$, the Jacobian is approximately equal to the identity matrix \mathbf{I} . Consequently, close to $\bar{\mathbf{y}}_0$, \mathbf{H}_∞ introduces almost no deformation.

To summarize these first steps, we have defined a transformation in the left image as

$$\mathbf{H}_L = \mathbf{H}_\infty \mathbf{T}_{\text{rot}} \mathbf{T}_{\text{trans}}. \quad (20.41)$$

It transforms the left epipole to a point at infinity, represented by $\hat{\mathbf{b}}_1$. This is the similar approach as the one outlined in Section 20.3.1, except it is done only in the left image. We have also been more careful about image deformation, at least around the point $\tilde{\mathbf{y}}_0$ in the left images.

Finding matching transformations

The final step is to also find \mathbf{H}_R , such that it, too, sends its corresponding epipole to infinity and is a matching transformation relative to \mathbf{H}_L .

Let \mathbf{H}_L and \mathbf{H}_R form a matching pair of transformations, in accordance with definition 20.6 on page 359, and let the underlying epipolar geometry be defined by the fundamental matrix \mathbf{F} . Based on observation 10.13, it must then be the case that

$$\mathbf{H}_L^{-\top} \mathbf{M}^{-\top} \mathbf{I}_R \sim \mathbf{H}_R^{-\top} \mathbf{I}_R, \quad (20.42)$$

for all epipolar lines \mathbf{I}_R in the right image. In this relation, \mathbf{M} is a mapping that satisfies $\mathbf{F} \sim [\mathbf{e}_L] \times \mathbf{M}$, in accordance with Section 10.3.5. A sufficient condition for the relation to be satisfied is $\mathbf{H}_L^{-\top} \mathbf{M}^{-\top} \sim \mathbf{H}_R^{-\top}$. Elimination of the inverses and transposes gives a sufficient condition for making \mathbf{H}_L and \mathbf{H}_R a matching pair:

$$\mathbf{H}_R \sim \mathbf{H}_L \mathbf{M}. \quad (20.43)$$

In fact, this condition is also necessary, which is left as an exercise to the reader.

Another interesting result is given by observation 10.10 on page 151:

$$\mathbf{H}_R \mathbf{e}_R \sim \mathbf{H}_L \mathbf{M} \mathbf{e}_R \sim \mathbf{H}_L \mathbf{e}_L \sim \hat{\mathbf{b}}_1. \quad (20.44)$$

This means that choosing \mathbf{H}_R according to Equation (20.43) assures not only that \mathbf{H}_L and \mathbf{H}_R are matching transformations, they are also rectifying homographies since each of them maps its epipole to $\hat{\mathbf{b}}_1$.

From observation 10.9, we know that \mathbf{M} is not unique. Any valid choice of \mathbf{M} can be replaced by $\mathbf{M} + \mathbf{e}_L \tilde{\mathbf{a}}^\top$, where $\tilde{\mathbf{a}} \in \mathbb{R}^3$ is arbitrary. Consequently, for a fixed choice of \mathbf{M} , there are multiple choices for \mathbf{H}_R that give a matching pair:

$$\mathbf{H}_R \sim \mathbf{H}_L (\mathbf{M} + \mathbf{e}_L \tilde{\mathbf{a}}^\top) = \mathbf{H}_L \mathbf{M} + \mathbf{H}_L \mathbf{e}_L \tilde{\mathbf{a}}^\top (\mathbf{H}_L \mathbf{M})^{-1} \mathbf{H}_L \mathbf{M} = (\mathbf{I} + \hat{\mathbf{b}}_1 \mathbf{a}^\top) \mathbf{H}_L \mathbf{M}. \quad (20.45)$$

Here, we have used $\mathbf{H}_L \mathbf{e}_L \sim \hat{\mathbf{b}}_1$, and substituted $\mathbf{a} = \mathbf{H}_L^{-\top} \mathbf{M}^{-\top} \tilde{\mathbf{a}}$. We now have $\mathbf{a} \in \mathbb{R}^3$ as arbitrary.

With this result at hand, we can reformulate Equation (20.43) as the more general relation

$$\mathbf{H}_R \sim \mathbf{A} \mathbf{H}_L \mathbf{M}, \quad \text{where} \quad \mathbf{A} = \mathbf{I} + \hat{\mathbf{b}}_1 \mathbf{a}^\top = \begin{pmatrix} a & b & c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad (20.46)$$

where any $\tilde{\mathbf{a}} = (a, b, c)$ produces a pair of matching transformations. To determine $\tilde{\mathbf{a}}$, we again minimize deformation. First, the initial set of corresponding image points are transformed by \mathbf{H}_L in the left image and by $\mathbf{H}_L \mathbf{M}$ in the right image:

$$\tilde{\mathbf{y}}_{Lk} \sim \mathbf{H}_L \mathbf{y}_{Lk}, \quad \tilde{\mathbf{y}}_{Rk} \sim \mathbf{H}_L \mathbf{M} \mathbf{y}_{Lk}, \quad k = 1, \dots, m. \quad (20.47)$$

After this mapping, corresponding image points $(\tilde{u}_{Lk}, \tilde{v}_{Lk})$ and $(\tilde{u}_{Rk}, \tilde{v}_{Rk})$ are always found on the same vertical coordinate, i.e., $\tilde{v}_{Lk} = \tilde{v}_{Rk}$, since the images are rectified. What remains is to apply the final transformation \mathbf{A} in the right image to reduce deformation. The disparity between corresponding points, also after the last transformation, reflects the depth of the corresponding 3D point. Although we never can expect the disparity to be zero, making it minimal over the entire point-set is one way to reduce the overall deformation. Thus, we minimize

$$\epsilon_{\text{rect}} = \sum_{k=1}^m d_{PP}(\mathbf{y}_{Lk}, \mathbf{A} \mathbf{y}_{Rk})^2 = \sum_{k=1}^m (a \tilde{u}_{Rk} + b \tilde{v}_{Rk} + c - \tilde{u}_{Lk})^2 \quad (20.48)$$

over a, b, c . Notice that the point-to-point distance only needs to involve the horizontal displacement between $\tilde{\mathbf{y}}_{Lk}$ and $\tilde{\mathbf{y}}_{Rk}$, i.e., their disparity, since they have of the same vertical coordinate.

Minimizing $\varepsilon_{\text{rect}}$ is a standard least squares problem, which can be solved by standard techniques⁴. For example, the corresponding normal equation is

$$\tilde{\mathbf{Y}}_R \tilde{\mathbf{Y}}_R^\top \bar{\mathbf{a}} = \tilde{\mathbf{Y}}_R \tilde{\mathbf{u}}_L^\top. \quad (20.49)$$

Here, $\tilde{\mathbf{Y}}_R$ is a $3 \times m$ matrix that holds the *canonical* homogeneous coordinates of the points $\tilde{\mathbf{y}}_{Rk}$ in its columns, and $\tilde{\mathbf{u}}_L$ is an m -dimensional row vector that holds the horizontal coordinates of the points $\tilde{\mathbf{y}}_{Lk}$.

Assuming $m \geq 3$ corresponding points in general configurations, Equation (20.49) has a unique solution $\bar{\mathbf{a}} = (a, b, c)$. Once $\bar{\mathbf{a}}$ is determined, \mathbf{H}_R can be computed from Equation (20.46). This concludes Hartley's rectification method, which is summarized in Algorithm 20.1.

A pair of unrectified stereo images are shown in Figure 20.2. The result of applying Hartley's rectification method on these images is shown in Figure 20.3. The epipoles in the rectified images are at infinity, which means that all epipolar lines are horizontal. Corresponding lines are found on the same vertical position.

Before we continue

To uniquely determine \mathbf{F} , we need a minimum of $m = 8$ correspondences, but the more, the better. The standard technique to estimate \mathbf{F} is the normalized 8-point algorithm, Algorithm 16.4 on page 285, but the Gold Standard method in Algorithm 16.6 is the preferred choice, should non-linear optimization estimation be available. The simplest way to set \mathbf{M} is to use and SVD of \mathbf{F} and then Equation (10.43).

There is a possibility that the two rectified images appear upside-down. Although this may look weird to a human observer, it is a perfectly valid result and should not affect further automatic processing of the images. The reason for this result is the rotation angle α in Equation (20.35), which here is defined such that it always rotates the epipole \mathbf{e}'_L to the positive side of the u -axis. To reduce large rotations, add or subtract π to α if $|\alpha| > \pi/2$.

Even though Hartley's rectification method tries to reduce image deformation, it does not always succeed in doing so. For example, it does not distribute the deformation evenly in the two images, a result of the fact that the algorithm does not treat both images in the same way. This can be seen in Figure 20.3, where the right image appears to significantly more deformed than the left one. This issue has been addressed in later rectification methods, but at the cost of a more complex algorithm. For an example, see the method by Loop & Zhang [47].

⁴See Toolbox Section 3.9.

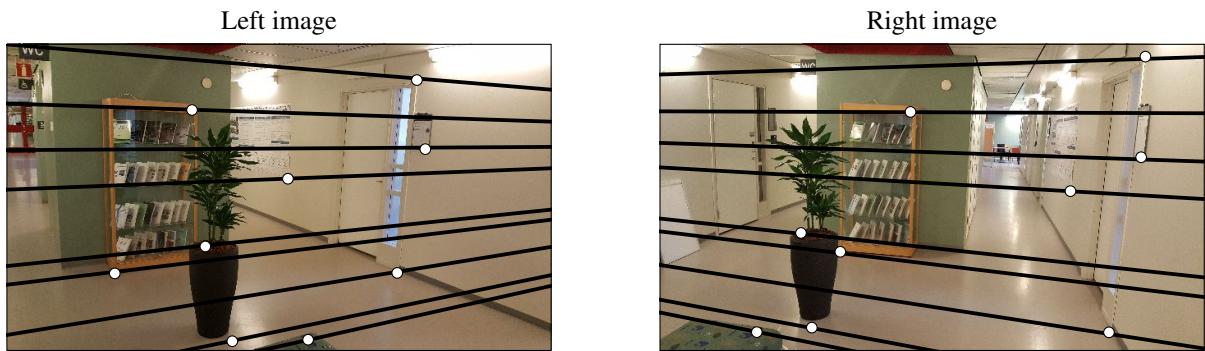


Figure 20.2: Two unrectified stereo images. The corresponding epipolar geometry (fundamental matrix) is estimated from a set of corresponding points (circles). For each point, its epipolar line is drawn in the other images. For each image, the epipolar lines intersect at the epipole. Since the images are not rectified, the epipoles are not lying at infinity.

Algorithm 20.1: Hartley's methods for stereo rectification

- Input:** A set of m corresponding image points $\mathbf{y}_{Lk}, \mathbf{y}_{Rk}, k = 1, \dots, m$, in the left and right image.
- Input:** A point $\bar{\mathbf{y}}_0 = (u_0, v_0)$, not equal to any epipole, where geometric distortion should be minimal.
- Output:** A pair of rectifying homographies \mathbf{H}_L and \mathbf{H}_R .
- 1 Estimate \mathbf{F} using, e.g., Algorithm 16.4 on page 285 or, even better, Algorithm 16.6
 - 2 Determine epipole \mathbf{e}_L from \mathbf{F}
 - 3 Determine matrix \mathbf{M} , e.g., using Equation (10.43)
 - 4 Determine translation $\mathbf{T}_{\text{trans}}$ from $\bar{\mathbf{y}}_0$, Equation (20.33)
 - 5 Determine translated epipole $\mathbf{e}'_L = \mathbf{T}_{\text{trans}} \mathbf{e}_L$
 - 6 From \mathbf{e}'_L , determine the corresponding rotation \mathbf{T}_{rot} , Equation (20.35)
 - 7 *Optional:* add π to angle if its magnitude is larger than $\pi/2$, to avoid upside-down images
 - 8 Compute $\hat{\mathbf{e}}_L = \mathbf{T}_{\text{rot}} \mathbf{e}'_L$. The left epipole is now on the u -axis
 - 9 From $\hat{\mathbf{e}}_L$, determine the corresponding transformation \mathbf{H}_∞ , Equation (20.37)
 - 10 Set $\mathbf{H}_L = \mathbf{H}_\infty \mathbf{T}_{\text{rot}} \mathbf{T}_{\text{trans}}$
 - 11 Transform all image points: $\tilde{\mathbf{y}}_{Lk} = \mathbf{H}_L \mathbf{y}_{Lk}$ and $\tilde{\mathbf{y}}_{Rk} = \mathbf{H}_L \mathbf{M} \mathbf{y}_{Lk}$, $k = 1, \dots, m$
 - 12 Form matrix $\tilde{\mathbf{Y}}_L$ with the *canonical* homogeneous coordinates $\tilde{\mathbf{y}}_{Lk}$ in its columns
 - 13 Form row vector $\tilde{\mathbf{u}}_R$ that contains the u -coordinates of points $\tilde{\mathbf{y}}_{Rk}$
 - 14 Solve $\bar{\mathbf{a}}$ from the normal equation $\tilde{\mathbf{Y}}_R \tilde{\mathbf{Y}}_R^\top \bar{\mathbf{a}} = \tilde{\mathbf{Y}}_R \tilde{\mathbf{u}}_L^\top$
 - 15 Form matrix \mathbf{A} from Equation (20.46), using the elements of $\bar{\mathbf{a}} = (a, b, c)$
 - 16 Set $\mathbf{H}_R = \mathbf{A} \mathbf{H}_L \mathbf{M}$

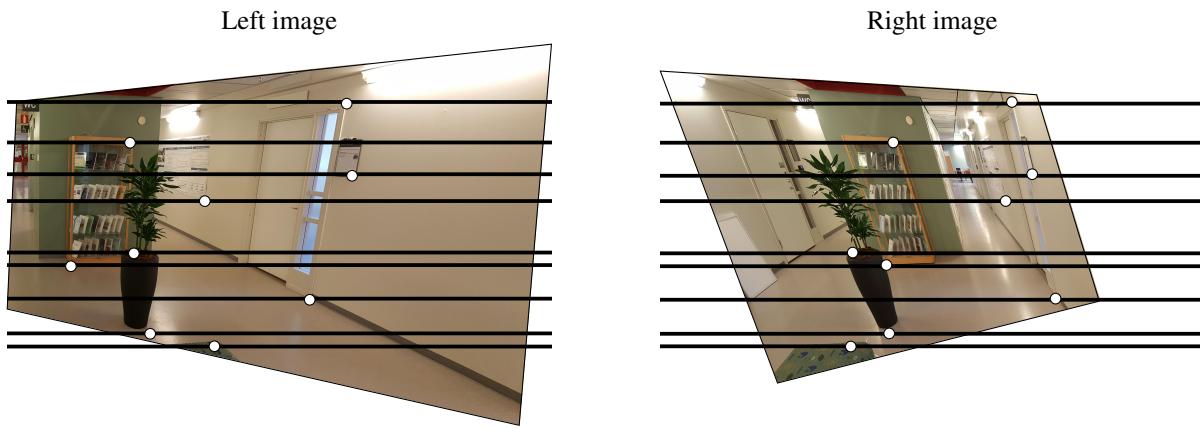


Figure 20.3: The result of applying Hartley's rectification method on the stereo images in Figure 20.2. Since the images are rectified, both epipoles are at infinity, the epipolar lines are parallel, and corresponding lines have the same vertical position.

Chapter 21

Structure from Motion

Before you read this chapter, you should have thorough understanding of the robust estimation, in particular RANSAC, described in Chapter 17. Furthermore, the non-linear estimation techniques described in Chapter 17 are used extensively in this chapter.

Given a pair of stereo images, we can extract a set of interest points from each image and apply robust estimation of the epipolar geometry related to the two images, using the approach described in Section 17.5.2. The result is a fundamental matrix \mathbf{F} , representing the epipolar geometry of the stereo images, and a consensus set of correspondences between interest points in the two images. If the camera matrices that refer to each of the two stereo images are known, we can then apply triangulation, as described in Section 10.4. This produces a set of 3D points that correspond to the interest points in the two images, a Euclidean reconstruction. Since a full calibration of the cameras often is impractical, this case is not very common. If the cameras are not known, it is possible to determine a pair of cameras that are consistent with \mathbf{F} and from them instead produce a projective reconstruction. Due to the geometric distortion implied by projective reconstruction, also this case is of limited practical application.

The remaining case is to calibrate the cameras, so that their internal parameters, \mathbf{K} , are known. From \mathbf{F} and \mathbf{K} we can then determine the essential matrix \mathbf{E} , Equation (10.50). From \mathbf{E} the relative camera pose can be deduced, but only up to an unknown scaling, as described in Section 10.5.3. Based on this information we can do a similarity reconstruction of 3D points, described in Section 10.4.1. The resulting reconstruction then has a scale ambiguity, which can be resolved by setting some distance in the reconstructed scene to a unit value.

Reconstruction of 3D points from only two views, however, has limitations. Obviously, the reconstruct points are restricted to lie on a 3D surface that is facing both cameras. Points on all sides of an object cannot be reconstructed, simply because they are not seen by the cameras. Another issue is how the reconstruction error depends on the length of the baseline of the stereo cameras. We want the cameras to be as far apart as possible to obtain a low reconstruction error, as discussed in Section 16.1.4. On the other hand, the further apart the cameras are, the smaller is the surface of the object that can be seen from both cameras, an effect that is emphasized the closer the object is to the cameras. This is illustrated in Figure 21.1.

To overcome these potential limitations, we need to extend the reconstruction to a set of images, taken by cameras with distinct centers. From multiple views of an object it should, in principle, be possible to reconstruct any point on the object's surface. To this end, we have to assume that any point can be seen in more than two or more of the images. Furthermore, we have to assume that observations in one image can be put into correspondence with observations in several other images. It should then be possible to reconstruct the corresponding 3D points, and to do so with a higher accuracy than what is possible from only two images. But how exactly do we achieve such a multi-view reconstruction?

21.1 Simplifying assumptions

In this chapter, we discuss two distinct approaches to this problem. They are based on a set of additional assumptions that enable us to simplify the problem, without making it trivial:

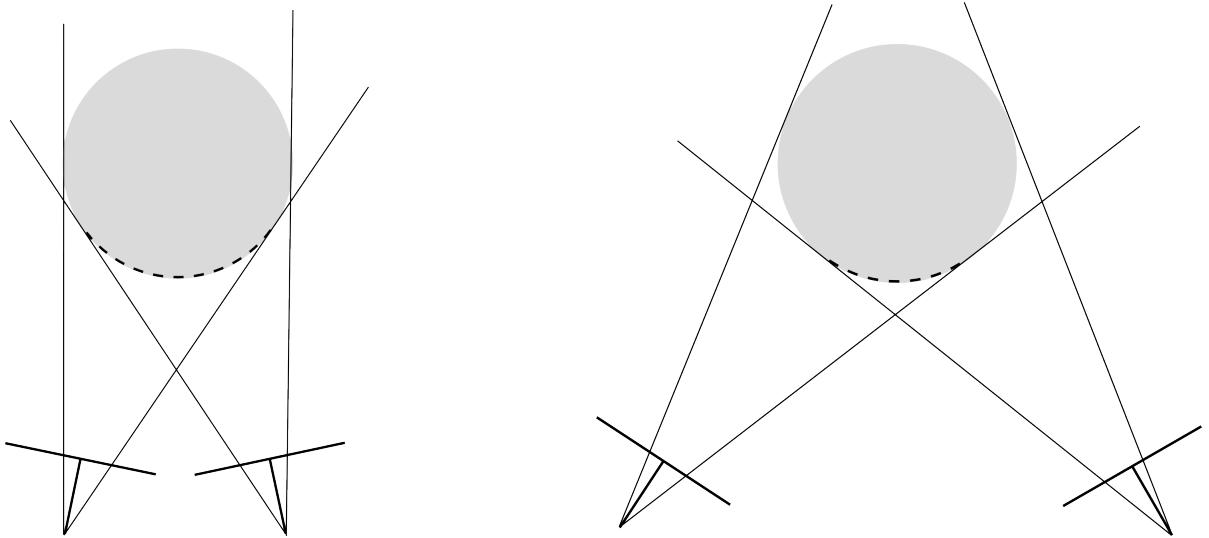


Figure 21.1: Two stereo cameras are observing an object (grey circle). Left: a narrow baseline makes the surface of the object that is visible in both cameras (dashed curve) larger. Right: a wider baseline makes the surface of the object that is visible in both cameras smaller.

1. The images are taken by cameras with centers that lie along a path, a curve, in 3D space. For example, this is the case if the images are produced by a single camera that is moving along the path. The camera can be a video camera taking images at a constant rate per second, or a still image camera taking the images at arbitrary time points. Alternatively, we can arrange a set of cameras along the path.
2. The 3D points depicted in the images are fixed relative to the world coordinate system. This means that if we are moving the camera along the path, all points in the scene depicted by the camera are stationary when the camera moves. If the images are produced by independent cameras, stationarity of the 3D points is not a requirement if the cameras can be synchronized to take their images at the same point in time.
3. The mapping from 3D space to images includes no significant lens distortion, which means that the pinhole camera model can be used. Furthermore, the camera, or cameras, which are used to produce the images have known internal calibration. This means that we have a known set of internal parameters \mathbf{K} for each camera that has produced every image. A simple approach is to use a single camera, with a fixed zoom and focus that make \mathbf{K} constant, and move it along the path and take the images. Either before or after the images are produced, we determine \mathbf{K} , for example by means of the method described in Chapter 18. If we instead use multiple cameras to produce the images, we need to determine the internal parameters of each individual camera.
4. There exist *putative correspondences* between image points in pairs of images along the path. They are putative in the sense that they may include correspondences that, upon further analysis, turn out to be incorrect. These putative correspondences can, for example, be the result of tracking interest points over an image sequences, or formed by comparing visual features of these points. The correspondences should normally stretch over several images, not just between adjacent images along the path.

The first assumption implies that we have an explicit relation of “closeness” between images along the path. It also implies that we can define an ordering of the images along the path, for example related to the temporal ordering in which they are taken. Thus, the images form a sequence, $\{I_k, k = 1, \dots, m\}$, where m is the number of images in the sequence, each with a corresponding camera matrix \mathbf{C}_k for $k = 1, \dots, m$. The cameras must have distinct centers, otherwise the epipolar geometry between pairs of cameras becomes degenerate. To avoid degeneracy, we must also assume that the scene contains 3D points in general configuration, e.g., they are not restricted to a plane.

Given an image I_k , we can explicitly refer to the preceding and succeeding images as I_{k-1} and I_{k+1} , respectively. We expect that images that are close along the path, with similar indices, are relative similar and contain many

common observations. The further away two images are in the sequence, i.e., the more different their indices are, the fewer points they have in common. The path of the cameras can be open or closed. In the case of a closed path, we expect that the last image is similar to the first image. Since assumption 1 often is realized by a single camera that is moving along the path, the reconstruction approach described here is often referred to *structure from motion*, or *SfM* for short. The term structure from motion is, however, not restricted to the case of a moving camera and is sometimes also applied to more general cases, where a common scene is depicted by multiple cameras from various viewpoints. In the literature, the same concept is also referred to as *structure and motion*.

If the 3D points do not have the same positions in 3D space when they are depicted in the different images, the reconstruction problem becomes much more complicated. The second assumption solves this issue, since it introduce a set of constraints for observations of corresponding points in several images; they are the projections of a point that has *one and the same 3D position*, but in different camera views.

The third assumption implies that we can do similarity reconstruction of 3D points. The origin and orientation of the coordinate system of the 3D space may be arbitrary defined, and typically the world coordinate system is aligned with the camera coordinate system of the first camera in the path. The benefit of this assumption is that it allows us to apply techniques based on calibrated epipolar geometry, defined by the essential matrix and normalized cameras. A consequence of this approach is that no information about the absolute translation between pairs of cameras can be given. Hence, we can only produce a similarity reconstruction, with an unknown scaling of the resulting points relative to a Euclidean reconstruction.

From all the m images we extract interest points, using any method that is suitable for this purpose, see Section 12.3.1. A typical situation is that we detect interest points in the first image, and then track them along the image sequence. This approach has the advantage of producing explicit correspondences between points that can stretch over multiple images. At suitable intervals, we can detect novel observations and start to track them, replacing points that have been lost during the tracking. Even though tracking is an efficient to establish correspondences over several images, most tracking methods use only the visual appearance of image regions in a set of images, and ignore the geometric constraint imposed by epipolar geometry. Tracking based correspondences are therefore still referred to as putative. An alternative to tracking is to generate putative correspondences by applying some type of matching of the visual appearance of a neighborhood around each image point, as described in Section 17.4.2. In particular, this applies to the case when the baseline between subsequent image views is too large to allow efficient tracking between the images.

21.2 Reconstruction from normalized cameras

Since we assume that the internal parameters \mathbf{K} are known for the camera that generate a particular image, we can transform pixel coordinates \mathbf{y}_p into C-normalized image coordinates in accordance with:

$$\mathbf{y}_n \sim \mathbf{K}^{-1} \mathbf{y}_p. \quad (21.1)$$

In the rest of this chapter, image points \mathbf{y} are assumed to be C-normalized, unless otherwise stated.

Given a set of interest points in each of the m images, and putative correspondences between these points, how can 3D reconstruction be done? A straightforward idea is to refine these correspondences using robust estimation of the epipolar geometry. Between pairs of images, I_k and I_l , the epipolar geometry can be estimated, as described in Section 17.5.2. Since we are using C-normalized image coordinates, the result is an estimate of the essential matrix \mathbf{E}_{kl} . We also get a consensus set C of correspondences. They are consistent with \mathbf{E}_{kl} , and are therefore more reliable than the initial putative correspondences.

From \mathbf{E}_{kl} , a one correspondence in C , we can determine the relative the relative rotation \mathbf{R}_{kl} and translation $\hat{\mathbf{t}}_{kl}$ between the two cameras, using Algorithm 10.3 on page 159. The rotation and the direction of the translation are unique, but the length of the translation cannot be determined. Hence, the translation is represented by the unit vector $\hat{\mathbf{t}}_{kl}$ that points in the direction of translation. This type of estimation of the relative camera pose can be applied to each pair of neighboring cameras in the sequence.

As a result of these operations, we now have a rough idea about the relative rigid transformation between pairs of cameras in the sequence. But since each relative translation can only be determined by its direction, the absolute poses of the cameras are still not well-defined at this point.

21.2.1 Accumulative reconstruction from triplets of normalized cameras

One way to deal with this ambiguity is to combine estimates of the relative poses among more than two cameras. For example, consider the three images I_1, I_2, I_3 . From these images, we can robustly estimate three essential matrices, $\mathbf{E}_{12}, \mathbf{E}_{23}$, and \mathbf{E}_{31} , and also three sets of correspondences that relate points in pairs of images. From the essential matrices, three sets of rigid transformations can be determined: $(\mathbf{R}_{12}, \hat{\mathbf{t}}_{12})$, $(\mathbf{R}_{23}, \hat{\mathbf{t}}_{23})$, $(\mathbf{R}_{31}, \hat{\mathbf{t}}_{31})$. As before, only the direction of each translation can be determined. Furthermore, an important observation to be made here is that each of these three rigid transformation are not absolute transformations that refer to a common world coordinate system. Instead they are relative transformations that transform the coordinate system of one camera to another camera. With this information at hand, we expect to see certain types of consistencies:

1. The rotation from the first to the second camera, and from the second to the third, and then from the third back to the first, should result in an identity transformation:

$$\mathbf{R}_{12}\mathbf{R}_{23}\mathbf{R}_{31} = \mathbf{I}. \quad (21.2)$$

2. Similarly, the translation from the first to the second camera, and from the second to the third, and then from the third back to the first, should result in a zero translation:

$$\bar{\mathbf{t}}_{12} + \mathbf{R}_{23}^\top \bar{\mathbf{t}}_{23} + \mathbf{R}_{12} \bar{\mathbf{t}}_{31} = \mathbf{0}. \quad (21.3)$$

Notice that we need to apply rotations in this expression to compensate for the fact that each translation vector $\bar{\mathbf{t}}_{12}, \bar{\mathbf{t}}_{23}, \bar{\mathbf{t}}_{31}$ refers to its own camera centered coordinate system, related to camera 1, 2, and 3, respectively.

3. If a point in image 1 corresponds to some other point in image 2, and that second point corresponds to a third point in image 3, then we expect that there is a correspondence also between the first and the third point.

The fact that we have estimated both poses and correspondences from noisy data implies that these three consistencies in practice never are exact. At best, these equalities are approximately correct. Within this approximation, however, the second consistency implies that we can determine the lengths of the translations, at least on a relative scale. We set

$$\bar{\mathbf{t}}_{12} = \lambda_{12}\hat{\mathbf{t}}_{12}, \quad \bar{\mathbf{t}}_{23} = \lambda_{23}\hat{\mathbf{t}}_{23}, \quad \bar{\mathbf{t}}_{31} = \lambda_{31}\hat{\mathbf{t}}_{31}, \quad (21.4)$$

where $\lambda_{12}, \lambda_{23}, \lambda_{31}$ are the absolute lengths of the translations of each of the three relative camera poses. Inserted into Equation (21.3), we get

$$\lambda_{12}\hat{\mathbf{t}}_{12} + \lambda_{23}\mathbf{R}_{23}^\top \hat{\mathbf{t}}_{23} + \lambda_{31}\mathbf{R}_{12}\hat{\mathbf{t}}_{31} = \mathbf{0}. \quad (21.5)$$

If we set $\lambda_{12} = 1$, we can then determine λ_{23} and λ_{31} by solving the three equations in Equation (21.5). Due to the noise in the data, we cannot expect to find an exact solution, and in this case we can instead determine a least squares estimate of λ_{23} and λ_{31} , which minimizes

$$\left\| \hat{\mathbf{t}}_{12} + \lambda_{23}\mathbf{R}_{23}^\top \hat{\mathbf{t}}_{23} + \lambda_{31}\mathbf{R}_{12}\hat{\mathbf{t}}_{31} \right\|^2. \quad (21.6)$$

The result is that we can determine the length of the translation $\bar{\mathbf{t}}_{23} = \lambda_{23}\hat{\mathbf{t}}_{23}$, measured in units defined by $\|\bar{\mathbf{t}}_{12}\| = 1$.

This type of computation can be applied on also the images I_2, I_3, I_4 where, in addition to the previous information, we now also obtain the camera poses $(\mathbf{R}_{34}, \hat{\mathbf{t}}_{34})$ and $(\mathbf{R}_{42}, \hat{\mathbf{t}}_{42})$. Since we already have an estimate for $\bar{\mathbf{t}}_{23}$ from before, we can now determine $\bar{\mathbf{t}}_{34} = \lambda_{23}\hat{\mathbf{t}}_{34}$ by minimizing

$$\left\| \bar{\mathbf{t}}_{23} + \lambda_{34}\mathbf{R}_{34}^\top \hat{\mathbf{t}}_{34} + \lambda_{42}\mathbf{R}_{23}\hat{\mathbf{t}}_{42} \right\|^2 \quad (21.7)$$

over λ_{34} and λ_{42} . Consequently, by estimating the relative poses between all triplets of cameras along the path, we can determine the absolute pose of each cameras, relative to a common 3D coordinate system. The length unit in this coordinate system is defined from $\|\bar{\mathbf{t}}_{12}\| = 1$, and we can iteratively compute the absolute pose of camera $\mathbf{C}_k \sim (\mathbf{R}_k | \bar{\mathbf{t}}_k)$ as

$$\mathbf{R}_k = \mathbf{R}_{(k-1)k}^\top \mathbf{R}_{k-1}, \quad \text{and} \quad \bar{\mathbf{t}}_k = \bar{\mathbf{t}}_{k-1} + \mathbf{R}_{(k-1)k}^\top \bar{\mathbf{t}}_{(k-1)k}. \quad (21.8)$$

We can set $\mathbf{R}_1 = \mathbf{I}$ and $\bar{\mathbf{t}}_1 = \mathbf{0}$, which implies that the world coordinate system refers to the camera coordinate system of the first camera.

Once the absolute camera poses are determined for $k = 1, \dots, m$, we can triangulate the 3D points that correspond to the observations in the m images. If a point is visible in more than two images, it can be triangulate based on its image coordinates in all the images where it is observed, for example using Algorithm 16.2 on page 278.

21.2.2 Analysis

This approach, where estimates of the relative camera poses are accumulated along the path, *is fragile*. Computations made by estimation from noisy data, leading to some amount of uncertainty in each camera pose. But since these camera poses are determined in accumulative fashion, by estimating relative camera poses from one view to the next, this uncertainty grows along the path. Eventually this will lead to significant errors in the absolute camera poses of the later images that are processed.

The errors in the absolute camera poses in turn lead to errors in the positions of the 3D points that we want to reconstruct. As a consequence, this accumulative approach is too simple to be of practical use. Both camera poses and coordinates of the 3D points are likely to be useless, even for very small levels of measurement noise.

21.1 Structure from motion based on the accumulative method in Section 21.2.1, which only establishes consistency between pairs of views, does not work in practice.

If we carry the analysis of the proposed method somewhat further, it clear that we have divided the reconstruction into two separate steps: first we determine the absolute poses of all cameras, and then we determine the 3D points by triangulation. In this way, we have no control of whether the estimates of absolute poses are consistent with all the reconstructed 3D points. In particular, no estimate of relative pose is based on observed points in more than two consecutive images. What is needed is a method that combines the estimation of the camera poses with the 3D reconstruction so that there is an optimal consistency between the poses and the 3D points. This is the basic idea of bundle adjustment, presented in the next section.

21.3 Bundle adjustment

Let us formulate the reconstruction problem in a less fragile way. We will do this by presenting an estimation problem in which both camera poses and 3D points are estimated simultaneously. As we will see, this estimation problem is relatively straight-forward to formulate, but it also corresponds to a non-linear optimization problem. The tricky part is then how to solve it. There are two key issues: how to find a useful initial solution, and how to deal with outliers. In this section, we will look at the principal formulation of this problem, and defer the tricky parts of its solution to Section 21.4.

We have m images, corresponding to m normalized camera matrices

$$\mathbf{C}_k \sim (\mathbf{R}_k | \bar{\mathbf{t}}_k), \quad k = 1, \dots, m. \quad (21.9)$$

Initially, all absolute camera poses $(\mathbf{R}_i, \bar{\mathbf{t}}_i)$ are unknown. We also have the C-normalized image coordinates

$$\mathbf{y}_{kj} \sim \mathbf{C}_k \mathbf{x}_j, \quad k = 1, \dots, m, \quad j = 1, \dots, p. \quad (21.10)$$

Here, \mathbf{x}_j are homogeneous coordinates of points in a set of p 3D points. You can think of this set as all 3D points that pertains to this problem, corresponding to any interest points that have been detected in the images. Initially, these 3D points are unknown, too.

Equation (21.10) implies that every 3D point is visible in every image, which is not correct. However, we can formulate an error function as

$$\epsilon_{BA} = \sum_{k=1}^m \sum_{j=1}^p w_{kj} d_{PP}(\mathbf{y}_{kj}, (\mathbf{R}_k | \bar{\mathbf{t}}_k) \mathbf{x}_j)^2. \quad (21.11)$$

Here, d_{PP} is the distance function between two image points in Equation (3.31), representing a *reprojection error* between image point \mathbf{y}_{ij} and 3D point \mathbf{x}_j . Furthermore, w is a *visibility function*, defined as

$$w_{kj} = \begin{cases} 1 & \text{if 3D point } \mathbf{x}_j \text{ is visible in image } k, \\ 0 & \text{otherwise,} \end{cases} \quad (21.12)$$

which implies that ϵ_{BA} only cares about reprojection errors from visible image points. Initially, the visibility function is not known.

The function w_{ij} provides an explicit representation of which 3D points are visible in a particular image. In practice, this information can instead be formulated implicitly, as associations between image points \mathbf{y}_{ij} and 3D points \mathbf{x}_j . In what follows, we will therefore use an equivalent but more practical formulation of ε_{BA} according to

$$\varepsilon_{\text{BA}} = \sum_{\mathbf{y}_{kj} \in T_{\text{obs}}} d_{\text{PP}}(\mathbf{y}_{kj}, (\mathbf{R}_k | \bar{\mathbf{t}}_k) \mathbf{x}_j)^2. \quad (21.13)$$

The notation $\mathbf{y}_{kj} \in T_{\text{obs}}$ is a shorthand to denote that the inner summation is made only over points \mathbf{y}_{kj} in image I_k , which can be associated with a 3D point \mathbf{x}_j . These observations will be collected in the set T_{obs} , together with their associations to 3D points. With this formulation, there is no need for the visibility function w_{ij} , and the summation is only made over points that actually contribute with well-defined reprojection errors to ε_{BA} . Initially, associations between image points and 3D points are unknown.

We are here dealing with data in the form of image coordinates in multiple images, and we know that they are perturbed by measurement noise of some magnitude η . Since the error function ε_{BA} is formulated as sum of squared reprojection errors, its minimization delivers an ML-estimate of the 3D points, based on the assumptions described in Section 12.6.

The reconstruction problem that we discuss here implies finding the global minimum of ε_{BA} over all the p 3D points \mathbf{x}_j and the m camera poses $(\mathbf{R}_k, \bar{\mathbf{t}}_k)$. In many applications, the interesting result is the set of 3D points, which means that the camera poses act as auxiliary variables in the optimization. However, if we are interested in determining the camera path, so called *ego-motion estimation* or *visual odometry*, we can use the same estimation method, but now with the 3D points as auxiliary variables.

The minimization of ε_{BA} is called *bundle adjustment*, and amounts to a highly non-linear optimization problem, which has to be solved by iterative methods. The “bundles” refer to the projection lines generated by the 3D points when projected onto the different image planes where they are visible. Minimization of ε_{BA} implies that we adjust these bundles, by modifying both the 3D points and the camera poses, such that the reprojected image points end up as close as possible to the observed interest points. The bundle adjustment problem was first presented and analyzed by Brown [7].

Since bundle adjustment relies on non-linear optimization of ε_{BA} , it requires an initial solution: initial camera poses and positions of the 3D points. In principle, we could use the absolute camera poses and 3D points that are produced by the accumulative method presented in Section 21.2.1. If this initial solution is sufficiently close to the global optimum, it will generate a reliable solution in a reasonable number of iterations. In general, however, there is no guarantee for this and, as a result, we may have to apply a large number of iterations in order to make the optimization converge. And even then, it may not converge to the global optimum since ε_{BA} in general has an abundance of local minima. A better approach is to use an incremental strategy to the initialization of the bundle adjustment. This method is presented in the next section.

21.4 Incremental bundle adjustment

Instead of trying to minimize ε_{BA} in Equation (21.13) directly, over all m images, we will partition the problem into smaller sub-problems. Each sub-problem makes a bundle adjustment on its own parameters. These sub-problems are then combined or extended, eventually including all the m images. How this is done can be formulated in many different ways, and this section presents one specific approach. To a large extent, it is inspired by [59] and is here referred to as *incremental bundle adjustment*. It starts with only two images, and first minimizes ε_{BA} over the data pertaining to these two images. This produce an initial solution, valid for the two camera poses and for the 3D points that are visible in the two images.

The remaining images are then added, one at a time, to the problem. Each time a new image is added, we get one more camera pose, and typically some additional 3D points that now become visible in (at least) two images. With these new free parameters added to the problem, we minimize ε_{BA} over all current poses and 3D points. Eventually, all m images have been included into the problem, and we are then minimizing ε_{BA} over all cameras and all 3D points. At this point, a relative accurate initial solution has already been provided from the earlier steps of the optimization.

Based on this idea, we can think of ε_{BA} a “partial” error function, which only considers reprojection errors in the l first images. We can still use the formulation of ε_{BA} in Equation (21.13), but with the modification that the set T_{obs} grows incrementally. Each time a new image I_k is added to the problem, image points \mathbf{y}_{kj} is added to T_{obs} , but only if they can be associated with 3D points \mathbf{x}_j .

Once we have minimized ε_{BA} over the l first camera poses and the 3D points that can be seen in at least two of the first l images, we add image I_{l+1} to the minimization problem and minimize ε_{BA} over the $l+1$ first camera poses and the corresponding 3D points. Eventually, we add the last of the m images, and as a consequence, the error function ε_{BA} in Equation (21.13) then corresponds to the original bundle adjustment problem, formulated in Section 21.3.

To get this incremental approach operational, we need to understand how to initialize the bundle adjustment, for example by considering only the first two images, and also look more in detail on how to extend the problem, from l to $l+1$ images. We will do this in the following sections, where we also describe the computational steps that are performed in each iteration more in detail.

21.4.1 Bookkeeping

Before going into the details, we need to describe the data that the method maintains to perform the computations. In principle, the implementation and organization of this data can be made in many different ways, but there are three key classes, or types of data, which are involved.

Observations

Each image contains a set of interest points, and these are sometimes also referred to as observations. In this presentation, we will give the concept *observation* a more specific meaning. It is associated not only to an image, but an image for which the camera pose has been determined. Furthermore, an observation is always associated with a corresponding 3D point.

You can think of an observation as something that initially is just an interest point in a specific image, represented by its Cartesian, or homogeneous, coordinates. These coordinates should be C-normalized. By processing the interest points of a specific image, together with points in other images, it will eventually be possible to estimate both the camera pose of image in which the points are found, as well as the corresponding 3D points. The interest point then becomes an observation, and is added to a table dedicated to observations, T_{obs} , which is kept to support further computations.

Views

A view is a combination of an image and the camera pose associated with the image. A view is also associated with the set of observations that pertain to this image. As soon the image points have been processed to estimate the camera pose of an image I_k , the image becomes a view, which stores the associated $(\mathbf{R}_k, \bar{\mathbf{t}}_k)$, as well as references to all observations that have emerged for the corresponding image. It is then added to a dedicated table of views, T_{views} , which is kept to support further processing.

3D points

A new 3D point is created by triangulation, from points in two or more images. Basically, a 3D point stores the position of the point, represented by its Cartesian, or homogeneous, coordinates. An important issue is that all 3D coordinates refer to *one and the same* coordinate system. It is not important exactly how this coordinate system is chosen, but a common approach is to use the coordinate system of the first camera along the path.

Apart of its coordinates, a 3D point is also associated with all corresponding observations, i.e., all image points that are projections of this 3D point. This does not mean that these observations are exact projections of the 3D point: we have to accept some amount of measurement noise. Once a new 3D point has been triangulated, it is added to a dedicated table for these points, T_{points} , which is kept to support further processing.

Tables

In summary, we need three tables for the bookkeeping in this method: T_{views} which contains the views that have been processed so far, T_{obs} which contains all observations that pertain to these views, and T_{points} which contains all 3D points that have been triangulated so far. We refer to these data structures as *bookkeeping tables*.

Each time a bundle adjustment is made, the corresponding computations use data that comes more or less exclusively from these bookkeeping tables. There is no need for the visibility function w_{kj} , defined in Equation (21.12),

it is implicitly represented by the bookkeeping tables. ϵ_{BA} adds squared reprojection errors of all 3D points that are visible in any of the images processed so far. Another way to put this is: it adds squared reconstruction errors of all image points for which there is a corresponding 3D point. These image points are exactly what here are referred to as observations, the entries of T_{obs} . Consequently, each time bundle adjustment is made, after adding image l , the error function ϵ_{BA} can be specified by iterating over the entries of T_{obs} .

We refer to this data as tables, but this does not suggest that they have to be implemented in any particular way. But they have to be dynamic: it must be possible to add new entries as the algorithm progresses, while some entries instead are removed when deemed as outliers. Depending on the data types supported by the programming language used, these tables can be implemented as dynamic arrays, as sets, or as linked lists. References from different types of data, e.g., between observations and views, can be in the form of indices to arrays, or as pointers (even smart pointers). The entries of these tables are formally unordered, but may be ordered as elements of arrays, where the positions reflect the order in which they were added.

Adding new entries

The presentation made here assumes that new entries are added to the bookkeeping tables as follows.

- A new entry can be added to T_{views} only when its corresponding camera pose is determined. We will use `addView(image, pose)` to denote a function that creates a new entry in T_{views} , referring to a specific image and camera pose. The function returns this new entry, which has an empty table of observations associated with it.
- A new entry is added to T_{points} only when a new 3D point has been computed using triangulation from image points. We will use `addPoint(coord)` to denote a function that creates a new entry in T_{points} , which stores the given 3D coordinate. The function returns this new entry, which has an empty table of observations associated with it.
- A new entry is added to T_{obs} only after its corresponding view and 3D point exist in T_{views} and T_{points} , respectively. We will use `addObs(coord, view, point)` to denote a function that creates a new entry in T_{obs} , which stores the given image coordinate and also establishes references between the new observation and its camera view as well as its corresponding 3D point. These references are bidirectional: the new entry is added to the view's table of observations as well as to the 3D point's table of observations.

21.4.2 Initialization and first bundle adjustment

The initial bundle adjustment is based on only two images, here referred to as I_1 and I_2 . These image can be the first two from the camera path described in Section 21.1, but they can also be chosen in accordance to the principles outlined in relation to Figure 21.1.

The partial error function ϵ_{BA} initially contains the reprojection errors for the 3D points that are visible only in the first two images. In fact, minimizing ϵ_{BA} at this point is equivalent to the gold standard estimation of the epipolar geometry between the first two images, described in Section 16.2.4. To carry out this step, the bookkeeping tables need to be initialized with data that refers to I_1 and I_2 . There should be initial estimates of the two camera poses, of 3D points, and of observations that are approximate projections of the 3D points. This information is then refined using bundle adjustment.

We assume that putative correspondences between I_1 and I_2 exists. From these we apply robust estimation of the epipolar geometry between the first two image, as described in Section 17.5.2. Since C-normalized coordinates are used, the result is an essential matrix \mathbf{E} , together with a consensus set C . From \mathbf{E} in combination and a pair of correspondences in C , a relative camera pose $(\mathbf{R}, \hat{\mathbf{t}})$ can be determined with Algorithm 10.3 on page 159.

The translation component $\hat{\mathbf{t}}$ has an undetermined scale, which is the root of the fact that we will be able to only make a similarity reconstruction of 3D structure. As simple approach is to choose a translation length of 1 unit, i.e., $\bar{\mathbf{t}} = \hat{\mathbf{t}}$. This allows us to set the camera pose of I_1 as $\mathbf{C}_1 = (\mathbf{I} | \mathbf{0})$, and that of I_2 as $\mathbf{C}_2 = (\mathbf{R} | \hat{\mathbf{t}})$, as the initial entries of T_{views} .

From the correspondences that remain in the consensus set C it is now possible to triangulate a set of corresponding 3D points, the initial entries of T_{points} . Finally, all image points in C can now be added to T_{obs} , which sets up the relations between them and their views and their 3D points. This initialization of the bookkeeping tables is presented in Algorithm 21.1.

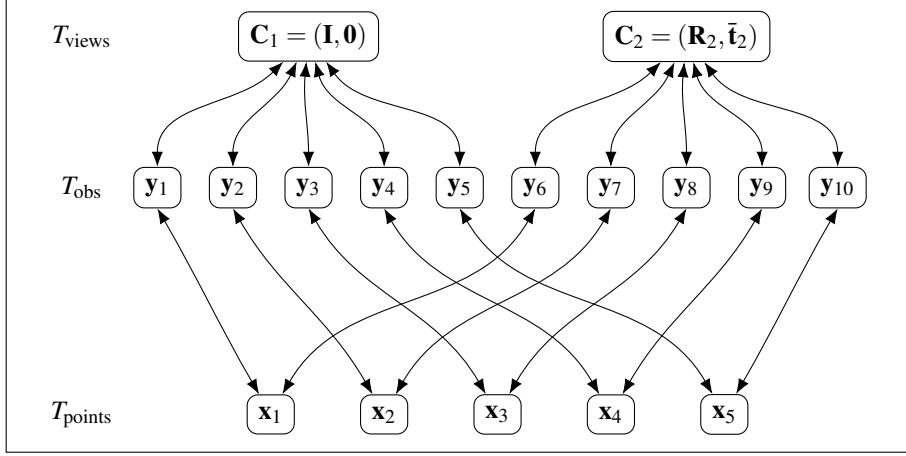


Figure 21.2: An example that shows the result of initialization of the bookkeeping tables from images I_1 and I_2 . The tables are here represented as lists, with double links between associated entries. In this example, T_{views} contains two initial camera poses, corresponding to I_1 and I_2 . T_{obs} contains 10 observations, 5 from each image. T_{points} contains 3D points corresponding to the 5 pairs of observations.

The result of this initial step, the three bookkeeping tables, are illustrated by an example in Figure 21.2. The tables are here represented as array, where each entry in a table has a unique index. References between the tables are represented by these indices. T_{views} contains two camera poses, corresponding to I_1 and I_2 , as well as references to the observations related to each view. T_{obs} contains 10 observations, 5 from each view. T_{points} contains 3D points corresponding to the 5 pairs of observations.

Based on this initial data, we can now minimize ϵ_{BA} . This refines the positions of the 3D points in T_{points} as well as the relative pose of the first two cameras in T_{views} . By minimizing ϵ_{BA} , we obtain an ML-estimate of the camera poses for the first two images, and of the 3D points corresponding to the observations in these images. So far, the SfM problem has taken into account information only in the first two images. In the next step, we will add an image to the optimization problem.

21.4.3 Adding an image

In the preceding section we discussed the initialization of the incremental SfM problem, where bundle adjustment is made first on information extracted only from the first two images in the sequence. Now we want to add a third image to the problem. In general, we can assume that the bookkeeping tables have been refined from data extracted from the $l - 1$ first images, and we want to extend the problem to l images.

In principle, adding the new image implies the same steps as for the initial part. We need to find a reasonable estimate of the camera pose for I_l , and add this as a new entry in T_{views} . Some 3D points may be triangulated, and added to T_{points} . Finally, new observations that are found in I_l are added to T_{obs} . This time, some bookkeeping is already available, which allows us to modify the processing to use this data.

Adding a view

The camera pose of image I_l can be determined based on the essential matrix between images I_{l-1} and I_l , in a similar way as in Section 21.4.2. In this case, robust estimation of \mathbf{E} is done with a minimum of 5 correspondences in the trial set of each RANSAC iteration. As an alternative, we can instead use the fact that (normally) a subset of putative correspondences between images I_{l-1} and I_l already have established 3D points in T_{points} . Using the image points in I_l from this subset and the corresponding 3D points from T_{points} , we can then do robust estimation of the l -th camera pose directly using robust PnP, Algorithm 17.7 on page 311.

The P3P approach has the advantage that each RANSAC iteration requires only 3 correspondences in the trial set. This means that the chance of selecting only inliers increases significantly compared to the former approach, thereby allowing a lower number of RANSAC iterations for the same success probability. Consequently, the P3P

Algorithm 21.1: Initialization of an incremental approach to SfM.

Input: Two initial images I_1 and I_2 , with putative correspondences between the images.
Input: Various parameters that control the processing in the different steps.
Output: Initial bookkeeping table for the views, T_{views} .
Output: Initial bookkeeping table for the 3D points, T_{points} .
Output: Initial bookkeeping table for the observations, T_{obs} .

```

1 Set  $T_{\text{points}} = \emptyset$ ,  $T_{\text{views}} = \emptyset$ ,  $T_{\text{obs}} = \emptyset$  /* Start with empty bookkeeping tables */.
2 Set  $U$  = putative correspondences between image  $I_1$  and  $I_2$ .
3 Robustly estimate the essential matrix  $\mathbf{E}$  from  $U$ , Algorithm 16.7.
4 Produces a consensus set  $C$  of correspondences that are consistent with  $\mathbf{E}$ .
5 From  $\mathbf{E}$  and a pair of correspondences in  $C$ , determine the relative pose  $(\mathbf{R}, \mathbf{t})$  between camera 1 and 2.
6 Use Algorithm 10.3 on page 159.
7 Set camera pose  $\mathbf{C}_1 = (\mathbf{I}, \mathbf{0})$  for image  $I_1$  and  $\mathbf{C}_2 = (\mathbf{R}, \mathbf{t})$  for image  $I_2$  in  $T_{\text{views}}$ .
8  $\text{view1} = \text{addView}(I_1, \mathbf{C}_1)$ ;  $\text{view2} = \text{addView}(I_2, \mathbf{C}_1)$ .
9 foreach correspondence  $(y_1, y_2) \in \text{consensus set } C$  do
10   Triangulate 3D point  $\mathbf{x}$  from  $y_1, y_2, \mathbf{C}_1$  and  $\mathbf{C}_2$ .
11    $\text{point} = \text{addPoint}(\mathbf{x})$ .
12    $\text{addObs}(y_1, \text{view1}, \text{point})$ ;  $\text{addObs}(y_2, \text{view2}, \text{point})$ .
13 end
```

approach is proposed here as the more efficient alternative. The result of the robust pose estimation is also a consensus set C of image points in I_l and 3D points in T_{points} consistent with the estimated pose.

In the end, the estimated camera pose is added as a new view to T_{views} for image I_l . For each correspondence in C , an image point in I_l and a 3D point in T_{points} , the image point is added as a new observation in T_{obs} , referring to the new view. The computations for adding a new view to T_{views} are summarized in Algorithm 21.2.

An example of the result of applying Algorithm 21.2 to the bookkeeping is presented in Figure 21.3, where the tables in Figure 21.2 have been updated. As seen, three new observations from the third view are added to T_{obs} . Since they refer to existing 3D points, no new 3D points are added to T_{points} . In the next step, we will extend also T_{points} with new 3D points that have not been visible before.

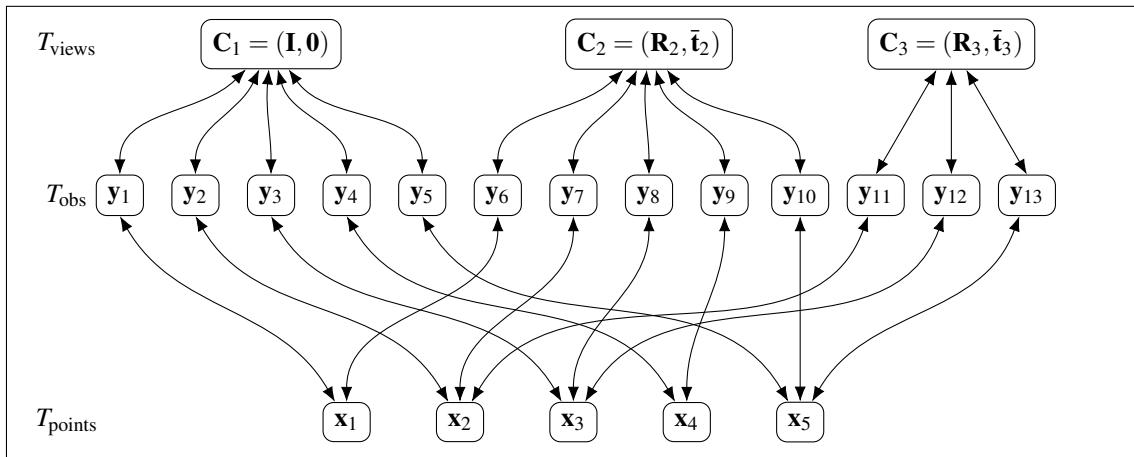


Figure 21.3: An example that shows the result of adding a third view to the bookkeeping tables in Figure 21.2. Three new observations from the third view are added to T_{obs} . But since they refer to existing 3D points, no new 3D points have been added to T_{points} .

Algorithm 21.2: Add new view to T_{views} .

Input: Images I_{l-1} and I_l , with putative correspondences between the two images.
Input: Current bookkeeping table for the views, T_{views} .
Input: Current bookkeeping table for the observations, T_{obs} .
Input: Various parameters that control the processing in the different steps.
Output: Updated bookkeeping table for the views, T_{views} .
Output: Updated bookkeeping table for the observations, T_{obs} .
Output: A , a set of putative correspondences between the two images so far without 3D points.

```

1 Set  $U$  = putative correspondences between image  $I_{l-1}$  and  $I_l$ .
2 Set  $D = \emptyset$  and  $A = \emptyset$ .
3 foreach putative correspondence  $(y_1, y_2)$  in  $U$  do
4   if  $y_1 \in T_{\text{obs}}$  then
5     There is a corresponding 3D point  $x$  in  $T_{\text{points}}$ . Add  $(y_2, x)$  to  $D$ .
6   else add  $(y_1, y_2)$  to  $A$ , a subset of  $U$  without 3D correspondence.
7 end
8 Robustly estimate camera pose  $(R_l, t_l)$  from  $D$  using PnP, Algorithm 17.7 on page 311.
9   Produces a consensus set  $C$  of correspondences that agree with the estimated camera pose.
10 Set camera pose  $C_l = (R_l, t_l)$  for image  $I_l$ .
11 view_l = addView( $I_l$ ,  $C_l$ ). /* Add new view to  $T_{\text{views}}$  */
12 foreach  $(y_2, x) \in C$  do addObs( $y_2$ , view_l,  $x$ ) /* Add all image points in  $C$  to  $T_{\text{obs}}$  */

```

Adding new 3D points

Once an initial estimate of the camera pose for image I_l has been determined by the previous step, we can use the poses of I_{l-1} and I_l to triangulate all putative correspondences between these two images that have not already been investigated. These correspondences are collected in Algorithm 21.2, in the form of set A .

Before further processing, these correspondences are first matched against the epipolar geometry between the two images. Only correspondences that match the epipolar constraint within a reasonable limit, given by the measurement noise, should be considered as reliable candidates for correspondence. Those that satisfy this criterion are then triangulated to give new 3D points. These new 3D points are added to T_{points} , after which also the image points can be added to T_{obs} as new observations. The computations for adding new 3D points to T_{points} are summarized in Algorithm 21.3.

An example of the result of applying Algorithm 21.3 is presented in Figure 21.4, where the tables in Figure 21.3 have been updated. As seen, two new observations each from the second and third view are added to T_{obs} . The corresponding 3D points are added to T_{points} .

All new points that appear in this step are based on putative correspondences that meet the epipolar constraint. Since this condition is not sufficient to determine correspondence, some of them may still be incorrect. This problem can, to some extent, be mitigated in the next step, where potential outliers are removed.

Inlier selection

At this point in the algorithm, all three bookkeeping tables have been updated with information from the new image added to the SfM problem. In principle, the next step is to apply bundle adjustment to this data, which refines the camera poses in T_{views} and the 3D points in T_{points} .

Bundle adjustment is to estimate parameters from observed data. As mentioned in Chapter 17, the quality of the resulting parameters from any estimation procedure is to a high degree dependent on reliable data, only consisting of inliers. Even one single outlier in a large dataset can severely corrupt the result. Therefore, it is important that bundle adjustment is applied to data that has a very low probability of containing outliers.

In the previous steps, we have added new 3D points to the problem, points which have only been seen in the last two images. They are accepted as inliers only by satisfying the corresponding epipolar geometry, which itself is not a sufficient condition. Therefore, these points may not be completely trusted as inliers. To facilitate the bundle adjustment, it is therefore a good idea to not use all 3D points in T_{points} in the formulation of the error function ε_{BA} . Instead, this is done based on a selection of 3D points from T_{points} , here denoted T'_{points} .

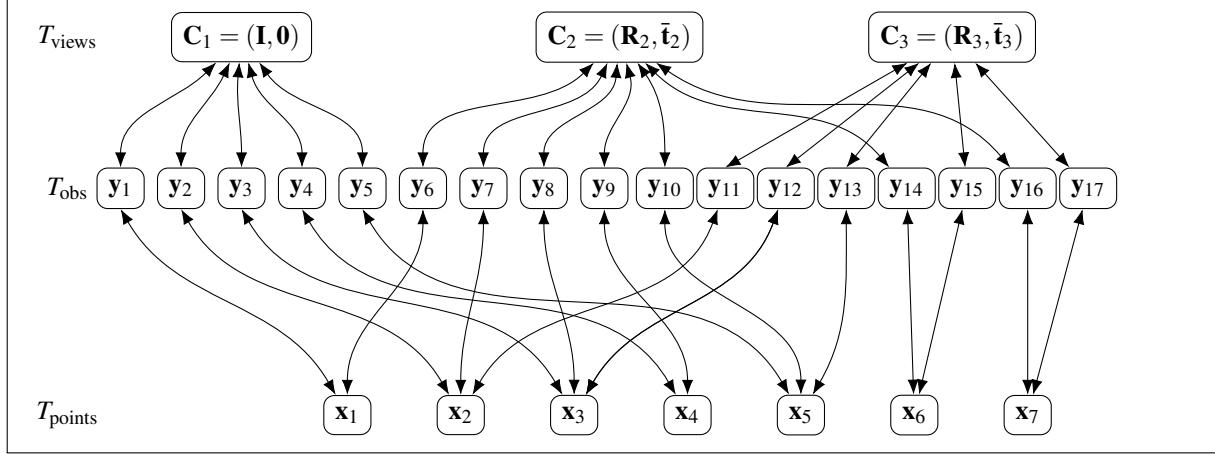


Figure 21.4: An example that shows the result of adding new points to the bookkeeping tables in Figure 21.3. Some new observations from both the second and third views have been added to T_{obs} , together with their corresponding 3D points that are added to T_{points} . T_{views} remains the same, except that references to observations have been updated.

A reasonable criterion for selection is that the 3D point has existed in T_{points} during many iterations. This excludes all points just added by Algorithm 21.3, which may be a good idea in some cases, but can also be going too far. If each putative correspondence between images has an associated measure of inlier probability, as discussed in Section 17.4.2, this measure can be used refine the selection. In this case, it makes sense to include this step already in the first iteration of this method, when the first two images are processed.

The literature offers additional criteria, but we will not go through every aspect of this issue here. In the end, this step implies that from T_{points} , we have selected a subset $T'_{\text{points}} \subset T_{\text{points}}$ that will be used to specify the error function in the next step, which is bundle adjustment.

Algorithm 21.3: Add new 3D points to the SfM problem.

Input: Set A of correspondences between I_{l-1} and I_l , still without corresponding 3D points.

Input: References view1 and view2 to views corresponding to images I_{l-1} and I_l .

Input: Camera matrices C_1 and C_2 corresponding to images I_{l-1} and I_l .

Input: Current bookkeeping table for the 3D points, T_{points} .

Input: Current bookkeeping table for the observations, T_{obs} .

Input: Various parameters that control the processing in the different steps.

Output: Updated bookkeeping table for the 3D points, T_{points} .

Output: Updated bookkeeping table for the observations, T_{obs} .

1 Compute the essential matrix E between I_{l-1} and I_l , Equation (10.53)

2 **foreach** correspondence $(y_1, y_2) \in A$ **do**

3 **if** (y_1, y_2) satisfy the epipolar constraint relative to E **then**

4 Triangulate 3D point x from y_1, y_2, C_1 and C_2 .

5 point = addPoint(x).

6 addObs(y_1 , view1, point); addObs(y_2 , view2, point).

7 **end**

8 **end**

Bundle adjustment

Given the three bookkeeping tables, now with the selection T'_{points} instead of T_{points} , we now do bundle adjustment. This corresponds to a non-linear estimation problem in which ϵ_{BA} in Equation (21.13) is minimized over the camera poses in T_{views} and the 3D points in T'_{points} . The error function is constructed by iterating over the 3D points in T'_{points} and, for each such point, iterate over the corresponding observations in T_{obs} . To each such observation, there is a corresponding term in ϵ_{BA} . This approach means that ϵ_{BA} has to be dynamically specified for each new iteration, which typically extends the error function with more terms corresponding to new observations.

Once the error function is specified, a general or dedicated non-linear optimization method is applied to it. The Levenberg-Marquardt algorithm, described in Toolbox Section 9.2.2, is a common choice for this application. The free parameters are the positions of the 3D points in T'_{points} and the camera poses in T_{views} . Initial values of these parameters are the existing values found in the tables. After the bundle adjustment step, the refined parameters have to be returned to the tables.

Re-triangulation

The bundle adjustment refines the camera poses and the 3D points, but since we only use a subset of the 3D points, in T'_{points} , not all entries in T_{points} are refined. This means that the 3D coordinates of the excluded points are not consistent with their corresponding observations and camera poses, since the latter have been adjusted.

As a consequence of this observation, we should re-triangulate all 3D points in T_{points} that are not selected as inliers in T'_{points} . For each such 3D point, the tables contain information about which are the supporting observations, and which are their associated camera poses. Based on this information, the triangulation is made in the same way as before, but now with updated camera poses.

Outlier removal

After bundle adjustment and re-triangulation, it is a good idea to review the refined data, primarily to remove potential outliers. The exact criteria that are used to designate a 3D point as an outlier may vary, but some common alternatives are presented below.

- A primary outlier criterion is to look at the reprojection error, i.e., the terms in ϵ_{BA} . An outlier typically has a large reprojection error in at least one images. Exactly what “large” means should be related to the expected measurement noise in the images. If the reprojection error is large in several images, the 3D point is likely an outlier. If the reprojection error is large only in one or a few images, it may suffice to exclude it from the inlier set, T'_{points} in the next iteration. NOTE: reprojection errors should be computed **after** the re-triangulation step.
- Another obvious outlier criterion is when the 3D point ends up *behind* any of the cameras from which it is visible. To this end, we compute the point’s position relative to the 3D coordinate system of each camera in which the point is visible, and investigate the depth coordinate. If it is negative, the point lies behind the camera and can in practice not be visible.
- Depending on the application, it may be reasonable to look at the configuration of the 3D points. If we know that they should form a cluster of neighboring points, e.g., lying on the surface of an object, any point that lies “far” from the other points is likely an outlier.
- A 3D point that makes a significant motion in position before and after the bundle adjustment step is likely an outlier.
- A 3D point that is visible only in two or a few images is likely an outlier. This criterion should be combined with the reprojection error: even if the point is visible in few images, but has a very low reprojection error in these images, it can still be an inlier.

Any 3D point classified as an outlier should be removed from T_{points} . This applies also to its corresponding observations in T_{obs} which, in turn, affects also the tables of observations for the views in T_{views} . We will use `delete(point)` to denote a function that removes the entry in T_{points} of the given 3D point, including removing all corresponding entries in T_{obs} and in T_{views} .

Iterate

Based on the preceding steps, we continue to add a new image to the bundle adjustment problem and optimize again. In each iteration, the bookkeeping tables extend by one new entry in T_{views} , and a set of new observation in T_{views} and 3D points in T_{points} . These iterations continue until all views have been processed.

Outliers are removed after the bundle adjustment step. The effect of this removal will be noticed the next time bundle adjustment is applied, when the free parameters are refined, now without the outliers. This, however, does not happen after the final iteration. It therefore makes sense, if outliers have been removed, to bundle adjust one more time after the final iteration.

Each time that we apply bundle adjustment, the initial state is expected to be relatively close to the global optimum. The “old” camera poses and 3D points should only have to be moderately adjusted when we take the new data into account. In particular, entries in the tables that have existed for several iterations should be relatively stable.

The final increment of this method, where ε_{BA} is minimized over all m images, can be seen as the actual bundle adjustment problem that we want to solve. To do this, we initiate the minimization by first minimizing ε_{BA} over $m - 1$ images, which means that the initial solution to the final problem already lies relatively close to the global solution. This is reasonable since all camera views except one, and all 3D points except those that appear in the final image, already have been optimized. Similarly, the initiation of the minimization of ε_{BA} over $m - 1$ images is found by first solving the corresponding problem over $m - 2$ images, and so on. As a consequence, the optimization progresses along a relatively “safe” trajectory in the state space of all the 3D points and camera poses.

The proposed method for incremental SfM is summarized in Algorithm 21.4. A graphical representation of the different steps of this SfM pipeline is presented in Figure 21.5. This algorithm has several parameters that need to be carefully tuned, for example there are thresholds in the two different RANSAC algorithms that are used for robust estimation of the initial epipolar geometry between I_1 and I_2 , and for the robust estimation of the camera pose of the new image I_l .

21.5 Practical issues

The basic SfM-algorithm outlined in Algorithm 21.4 is relatively simple to implement, at least when the bundle adjustment step is supported. The algorithm has a medium computational complexity. For example, the reconstruction of approximately 600 3D points in approximately 30 views takes a few minutes when implemented in a standard platform for numerical computations, and can be much faster with dedicated SfM software. In general, the algorithm also gives reasonable results.

The resulting 3D points in T_{points} form a *point-cloud*, i.e., an unordered set of 3D points. To be more useful, and perhaps give a better visual impression, we may apply further processing on this point cloud. For example, the points can be connected into a 3D surface mesh, onto which a texture can be mapped. Finally, this data can be rendered in a 3D graphics visualization platform, e.g., OpenGL. These later steps, however, are not discussed here.

In this section, we discuss some implementation issues and possible extensions of Algorithm 21.4. These may be relevant in the case of significant measurement noise, or if the underlying assumptions described in the beginning of this chapter are not completely satisfied. A deeper analysis and additional hints for improvements are presented in the extended paper by Triggs *et al.* [63], which also gives an overview of the historic development of the bundle adjustment method.

21.5.1 Parameterization of the camera poses

In the presentation of the proposed SfM-method, each time that we minimize ε_{BA} this is done over the free variables in the form of the coordinates of the 3D points \mathbf{x}_j in T_{points} , and camera poses $(\mathbf{R}_k | \bar{\mathbf{t}}_k)$ in T_{views} . From the outset, it may then seem natural that the parameterization of the poses is made in terms of $\mathbf{R}_k \in SO(3)$ and $\bar{\mathbf{t}}_k \in \mathbb{R}^3$.

As has been pointed out in Section 8.2.1, $\bar{\mathbf{t}}_k$ is not the center of camera \mathbf{C}_k . The camera center is instead located at the point with Cartesian coordinates $\bar{\mathbf{n}}_k = -\mathbf{R}^\top \bar{\mathbf{t}}_k$. Given the geometric structure of this estimation problem, what we need to determine are the camera centers, $\bar{\mathbf{n}}_k$, and the absolute rotation of each camera, \mathbf{R}_k , together with all the 3D points. During the course of the incremental bundle adjustment algorithm we expect to see only small adjustments in all these parameters. In particular, the positions of the camera centers of the “old cameras”, which

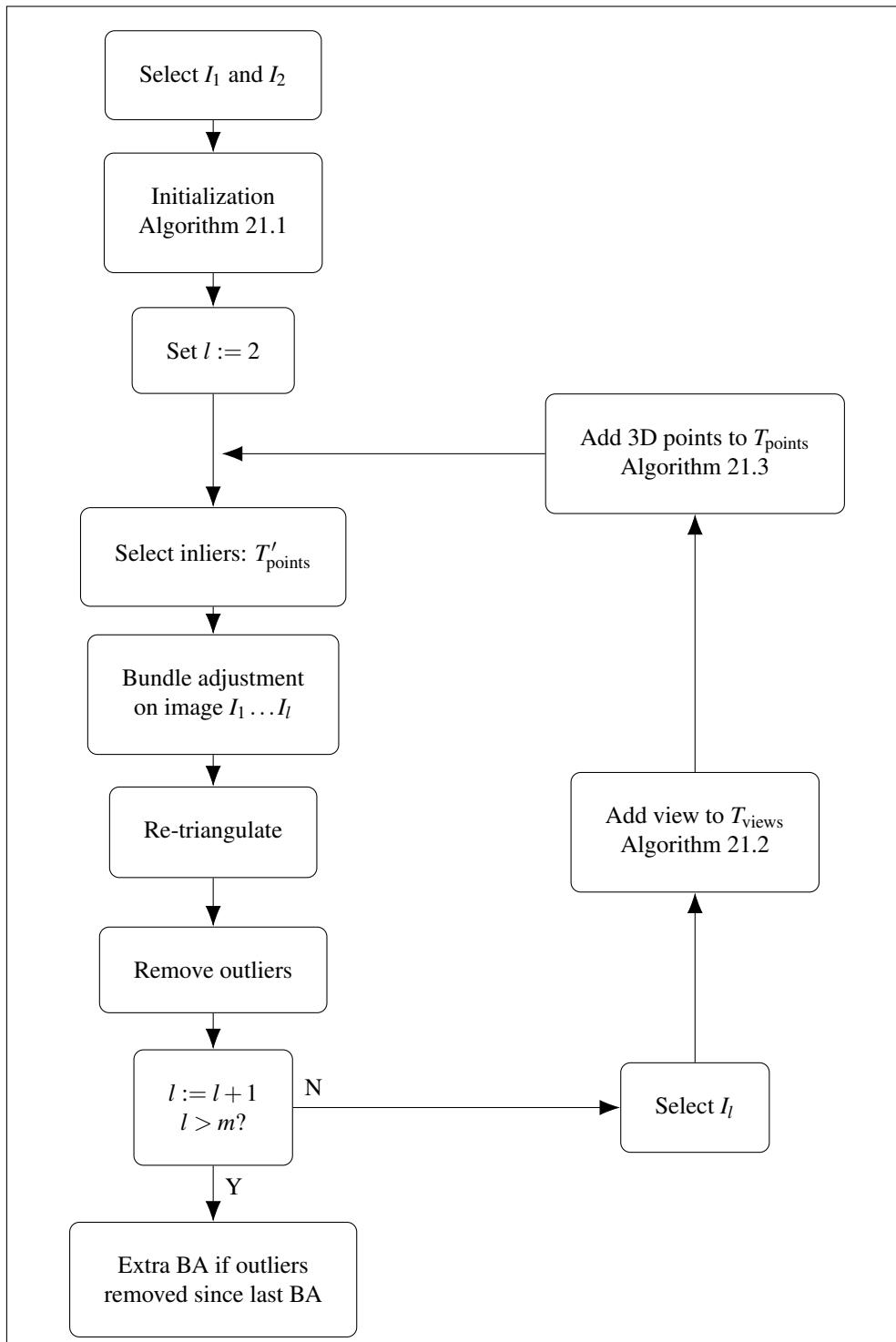


Figure 21.5: The SfM pipeline described in Algorithm 21.4.

Algorithm 21.4: An incremental approach to SfM.

```

Input: A sequence of  $m$  images  $I_k$ , with putative correspondences between consecutive pairs of images.  

    Each image point is represented in C-normalized coordinates.  

Input: Various parameters that control the processing in the different steps.  

Output: Final set of camera views in  $T_{\text{views}}$ .  

Output: Final set of 3D points in  $T_{\text{points}}$ .  

Output: Final set of observations in  $T_{\text{obs}}$ .  

1 /* Initialize */  

2 Select the first two images  $I_1$  and  $I_2$  to process.  

3 Initiate the bookkeeping tables from  $I_1$  and  $I_2$ , Algorithm 21.1.  

4 Set  $l := 2$ .  

5 /* Main loop, repeats until all images are processed. The loop breaks on line 21. */  

6 repeat  

7     Select inlier 3D points:  $T'_{\text{points}}$ .  

8     Bundle adjustment: minimize  $\varepsilon_{\text{BA}}$  over the  $l$  camera poses in  $T_{\text{views}}$  and the 3D points in  $T'_{\text{points}}$ .  

9     /* Re-triangulate excluded 3D points */  

10    foreach 3D point  $\mathbf{x} \in T_{\text{points}} \setminus T'_{\text{points}}$  do  

11        Get corresponding observations  $\mathbf{y}_1$  and  $\mathbf{y}_2$  from  $T_{\text{obs}}$  and their camera poses  $\mathbf{C}_1$  and  $\mathbf{C}_2$  from  $T_{\text{views}}$ .  

12        Triangulate  $\mathbf{x}$  from  $\mathbf{y}_1, \mathbf{y}_2, \mathbf{C}_1$ , and  $\mathbf{C}_2$ . Update 3D point in  $T_{\text{points}}$ .  

13    end  

14    /* Remove potential outliers from  $T_{\text{points}}$  after bundle adjustment */  

15    foreach 3D point  $\mathbf{x} \in T_{\text{points}}$  do  

16        if its reprojection errors in images  $I_1, \dots, I_l$  are large OR some other outlier criterion is met then  

17            delete( $\mathbf{x}$ )  

18        end  

19    end  

20    Increment to next image:  $l := l + 1$ .  

21    if  $l > m$  then break /* Break main loop when all images are processed */  

22    Select next image  $I_l$  that is added to the problem, see Section 21.5.2.  

23    Add the new view, corresponding to  $I_l$ . Algorithm 21.2.  

24    Produces a set  $A$  of image correspondences that still have no corresponding 3D points.  

25    Add new 3D points, Algorithm 21.3.  

26 until false  

27 if outliers where removed after last bundle adjustment then  

28     Bundle adjustment: minimize  $\varepsilon_{\text{BA}}$  over all camera poses and 3D points  

29 end
```

have been introduced into the optimization early on in the process, should be subject to minute adjustments when new cameras are added to the problem.

If we use the parameterization based on $(\mathbf{R}_k, \bar{\mathbf{t}}_k)$, however, the center $\bar{\mathbf{n}}_k$ of camera \mathbf{C}_k is affected both by adjustments in \mathbf{R}_k and in $\bar{\mathbf{t}}_k$. More precisely, a small change in $\bar{\mathbf{t}}_k$ corresponds to a small change in $\bar{\mathbf{n}}$, but a small change in \mathbf{R}_k may still generate a large change in $\bar{\mathbf{n}}_k$ if $\bar{\mathbf{t}}_k$ is large. Intuitively, the derivative of ε_{BA} with respect to \mathbf{R}_k is amplified by the norm of $\bar{\mathbf{t}}_k$, and the only way to keep the center approximately stable when small adjustments are made to \mathbf{R}_k is to have a relatively large adjustment in $\bar{\mathbf{t}}_k$ if $\bar{\mathbf{t}}_k$ is large. In the end, this implies that the minimization of ε_{BA} converges more slowly with this parameterization.

The natural remedy of this problem is to parameterize each normalized camera in terms of \mathbf{R}_k and $\bar{\mathbf{n}}_k$, instead of using \mathbf{R}_k and $\bar{\mathbf{t}}_k$. In accordance with Equation (8.22), we then write the normalized cameras as

$$\mathbf{C}_k \sim \mathbf{K}(\mathbf{R}_k | -\mathbf{R}_k \bar{\mathbf{n}}_k). \quad (21.14)$$

As discussed in Section 15.3, the rotations \mathbf{R}_k can conveniently be parameterized in a consistent way by means of unit quaternions. Alternatively, the axis-angle representation can be used if the singularities for rotations close

to \mathbf{I} are taken care of.

21.2 Parameterize the camera poses in terms of the position of each camera center, $\bar{\mathbf{n}}_k$, and the orientation of each camera, \mathbf{R}_k .

21.5.2 Flexible scheme for adding cameras

As explained already in the introduction of this chapter, there is a trade-off between using camera views that have a short or a large baseline. A shorter baseline means a larger set of correspondences between two images, but also gives a larger reconstruction error in 3D space. For a larger baseline, it is the other way around. This means that it may not be an ideal choice to use the first two images in a sequence for the initialization of the computations. Instead it may be a better option to choose two images that are further apart but, again, not too far apart.

Another issue in the selection of images relates to the fact that once there is a sufficient amount of relatively reliable camera poses and reconstructed 3D points, we may add two (or a few) new views for each iteration instead of only one. This means that T_{views} increases with more than one entry per iteration. This strategy may reduce the total amount of computations, since fewer bundle adjustment steps are made.

21.5.3 Image coordinate system

In this chapter, we have assumed that image points are represented in C-normalized coordinates. This is necessary for several of the computations in the proposed algorithm, e.g., for camera pose estimation using PnP and for estimation of the essential matrix. This observation, does not exclude the possibility that image points can be represented in pixels coordinates as well. Transformations from pixel coordinates to C-normalized coordinates, and vice versa, can be made easily based on Equation (21.1). This means that image points can be represented in any suitable coordinate system, as long as it is possible to transform these points to C-normalized coordinates.

Pixel coordinate may be more intuitive to use when errors are processed. Expected levels the measurement noise is typically given in a pixel-based system. This means either that noise levels must be transformed to a C-normalized system, or that image points are available in pixel coordinates when reprojection errors are computer.

Finally, pixel coordinates of image points are needed when we want to visualize results of the computations. Consequently, it is a good idea to support both pixel coordinates and C-normalized coordinates for the image points.

21.5.4 The residual vector \mathbf{r}

A closer look at the cost function ε_{BA} in Equation (21.13) shows that it can be optimized using non-linear least squares methods, as discussed in Toolbox Section 9.2 and Chapter 17. Hence, there is a corresponding residual vector \mathbf{r} , and minimizing ε_{BA} is equivalent to minimizing the norm of \mathbf{r} . For this problem, we can formulate the residual vector \mathbf{r} as

$$\mathbf{r} = \begin{pmatrix} \vdots \\ \left(u_{kj} - \frac{\mathbf{c}_{k1} \cdot \mathbf{x}_j}{\mathbf{c}_{k3} \cdot \mathbf{x}_j} \right) \\ \left(v_{kj} - \frac{\mathbf{c}_{k2} \cdot \mathbf{x}_j}{\mathbf{c}_{k3} \cdot \mathbf{x}_j} \right) \\ \vdots \end{pmatrix}, \quad \text{where } \bar{\mathbf{y}}_{kj} = (u_{kj}, v_{kj}), \quad \text{and} \quad (\mathbf{R}_k | \bar{\mathbf{t}}_k) = \begin{pmatrix} - & \mathbf{c}_{k1} & - \\ - & \mathbf{c}_{k2} & - \\ - & \mathbf{c}_{k3} & - \end{pmatrix} \in \mathbb{R}^{3 \times 4}. \quad (21.15)$$

Here, we are using the approach advocated in observation 14.6 on page 250, where the elements of \mathbf{r} are differences in vertical and horizontal coordinates, rather than Euclidean distances.

If we are using the approach suggested in Equation (21.13), where only observations that can be associated with some 3D points are included in the error function, then \mathbf{r} has $2n_{\text{obs}}$ elements, where n_{obs} is the number of observations in T_{obs} . In incremental bundle adjustment, T_{obs} grows as new images are added to the problem, and therefore \mathbf{r} grows as well.

21.5.5 Sparsity of \mathbf{J}

Another important observation to be made in relation to \mathbf{r} in Equation (21.15) is that its Jacobian matrix, \mathbf{J} , is very sparse. Each element in \mathbf{r} is a function only of *one* 3D point \mathbf{x}_j and *one* camera pose $(\mathbf{R}_k, \mathbf{\tilde{t}}_k) = (\mathbf{R}_k, -\mathbf{R}_k \mathbf{\bar{n}}_k)$. Another way to put it is: each element in \mathbf{r} depends only on a small subset of the free parameters, the other parameters do not affect that element. More precisely, each one depends on the coordinates of one particular 3D point and the pose of one particular camera. This means that when we form the derivatives of \mathbf{r} with respect to the free parameters in \mathbf{z} , which constitute the elements of \mathbf{J} , we get zero values almost everywhere except for the parameters that control the pose of camera \mathbf{C}_k and the position of \mathbf{x}_j . As a consequence, only a small fraction of the elements in \mathbf{J} need to be computed in practice, which in a significant way saves computational costs each time \mathbf{J} is determined. In particular, this is the case if \mathbf{J} is estimated by numerical approximations of derivatives.

Many software for non-linear least squares optimization offer the option to use a *Jacobian mask*. It is a fixed boolean matrix of the same size as \mathbf{J} , which makes explicit which elements in \mathbf{J} that we know are = 0, and which can be non-zero. This option should be used whenever available, as it significantly reduces the computation cost at the small effort of specifying the mask. The mask can be specified directly based on the bookkeeping tables.

The sparseness of \mathbf{J} is illustrated in Figure 21.6, left. In this simple example $p = 10$ points are observed in $m = 4$ camera views, and the rows in \mathbf{r} corresponding to $w_{kj} = 0$ have been removed. Each row in \mathbf{J} , with index i , is the gradient of element r_i in \mathbf{r} , and all elements of these gradients that are = 0 are blank in the figure, while elements which in general are non-zero are colored. Notice that the rows in \mathbf{J} come in pairs, in a similar way as the elements of \mathbf{r} , one for u and one for v .

Each camera pose is parameterized by the corresponding camera center and a unit quaternion which represents the orientation of the camera ($3 + 4 = 7$ parameters), and each 3D point is parameterized by its Cartesian coordinates relative to some specific coordinate system (3 parameters). The first $n_c = 7 \times l = 28$ columns of \mathbf{J} contain the derivatives of the elements in \mathbf{r} with respect to the external camera parameters, and the following $n_p = 3 \times p = 30$ columns of \mathbf{J} contain the derivatives with respect to the 3D coordinates. In this particular example, more than 80% of the elements in \mathbf{J} must be zero. In a more realistic example, with many more points and views, this fraction can be close to 100%.

21.3 A Jacobian mask should be used in the non-linear least squares estimation methods related to bundle adjustment problems. This gives a significant reduction in the computational cost of the bundle adjustment problem.

21.5.6 The Schur complement trick

In the end, bundle adjustment leads to a non-linear optimization of ε_{BA} , using refinement steps to improve an initial solution, see [54] Chapter 9. In each such step, we want to solve the equation $\mathbf{H}\mathbf{h} = -\mathbf{g}$, where \mathbf{H}, \mathbf{g} are the Hessian and gradient, respectively, and \mathbf{h} is the refinement of the free parameters. We have now made 2 important observations that pertain to the solution of this equation. First, the corresponding Jacobian \mathbf{J} can be seen as the concatenation of two Jacobian matrices, one that contains the derivatives with respect to the n_c camera pose parameters, and one that contains the derivatives with respect to the n_p point parameters:

$$\mathbf{J} = (\mathbf{J}_c \ \mathbf{J}_p). \quad (21.16)$$

The corresponding Hessian approximation is then given as

$$\mathbf{H} = \mathbf{J}^\top \mathbf{J} + \mu \mathbf{I} = \begin{pmatrix} \mathbf{J}_c^\top \mathbf{J}_c + \mu \mathbf{I} & \mathbf{J}_c^\top \mathbf{J}_p \\ \mathbf{J}_p^\top \mathbf{J}_c & \mathbf{J}_p^\top \mathbf{J}_p + \mu \mathbf{I} \end{pmatrix} = \begin{pmatrix} \mathbf{A} & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{D} \end{pmatrix}, \quad (21.17)$$

where $\mu = 0$ for the Gauss-Newton method. Hence, the Hessian can be characterized as a 2×2 block matrix where \mathbf{A} is a symmetric $n_c \times n_c$ matrix, \mathbf{D} is a symmetric $n_p \times n_p$ matrix, and \mathbf{B} is $n_p \times n_c$. This is illustrated in Figure 21.6, right, for the simple example discussed in Section 21.5.5. There is a great deal of structure also in \mathbf{H} : both \mathbf{A} and \mathbf{D} are block diagonal matrices and, in particular, the diagonal blocks of \mathbf{D} are of size 3×3 . This observation becomes important shortly.

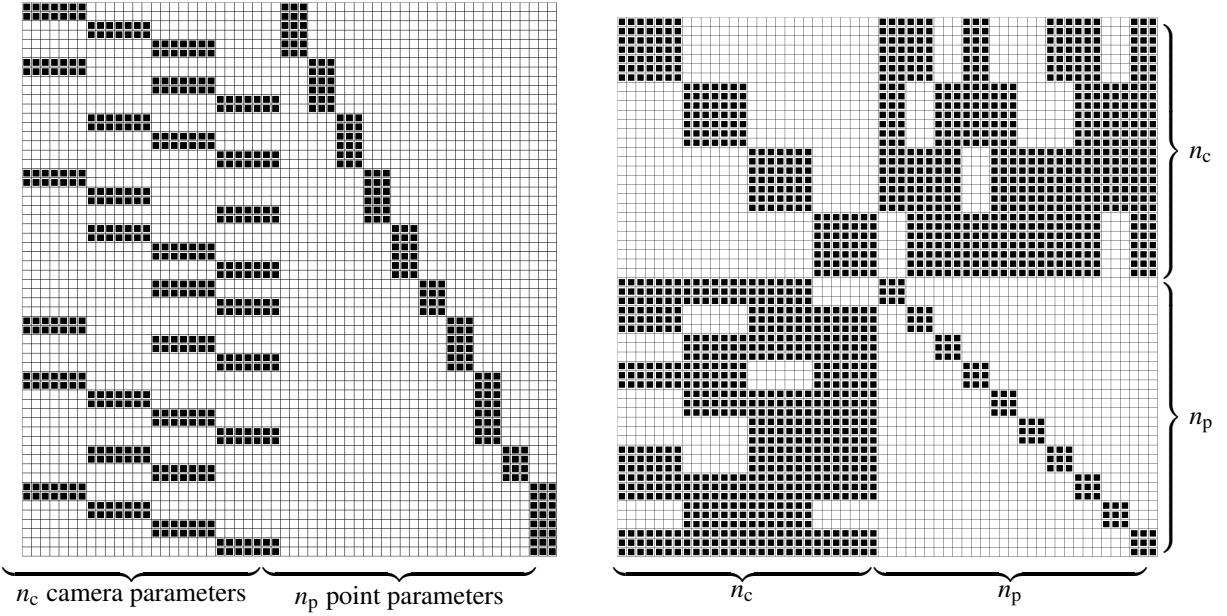


Figure 21.6: Left: the Jacobian matrix \mathbf{J} for a simple example. Zero elements are white, non-zero elements are colored. In this example, more than 80% of the elements in \mathbf{J} are zero. Right: the corresponding approximation of the Hessian, in terms of $\mathbf{J}^\top \mathbf{J}$.

The partitioning of \mathbf{J} into two parts carries over also the vector \mathbf{h} . It consists of a refinement of the n_c camera parameters, here denoted as \mathbf{h}_c , and of the n_p point parameters, \mathbf{h}_p . In combination with the recent results, these are found as the solution of

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{D} \end{pmatrix} \begin{pmatrix} \mathbf{h}_c \\ \mathbf{h}_p \end{pmatrix} = - \begin{pmatrix} \mathbf{g}_c \\ \mathbf{g}_p \end{pmatrix} = -\mathbf{g}. \quad (21.18)$$

Consequently, the refinement step can be computed in accordance with

$$\begin{pmatrix} \mathbf{h}_c \\ \mathbf{h}_p \end{pmatrix} = - \begin{pmatrix} \mathbf{A} & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{D} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{g}_c \\ \mathbf{g}_p \end{pmatrix}. \quad (21.19)$$

Based on Toolbox Section 8.1.4, the inverse of the 2×2 block matrix \mathbf{H} can be implemented in terms of the Schur complement, and Equation (21.19) can then be reformulated as

$$\begin{pmatrix} \mathbf{h}_c \\ \mathbf{h}_p \end{pmatrix} = - \begin{pmatrix} \mathbf{S}^{-1} & -\mathbf{S}^{-1}\mathbf{B}^\top\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{B}\mathbf{S}^{-1} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{B}\mathbf{S}^{-1}\mathbf{B}\mathbf{D}^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{g}_c \\ \mathbf{g}_p \end{pmatrix}, \quad (21.20)$$

where \mathbf{S} is the Schur complement of \mathbf{D} :

$$\mathbf{S} = \mathbf{A} - \mathbf{B}^\top \mathbf{D}^{-1} \mathbf{B}. \quad (21.21)$$

Let us now look at this result more in detail. As a start, Equation (21.20) implies that \mathbf{h}_c is the solution of the linear equation

$$\mathbf{S} \mathbf{h}_c = -\mathbf{g}_c + \mathbf{B}^\top \mathbf{D}^{-1} \mathbf{g}_p. \quad (21.22)$$

In this expression, the block diagonal matrix \mathbf{D} has an inverse \mathbf{D}^{-1} that can be determined very efficiently by only inverting p diagonal blocks, each of size 3×3 . The computational cost for this step is linear in p , the number of 3D points. What remains, is to solve the linear equation that is determined by the $n_c \times n_c$ symmetric matrix \mathbf{S} , the Schur complement of \mathbf{D} . This can be done using standard techniques, e.g., based on the Cholesky factorization, described in Toolbox Section 8.3. The cost for this step is $O(l^3)$, where l is the number of views.

We also see that, once \mathbf{h}_c is determined, Equation (21.20) means that \mathbf{h}_p can be computed as

$$\mathbf{h}_p = -\mathbf{D}^{-1} (\mathbf{g}_p + \mathbf{B} \mathbf{h}_c). \quad (21.23)$$

Hence, in this expression for \mathbf{h}_p we can reuse \mathbf{D}^{-1} that has already been determined for the computation of \mathbf{h}_c , and it is not really necessary to solve an equation.

In summary, the computational cost of determining the refinement vector \mathbf{h} is typically dominated by solving Equation (21.22), since the computation of \mathbf{D}^{-1} is linear in p . Consequently, instead of solving $\mathbf{H}\mathbf{h} = -\mathbf{g}$, which has a computational cost that is $O((n_c + n_p)^3)$, where typically $n_p > n_c$, we can reduce the cost to $O(n_c^3)$. This makes the computation of \mathbf{h} manageable even for a very large number of 3D points, as long as the number of views, m , is limited. This approach to determine \mathbf{h} is referred to as the *Schur complement trick*.

The sparsity of \mathbf{J} and how it leads to the Schur complement trick for simplifying the computation of \mathbf{h} is further explained in the articles by Agarwal *et al.* [3], and by Triggs *et al.* [63]. The Schur complement trick is not a technique that can be applied to general optimization problem and, therefore, it is typically not used by software for general optimization. Instead, it is found in dedicated bundle adjustment software.

21.5.7 Scale ambiguity

As was mentioned in Section 21.1, the 3D structure generated in this way is a similarity reconstruction: there is a scale ambiguity in the resulting 3D space.

An arbitrary scaling can be chosen without validating the geometric constraints imposed by the data. Often this is done by setting the (initial) relative translation between the first two views equal to, for example, one unit of length. In some applications this produces a useful result, but sometimes we are interested also in the absolute scale. There are several approaches to this in the literature, and a few are briefly described here.

The most accurate solution to the scaling problem is to measure the absolute translation between the camera views directly when the images are generated. This requires a calibration procedure of the entire setup that produces the images, and requires proper planning in combination with the image acquisition.

As an alternative, the proper scale can be determined if there is some distance, or distances, in the reconstructed scene that are known. In particular, if the real distance between two 3D points in T_{points} is known, we can determine the fraction between the real distance and the reconstructed distance given by the coordinates in T_{points} . This fraction is then used to give a correct scale to the reconstructed points. In this way, we obtain a metric reconstruction. If several real distances between points in T_{points} are known, the scale factor can instead be determined by least squares estimation.

A combination of the last two approaches is to capture the images using a stereo rig, with a fixed and known relative pose of the cameras. This gives an even stronger set of constraints than just a known distance, which to some extent can simplify the estimation problem. The basic incremental approach to bundle adjustment presented here can still be used, with one important modification. Every time a new image is added to the problem, and its companion stereo image already exists in T_{views} , the camera pose of this new image is locked to its companion by the calibrated relative pose, and is kept that way during the bundle adjustment step. As a consequence, there will be only half as many parameters to adjust for the camera poses, compared to the mono camera case.

Another possibility is to combine the information extracted from the images with other data that measures the positions of the camera views. For example, modern cell phones are often equipped with inertial sensors, which then can provide measurements of the relative motion between each view. These sensors, however, have limited accuracy and can only give rough estimates of the scale factor for the reconstructed points.

21.5.8 The first camera pose

The proposed method for 3D reconstruction presented here sets the first camera pose to $(\mathbf{I}, \mathbf{0})$. This means that the world coordinate system has its origin at the corresponding camera center, and that its coordinate axes are aligned with the camera axes.

During the bundle adjustment step, camera poses and 3D points are adjusted to minimize ε_{BA} . In principle, this includes the pose of the first camera, and doing so may not be wrong. However, there are some reasons why it can be a good idea to keep the first camera pose constant, at the origin of the world coordinate system.

The discussion about unknown scale in Section 21.5.7 does not mention the fact that also the absolute position and rotation of the 3D world coordinate system is undetermined. Leaving the first camera pose fixed, implies that the reconstructed point-cloud will “stay put” during the iterations. They cannot translate or move around too much relative to the first camera view, without increasing the corresponding reconstruction errors in that view too much. This makes the intuitive understanding of how the solutions evolve over the iterations easier.

A second observation is that, by keeping the first camera pose constant, there will be fewer parameters to optimize over. In particular, the redundancy introduced by optimizing over all camera poses by itself may introduce additional local minima that can disturb the optimization process. If there are many images, corresponding to many camera poses to optimize over in the final stages of the algorithm, this may not be a big issue. During the initial iterations, when there is a larger uncertainty about the free parameters, removing redundant parameters enables the optimization to be made with fewer iterations and also with less chance of getting stuck at a non-global minimum.

21.6 Further reading

There is an extensive body of literature on the SfM problem. A recent overview and tutorial on the subject can be found in an article by Furukawa & Hernandez [22]. An example of how to reconstruct a 3D scene from an unordered collection of images is presented in an article of Agarwal *et al.* [2].

Bibliography

- [1] Y. I. Abdel-Aziz and H. M. Karara. Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry. In *Proceedings of the Symposium on Close-Range Photogrammetry, American Society of Photogrammetry*, pages 1–18, 1971.
- [2] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building Rome in a Day. In *Proceedings of International Conference on Computer Vision, 2010*, pages 72–79, 2009. doi: 10.1109/ICCV.2009.5459148.
- [3] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle Adjustment in the Large. In *Proceedings of European Conference on Computer Vision, 2010*, pages 29–42, 2010. doi: 10.1007/978-3-642-15552-9_3.
- [4] A. Basu and S. Licardie. Alternative models for fish-eye lenses. *Pattern Recognition Letters*, 16(4):433–441, 1995. doi: 10.1016/0167-8655(94)00115-J.
- [5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, pages 346–359, 2008. doi: 10.1016/j.cviu.2007.09.014. First published at ECCV 2006.
- [6] P. J. Besl and N. D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. doi: 10.1109/34.121791.
- [7] D. C. Brown. A solution to the general problem of multiple station analytical stereotriangulation. Technical report, Patrick Airforce Base, Florida, 1958. RCA-MTP Data Reduction Technical Report No. 43.
- [8] D. C. Brown. Decentering Distortion of Lenses. *Photogrammetric Engineering*, 32(3):444–462, 1966.
- [9] M. Brown and D. G. Lowe. Recognising Panoramas. In *Proceedings of the International Conference on Computer Vision*, pages 1218–1225, 2003. doi: 10.1109/ICCV.2003.1238630.
- [10] R. Burtch. History of photogrammetry. Web page, retrieved Jan 23, 2016:
<https://spatial.curtin.edu.au/local/docs/HistoryOfPhotogrammetry.pdf>.
- [11] Y. Chen and G. Medioni. Object Modeling by Registration of Multiple Range Images. In *Proceedings of the International Conference on Robotics and Automation*, pages 2724–2729, 1991. doi: 10.1016/0262-8856(92)90066-C.
- [12] A. Conrad. Projective Geometry - The early years, 2000. Web page, retrieved Jan 23, 2016:
<http://www.math.rutgers.edu/~cherlin/History/Papers2000/conrad.html>.
- [13] A. E. Conrady. Decentered Lens-Systems. *Monthly notices from the Royal Astronomical Society*, 79(5):384–390, 1919.
- [14] P. Corke. *Robotics, Vision and Control*. Springer, 2011.
- [15] H. S. M. Coxeter. *Projective Geometry*. Springer, 2003. ISBN 978-0387406237. 2nd edition.
- [16] F. Dervernay and O. Faugeras. Straight lines have to be straight. *Machine Vision and Application*, 13:14–24, 2001. doi: 10.1007/PL00013269.

- [17] O. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? In *Proceedings of the European Conference of Computer Vision*, pages 563–578, 1992. doi: 10.1007/3-540-55426-2_61.
- [18] O. Faugeras. *Tree-dimensional computer vision : a geometric viewpoint*. The MIT Press, 1993.
- [19] J. V Field. Two Mathematical Inventions in Kepler's 'Ad Vitellionem Paralipomena'. *Studies in History and Philosophy of Science*, 17(4):449–468, 1986.
- [20] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the Association for Computing Machinery*, 24(6):381–395, 1981. doi: 10.1145/358669.358692.
- [21] A. W. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol 1*, pages 125–132, 2001. doi: 10.1109/CVPR.2001.990465.
- [22] Y. Furukawa and C. Hernández. Multi-View Stereo: A Tutorial. *Foundations and Trends in Computer Graphics and Vision*, 9(1-2):1–148, 2015. doi: 10.1561/0600000052.
- [23] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete Solution Classification for the Perspective-Three-Point Problem. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 25(8):930–943, 2003. doi: 10.1109/TPAMI.2003.1217599.
- [24] R. C. Gozalez and R. E. Woods. *Digital Image Processing*. Pearson Prentice Hall, 2008. 3rd ed.
- [25] J. A. Grunert. Das Pothenotische Problem in erweiterter Gestalt nebst über seine Anwendungen in der Geodäsie. In *Grunerts Archiv für Mathematik und Physik*, 1841.
- [26] R. I. Hanson and M. J. Norris. Analysis of measurement based on the singular value decomposition. *SIAM Journal of Scientific Statistical Computing*, 2(3):363–373, 1981. doi: 10.1137/0902029.
- [27] C. Harris and M. Stevens. A Combined Corner and Edge Detector. In *Proceedings of Fourth Alvey Vision Conference*, pages 147–151, 1988. Web page retrieved Aug 9, 2017: <http://www.bmva.org/bmvc/1988/avc-88-023.pdf>.
- [28] R. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Proceedings of the European Conference of Computer Vision*, pages 579–587, 1992. doi: 10.1007/3-540-55426-2_62.
- [29] R. Hartley. Theory and practice of projective rectification. *International Journal of Computer Vision*, 35(2):115–127, 1999.
- [30] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge, 2003. ISBN 978-0-521-54051-3. 2nd edition.
- [31] R. I. Hartley. In Defence of the 8-point Algorithm. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 19(6):580–593, 1997. doi: 10.1109/ICCV.1995.466816.
- [32] R. I. Hartley and J. L. Mundy. The relationship between photogrammetry and computer vision. In *Proceedings of SPIE 1944*, pages 92–105, 1993. doi: 10.1117/12.155818.
- [33] R. I. Hartley and P. Sturm. Optimal Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.
- [34] G. Hauck. Neue Constructionen der Perspektive und Photogrammetrie (Theorie der trilinearen Verwandtschaft ebener Systeme). *Journal für die reine und angewandte Mathematik*, 95(1):1–35, 1883. http://gdz.sub.uni-goettingen.de/dms/load/img/?PID=PPN243919689_0095&physid=PHYS_0006.
- [35] N. J. Higham. *Function of Matrices: Theory and Computation*. SIAM, 2008. doi: 10.1137/1.9780898717778.
- [36] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987. doi: 10.1364/JOSAA.4.000629.

- [37] W. Hugemann. Correcting Lens Distortions in Digital Photographs. In *EVU*, 2010.
- [38] P-E. Forssén J. Hedborg and M. Felsberg. Fast and Accurate Structure and Motion Estimation. In *Proceedings of international Symposium on Visual Computing*, pages 211–222, 2009. doi: 10.1007/978-3-642-10331-5_20.
- [39] B. Jähne. *Digital Image Processing*. Springer, 2005.
- [40] E. M. Jones and P. Fjeld. Gimbal Angles, Gimbal Lock, and a Fourth Gimbal for Christmas. Web page retrieved April 14, 2017: <https://www.hq.nasa.gov/alsj/gimbals.html>.
- [41] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica*, A32:922–923, 1976. doi: 10.1002/jcc.20110.
- [42] W. Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica*, A34: 827–828, 1978. doi: 10.1107/S0567739478001680.
- [43] L. Kneip, D. Scarmuzza, and R. Siegwart. A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4546–4553, 2011. doi: 10.1109/CVPR.2011.5995464.
- [44] L. Kneip. <http://www.laurentkneip.de>.
- [45] C. H. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981. doi: 10.1038/293133a0.
- [46] C. H. Longuet-Higgins. The reconstruction of a plane surface from two perspective projections. *Proc. of R. Soc. of Lond. B*, 227:399–410, 1986. doi: 10.1098/rspb.1986.0030.
- [47] C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 125–131, 1999. doi: 10.1109/CVPR.1999.786928.
- [48] D. Lowe. Object Recognition from Local Scale-Invariant Features. In *Proceedings from the International Conference on Computer Vision, Vol 2*, pages 1150–1157, 1999. Web page retrieved Aug 9, 2017: <http://www.cs.ubc.ca/~lowe/papers/iccv99.pdf>.
- [49] M. Mühlbach and R. Mester. A considerable improvement in non-iterative homography estimation using TLS and equilibrium. *Pattern Recognition Letters*, 22:1181–1189, 2001. doi: 10.1016/S0167-8655(01)00051-4.
- [50] G. Miller. The Huge Unseen Operation Behind the Accuracy of Google Maps. Published in *Wired* on Dec. 8, 2014. Web page retrieved June 23, 2017: <https://www.wired.com/2014/12/google-maps-ground-truth/>.
- [51] M. E. Muller. A Note on a Method for Generating Points Uniformly on N-Dimensional Spheres. *Communication of the ACM*, 2(4):19–20, 1959. doi: 10.1145/377939.377946.
- [52] D. Nistér. An Efficient Solution to the Five-Point Relative Pose Problem. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages II–195–202, 2003. doi: 10.1109/CVPR.2003.1211470.
- [53] D. Nistér. An Efficient Solution to the Five-Point Relative Pose Problem. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004. doi: 10.1109/TPAMI.2004.17.
- [54] K. Nordberg. *Mathematical Toolbox for studies in Visual Computation at Linköping University*. Linköping University, 2018. Version 0.40.
- [55] S. J. D. Prince. *Computer Vision : Models, Learning and Inference*. Cambridge, 2012. ISBN 978-1-107-01179-3.
- [56] R. J. Radke. *Computer Vision for Visual Effects*. Cambridge University Press, 2013. ISBN 978-0521766876.
- [57] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proceedings from International Conference on Computer Vision*, pages 2564–2571, 2011. doi: 10.1109/ICCV.2011.6126544.
- [58] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *3-D Digital Imaging and Modeling*, pages 145–152, 2001. doi: 10.1109/IM.2001.924423.
- [59] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *SIGGRAPH Conference Proceedings*, pages 835–846, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-364-6.

- [60] M. Sonka, V. Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Thomson, 2008. 3rd edition.
- [61] P. Sturm. A historical survey of geometric computer vision. In *International Conference on Computer Analysis of Images and Patterns*, pages 1–8, 2011. doi: 10.1007/978-3-642-23672-3_-1.
- [62] R. Szeliski. *Computer Vision : Algorithms and Applications*. Springer, 2011.
- [63] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle Adjustment – A Modern Synthesis. In *International Conference on Computer Vision, Workshop on Vision Algorithms: Theory and Practice*, pages 298–372, 1999. doi: 10.1007/3-540-44480-7_-21.
- [64] G. Wahba. Problem 651: A Least Squares Estimate of Spacecraft Attitude. *SIAM Review*, 7(3), 1965. doi: 10.1137/1007077.
- [65] G. Xu and Z. Zhang. *Epipolar Geometry in Stereo, Motion and Object Recognition*. Kluwer Academic Publishers, 1996.
- [66] Z. Zhang. Determining the Epipolar Geometry and its Uncertainty: A Review. *International Journal of Computer Vision*, 27(2):161–198, 1998. doi: 10.1023/A:1007941100561.
- [67] Z. Zhang. A Flexible New Technique for Camera Calibration. Technical report, 1998. Technical Report, Microsoft Research, MSR-TR-98-71.
- [68] Z. Zhang. A Flexible New Technique for Camera Calibration. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 22(11):1330–1334, 2000. doi: 10.1109/34.888718.

Index

- 3D reconstruction, 139
- 5+1-point algorithm, 310
- 5-point algorithm, 290
- 7-point algorithm, 285
- 8-point algorithm, 283
- affine point, 47
- affine transformation, 59
- affine transformation group, 59
- affine vector, 47
- algebraic distance, 42
- algebraic error, 42, 204, 205
- algebraic estimation, 208
- algebraic representation of $SO(3)$, 163, 165
- algebraically independent, 118
- analytic geometry, 16
- aperture, 97, 317
- automatic camera calibration, 316
- auxiliary variables, 221, 243, 289, 317, 328
- axis-angle representation of $SO(3)$, 166
- backward interpolation, 337
- barrel distortion, 320
- barycentric coordinates, 264
- baseline, 140
- blending, 338
- blind plane, 356
- bookkeeping tables, 371
- bounding box, 112, 332
- Brown-Conrady distortion, 322
- bundle adjustment, 17, 370
 - incremental, 370
- C-normalized image coordinates, 102
- calibrated epipolar geometry, 137, 282
- calibration device, 325
- camera, 97
 - rotating, 131
- camera (projection) matrix, 96
 - from \mathbf{F} , 149
- camera calibration, 111, 240, 315
- camera center, 96
- camera centered coordinate systems, 98
- camera normalized image coordinates, 102
- camera obscura, 97
- camera resectioning, 110
- cameras observe a plane, 126
- canonical form of homogeneous coordinates
 - 2D line, 34
 - 2D point, 32
 - 3D point, 67
 - plane, 68
- canonical form of Plücker coordinates, 72, 75
- canonical set of homogeneous coordinates, 92
- Cardano angles, 179
- Cartesian coordinate system, 21
- Cartesian representation, 22
- Cayley transform, 172
- central perspective, 95
- central projection, 95
- centroid, 193
- co-linear planes, 27
- co-linear points, 22, 27
- co-planar lines, 27
- co-planar points, 26
- collineation, 87
- complete representation, 165
- concatenation of rotations, 166
- consensus set, 299
- consistency, 206
- consistent parameterization, 244
- consistent Plücker coordinates of 3D line, 74
- constraint enforcement, 244
 - fundamental matrix, 284
 - normalized camera matrix, 270
 - rotation matrix, 261
- corner point, 122
- correspondence, 137
- correspondence problem, 124
- corresponding lines, 122
- corresponding points, 122, 137, 219
- cost function, 192
- cross-correlation, 193
- D-normalization, 35
- data constraint, 207
- data degeneracy, 207, 224, 225, 282
- data error, 196
- data matrix, 38, 206
- decentering distortion, 322
- degeneracy
 - epipolar geometry estimation, 286

homography estimation, 224
 method, 210
 of data, 207
 degenerate configuration, 54
 degenerate data, 37
 degrees of freedom, 207
 dense disparity estimation, 353
 depth, 101, 349
 depth-of-field, 317
 direct linear transformation, 118, 221
 directed line, 43
 direction, 47
 disparity, 349, 353
 distortion center, 323
 DL-normalization, 75
 DLT, 221
 double embedding of $SO(3)$, 176
 dual homogeneous coordinates, 34, 69
 dual line normalization, 35, 75
 dual Plücker coordinates, 51, 73
 dual transformation, 64
 duality mapping, 74
 duality of points and lines, 52

 ego-motion estimation, 131, 370
 energy function, 192
 epipolar constraint, 118, 125, 137, 144
 epipolar geometry, 121, 137

- degenerate, 286

 epipolar line, 141, 145
 epipolar plane, 142
 epipolar point, 140
 epipole, 140
 equation of the line, 22
 equation of the plane, 26
 equilibrating transformation, 234
 error function, 192
 essential matrix, 137, 156

- estimation, 290
- robust, 308
- internal constraint, 157

 estimation

- camera matrix, 239
- epipolar geometry, 275
- fundamental matrix, 282
- homography, 219
- introduction, 191
- MAP, 216
- maximum a posteriori probability, 216
- rigid transformation, 263
- rotations, 259
- transformations, 219

 Euclidean geometry, 15
 Euclidean reconstruction, 154

Euler angles, 179
 exhaustive search, 125
 exterior orientation, 108, 263
 external camera parameters, 108

 feasibility constraint, 101
 FET, 320
 field of view, 119
 field of view model, 321
 fish-eye lens, 318
 fish-eye transform, 320
 focal length, 98
 focal point, 96
 forward interpolation, 337
 FOV, 321
 fundamental matrix, 137, 143

- camera matrices from, 149
- estimation, 282
- robust, 307
- internal constraint, 143

 general configuration of points, 54
 geometric distortion, 318
 geometric estimation, 200
 geometric objects, 31
 gimbal lock, 187
 gold standard method, 288

 H-normalization, 235
 hand-eye calibration, 263
 Hartley normalization, 235
 Hesse normal form, 24
 Hesse parameters, 24
 homogeneous coordinates, 17, 30

- 2D point, 32
 - canonical form, 32
- 3D point, 67
 - canonical form, 67
- line, 34
 - 2D canonical form, 34
- plane, 69
 - canonical form, 68

 homogeneous method, 212
 homogeneous representation, 22
 homography, 87

- planar, 121, 126
- rotational, 133

 homography estimation

- degeneracy, 224
- robust, 307

 homography group, 89
 homography transformation, 87

 IAC, 134
 ICP, 265

ideal data, 197
 ideal point, 16, 47
 identity rotation, 166
 IID, 197
 image coordinate system, 105
 image coordinates, 96
 image mosaic, 332
 image mosaicking, 332
 image of the absolute conic, 134
 image plane, 96
 image point, 96
 image stitching, 332
 incidence relations, 36
 incremental bundle adjustment, 370
 independent and identical distributions, 197
 indeterministic estimation method, 299
 inhomogeneous method, 210
 inliers, 199, 296
 inner derivatives, 246
 intercept, 23
 interest point, 122
 detection, 124
 interior orientation, 108
 internal camera parameters, 108
 internal constraint, 133, 206
 essential matrix, 157
 fundamental matrix, 143
 normalized camera matrix, 103
 of a rotation representation, 166
 Plücker coordinates, 74
 rotation matrix, 164
 invariances, 53
 inverse Cayley transform, 174
 inverse of a rotation, 166
 involution, 61
 iterative closest point, 265

 Jacobian mask, 257, 382

 keystone effect, 90

 L-normalization, 72
 L_1 error, 202
 L_2 error, 201
 lens distortion, 318
 lens distortion function, 318
 lens effect, 318
 likelihood, 217
 line at infinity, 48, 71
 line normalization, 72

 MAP, 216
 match, 122
 matched transformations, 359
 matching points, 122

 max-error, 202
 maximum a posteriori probability, 216
 maximum likelihood estimation, 216, 217
 mean, 193
 measurement error, 196
 measurement noise, 114, 197
 median-error, 202
 method degeneracy, 210, 225, 281
 mid-point method, 276
 minimal case estimation, 223
 minimal case of model estimation, 208
 minimal parameterization, 108, 149
 ML-estimation, 216, 217
 model error, 198
 model estimation
 minimal case, 208
 over-determined case, 208
 under-determined case, 207
 model parameters, 193, 205
 motion stereo, 139

 nearest neighbor interpolation, 115, 337
 non-linear optimization, 201
 normalization constraint, 206
 normalized 8-point algorithm, 285
 normalized camera matrix, 101, 103
 internal constraint, 103
 normalizing transformation, 231, 234

 objective function, 192
 observation, 197
 omni-directional camera, 332
 one-first enumeration of coordinates, 114
 optical axis, 98
 optical center, 96
 optical ray, 115
 optimal triangulation, 277
 orientation of a line, 46
 outlier, 199, 203, 296
 over-determined model estimation, 208
 over-fitting, 198

 P-normalization, 33
 P3P, 272
 panorama image, 332
 panorama stitching, 332
 parameter space, 165
 parameterization of a transformation, 54
 parametric representation, 25
 perspective n -point problem, 270
 photogrammetry, 17
 pin-cushion distortion, 320
 pinhole camera, 95, 97
 pinhole perspective, 95
 pixel coordinates, 105

pixel density, 108
 pixel grid, 112
 pixel value, 113
 pixels, 97
 Plücker coordinates of 2D line, 51
 Plücker coordinates of 3D line, 71
 canonical form, 72, 75
 consistent, 74
 internal constraint, 74
 sandwich product, 86
 planar homography, 121, 126
 plane observed by two cameras, 126
 plumb bob distortion, 322
 PnP, 270
 point at infinity, 16, 46
 point correspondence, 122
 point normalization, 33
 point of interest, 122
 point-cloud, 378
 pose, 108
 pre-conditioning, 234
 principal axis, 98
 principal line, 98
 principal plane, 98
 principal point, 98
 probability density, 214
 probability density function, 197
 projection, 87
 projection line, 87, 115
 projection plane, 117
 projection point, 87
 projective geometry, 16, 48
 projective group, 89
 projective plane, 16, 48
 projective reconstruction, 155
 projective spaces, 17
 projectivity, 87
 proper line, 45, 71
 proper point, 45
 putative correspondence, 130, 366

 quaternion sandwich product, 175
 quaternionic embedding of $SO(3)$, 175
 quaternionic embedding of \mathbb{R}^3 , 175

 radial distortion function, 320
 Random Sample Consensus, 300
 RANSAC, 300
 re-mapping of the residual vector, 247
 re-normalize, 231
 re-parameterization, 245
 reconstruction, 153
 rectified stereo, 349
 synthetic, 349
 rectified stereo cameras, 349

 rectified stereo rig, 349
 rectifying homographies, 349, 357
 reference image, 333
 registration, 265
 reprojection error, 369
 residual error, 193
 residual vector, 206, 242
 rigid transformation, 57
 estimation, 263
 rigid transformation group, 57
 robust errors, 297
 robust estimation, 295, 296
 essential matrix, 308
 fundamental matrix, 307
 homography, 307
 Rodrigues' rotation formula, 167, 168
 rotating camera, 131
 rotation, 164
 rotation matrix
 internal constraint, 164
 rotational homography, 121, 131, 133

 SAD, 202
 sandwich product of Plücker coordinates, 86
 Schur complement trick, 384
 $SE(2)$, 57
 $SE(3)$, 83
 SfM, 367
 shearing, 62
 signed distance, 41
 similarity reconstruction, 154
 similarity transformations, 58
 singularity, 166
 skewing, 62
 slope, 23
 solution space, 207, 225
 SOPP, 261
 sparse disparity estimation, 353
 special Euclidean transformation group
 in \mathbb{E}^2 , 57
 in \mathbb{E}^3 , 83
 special orthogonal Procrustes problem, 261
 SSD, 201
 standard Plücker coordinates, 73
 stereo cameras, 137, 139
 stereo images, 139
 stereo rig, 139
 structure and motion, 367
 structure from motion, 367
 sum of absolute differences, 202
 sum of square differences, 201
 SVD profile, 226
 synthetic stereo rectification, 349, 356, 357
 synthetically rectified stereo images, 357

Tait-Bryan angles, 179
thin-prism distortion, 322
three-angle representation, 179
total least squares, 195
transfer, 146
 of epipolar lines, 146
 of points, 146
translation group, 56
trial set, 299
triangulation, 17, 118, 139, 153, 275
twisted pair, 185
twisted rotations, 159, 185

unbiased estimate, 237
uncalibrated epipolar geometry, 137, 149
under-determined model estimation, 207
unique representation, 165
unperturbed data, 197

variances, 193
vectorization, 206, 222, 239
virtual image plane, 101
visibility function, 369
visual appearance, 305
visual odometry, 370

Wahba's problem, 260
weak stereo rectification, 350
world coordinate system, 102

zero-first enumeration of coordinates, 114

List of Algorithms

3.1 Determine if a 2D point \mathbf{y} lies inside a convex region Ω bounded by a polygonal chain.	45
8.1 Camera resectioning	111
9.1 Determine $\mathbf{R}, \hat{\mathbf{t}}, \hat{\mathbf{p}}$, and γ from a planar homography \mathbf{H}	131
10.1 Determine $\mathbf{C}_1, \mathbf{C}_2$ that are consistent with \mathbf{F} , based on epipoles	150
10.2 Determine $\mathbf{C}_1, \mathbf{C}_2$ that are consistent with \mathbf{F} , based on SVD	151
10.3 Determine a relative camera pose $(\mathbf{R}, \hat{\mathbf{t}})$ that is consistent with \mathbf{E}	159
11.1 Finding rotation axis $\hat{\mathbf{n}}$ and rotation angle α given a rotation matrix \mathbf{R} . No EVD.	170
11.2 Find a unit quaternion that represents a given 3D rotation.	178
11.3 Find Euler angles corresponding to $\mathbf{R} \in SO(3)$	181
11.4 Generates a random element of $SO(3)$ with a uniform PDF.	185
12.1 The inhomogeneous method for minimizing algebraic errors	210
12.2 The homogeneous method for minimizing algebraic errors	212
13.1 Normalization of 2D points using Hartley's approach.	235
13.2 Basic algorithm for algebraic estimation of a homography	238
13.3 Algebraic estimation of \mathbf{H} using the homogeneous method with data normalization.	238
15.1 Strict version of OPP.	261
15.2 Solving the special orthogonal Procrustes problem (SOPP).	262
15.3 Estimation of 3D rotations based on quaternions.	262
15.4 Estimation of 3D rigid transformations.	264
15.5 The Iterative Closest Point algorithm (ICP).	265
15.6 PnP based on algebraic minimization.	271
16.1 The mid-point method for 2-view triangulation	276
16.2 Algebraic method for triangulation of a 3D point from multiple views.	278
16.3 Optimal triangulation	280
16.4 The normalized 8-point algorithm	285
16.5 The 7-point algorithm	286
16.6 The gold standard method for estimation of \mathbf{F}	289
16.7 An adaptation of the 8-point algorithm to estimation of \mathbf{E}	291
16.8 Algebraic estimation of ω and \mathbf{K}	293
17.1 The generic RANSAC algorithm	304
17.2 The RANSAC algorithm adapted to the correspondence problem	306
17.3 RANSAC algorithm for estimation of a homography	307
17.4 RANSAC algorithm for estimation of epipolar geometry	308
17.5 RANSAC algorithm for estimation of essential matrix based on the 5-point algorithm.	309
17.6 RANSAC algorithm for estimation of essential matrix based on the 5+1-point algorithm.	309
17.7 RANSAC algorithm for robust estimation of camera pose.	311

18.1	Zhang's calibration method based on algebraic minimization	329
18.2	Zhang's calibration method based on geometric minimization	330
19.1	Extending a reference image with a single image.	341
19.2	Building a mosaic from a set of images that are related by homographies.	342
19.3	Adding an image to the reference image. Based on spherical coordinates.	347
19.4	Building a panorama from a set of images. Based on spherical coordinates.	348
20.1	Hartley's methods for stereo rectification	363
21.1	Initialization of an incremental approach to SfM.	374
21.2	Add new view to T_{views}	375
21.3	Add new 3D points to the SfM problem.	376
21.4	An incremental approach to SfM.	380

Video References

A number of videos that are available from the internet can be used to illustrate or explain certain concepts that are used in this compendium.

1. <https://www.youtube.com/watch?v=WqzK3UAXaHs&index=1&list=PL26812DF9846578C3>

 An introduction to Euclid's work and its impact on mathematics, as well as on American presidents!
Production: Kahn Academy



2. <http://smarthistory.org/linear-perspective-brunelleschis-experiment/>

 A presentation of the experiment on linear projection that was performed by Brunelleschi in the early 1400's at the Dome of Florence.
Production: Smart History / Kahn Academy

