

Robot Vision

Lecture Notes for TPK4171 Advanced Industrial Robotics

Professor Olav Egeland
Department of Mechanical and Industrial Engineering
NTNU

January 17, 2024

Contents

1	Introduction	11
2	Camera model	11
2.1	Introduction	11
2.2	The pinhole camera model	11
2.3	Object frame	17
2.4	Camera parameters	19
2.5	Calculation of pixel coordinates	20
2.6	Calculation of normalized coordinates from pixel coordinates	21
2.7	Base frame, camera frame and object frame	21
2.8	The image Jacobian*	22
3	RANSAC and the Hough transform	23
3.1	Determination of a line from noisy points	23
3.2	Linear regression to determine the line	24
3.3	RANSAC for determination of the line	25
3.4	Determination of a circle from noisy points	28
3.5	The Hough transform of a line	30
4	Projective geometry in two dimensions	34
4.1	Introduction	34
4.2	Homogeneous coordinates 2D	34
4.3	Rotation of a vector	35
4.4	Displacement of a vector	35
4.5	Lines in 2D	35
4.6	Homogeneous description of lines	37
4.7	Distance from a point to a line	38
4.8	Lines through the origin	41
4.9	A closer look at the homogeneous representation of point and lines in 2D*	41
4.10	Linear dependence of homogeneous and Euclidean points*	42

4.11	Points and lines in terms of determinants*	42
4.12	Point at infinity	44
4.13	Intersection of two parallel lines	45
4.14	Line at infinity	45
4.15	Intersection of a line and a line at infinity	47
4.16	Fitting a line to a set of two points	47
4.17	Fitting a line through the origin to two points with SVD	48
4.18	Line regression gives a line through the centroid	50
4.19	Fitting a line through the origin to a set of noisy points	50
4.20	Line regression with least squares versus total least squares	54
5	Homographies in two dimensional space	57
5.1	Introduction	57
5.2	Homography	57
5.3	Independent parameters of a homography in 2D	57
5.4	Homographies in two dimensions	58
5.5	Transformation of lines	58
5.6	Homographic mapping of pixel values	58
5.7	Types of homographies	60
5.8	Euclidean transformation	61
5.9	Similarity transformation	62
5.10	Affine transformations	63
5.11	The effect of an affine transformation	64
5.12	Affine transformation of points at infinity and the line at infinity	66
5.13	Projective transformations	66
5.14	Transformation of lines with a projective homography	66
5.15	Transformation of points at infinity and the line at infinity	67
5.16	Decomposition of a projective transformation	67
5.17	Detailed expressions for the decomposition of projective homographies	68
5.18	A closer look at the projective property of a transformation	69
5.19	Projective transformations: Case 2	72
5.20	The projective transformation of the camera model	75
6	Computation of homographies in 2D	79
6.1	Computation of homographies from point mappings by DLT	79
6.2	Computation from 4 point mappings	81
6.3	Computation from more than 4 point mappings	82
6.4	Conditions from point mappings: Method 1	82
6.5	A simplified set of equations	83
6.6	Normalization	84
6.7	Conditions from point mappings: Method 2	84
6.8	Pose estimation with PnP using 4 points in a plane	89
7	Optimization in the computation of homographies	93
7.1	Introduction	93
7.2	The algebraic error of a DLT problem	93
7.3	Transfer error in the second image	94

7.4	Transfer error in the first image	95
7.5	Minimization of the transfer error in the second image	96
7.6	The reprojection error	98
7.7	The Sampson error in Euclidean 2D	99
7.8	Sampson's error in higher dimension problems	100
7.9	Sampson's error for the reprojection error	101
7.10	Jacobian	102
7.11	Optimization of the homography with Sampson's approximation of the reprojection error	102
7.12	Optimization of the reprojection error with Sampson's error by iteration	104
7.13	The gold standard	104
7.14	Invariance of the geometric error	106
7.15	The DLT method is not invariant to Euclidean transformations	107
8	Points and planes in 3D geometry	107
8.1	Points in 3D	107
8.2	Normalization of homogeneous vectors in the camera model	108
8.3	Point at infinity	108
8.4	Planes in 3D geometry	109
8.5	Distance from a point to a plane	109
8.6	Plane at infinity	110
8.7	Determining a plane from 3 points	110
8.8	Fitting a plane to a set of points	111
9	Points, lines and planes in 3D using Plücker coordinates	112
9.1	Introduction	112
9.2	Degrees of freedom for a line in 3D	112
9.3	Description of a line with a point on the line and the direction vector	112
9.4	Description of a line with the direction vector and the moment	113
9.5	Lines with unit direction vector	115
9.6	Change of reference point for a line in 3D	115
9.7	Plücker coordinates for a line	116
9.8	Geometric interpretation of the Plücker coordinates	118
9.9	Invariance of Plücker coordinates to selection of points on the line	119
9.10	Condition for a point to be on a line	119
9.11	The line defined by a finite point and a point at infinity	120
9.12	A line defined by two points at infinity	120
9.13	Condition for a point to be at a line at infinity	120
9.14	The Plücker coordinates of a plane	121
9.15	The Plücker coordinates of a plane from the normal vector and a point in the plane	122
9.16	Calculation of points on a plane	124
9.17	More on the geometric interpretation of the Plücker coordinates of a plane	125
9.18	A plane defined by a finite point and a line at infinity	127
9.19	A plane defined by three points at infinity	127
9.20	A line determined by the intersection of two planes	127
9.21	Derivation of the expression for a line as the intersection of two planes	128

9.22 The line at the intersection of a finite plane and the plane at infinity	129
9.23 A point as the intersection of a line and a plane	129
9.24 A point as the intersection of a line and the plane at infinity	130
9.25 Condition for a line to be in a plane	130
9.26 Plane spanned by two intersecting lines	131
9.27 Point as intersection of three planes	131
9.28 The distance from a line to a point	132
9.29 Plücker matrices	133
9.30 The intersection of a line and the plane at infinity	133
9.31 Dual Plücker matrices	134
10 Intersecting lines and non-intersecting lines	134
10.1 Condition for two lines to intersect	134
10.2 Point as intersection of two lines: Geometric method	135
10.3 Least-squares geometric error for the intersection point of lines	139
10.4 The common normal of two lines	142
10.5 The distance between two nonparallel lines	142
10.6 The midpoint of the common normal of two nonparallel lines	143
10.7 Point of intersection between a set of lines by minimization of algebraic error *	144
11 Images of geometric objects	147
11.1 The image of a line	147
11.2 The plane defined by a 3D line and the camera center	147
11.3 The image of a line defined by two planes	147
11.4 Triangulation of a point in a known plane	148
12 The geometry of a laser line scanner	148
12.1 The geometry of the laser plane	148
12.2 The image of a point in the laser plane	148
12.3 The image of a line at the intersection of a plane and the laser plane	149
12.4 Reconstruction of a 3D line from a laser scanner system	149
12.5 Laser scanner image of plane with weld groove	149
13 Homographies in 3D*	155
13.1 Transformation of points	155
13.2 Transformation of planes	155
13.3 Types of homographies	155
13.4 Affine transformations	156
13.5 Affine transformation of points at infinity and planes at infinity	157
13.6 Projective transformations	157
13.7 Decomposition of a projective homography	158
13.8 Affine reconstruction	158
13.9 Projection in terms of a homography	159
13.10 Orthographic projections	160
13.11 The camera projection*	161
13.12 The absolute conic	162

14 Quadratics*	162
14.1 Introduction	162
14.2 Dual quadric	162
14.3 Transformation of quadrics	163
15 Recovery of affine and metric properties	163
15.1 Recovering affine properties using the line at infinity	164
15.2 Recovery of metric properties	168
16 Calibration	176
16.1 Introduction	176
16.2 Vanishing points in an image	176
16.3 The image of a vanishing point	176
16.4 The image of a vanishing line	178
16.5 Calibration from scene constraints	179
16.6 Expressions for the image of the absolute conic	180
16.7 Intrinsic calibration with checkerboard planes: Zhang's method	182
16.8 Intrinsic calibration with checkerboard planes: Bouguet's method	191
16.9 Calibration from three vanishing points	192
16.10 Calibration by decomposition of the camera matrix	192
17 Stereo vision with calibrated cameras	196
17.1 Introduction	196
17.2 The geometry of a stereo arrangement	196
17.3 Epipolar constraints and the essential matrix	198
17.4 The essential matrix and epipolar lines	198
17.5 The essential matrix for standard camera matrices	203
17.6 Recovery of rotation and translation from the essential matrix	204
17.7 A closer look at the recovery of displacement from the essential matrix	205
17.8 The twisted pair ambiguity	207
17.9 Comments on the SVD of the essential matrix	207
17.10 Selection of the recovered displacement from the essential matrix	209
17.11 Calculation of the essential matrix with the eight-point algorithm	209
17.12 Triangulation to determine the valid displacement of the recovery problem	212
17.13 Triangulation using the midpoint	214
17.14 Linear triangulation by minimizing the algebraic error	215
17.15 Triangulation using the reprojection error	216
17.16 3D structure recovery	216
17.17 Triangulation with n known camera matrices	217
17.18 Determination of a line from stereo vision	218
17.19 Geometric interpretation of the determination of a line from stereo vision	219
17.20 Planar homography in stereo vision	220
17.21 Computation of the planar homography	221
17.22 Recovery of the displacement from the planar homography	222
17.23 The planar homography and the essential matrix	226
18 Stereo vision with uncalibrated cameras*	226

18.1 The fundamental matrix	226
18.2 Transformation of skew symmetric matrices	228
18.3 The image of a line in Plücker coordinates with an uncalibrated camera	230
18.4 Transformation of a line in Plücker coordinates by a homography	230
18.5 Fundamental matrix for standard cameras	232
18.6 Fundamental matrix for standard cameras	232
18.7 Calculation of the fundamental matrix from point correspondences	233
18.8 Sampson error of the uncalibrated epipolar constraint	234
18.9 The ambiguity in the reconstruction from two uncalibrated cameras	234
18.10 Decomposition of projective homography	235
18.11 Camera models in the uncalibrated reconstruction	236
18.12 The fundamental matrix and the projective ambiguity	237
18.13 A projective camera model from the fundamental matrix	238
18.14 Projective reconstruction from uncalibrated views	238
18.15 Affine reconstruction from projective reconstruction	239
18.16 Affine reconstruction from uncalibrated views	239
18.17 Euclidean reconstruction	240
18.18 Calibration from three sets of orthogonal parallel lines	240
19 Conics*	241
19.1 Introduction	241
19.2 Description of conics	241
19.3 Determination of an ellipse from 5 points	245
19.4 Determination of a circle from 3 points	246
19.5 Determination of a conic from determinants	248
19.6 Determination of a circle from determinants	250
19.7 Least-squares determination of a conic from point observations	251
19.8 Least-squares determination of a circle from point observations	252
19.9 The Kåsa fit of a circle to a set of points	253
19.10 The original Kåsa fit*	255
19.11 The Rayleigh quotient	256
19.12 The generalized eigenvalue problem	257
19.13 Eigenvalues and the Rayleigh quotient	258
19.14 The Pratt fit	258
19.15 The Taubin fit of a circle	259
19.16 The Taubin fit for an ellipse	260
19.17 Example of fitting a circle to a set of points	262
19.18 Lines and conics	266
19.19 Homographies and conics	267
19.20 Degenerate conics	267
19.21 Circular points*	268
19.22 Degenerate dual conic of the circular points*	268
19.23 Transformation of the dual conic of the circular points*	269
20 Quaternions	270
20.1 Introduction	270
20.2 Quaternion sum, difference and product	270

20.3	Quaternion norm, conjugate and inverse	271
20.4	Vectors as quaternions	272
20.5	The quaternion as a 4-dimensional column vector	273
20.6	Matrix representation of the quaternion product*	273
20.7	Hamilton's Representation in complex form*	274
20.8	Unit quaternions	275
20.9	Representation of a rotation matrix by a unit quaternion	276
20.10	Shepperd's method for calculation of the unit quaternion	276
20.11	Rotation of a vector by quaternion conjugation	279
20.12	Vector form of the unit quaternion	279
20.13	Matrix representation of quaternion product for unit quaternions*	280
20.14	Composite rotations in terms of quaternions	280
20.15	The deviation between two quaternions	281
20.16	Kinematic differential equations for quaternions	281
20.17	Kinematic differential equations for composite rotations	283
20.18	Normalization of unit quaternions	284
20.19	Logarithm of a unit quaternion	284
20.20	Computation of the quaternion from the logarithm	285
20.21	Computation of the logarithm	285
20.22	Numerical integration of quaternions with Euler's method	286
20.23	Numerical integration of quaternions with exponential methods	287
20.24	The power of a quaternion	287
20.25	Interpolation of unit quaternions with SLERP	287
20.26	Derivation of the SLERP formula	288
20.27	Composite quaternion in the matrix form	289
20.28	The $\sin \theta \mathbf{k}$ vector	289
20.29	The Rodrigues vector	290
20.30	The modified Rodrigues vector	291
20.31	Kinematic differential equations for the Rodrigues vector	291
20.32	Kinematic differential equations for the modified Rodrigues parameters	291
20.33	Kinematic differential equations for the $\mathbf{k} \sin \theta$ vector	292
21	Exponential description of rotations	292
21.1	Exponential description of rotation matrices from angle and axis	292
21.2	Rotation of the logarithm	294
21.3	The norm of the logarithm	294
21.4	Computation of the exponential the rotation matrix	294
21.5	Computation of the logarithm the rotation matrix for $ \theta < \pi$	295
21.6	Computation of the logarithm the rotation matrix for $ \theta \leq \pi$	296
21.7	Position and velocity of a point mass	299
21.8	Rotation and angular velocity in $SO(3)$	299
21.9	The kinematic differential equation of the logarithm	300
22	Gradient search on $SO(3)$	303
22.1	Distances in $SO(3)$	303
22.2	Invariance of distance measures	304
22.3	Norms of the logarithm	304

22.4	Angular distance	305
22.5	Chordal distance	305
22.6	Quaternion distance*	306
22.7	The gradient in Euclidean space	306
22.8	The Riemannian metric on $SO(3)$	307
22.9	Distance induced by the Riemannian metric*	308
22.10	The gradient in $SO(3)$ based on the Riemannian metric	309
22.11	The gradient of the angular distance	310
22.12	The gradient of the angular distance of an error rotation	310
22.13	More on the gradient of the angular distance*	312
22.14	Rotation averaging for the angular distance	313
22.15	Rotation averaging for two rotation matrices	315
22.16	Rotation averaging for rotation matrices with the same rotation vector	315
23	Exponential description of displacements in $SE(3)$	316
23.1	Exponential description of a displacement	316
23.2	The exponential function on $SE(3)$	317
23.3	Chasles' theorem	318
24	Distance metrics on $SE(3)$	320
24.1	Introduction	320
24.2	$SO(3)$ as a Lie group	320
24.3	Bi-invariance of the Riemannian metric on $SO(3)$ *	321
24.4	Riemannian metrics on $SE(3)$	322
24.5	Distance functions on $SE(3)$	324
24.6	A closer look at the left-invariant distance metric on $SE(3)$	326
24.7	A closer look at the right-invariant distance metric on $SE(3)$	327
25	Determination of rotations and displacements from measurements	328
25.1	Calculation of rotation matrices in the orthogonal Procrustes problem	328
25.2	Derivation of the solution of the Procrustes problem	329
25.3	Number point correspondences in the Procrustes problem	332
25.4	A vector formulation of the Procrustes problem	333
25.5	The weighted Procrustes problem	333
25.6	Reflection and rotation matrices in the optimal Procrustes problem	334
25.7	Calculation of rigid displacement from two sets of 3D data	336
25.8	Calculation of displacement with weighting	339
25.9	Calculation of a similarity transform from two sets of 3D data	341
25.10	Rotation averaging for the chordal distance	343
25.11	Orthogonalization of an approximation of a rotation matrix	344
25.12	Recovery of rotation matrix from an estimate of two columns	344
25.13	Gradient search for the Procrustes problem	344
25.14	Kinematics of the hand-eye calibration problem	347
25.15	The Park-Martin solution to the hand-eye calibration problem	349
26	Point clouds	353
26.1	Computation of surface normals	353

26.2	Iterative closest point	355
26.3	Point-Feature Histogram (PFH)	359
26.4	Viewpoint feature histogram (VFH)	360
26.5	Radius-based Surface Descriptor (RSD)	360
27	PnP and PnL	362
27.1	P4P with 4 points in a plane	362
27.2	PnP with control points: EPnP	362
27.3	DLT-Plücker-Lines	365
28	Graduated non-convexity for outlier rejection	366
28.1	Introduction	366
28.2	Robust cost function	367
28.3	Truncated least squares	368
28.4	Graduated non-convexity cost function for truncated least squares	369
28.5	Minimization of the surrogate cost function with respect to the weights	370
28.6	Graduated non-convexity for the average of a set of real numbers	372
28.7	Graduated non-convexity for the Procrustes problem	374
29	Latent space	376
29.1	Introduction	376
29.2	Autoencoder	376
29.3	Principal Component Analysis (PCA)	376
29.3.1	Introduction	376
29.3.2	The empirical covariance matrix	377
29.3.3	Singular value decomposition	377
29.3.4	Singular value decomposition in matrix form	378
29.3.5	Latent space from principal components analysis	379
29.3.6	PCA in 3-dimensional space	380
A	Numerical optimization	381
A.1	Linear least-squares in the underdetermined case	381
A.2	Linear least-squares in the overdetermined case	382
A.3	Weighted least-squares	383
A.4	Damped linear least-squares	383
A.5	Formulation of the nonlinear least squares problem	385
A.6	Gradient search	387
A.7	The Gauss-Newton method for nonlinear least squares	387
A.8	Gauss-Newton method for a quadratic objective function	388
A.9	Gradient method for a quadratic objective function	389
A.10	The Levenberg-Marquardt method for nonlinear least squares	393
A.11	The Rosenbrock function	393
A.12	Optimization of Rosenbrock's function with scipy	396
B	Results from linear algebra	397
B.1	Singular value decomposition	397
B.2	Nullspace and range space	398

B.3	The singular value decomposition and matrix inversion	400
B.4	The thin SVD	400
B.5	Singular values and eigenvalues for a symmetric matrix	402
B.6	Polar decomposition	402
B.7	The square root of a matrix	403
B.8	The trace of a matrix	403
B.9	The trace of a quadratic form	404
B.10	The Frobenius inner product	405
B.11	The Frobenius norm	406
B.12	Vector norm	407
B.13	The induced matrix 2-norm	407
B.14	Quadratic forms and positive definite matrices	408
B.15	Linear least-squares solution in terms of the normal equations for the full rank case	408
B.16	The normal equations in terms of the singular value decomposition for the full rank case	409
B.17	The Rayleigh quotient	410
B.18	Upper and lower triangular matrices	411
B.19	The Cholesky decomposition	411
B.20	QR decomposition	412
B.21	Cross product relations	413

1 Introduction

These lecture notes are written for the course TPK4171 on Advance Industrial Robotics at The Norwegian University of Science and Technology (NTNU). The notes present geometric tools and methods for robot vision, where image data are used to determine the position and orientation of geometric objects in a scene. This includes the camera model, calibration techniques, perspective effects and stereo vision. Also mathematical tools for the description of points, lines and planes are presented. A detailed description of rotation and rigid body motion is presented, and optimization techniques for of rotation, rigid body motion and projective transformations are described. Matlab and Python scripts are included to demonstrate how computations can be implemented. Background material on linear algebra and numerical optimization methods is found in the appendix. The selection of material is based on the standard textbooks of Hartley and Zissermann [30] and Ma, Soatto, Košecká and Sastry [41] in combination with additional material on the underlying geometric methods, including homogeneous description of 2-dimensional and 3-dimensional projective geometry, and Plücker coordinates in the description of points, lines and planes based on [61] and [54]. In addition material on optimization techniques for use in registration problems and vision sensor systems is included.

2 Camera model

2.1 Introduction

A camera makes an image where a point in a 3-dimensional scene is imaged to a 2-dimensional point in an image. The scene is described as 3-dimensional Euclidean space, which is the real world we experience it. The camera model is the mapping from a point in the 3-dimensional scene to the 2-dimensional image plane. The mapping to a 2-dimensional image plane has two important effects. Firstly, there are perspective effects, which means that parallel lines in the 3-dimensional scene are not necessarily parallel in the image. Secondly, one dimension is lost, which typically means that the distance to the point in the scene is not possible to determine from one image, unless there is additional information about the object in the scene. The camera model used here is the pinhole model. The pinhole model is the standard camera model in computer vision, and captures the essential geometric transformations involved in the image formation using a camera.

2.2 The pinhole camera model

Consider a camera which is used to take images of a scene in the 3-dimensional real world. The camera frame c is fixed in the camera with the $x_c y_c$ plane as the focal plane containing the camera lens, and the z_c axis as the optical axis pointing out of the camera lens. The origin of the c frame is called the optical center. Let the position of a point P in the 3-dimensional scene be given by \vec{r}_{cp} relative to the origin of c . The coordinates of P in the c frame is given by

$$\mathbf{r}_{cp}^c = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \tilde{\mathbf{r}}_{cp}^c = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (1)$$

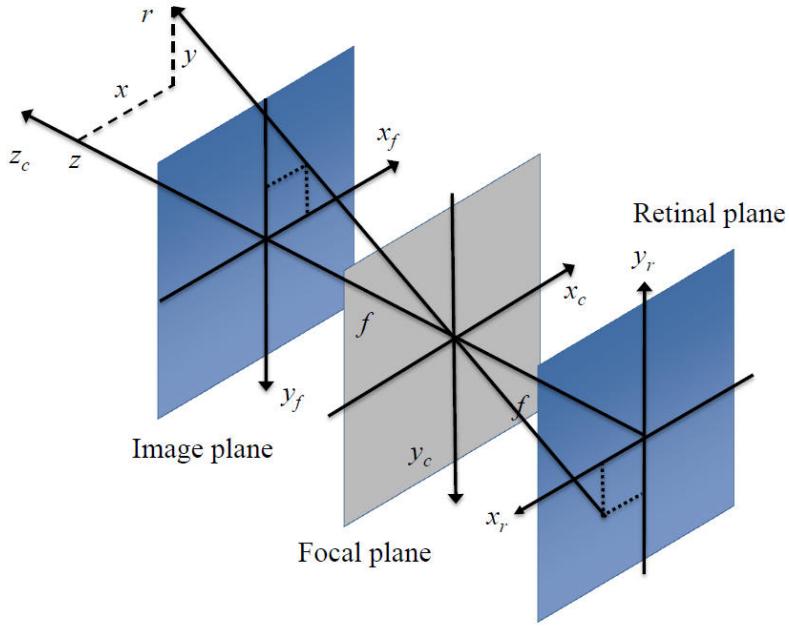


Figure 1: Retinal, focal and image planes. The image sensors are on the retinal plane, while the image plane is a virtual plane used to simplify the geometric model.

where \mathbf{r}_{cp}^c is the usual Euclidean coordinate vector, and $\tilde{\mathbf{r}}_{cp}^c$ is the corresponding homogeneous coordinate vector. When a camera makes an image of the point P , there are two effects, the perspective transformation from 3-dimensional space to the position in the 2-dimensional image plane, and the mapping from the position in the image plane to the pixel coordinates. The pixels are the smallest elements of the digital image, and each pixel is identified by its pixel coordinates. The pixel coordinates are integers that are the indices of the pixel in the image.

The camera model used here is the pinhole model. In this model the light is supposed to pass through the optical center, which is the origin of the camera frame, and is then projected on the retinal plane where the image sensors are located. The retinal plane is parallel to the focal plane, and is placed at the focal distance f in the negative z_c direction. This is in agreement with the thin-lens theory of optics. To simplify the geometric model, a virtual image plane is introduced. The image plane is in front of the camera frame, and is parallel to the focal plane at a distance f in the positive direction of the z_c axis. It is noted that the image plane is a virtual plane as opposed to the physical retina plane, which is behind the optical center. The image in the image plane will be the same as the image in the retinal plane, except that the image is rotated by an angle π about the z_c axis. The image plane, focal plane and retinal plane are shown in Figure 1.

The image of a point P with coordinates (x, y, z) in the camera frame will be a point with coordinates (x_f, y_f) in the image plane. Basic geometric considerations from Figure 2 leads to

$$\frac{x_f}{f} = \frac{x}{z}, \quad \frac{y_f}{f} = \frac{y}{z} \quad (2)$$

where f is the focal length, and z is the depth coordinate of the point. The expressions for

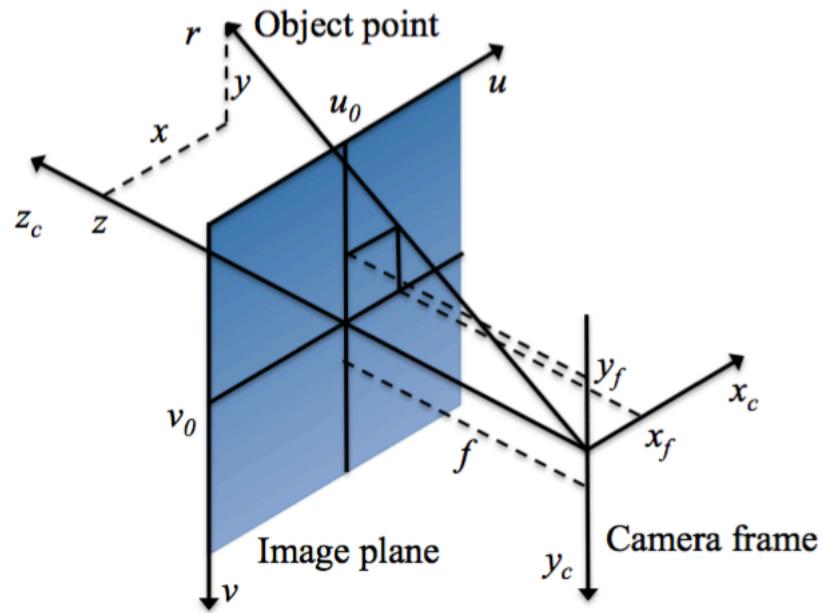


Figure 2: Image frame and camera frame c with image coordinates (x_f, y_f) .

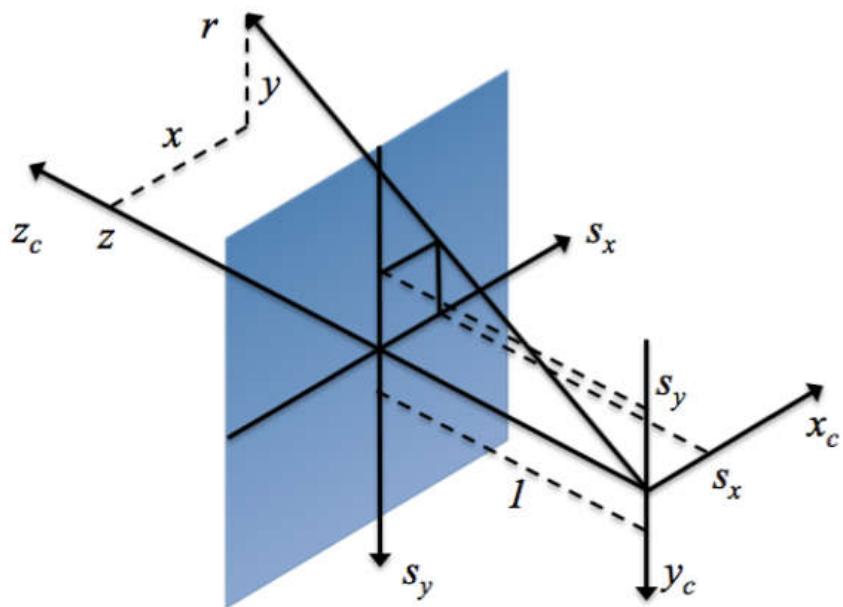


Figure 3: Normalized image coordinates $\tilde{s} = (s_x, s_y, 1)^T$.

the image coordinates in the image plane is therefore

$$x_f = f \frac{x}{z} \quad (3)$$

$$y_f = f \frac{y}{z} \quad (4)$$

$$z_f = f \quad (5)$$

where it is seen that the inverse of the depth coordinate z scales the image points. The mapping from the 3-dimensional scene to the 2-dimensional image plane is said to be a perspective projection, as will be explained in detail in a later chapter.

It is common practice to normalize the image coordinates by introducing yet another plane, which is called the normalized image plane. This plane is parallel to the image plane, and is positioned at a distance equal to 1 along the z_c axis as shown in Figure 3. The coordinates in this plane are called the normalized image coordinates which are given by

$$s_x = \frac{x}{z} \quad (6)$$

$$s_y = \frac{y}{z} \quad (7)$$

$$s_z = 1 \quad (8)$$

The position in the normalized image plane is given by the normalized image vector $\mathbf{s} = (s_x, s_y)^T$, which has the homogeneous form $\tilde{\mathbf{s}} = (s_x, s_y, 1)^T$ which is given by

$$\tilde{\mathbf{s}} = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} = \frac{1}{z} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{1}{z} \mathbf{r}_{cp}^c \quad (9)$$

Note that the homogeneous normalized image vector $\tilde{\mathbf{s}}$ is the Euclidean position \mathbf{r}_{cp}^c of the point in the scene scaled by the inverse of the depth coordinate z .

The pixel coordinates are integer coordinates for the image plane shown in Figure 4, and are given by

$$u = \frac{f}{\rho_x} \frac{x}{z} + u_0 \quad (10)$$

$$v = \frac{f}{\rho_y} \frac{y}{z} + v_0 \quad (11)$$

where ρ_x is the horizontal width of a pixel, ρ_y is the vertical height of a pixel, and u_0 and v_0 are the pixel coordinates of the z_c axis, which is the center of the image plane. The pixel coordinates are defined so that the origin is in the upper left corner of the image plane. This is shown in Figure 5. The pixel coordinates are given in vector form as

$$\mathbf{p} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad \tilde{\mathbf{p}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (12)$$

where the homogeneous form $\tilde{\mathbf{p}}$ is introduced. It will be seen that the homogeneous form is well suited for use in computations.

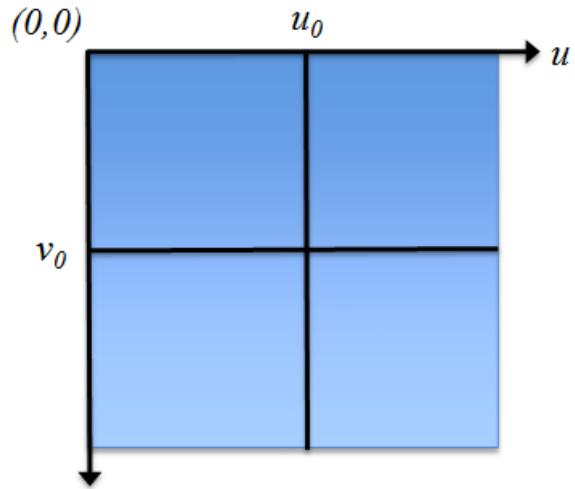


Figure 4: Pixel coordinates $\tilde{\mathbf{p}} = (u, v, 1)^T$.

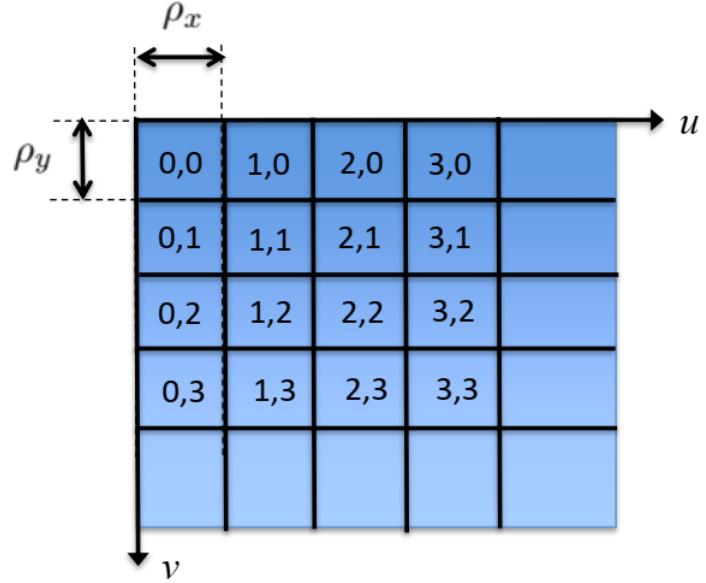


Figure 5: Pixels elements in the image plane. The pixels are light sensors arranged in a matrix. The width of a pixel is ρ_x and the height is ρ_y . The pixels are numbered as (u, v) .

The pixel coordinates are given by the normalized image coordinates as

$$u = \frac{f}{\rho_x} s_x + u_0 \quad (13)$$

$$v = \frac{f}{\rho_y} s_y + v_0 \quad (14)$$

Then it is possible to describe the transformation from the normalized image coordinates to the pixel coordinates as a linear transformation by using the homogeneous vectors $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{s}}$. The transformation is written

$$\tilde{\mathbf{p}} = \mathbf{K} \tilde{\mathbf{s}} \quad (15)$$

where $\tilde{\mathbf{p}} = (u, v, 1)^T$ is the pixel coordinate vector, $\tilde{\mathbf{s}} = (s_x, s_y, 1)^T$ is the normalized image coordinate vector and

$$\mathbf{K} = \begin{bmatrix} \frac{f}{\rho_x} & 0 & u_0 \\ 0 & \frac{f}{\rho_y} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (16)$$

is the camera parameter matrix. The elements of \mathbf{K} are given by the parameters f , ρ_y , ρ_x , u_0 and v_0 . These parameters are specific to the camera, and are referred to as intrinsic parameters. The mapping from $\tilde{\mathbf{s}}$ to $\tilde{\mathbf{p}}$ is an affine transformation, as will be explained later in this note.

To sum up, the camera model from the position vector $\mathbf{r}_{cp}^c = (x, y, z)^T$ to the homogeneous pixel coordinates $\tilde{\mathbf{p}} = (u, v, 1)^T$ is given by the two equations

$$\tilde{\mathbf{s}} = \frac{1}{z} \mathbf{r}_{cp}^c \quad (17)$$

$$\tilde{\mathbf{p}} = \mathbf{K} \tilde{\mathbf{s}} \quad (18)$$

The first equation is the nonlinear perspective projection from the the 3-dimensional scene to the normalized image plane. The second equation is the linear affine mapping from the normalized image coordinates to the pixel coordinates through the camera parameter matrix.

Example

Consider a point $\mathbf{r}_{cp1}^c = [x, y, z]^T = [1, 1, 1]^T$. Then $\tilde{\mathbf{s}}_1 = [\frac{x}{z}, \frac{y}{z}, 1]^T = [1, 1, 1]^T$.

Next, consider a point $\mathbf{r}_{cp2}^c = [1, 1, 2]^T$. Then $\tilde{\mathbf{s}}_2 = [\frac{x}{z}, \frac{y}{z}, 1]^T = [0.5, 0.5, 1]^T$.

The point $\mathbf{r}_{cp3}^c = [1, 1, 10]^T$ gives $\tilde{\mathbf{s}}_3 = [\frac{x}{z}, \frac{y}{z}, 1]^T = [0.1, 0.1, 1]^T$.

Finally, the point $\mathbf{r}_{cp4}^c = [0.1, 0.1, 1]^T$ gives $\tilde{\mathbf{s}}_4 = [\frac{x}{z}, \frac{y}{z}, 1]^T = [0.1, 0.1, 1]^T$.

Note that $\tilde{\mathbf{s}}_3 = \tilde{\mathbf{s}}_4$, even though $\mathbf{r}_{cp3}^c = 10\mathbf{r}_{cp4}^c$. This is an example of the general result that two points that are proportional in the camera frame cannot be distinguished based on their normalized image coordinates. \square

Comment

Note that the image of the Euclidean point $\mathbf{r}_{cp}^c \in \mathbb{R}^3$ in the scene has normalized image coordinates $\tilde{\mathbf{s}} = \frac{1}{z} \mathbf{r}_{cp}^c$. This means that if $\tilde{\mathbf{s}}$ is given and z is unknown, then the point \mathbf{r}_{cp}^c will be on the line $\mathbf{r}(\lambda) = \lambda \tilde{\mathbf{s}}$ where $\mathbf{r}(\lambda)$ is given in the coordinates of the camera frame c . \square

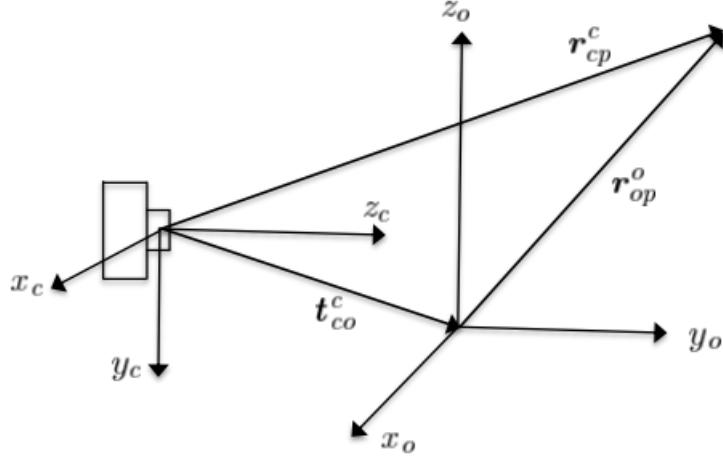


Figure 6: The camera frame c and the object frame o .

2.3 Object frame

The position of the object may be given in the object frame o by the position vector $\mathbf{r}_{op}^o = (x_o, y_o, z_o)^T$ from the origin of frame o to the point P in the coordinates of the o frame. Let the homogeneous transformation matrix from the camera frame to the object frame be

$$\mathbf{T}_o^c = \begin{bmatrix} \mathbf{R}_o^c & \mathbf{t}_{co}^c \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (19)$$

Here \mathbf{R}_o^c is the rotation matrix from c to o , and \mathbf{t}_{co}^c is the translation from the origin of the c frame to the origin of the o frame given in the coordinates of the c frame. This matrix can be used to calculate the position in camera frame as

$$\tilde{\mathbf{r}}_{cp}^c = \mathbf{T}_o^c \tilde{\mathbf{r}}_{op}^o \quad (20)$$

where $\tilde{\mathbf{r}}_{op}^o = (x_o, y_o, z_o, 1)^T$ is the homogeneous form of \mathbf{r}_{op}^o . This is shown in Figure 6.

It is recalled from (17) that the normalized image coordinates are given by $z\tilde{\mathbf{s}} = \mathbf{r}_{cp}^c$, where the non-homogeneous form \mathbf{r}_{cp}^c of the position vector is used. The transformation from the homogeneous vector $\tilde{\mathbf{r}}_{cp}^c$ to the corresponding Euclidean vector \mathbf{r}_{cp}^c is described with the projection matrix

$$\mathbf{\Pi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (21)$$

This matrix gives a transformation which simply removes the fourth element of a homogeneous vector. It is then possible to write

$$\mathbf{r}_{cp}^c = \mathbf{\Pi} \tilde{\mathbf{r}}_{cp}^c \quad (22)$$

which in coordinate form is written

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (23)$$

The normalized image coordinates can then be found from

$$\tilde{s} = \frac{1}{z} \mathbf{r}_{cp}^c = \frac{1}{z} \mathbf{\Pi} \tilde{\mathbf{r}}_{cp}^c = \frac{1}{z} \mathbf{\Pi} \mathbf{T}_o^c \tilde{\mathbf{r}}_{op}^o \quad (24)$$

In this case, the pinhole camera model is given by the two equations

$$\tilde{s} = \frac{1}{z} \mathbf{\Pi} \mathbf{T}_o^c \tilde{\mathbf{r}}_{op}^o \quad (25)$$

$$\tilde{p} = \mathbf{K} \tilde{s} \quad (26)$$

where z is the third coordinate of $\mathbf{r}_{cp}^c = \mathbf{\Pi} \mathbf{T}_o^c \tilde{\mathbf{r}}_{op}^o$. Then the elements of \mathbf{T}_o^c in the first equation are called the extrinsic parameters of the camera model, while the intrinsic parameters in \mathbf{K} appear in the second equation. The two equations of the camera model are often combined in one equation as

$$z \tilde{p} = \mathbf{K} \mathbf{\Pi} \mathbf{T}_o^c \tilde{\mathbf{r}}_{op}^o \quad (27)$$

This equation can be simplified by observing that $\mathbf{\Pi} \mathbf{T}_o^c = [\mathbf{R}_o^c \ \mathbf{t}_{co}^c]$ which gives the formulation

$$z \tilde{p} = \mathbf{C} \tilde{\mathbf{r}}_{op}^o \quad (28)$$

where

$$\mathbf{C} = \mathbf{K} [\mathbf{R}_o^c \ \mathbf{t}_{co}^c] \in \mathbb{R}^{3 \times 4} \quad (29)$$

is called the camera matrix. Note that the camera matrix \mathbf{C} contains both the intrinsic parameters in \mathbf{K} and the extrinsic parameters \mathbf{R}_o^c and \mathbf{t}_{co}^c .

Comment Some authors, e.g. [30], describe the translation between the camera frame and the object frame in terms of the vector \mathbf{t}_{oc}^o , which is the position of the camera frame relative to the object frame. Then $\mathbf{t}_{co}^c = -\mathbf{R}_o^c \mathbf{t}_{oc}^o$, and the camera matrix is $\mathbf{C} = \mathbf{K} [\mathbf{R}_o^c \ -\mathbf{R}_o^c \mathbf{t}_{oc}^o]$.

□

Example

Let the position of the point P in the object frame is $\mathbf{r}_{op}^o = (0, 0, 1)^T$ while the displacement of the object frame o relative to the camera frame c is

$$\mathbf{T}_o^c = \begin{bmatrix} \mathbf{R}_o^c & \mathbf{t}_{co}^c \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{R}_o^c = \mathbf{R}_x\left(\frac{\pi}{6}\right) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.866 & -0.5 \\ 0 & 0.5 & 0.866 \end{bmatrix}, \quad \mathbf{t}_{co}^c = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \quad (30)$$

The position of the object in the camera frame is

$$\tilde{\mathbf{r}}_{cp}^c = \mathbf{T}_o^c \tilde{\mathbf{r}}_{op}^o = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.866 & -0.5 & 1 \\ 0 & 0.5 & 0.866 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.5 \\ 2.866 \\ 1 \end{bmatrix} \quad (31)$$

It is seen that the depth coordinate, which is the z coordinate of $\tilde{\mathbf{r}}_{cp}^c$, is $z = 2.866$. Then the normalized image coordinates are found to be

$$\tilde{s} = \frac{1}{z} \mathbf{r}_{cp}^c = \frac{1}{2.866} \begin{bmatrix} 0 \\ 0.5 \\ 2.866 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.1745 \\ 1 \end{bmatrix} \quad (32)$$

```

# Computation of normalized coordinates
import numpy as np

def skewm(r):
    return np.array([[0, -r[2], r[1]], [r[2], 0, -r[0]], [-r[1], r[0], 0]])

def angle_axis_2_R(r, theta):
    S = skewm(r/np.linalg.norm(r))
    return np.eye(3) + np.sin(theta)*S + (1-np.cos(theta))*S @ S

def rot_trans_2_T(R, t):
    return np.block([[R, t.reshape(3, 1)],
                    [np.array([0, 0, 0, 1])]])

# Input data
ro = np.array([0, 0, 1])
Rco = angle_axis_2_R(np.array([1, 0, 0]), np.pi/6)
tco = np.array([0, 1, 2])
# Computation
ro_h = np.block([ro, 1])
Tco = rot_trans_2_T(Rco, tco)
rc_h = Tco @ ro_h
s_h = rc_h[0:3]/rc_h[2]
print(s_h)

```

2.4 Camera parameters

Consider a digital camera with focal length $f = 15.0 \text{ mm} = 0.0150 \text{ m}$, and square pixels of dimension $10 \mu\text{m}$, that is, $\rho_y = \rho_x = 10 \times 10^{-6} \text{ m}$. The camera has 1280 pixels in the horizontal direction and 1024 pixels in the vertical direction, and the pixel coordinates are $(u, v) = (0, 0)$ in the upper left corner of the image plane. The optical axis goes through the center of the image plane, which means that the optical axis has pixel coordinates $u_0 = 640$ and $v_0 = 512$. In this case the camera parameter matrix is

$$\mathbf{K} = \begin{bmatrix} 1500 & 0 & 640 \\ 0 & 1500 & 512 \\ 0 & 0 & 1 \end{bmatrix} \quad (33)$$

This camera has 1,310,720 pixels, which is about 1.3 megapixels. The VGA standard has 640 by 480 pixels, which gives an aspect ratio of 4:3 and approximately 0.3 megapixels. In comparison a HD video camera can have a vertical resolution of 720 and a horizontal resolution of 1280, which gives an aspect ratio of 16:9 and 921,600 pixels, that is, just below 1 megapixel. This HD camera will have three image planes in an RGB arrangement where one is for red (R), one for green (G) and one for blue (B), and each image plane channel will have resolution 720×1280 . A HD standard with higher resolution is the HD video format with a vertical resolution of 1080 pixels and a horizontal resolution of 1920 pixels, which again is an aspect ratio of 16:9. This gives 2,073,600 pixels for each color, which gives about 6 megapixels for the complete RGB image.

2.5 Calculation of pixel coordinates

Consider the following numerical example: It is given that the camera parameter matrix is

$$\mathbf{K} = \begin{bmatrix} 1500 & 0 & 640 \\ 0 & 1500 & 512 \\ 0 & 0 & 1 \end{bmatrix} \quad (34)$$

This is the camera parameter matrix for a camera with 1280×1024 pixels. The position of the object in the object frame is $\mathbf{r}_{op}^o = (0.1, 0, 0)^T$ while the displacement of the object frame o relative to the camera frame c is

$$\mathbf{T}_o^c = \begin{bmatrix} \mathbf{R}_o^c & \mathbf{t}_{co}^c \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{R}_o^c = \mathbf{I}, \quad \mathbf{t}_{co}^c = \begin{bmatrix} 0 \\ 0 \\ 0.5 \end{bmatrix} \quad (35)$$

The position of the object in the camera frame is

$$\tilde{\mathbf{r}}_{cp}^c = \mathbf{T}_o^c \tilde{\mathbf{r}}_{op}^o = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0 \\ 0.5 \\ 1 \end{bmatrix} \quad (36)$$

It is seen that $z = 0.5$. Then the normalized image coordinates are found to be

$$\tilde{\mathbf{s}} = \frac{1}{z} \mathbf{r}_{cp}^c = \frac{1}{0.5} \begin{bmatrix} 0.1 \\ 0 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0 \\ 1 \end{bmatrix} \quad (37)$$

Finally, the pixel coordinates are found from

$$\tilde{\mathbf{p}} = \mathbf{K} \tilde{\mathbf{s}} = \begin{bmatrix} 1500 & 0 & 640 \\ 0 & 1500 & 512 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 940 \\ 512 \\ 1 \end{bmatrix} \quad (38)$$

```
# Computation of pixel coordinates
import numpy as np
# Input data
ro = np.array([0.1, 0, 0])
Tco = np.eye(4)
Tco[2][3] = 0.5
K = np.array([[1500, 0, 640], [0, 1500, 512], [0, 0, 1]])
# Computation
ro_h = np.block([ro, 1])
rc_h = Tco @ ro_h
s_h = rc_h[0:3]/rc_h[2]
p_h = K @ s_h
print(p_h)
```

2.6 Calculation of normalized coordinates from pixel coordinates

If the pixel coordinates are given, then the homogeneous normalized coordinates can be computed from

$$\tilde{s} = \mathbf{K}^{-1} \tilde{p} \quad (39)$$

The inverse camera parameter matrix is given by

$$\mathbf{K}^{-1} = \frac{1}{f} \begin{bmatrix} \rho_x & 0 & -\rho_x u_0 \\ 0 & \rho_y & -\rho_y v_0 \\ 0 & 0 & f \end{bmatrix} \quad (40)$$

Let the inverse camera parameter matrix be

$$\mathbf{K}^{-1} = \begin{bmatrix} 1500 & 0 & 640 \\ 0 & 1500 & 512 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 0.0007 & 0 & -0.4267 \\ 0 & 0.0007 & -0.3413 \\ 0 & 0 & 1 \end{bmatrix} \quad (41)$$

Insertion of the corner points $\tilde{p}_1 = (0, 0, 1)^T$ and $\tilde{p}_2 = (1280, 1024, 1)^T$ in $\tilde{s} = \mathbf{K}^{-1} \tilde{p}$ gives $\tilde{s}_1 = (-0.4267, -0.3413, 1)^T$ and $\tilde{s}_2 = (0.4267, 0.3413, 1)^T$, which shows that the range of the normalized image coordinates are

$$-0.4267 \text{ m} \leq s_x \leq 0.4267 \text{ m} \quad (42)$$

$$-0.3413 \text{ m} \leq s_y \leq 0.3413 \text{ m} \quad (43)$$

```
# Calculation of corner points
import numpy as np
# Input data
p1_h = np.array([0, 0, 1])
p2_h = np.array([1280, 1024, 1])
K = np.array([[1500, 0, 640], [0, 1500, 512], [0, 0, 1]])
# Computation
s1_h = np.linalg.solve(K, p1_h)
s2_h = np.linalg.solve(K, p2_h)
print(s1_h, s2_h)
```

2.7 Base frame, camera frame and object frame

The position and orientation of the camera frame with respect to the base frame b is given by the homogeneous matrix

$$\mathbf{T}_c^b = \begin{bmatrix} \mathbf{R}_c^b & \mathbf{t}_{bc}^b \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (44)$$

where \mathbf{R}_c^b is the rotation matrix from frame b to frame c , and \mathbf{t}_{bc}^b is the position of frame c with respect to frame b in the coordinates of the b frame.

The position and orientation of the object frame o with respect to the camera frame c is given by

$$\mathbf{T}_o^c = \begin{bmatrix} \mathbf{R}_o^c & \mathbf{t}_{co}^c \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (45)$$

where \mathbf{R}_o^c is the rotation matrix from frame c to frame o , and \mathbf{t}_{co}^c is the position of frame o with respect to frame c in the coordinates of the c frame.

The position and orientation of the object frame o is also given with respect to the base frame b as

$$\mathbf{T}_o^b = \begin{bmatrix} \mathbf{R}_o^b & \mathbf{t}_{bo}^b \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (46)$$

Then the position and orientation of the object frame o with respect to the origin of the base frame b is found from

$$\mathbf{T}_o^b = \mathbf{T}_c^b \mathbf{T}_o^c \quad (47)$$

In the case that the object is given as a point P , then only the homogeneous position vector $\tilde{\mathbf{P}}_{bp}^b$ is used to describe the object. The position of the point P with respect to the origin of the base frame b is then found to be

$$\tilde{\mathbf{r}}_{bp}^b = \mathbf{T}_c^b \tilde{\mathbf{r}}_{cp}^c \quad (48)$$

2.8 The image Jacobian*

Consider the equation

$$z\tilde{\mathbf{s}} = \mathbf{r}_{cp}^c \quad (49)$$

Time differentiation gives

$$\dot{z}\tilde{\mathbf{s}} + z\dot{\tilde{\mathbf{s}}} = \dot{\mathbf{r}}_{cp}^c \quad (50)$$

The velocity of the point P relative to the camera frame c is the time derivative of the position vector \mathbf{r}_{cp}^c in the non-moving base frame b , that is,

$$\mathbf{v}_{cp}^b = \frac{d}{dt}(\mathbf{r}_{cp}^b) = \dot{\mathbf{r}}_{cp}^b \quad (51)$$

In equation (50) the position vector is differentiated in the c frame as

$$\frac{d}{dt}(\mathbf{r}_{cp}^c) = \dot{\mathbf{r}}_{cp}^c \quad (52)$$

which is not a velocity, but merely the time derivative of a position vector in a moving frame. To connect the equations it is noted that $\mathbf{r}_{cp}^c = \mathbf{R}_b^c \mathbf{r}_{cp}^b$. Time differentiation gives

$$\dot{\mathbf{r}}_{cp}^c = \mathbf{R}_b^c \dot{\mathbf{r}}_{cp}^b + \dot{\mathbf{R}}_b^c \mathbf{r}_{cp}^b = \mathbf{R}_b^c \mathbf{v}_{cp}^b + \mathbf{S}(\boldsymbol{\omega}_{cb}^c) \mathbf{R}_b^c \mathbf{r}_{cp}^b \quad (53)$$

Insertion of $\boldsymbol{\omega}_{bc}^c = -\boldsymbol{\omega}_{cb}^c$ gives.

$$\dot{\mathbf{r}}_{cp}^c = \mathbf{v}_{cp}^c - \mathbf{S}(\boldsymbol{\omega}_{bc}^c) \mathbf{r}_{cp}^c \quad (54)$$

The velocity can be expressed by the velocity of the point P and the velocity of the camera as

$$\mathbf{v}_{cp}^c = \mathbf{v}_p - \mathbf{v}_c^c \quad (55)$$

In the case that the object is at rest and the camera is moving, the velocity will be

$$\mathbf{v}_{cp}^c = -\mathbf{v}_c^c \quad (56)$$

and

$$\dot{\mathbf{r}}_{cp}^c = -\mathbf{v}_c^c - \mathbf{S}(\boldsymbol{\omega}_{bc}^c)\mathbf{r}_{cp}^c \quad (57)$$

The coordinate forms are

$$\mathbf{v}_c^c = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}, \quad \boldsymbol{\omega}_{bc}^c = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (58)$$

Insertion in (50) this gives

$$\dot{z} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} + z \begin{bmatrix} \dot{x} \\ \dot{y} \\ 0 \end{bmatrix} = - \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} - \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \begin{bmatrix} zx \\ zy \\ z \end{bmatrix} \quad (59)$$

The last row gives

$$\dot{z} = -(v_z + \omega_x zy - \omega_y zx) \quad (60)$$

Insertion of this equation in the the two first rows gives

$$z\dot{x} = -(v_x + \omega_y z - \omega_z zy) + x(v_z + \omega_x zy - \omega_y zx) \quad (61)$$

$$z\dot{y} = -(v_y + \omega_z zx - \omega_x z) + y(v_z + \omega_x zy - \omega_y zx) \quad (62)$$

which is written

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \mathbf{L}(\mathbf{s}, z) \begin{bmatrix} \mathbf{v}_c^c \\ \boldsymbol{\omega}_{bc}^c \end{bmatrix} \quad (63)$$

where $\mathbf{L}(\mathbf{s}, z)$ is the interaction matrix

$$\mathbf{L}(\mathbf{s}, z) = \begin{bmatrix} -\frac{1}{z} & 0 & \frac{x}{z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{z} & \frac{y}{z} & (1+y^2) & -xy & -x \end{bmatrix} \quad (64)$$

3 RANSAC and the Hough transform

3.1 Determination of a line from noisy points

Point	1	2	3	4	5	6	7
x	0	1	2	3	3	4	10
y	0	1	2	2	3	4	2

Table 1: Points in the dataset of Fischler and Bolles [24].

RANSAC is a powerful tool for fitting data with models. The method was introduced by [24], and was called Random Sample Consensus, which has been abbreviated to RANSAC. The introductory example used in [24] is line fitting with least-squares to the 7 data points in Table 1, which are plotted in Figure 7. Here the obvious interpretation is that the points are on the line $y = x$, except for a small error in point 4, and a gross error in point 7, which can be called a wild point. It turns out that the usual linear regression methods based on least-squares have problems with these data points due to the wild point. Regression methods will usually have a heuristic method for eliminating wild points. In the following it is shown that an apparently clever method for wild-point elimination fails for this data-set, and the resulting solution gives a bad fit to the data.

3.2 Linear regression to determine the line

To demonstrate the potential problems of the data-set a linear regression with least-squares is done. The line is assumed to be given by

$$y = ax + b \quad (65)$$

It is assumed that there is some noise or inaccuracy involved, so each of the n data-point is supposed to satisfy

$$y_i - ax_i - b = v_i \quad (66)$$

Linear regression is then used to minimize the cost function

$$C = \sum_{i=1}^n v_i^2 = \sum_{i=1}^n (y_i - ax_i - b)^2 \quad (67)$$

Let the data be represented by

$$\mathbf{X} = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad (68)$$

and let the line parameters be represented by $\boldsymbol{\alpha} = [a, b]^T$. Then the cost function is given by

$$C = (\mathbf{X}\boldsymbol{\alpha} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\alpha} - \mathbf{y}) \quad (69)$$

The least squares solution is then found from the normal equations $(\mathbf{X}^T\mathbf{X})\boldsymbol{\alpha} = \mathbf{X}^T\mathbf{y}$, which gives

$$\boldsymbol{\alpha} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \quad (70)$$

A nominal solution is calculated using all the points, and then the point that is furthest away from the line is eliminated. Then a second solution is then calculated from the remaining points. As shown in Table 2, the first iteration eliminates point 6, and not the wild point 7. In the next iteration the point 5 is eliminated, and then point 1 is eliminated. The resulting solution is based on the points 2, 3, 4 and 7, which includes the wild point 7. The solution gives a bad fit to the data set.

Jupyter code

```
# %%
# Script for regression in the line example of Fischler & Bolles (1981)
import numpy as np
import matplotlib.pyplot as plt

# %%
x0 = np.array([0., 1., 2., 3., 3., 4., 10.])
y0 = np.array([0., 1., 2., 2., 3., 4., 2.])

# %%
x = np.array([0., 1., 2., 3., 3., 4., 10.])
```

```

y = np.array([0., 1., 2., 2., 3., 4., 2.])
Xr = np.vstack([x, np.ones(x.shape)]).T
alpha = np.linalg.inv(Xr.T @ Xr) @ Xr.T @ y
alpha

# %%
x = np.array([0., 1., 2., 3., 3., 10.])
y = np.array([0., 1., 2., 2., 3., 2.])
Xr = np.vstack([x, np.ones(x.shape)]).T
alpha = np.linalg.inv(Xr.T @ Xr) @ Xr.T @ y
alpha

# %%
x = np.array([0., 1., 2., 3., 10.])
y = np.array([0., 1., 2., 2., 2.])
Xr = np.vstack([x, np.ones(x.shape)]).T
alpha = np.linalg.inv(Xr.T @ Xr) @ Xr.T @ y
alpha

# %%
plt.figure(1)
plt.figure(1).clear()
plt.plot(x0, y0, 'rd')
x = np.arange(-1,12)
plt.plot(x, alpha[0]*x + alpha[1], '-b')
x_axis = np.block([[[-1,12], [0,0]]])
y_axis = np.block([[0,0], [-1,6]]);
plt.plot(x_axis[0,:], x_axis[1,:], 'k', y_axis[0,:], y_axis[1,:], 'k')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.show()

```

3.3 RANSAC for determination of the line

In the RANSAC approach two points are selected at random to define a candidate line, and then it is computed how many points are sufficiently close to this candidate line. Points that are close to the line are termed inlier points on the line. Then, the candidate line with the most inlier is selected as the solution. It is seen from Table 3 that iteration 1 and 2 has the most inliers, and the resulting solution $y = x$ is therefore selected.

MATLAB example

In this example a line is found from two points (x_1, y_1) and (x_2, y_2) as

$$y = ax + b \quad (71)$$

where

$$a = \frac{y_2 - y_1}{x_2 - x_1}, \quad b = -ax_1 + y_1 \quad (72)$$

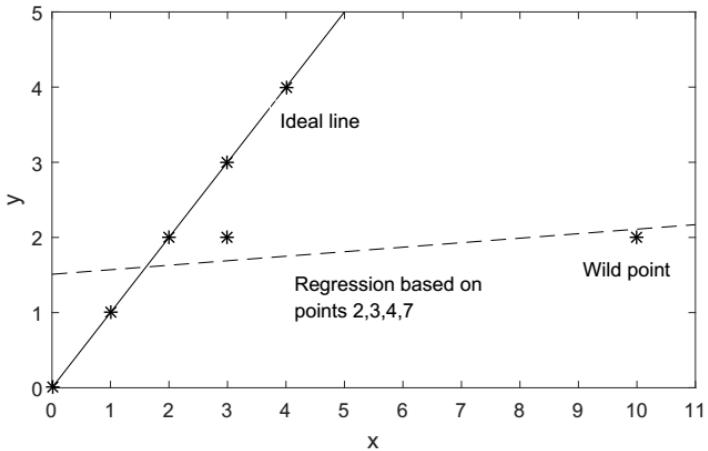


Figure 7: Least-squares regression of a line to 7 points, where one of points is a wild point. A heuristic method is used to eliminate the point with the largest residual. After 4 iterations points 1, 5 and 6 have been eliminated, and a poor regression results.

Iteration	Data set	Line from least-squares	Point furthest from line
1	1,2,3,4,5,6,7	$y = 1.48 + 0.16x$	6
2	1,2,3,4,5,7	$y = 1.25 + 0.13x$	5
3	1,2,3,4,7	$y = 0.96 + 0.14x$	1
4	2,3,4,7	$y = 1.51 + 0.06x$	-

Table 2: Solution of regression with linear least-squares in combination with wild point elimination

The distance from a point (x_i, y_i) to a line $y = ax + b$ is found from

$$\delta = \frac{ax_i - y_i + b}{\sqrt{a^2 + 1}} \quad (73)$$

This is explained in detail in Section 4.5. For a candidate line computed from a pair of points, a point will be an inlier if the distance is less than a given tolerance. The candidate line that has the most inliers is selected as the line to be found.

```
# Script for the line example in Fischler & Bolles (1981)
import numpy as np
import matplotlib.pyplot as plt

X = np.array([[0,0], [1,1], [2,2], [3,2], [3,3], [4,4], [10,2]]).T
ind = np.array([[0,1], [0,2], [0,3], [0,4], [1,2], [1,6], [2,3]]).T

npoint = X.shape[1]; ncomb = ind.shape[1]
X_inlier = np.zeros((npoint, ncomb))
a = np.zeros(ncomb); b = np.zeros(ncomb)
N_inlier = np.zeros(ncomb)
```

Iteration	Initial data set	Generated line	Inliers	# inliers
1	1,2	$y = x$	1,2,3,5,6	5
2	1,3	$y = x$	1,2,3,5,6	5
3	1,4	$y = 0.67x$	1,2,4	3
4	1,7	$y = 0.2x$	1,7	2
5	3,4	$y = 2$	3,4 7	3

Table 3: Solution of regression using RANSAC. Iteration 1 and 2 has the most inliers, and the resulting solution $y = x$ is therefore selected.

```

maxinliers = 0; j_maxinliers = 0
for j in range(ncomb):
    a[j] = (X[1,ind[1,j]] - X[1,ind[0,j]]) / (X[0,ind[1,j]] - X[0,ind[0,j]])
    b[j] = -a[j] * X[0,ind[0,j]] + X[1,ind[0,j]]
    i_inlier = 0
    for i in range(npoint):
        delta = a[j] * X[0,i] - X[1,i] + b[j]/np.sqrt(a[j]**2 + 1)
        if abs(delta) < 0.1:
            X_inlier[i_inlier,j] = i + 1
            i_inlier = i_inlier + 1
    N_inlier[j] = i_inlier
    if i_inlier > maxinliers:
        maxinliers = i_inlier
        j_maxinliers = j
L = np.block([[a], [b]])
n_inliers = N_inlier[j_maxinliers]
Line = L[:,j_maxinliers]
inliers = X_inlier[:,j_maxinliers]
print('Line with the most inliers: \n', Line, '\n'
      + 'Inliers for best solution: \n', inliers)

plt.figure(1)
plt.figure(1).clear()
x = np.arange(-1,7,0.01)
plt.plot(X[0,:], X[1,:], 'rd')
plt.plot(x, x*Line[0] + Line[1])
x_axis = np.block([[-1,11], [0,0]])
y_axis = np.block([[0,0], [-1,7]]);
plt.plot(x_axis[0,:], x_axis[1,:], 'k', y_axis[0,:], y_axis[1,:], 'k')
plt.axis([-1, 11, -1, 7.5])
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.show()

```

3.4 Determination of a circle from noisy points

Consider the problem where n points \mathbf{x}_i , $i = 1, \dots, n$ in the plane are given. It is known that many of these points are on a circle in the plane. The data are noisy, so there are also a significant number of points that are not on the circle, and that may even be far away from the circle. It is not known which points are on the circle. Moreover, it is not known where the circle is, and what the radius is.

In the following example circle given by $\mathbf{c} = (a, d, e, f)$ is computed from 3 points on the circle with determinants as explained in Section 19.6. The circle can be drawn by computing the origin (x_0, y_0) from

$$x_0 = -\frac{d}{2a}, \quad y_0 = -\frac{e}{2a} \quad (74)$$

and the radius from

$$r = \sqrt{\frac{d^2 + e^2 - 4af}{4a}} \quad (75)$$

RANSAC can be used for this problem. The method is

Initialize: $i = 0$, $k = 0$
for $i = 1 : N$

- Randomly select 3 of the n points
- Determine a circle \mathbf{c}_i so that these points are on the circle
- Find the number k_i of inliers that are sufficiently close to the circle, and store the indices of inliers \mathbf{m}_i
- If $k_i > k$ then the circle is $\mathbf{c} = \mathbf{c}_i$, $k = k_i$ and $m = m_i$.

The result is the circle given by \mathbf{c} , which has inliers with indices given by the index vector \mathbf{m} .

Example

RANSAC implementation to demonstrate how a circle can be found. Input: 9 points where it is given that some of the points are on a circle that is to be found. The resulting circle is shown in Figure 8.

```
import numpy as np
import matplotlib.pyplot as plt

X = np.array([[1, 0, 1], [0, 1, 1], [0, -1, 1], [-2, -1.5, 1], [-1, 0, 1],
              [-1, 1, 1], [2, 2, 1], [0.5, 0.8660, 1], [0.7071, 0.7071, 1]]).T
ind = np.array([[1, 3, 4], [1, 2, 3], [1, 4, 5],
                [2, 3, 4], [2, 4, 5], [3, 4, 5], [3, 4, 6]]).T
ind = ind - 1

npoint = X.shape[1]; ncomb = ind.shape[1]
X_inlier = np.zeros((npoint, ncomb))
C = np.zeros((4,ncomb))
```

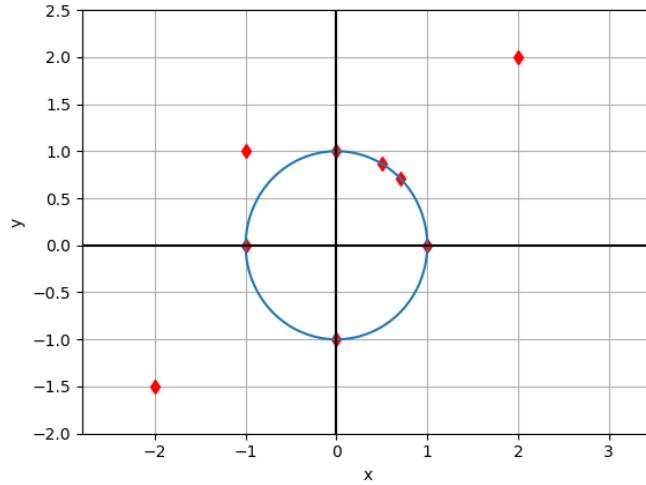


Figure 8: Circle found from 9 points with RANSAC.

```

N_inlier = np.zeros(ncomb)
maxinliers = 0; j_maxinliers = 0
for j in range(ncomb):
    A = np.zeros((3,4))
    for i in range(3):
        ii = ind[i,j]
        A[i,:] = np.array([X[0,ii]**2 + X[1,ii]**2, X[0,ii]*X[2,ii],
                           X[1,ii]*X[2,ii], X[2,ii]**2])
    c = np.array([np.linalg.det(np.delete(A,0,1)),
                  - np.linalg.det(np.delete(A,1,1)),
                  np.linalg.det(np.delete(A,2,1)),
                  - np.linalg.det(np.delete(A,3,1))])
    C[:,j] = c/c[0]

    i_inlier = 0
    for i in range(npoint):
        a = np.array([X[0,i]**2 + X[1,i]**2, X[0,i]*X[2,i],
                      X[1,i]*X[2,i], X[2,i]**2])
        delta = np.dot(a,C[:,j])
        if abs(delta) < 0.1:
            X_inlier[i_inlier,j] = i + 1
            i_inlier = i_inlier + 1
    N_inlier[j] = i_inlier
    if i_inlier > maxinliers:
        maxinliers = i_inlier
        j_maxinliers = j

n_inliers = N_inlier[j_maxinliers]

```

```

circle = C[:,j_maxinliers]
inliers = X_inlier[:,j_maxinliers]
print('Circle [a, d, e, f]:\n', circle, '\n'
      + 'Inliers for best solution: \n', inliers)

x0 = - circle[1]/(2*circle[0])
y0 = - circle[2]/(2*circle[0])
r2 = (circle[1]**2 + circle[2]**2 - 4*circle[0]*circle[3])/(4*circle[0])
r = np.sqrt(r2)

plt.figure(1)
plt.figure(1).clear()
x = np.arange(0,6.28,0.01)
plt.plot(X[0,:], X[1,:], 'rd')
plt.plot(r*np.cos(x), r*np.sin(x))
x_axis = np.block([[-3,4], [0,0]])
y_axis = np.block([[0,0], [-3,3]]);
plt.plot(x_axis[0,:], x_axis[1,:], 'k', y_axis[0,:], y_axis[1,:], 'k')
plt.axis([-1.4*2, 1.4*2.5, -2, 2.5])
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.show()

```

3.5 The Hough transform of a line

The Hough transform is an alternative method for determining a line from point data. Consider the line

$$y = ax + b \quad (76)$$

To explain the basic idea of the Hough transform we will first consider the idea of plotting a point in the (x, y) plane as a set of lines in the (a, b) plane. A point $\mathbf{p}_i = (x_i, y_i)^T$ will be on all the lines given by $y_i = ax_i + b$ where (x_i, y_i) is fixed and a and b are variables. This means that the point \mathbf{p}_i defines a set of lines $b = -x_i a + y_i$ in the a, b plane. Another point (x_j, y_j) will define another set of lines $b = -x_j a + y_j$. If one line with parameters (a_0, b_0) belongs to both sets, then this line must be through both points.

It turns out to be better to use another parameterization of a line. The reason for this is that the line description $y = ax + b$ is undefined for a vertical line $a \rightarrow \infty$. Instead the parameterization of a line is set to

$$\rho = x \cos \theta + y \sin \theta \quad (77)$$

where ρ is the distance from the origin to the line, and θ is the angle between the x axis and the normal \mathbf{n} to the line. This is based on the description

$$n_x x + n_y y - \rho = \mathbf{n}^T \mathbf{p} - \rho = 0 \quad (78)$$

of the line as shown in Figure 9, where $\mathbf{n} = [n_x, n_y]^T$ is the unit normal of the line, and $\rho = \mathbf{n}^T \mathbf{p}$ is the distance from the origin to the line in the direction of the unit normal \mathbf{n} . Then $\cos \theta = n_x$ and $\sin \theta = n_y$, which gives the expression (77) for the line.

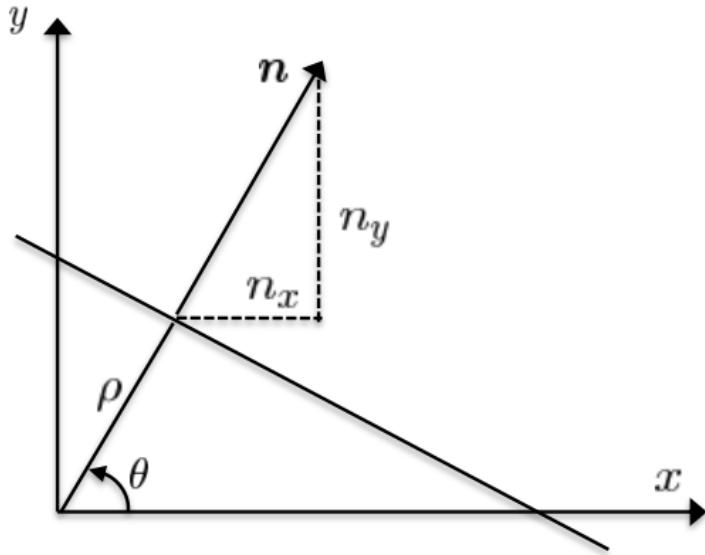


Figure 9: Definition of a line in terms of θ and ρ for use in the Hough transformation.

Then all the lines through the point $\mathbf{p}_i = [x_i, y_i]^T$ are given by the sinusoidal curve

$$\rho = x_i \cos \theta + y_i \sin \theta \quad (79)$$

This means that for a given angle θ , the line through $\mathbf{p}_i = [x_i, y_i]^T$ will have distance ρ from the origin given by (79). In the same way all the lines through a point $\mathbf{p}_j = [x_j, y_j]^T$ will be given by the curve

$$\rho = x_j \cos \theta + y_j \sin \theta \quad (80)$$

Then, if for a given angle θ_0 , the distance ρ_0 is the same for the points \mathbf{p}_i and \mathbf{p}_j , then the curves for both points in the parameter diagram will intersect at (θ_0, ρ_0) . This means that the line (θ_0, ρ_0) goes through both points. If the curves of n points intersect at (θ_0, ρ_0) , then that means that all the corresponding n points are on the line (θ_0, ρ_0) . The line with the most points on it can then be identified in the Hough diagram as the point where the curves from the most points intersect.

The points in Table 1 correspond to the following set of curves:

Point	Curve in parameter plane
1	$\rho = 0$
2	$\rho = \cos \theta + \sin \theta$
3	$\rho = 2 \cos \theta + 2 \sin \theta$
4	$\rho = 3 \cos \theta + 2 \sin \theta$
5	$\rho = 3 \cos \theta + 3 \sin \theta$
6	$\rho = 4 \cos \theta + 4 \sin \theta$
7	$\rho = 10 \cos \theta + 2 \sin \theta$

The intersections are given by

Intersection	ρ	θ	Line	Points
1	0	$\frac{3\pi}{4}$	$y = x$	1,2,3,5,6
2	2	$\frac{\pi}{2}$	$y = 2$	3,4,7

It is seen from Figure 10 that two lines are identified. The line with 5 points is $y = x$, while the line with 3 points is $y = 2$. The resulting lines are shown in Figure 11.

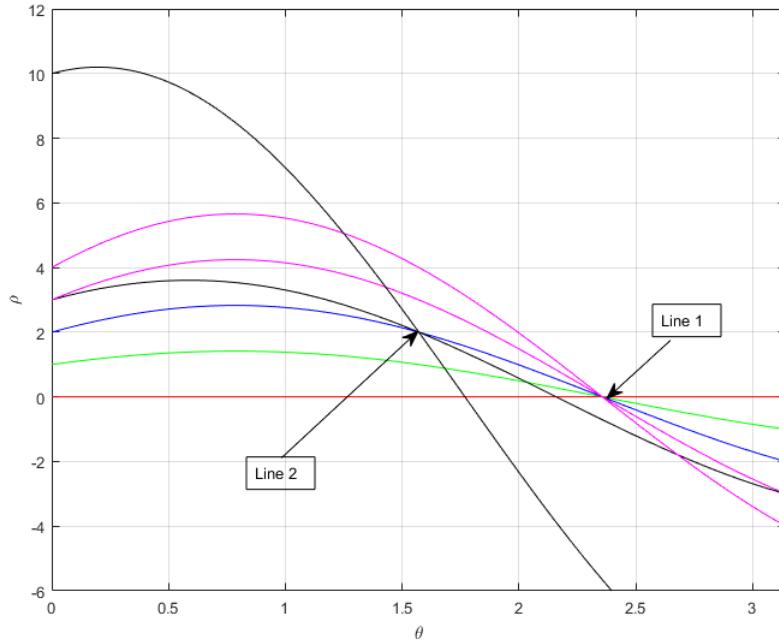


Figure 10: Curves in the parameter plane corresponding to the points in the spatial plane. The five colored curves of point 1, 2, 3, 5 and 6 intersect at $\rho = 0$ and $\theta = 3\pi/4$, which corresponds to the line $y = x$. The three curves of points 3, 4 and 7 intersect at $\rho = 2$ and $\theta = \pi/2$, which is the line $y = 2$.

MATLAB Example

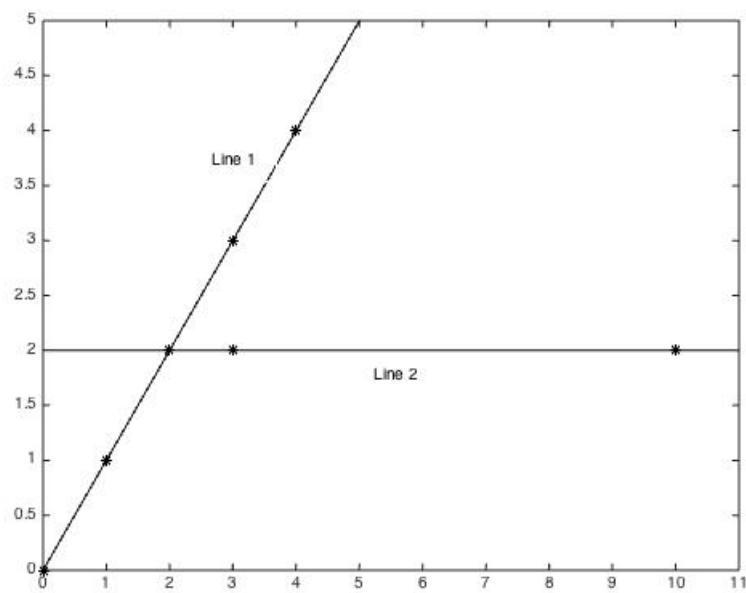


Figure 11: Two lines found from the Hough transformation. Line 1 has 5 points and Line 2 has 3 points on the line.

```

t, P(1,6)*cos(t)+P(2,6)*sin(t), 'm', ...
t, P(1,7)*cos(t)+P(2,7)*sin(t), 'k')
axis([0 pi -6 12]);
% The line parameters are found where the maximum number of curves
% intersect

```

4 Projective geometry in two dimensions

4.1 Introduction

Projective geometry is an important tool in computer vision. The usual geometric description that is used to characterize our surroundings is the Euclidean geometry, which dates back to Euclid's *Elements*, which was written around 300 BC in Alexandria. This is the familiar geometric description which is used in robot kinematics and dynamics. In robotic vision it turns out that it is useful to introduce the more general projective geometry. An early result in projective geometry is Pappus's theorem, which dates back to 300 AD. Projective geometry was then developed by renaissance artists who introduced perspective to have a correct representation of depth in their paintings. An important effect that was discovered was the concept of parallel which lines intersect at a vanishing point, as in *The School of Athens* by Raphael in 1508-1511 [41]. An algebraic treatment of projective geometry was developed in the 19th century based on the use of homogeneous coordinates, and the link between projective geometry and transformations was established in the *Erlangen Programme* of Felix Klein in 1872 [61]. Felix Klein worked with the Norwegian mathematician Sophus Lie, who became professor in Leipzig in 1886 when he replaced Felix Klein, who had moved to Göttingen. Lie developed the theory of continuous transformation groups, that became known as Lie groups. Projective geometry has developed in the last decades from being a theoretically interesting topic to a practical and widely used tool in computer vision. In the same way Lie groups have become widely used in robotics after the first adoption around 1990.

4.2 Homogeneous coordinates 2D

A point in the 2-dimensional Euclidean plane \mathbb{R}^2 can be described with the Euclidean vector $\mathbf{p} = [x, y]^T \in \mathbb{R}^2$ where x and y are the coordinates of the point. In vision algorithms it is useful to introduce the projective space \mathbb{P}^2 to represent the Euclidean geometry. Then a point is described in terms of homogeneous coordinates as

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathbb{P}^2 \quad (81)$$

This homogeneous point represents the Euclidean point

$$\mathbf{p} = \frac{1}{x_3} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2 \quad (82)$$

where $x = x_1/x_3$ and $y = x_2/x_3$. It is seen that the homogeneous point \mathbf{x} , which is given as a vector of homogeneous coordinates, will represent the same Euclidean point \mathbf{p} in \mathbb{R}^2 as long as x_1/x_3 and x_2/x_3 are unchanged, or, equivalently, as long as the ratios $x_1 : x_2 : x_3$ are unchanged. In particular, we may select $x_3 = 1$, which gives the usual homogeneous form $\tilde{\mathbf{p}}$ of \mathbf{p} , which is written

$$\mathbf{x} = \tilde{\mathbf{p}} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (83)$$

It is noted that all points $\mathbf{x} = \lambda[x, y, 1]^T$ will represent the same point $\mathbf{p} = [x, y]^T \in \mathbb{R}^2$ for all $\lambda \neq 0$. The homogeneous vector $[0, 0, 0]^T$ undefined.

4.3 Rotation of a vector

A Euclidean position vector $\mathbf{p} = [p_x, p_y]^T \in \mathbb{R}^2$ is rotated by an angle θ to a Euclidean position vector $\mathbf{q} = [q_x, q_y]^T \in \mathbb{R}^2$ according to

$$\mathbf{q} = \mathbf{R}\mathbf{p} \quad (84)$$

where \mathbf{R} is the rotation matrix

$$\mathbf{R} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \in SO(2) \quad (85)$$

Here $SO(2)$ is the special orthogonal group of order 2, which is a Lie group defined by

$$SO(2) = \{\mathbf{R} \in \mathbb{R}^{2 \times 2} \mid \det \mathbf{R} = 1 \text{ and } \mathbf{R}\mathbf{R}^T = \mathbf{I}\} \quad (86)$$

4.4 Displacement of a vector

The displacement in the form of a rigid motion of a Euclidean position vector $\mathbf{p} \in \mathbb{R}^2$ to a Euclidean position vector $\mathbf{q} \in \mathbb{R}^2$ is given by the affine transformation

$$\mathbf{q} = \mathbf{R}\mathbf{p} + \mathbf{t} \quad (87)$$

where $\mathbf{R} \in SO(2)$ is a rotation matrix and $\mathbf{t} \in \mathbb{R}^2$ is a translation. In this case it is useful to introduce the homogeneous vectors $\tilde{\mathbf{p}} = [\mathbf{p}^T, 1]^T$ and $\tilde{\mathbf{q}} = [\mathbf{q}^T, 1]^T$, which represent the Euclidean position vectors \mathbf{p} and \mathbf{q} . Then the displacement can then be written in terms of the homogeneous vectors as the linear transformation

$$\tilde{\mathbf{q}} = \mathbf{T}\tilde{\mathbf{p}} \quad (88)$$

where

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(2) \quad (89)$$

is a homogeneous transformation matrix, and $SE(2)$ is a Lie group.

4.5 Lines in 2D

It is well known from elementary geometry that a line ℓ in the Euclidean plane is given by the set of points $\mathbf{p} = [x, y]^T \in \mathbb{R}^2$ that satisfy

$$ax + by + c = 0. \quad (90)$$

Note that the description $ax + by + c = 0$ for a line is more general than the description $y = Ax + B$, where $A = -a/b$ and $B = -c/b$, which is not defined for $b = 0$.

The geometric interpretation is that the line is normal to the vector

$$\mathbf{n} = \begin{bmatrix} a \\ b \end{bmatrix}, \quad (91)$$

which is called the normal vector of the line. The distance from the origin to the line in the direction of \mathbf{n} is found from $\mathbf{n}^T \mathbf{p} = -c$ to be

$$d = \frac{\mathbf{n}^T \mathbf{p}}{|\mathbf{n}|} = -c/|\mathbf{n}|, \quad (92)$$

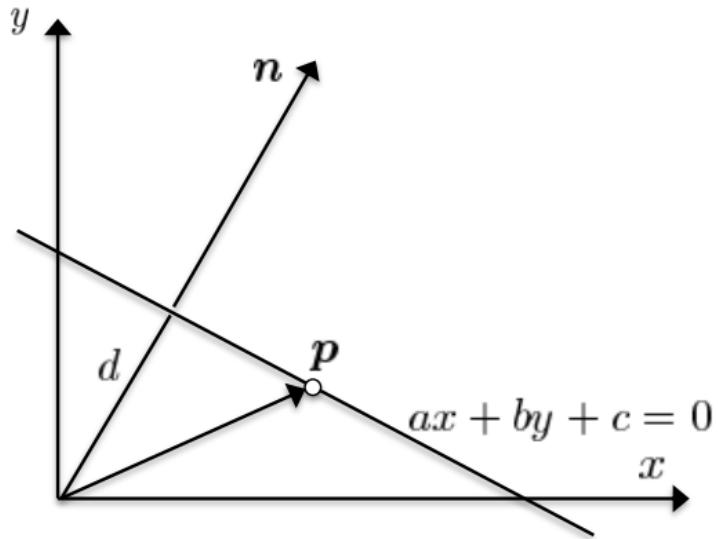


Figure 12: The distance $d = \mathbf{n}^T \mathbf{p} / |\mathbf{n}|$ from the origin to the line in the direction of the normal vector \mathbf{n} .

This is shown in Figure 12.

The direction vector along the line is

$$\mathbf{a} = \begin{bmatrix} b \\ -a \end{bmatrix} \quad (93)$$

which is orthogonal to \mathbf{n} .

Example 1

The line $y = x$ can be written $x - y = 0$. The normal vector is $\mathbf{n} = [1, -1]^T$, and $c = 0$. The direction vector is $\mathbf{a} = [1, 1]^T$. The distance from the origin is $d = 0$, which is illustrated in Figure 13a. \square

Example 2

The line $y = 2x + 1$ can be written $2x - y + 1 = 0$. The normal vector is $\mathbf{n} = [2, -1]^T$, which has the length $|\mathbf{n}| = \sqrt{5}$. The direction vector is $\mathbf{a} = [1, 2]^T$, and $c = 1$. It is found that $\mathbf{n}^T \mathbf{p} = -1$ for any point \mathbf{p} on the line, e.g., for $\mathbf{p} = [0, 1]^T$. The distance from the origin to the line in the direction of \mathbf{n} is $d = -1/\sqrt{5} = -0.4472$. Note that d is negative as the distance is in the direction of $-\mathbf{n}$, which is seen from Figure 13b. \square

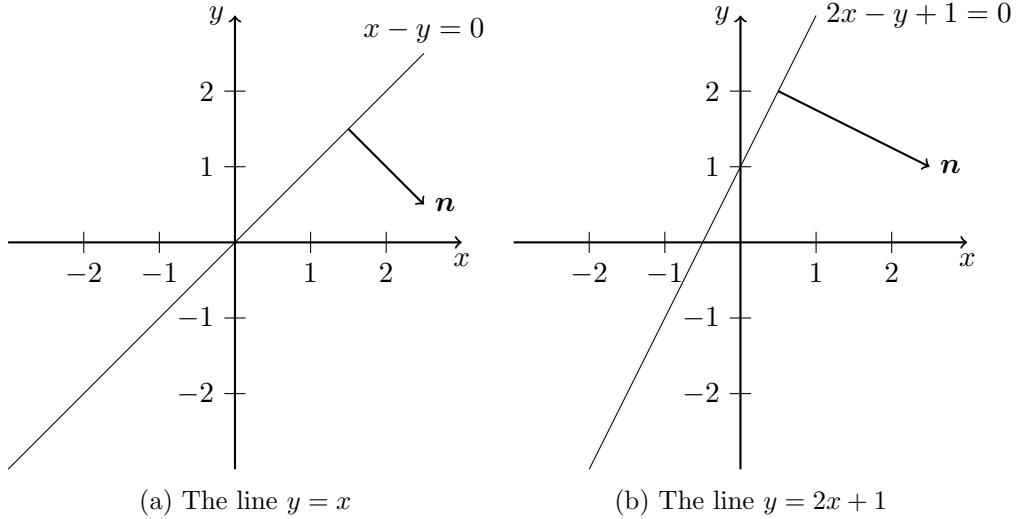


Figure 13: Examples of lines

4.6 Homogeneous description of lines

Additional results on lines in the plane can be developed by describing the line ℓ with homogeneous coordinates as

$$\ell = \begin{bmatrix} a \\ b \\ c \end{bmatrix}. \quad (94)$$

Then a homogeneous point $\mathbf{x} = [x, y, 1]^T$ is on the line if the point and the line are orthogonal in homogeneous coordinates, that is,

$$\ell^T \mathbf{x} = 0. \quad (95)$$

which is equivalent to (90). Obviously, this will still be valid if the vectors ℓ or \mathbf{x} are scaled by multiplication with a scalar.

The orthogonality of a line and the points on the line gives some interesting results. In particular, if the point \mathbf{x} is orthogonal to both of the two lines ℓ_1 and ℓ_2 , then the point is on both lines, which means that \mathbf{x} is the point of intersection between the two lines. This means that the point

$$\mathbf{x} = \ell_1 \times \ell_2$$

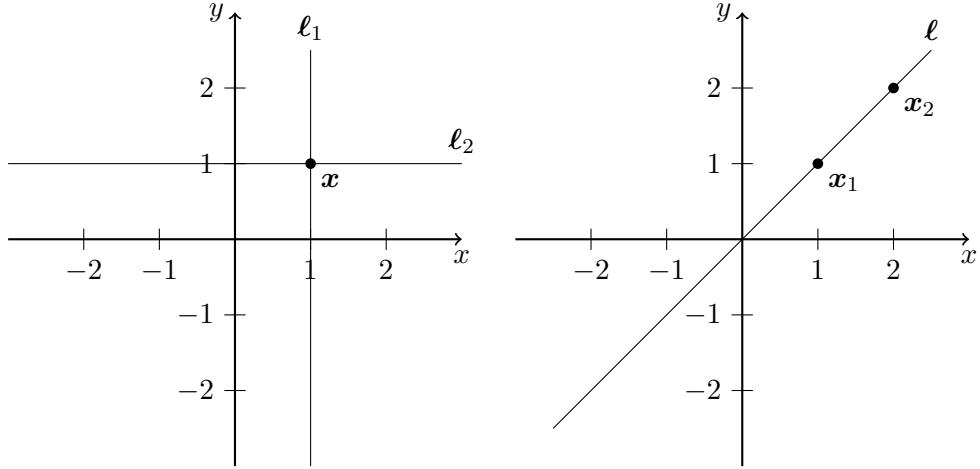
will be at the intersection point of the two lines because it is orthogonal to both of the lines ℓ_1 and ℓ_2 .

Consider the case where two points \mathbf{x}_1 and \mathbf{x}_2 are given in homogeneous coordinates. Then the two points will be on the line ℓ whenever the line is orthogonal to both points. This means that if the line is given by

$$\ell = \mathbf{x}_1 \times \mathbf{x}_2$$

then both points \mathbf{x}_1 and \mathbf{x}_2 will be on the line ℓ .

Example 1



(a) Intersection of the lines $x = 1$ and $y = 1$ at $x = 1, y = 1$. (b) The line through the points $\mathbf{x}_1 = [1, 1, 1]^T$ and $\mathbf{x}_2 = [2, 2, 1]^T$.

Figure 14: Intersection of lines and points

Consider the lines $x = 1$ and $y = 1$, which are given in homogeneous coordinates as $\ell_1 = [1, 0, -1]^T$ and $\ell_2 = [0, 1, -1]^T$. The two lines intersect at

$$\mathbf{x} = \ell_1 \times \ell_2 = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad (96)$$

which is the homogeneous form of the point $[x, y] = [1, 1]$. \square

Example 2

Consider the points $[x, y] = [1, 1]$ and $[x, y] = [2, 2]$, which are given in homogeneous coordinates as $\mathbf{x}_1 = [1, 1, 1]^T$ and $\mathbf{x}_2 = [2, 2, 1]^T$. The line through the two points is

$$\ell = \mathbf{x}_1 \times \mathbf{x}_2 = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \quad (97)$$

which is the homogeneous form of the line $-x + y = 0$ which can also be written in the form $y = x$. This is clearly the line through \mathbf{x}_1 and \mathbf{x}_2 . \square

4.7 Distance from a point to a line

A line $\ell = [a, b, c]^T$ where $\mathbf{n} = [a, b]^T$ is the normal vector of the line. Let $\mathbf{x} = [\mathbf{p}^T, 1]^T = [x, y, 1]^T$ be the homogeneous vector to the point \mathbf{p} on the line. The origin is given by the homogeneous vector $\mathbf{x}_0 = [0, 0, 1]^T$. Then

$$\ell^T \mathbf{x}_0 = c \quad (98)$$

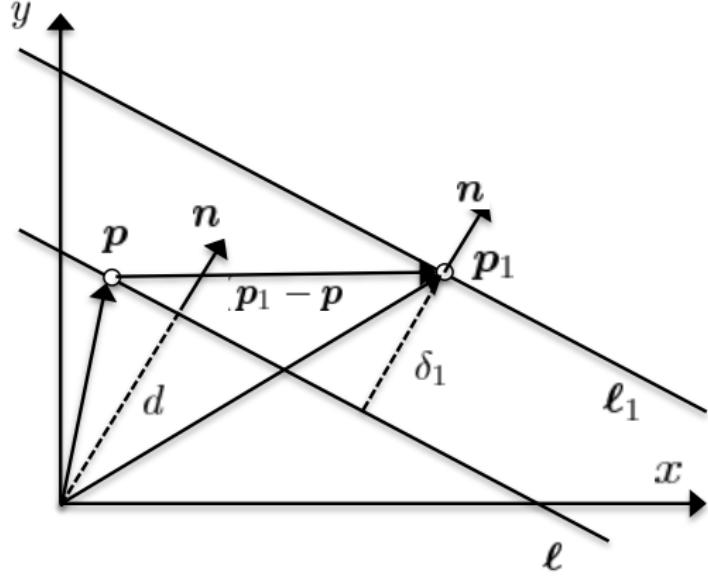


Figure 15: The distance $\delta_1 = \mathbf{n}^T(\mathbf{p} - \mathbf{p}_1)/|\mathbf{n}| = -\ell^T \mathbf{x}_1/|\mathbf{n}|$ from the point \mathbf{x}_1 to the line ℓ with the positive direction defined by the normal vector \mathbf{n} .

The distance d from the line to the origin in direction of \mathbf{n} is then

$$d = -\frac{\mathbf{n}^T \mathbf{p}}{|\mathbf{n}|} = \frac{c}{|\mathbf{n}|} = \frac{\ell^T \mathbf{x}_0}{|\mathbf{n}|} \quad (99)$$

A second line is given by $\ell_1 = [a, b, c_1]^T$, which is parallel to ℓ as shown in Figure 15. This line will be at a distance

$$d_1 = \frac{\mathbf{n}^T \mathbf{p}_1}{|\mathbf{n}|} = -\frac{c_1}{|\mathbf{n}|} \quad (100)$$

from the origin in the direction of \mathbf{n} . Next it is used that $\mathbf{n}^T \mathbf{p}_1 + c_1 = 0$, which gives

$$\ell^T \mathbf{x}_1 = \mathbf{n}^T \mathbf{p}_1 + c = c - c_1 \quad (101)$$

It can then be concluded that the distance δ_1 from the line ℓ to the point \mathbf{x}_1 in the direction of \mathbf{n} is given by

$$\delta_1 = d_1 - d = -\frac{c_1}{|\mathbf{n}|} + \frac{c}{|\mathbf{n}|} \quad (102)$$

This gives the result

$$\delta_1 = \frac{\ell^T \mathbf{x}_1}{|\mathbf{n}|} \quad (103)$$

which is the distance from ℓ to \mathbf{x}_1 in the direction of n .

Example 1:

An alternative expression which is seen from Figure 15 is

$$\delta_1 = \frac{\mathbf{n}^T(\mathbf{p} - \mathbf{p}_1)}{|\mathbf{n}|} \quad (104)$$

where $\mathbf{x} = [\mathbf{p}^T, 1]^T$ is an arbitrary point on the line. The distance δ_1 is positive in the direction of \mathbf{n} .

Example 2:

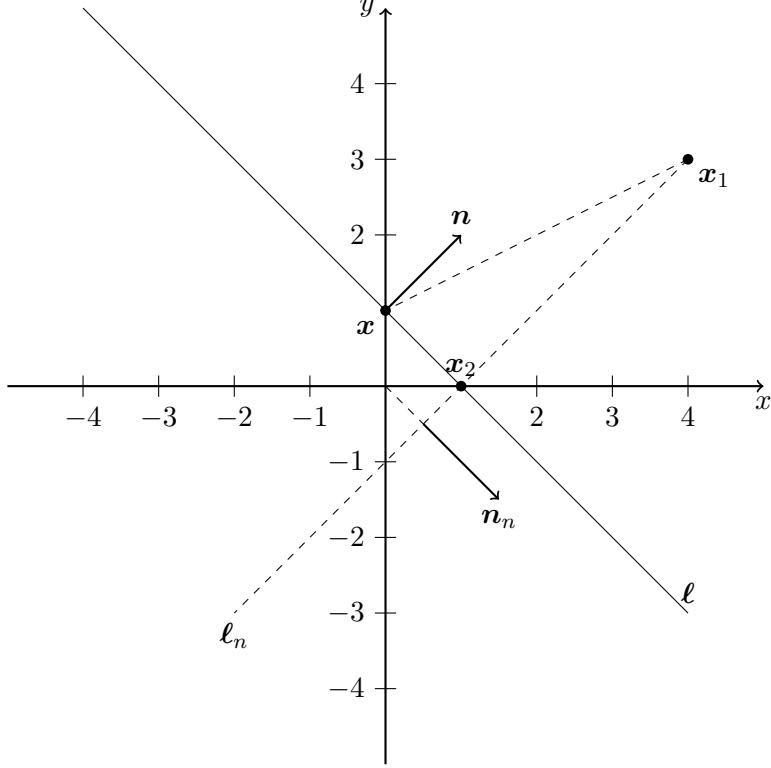


Figure 16: Intersection of lines and points

Consider the line $\ell = [1, 1, -1]^T$ which is the line $y = -x + 1$. The normal vector is $\mathbf{n} = [1, 1]^T$, which gives $|\mathbf{n}| = \sqrt{2}$. Then the point $\mathbf{p}_1 = [4, 3]^T$ corresponding to the homogeneous form $\mathbf{x}_1 = [4, 3, 1]^T$ has the distance

$$\delta_1 = \frac{\ell^T \mathbf{x}_1}{|\mathbf{n}|} = \frac{[1, 1, -1]^T [4, 3, 1]}{\sqrt{1+1}} = \frac{6}{\sqrt{2}} = 4.2426$$

from the line to the point in the direction of \mathbf{n} .

The line ℓ_n through \mathbf{x}_1 that is perpendicular to ℓ will have direction vector $\mathbf{a}_n = \mathbf{n} = [1, 1]^T$, since it is normal to ℓ . The normal vector of the line is therefore $\mathbf{n}_n = [1, -1]^T$. The distance from the origin to the line ℓ_n in the direction of \mathbf{n}_n will be

$$\frac{\mathbf{p}_1^T \mathbf{n}_n}{|\mathbf{n}_n|} = \frac{[4, 3][1, -1]^T}{|\mathbf{n}_n|} = \frac{1}{|\mathbf{n}_n|} \quad (105)$$

This gives the line $\ell_n = [1, -1, -1]^T$. The intersection between ℓ_n and $\ell = [1, 1, -1]^T$ is found to be

$$\mathbf{x}_2 = -\ell \times \ell_n = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \\ -2 \end{bmatrix} = -2 \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad (106)$$

where the homogeneous vector is scaled so that the third component is unity, which gives the Euclidean point of intersection $\mathbf{p}_2 = [1, 0]^T$ as the two first components of the scaled vector $[1, 0, 1]^T$. \square

4.8 Lines through the origin

A line through the origin will be given by $\ell = [a, b, 0]^T$, where $\mathbf{n} = [a, b]^T$ is the normal vector of the line. Then a Euclidean point $\mathbf{p} = [x, y]^T$ with homogeneous representation $x = [\mathbf{p}^T, 1]^T$ will be on the line if and only if $\ell^T x = \mathbf{p}^T \mathbf{n} = 0$.

Let $\mathbf{p}_1 = [x_1, y_1]^T$ be a Euclidean point with homogeneous representation $\mathbf{x}_1 = [x_1, y_1, 1]^T$. The distance from the line $\ell = [\mathbf{n}^T, 0]^T$ to the point \mathbf{p}_1 is then

$$d_1 = \frac{\mathbf{n}^T \mathbf{p}_1}{|\mathbf{n}|} = \frac{\ell^T \mathbf{x}_1}{|\mathbf{n}|} \quad (107)$$

4.9 A closer look at the homogeneous representation of point and lines in 2D*

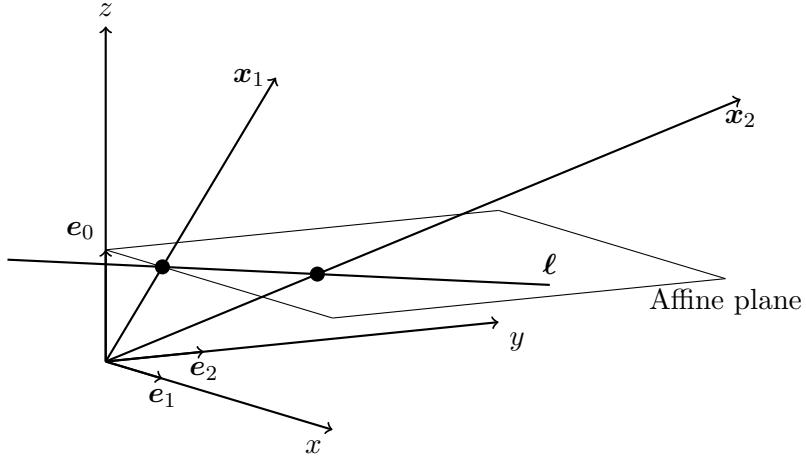


Figure 17: Two points are defined by the intersection of the vectors $\mathbf{x}_1 = \lambda_1[1, 0, 1]^T$ and $\mathbf{x}_2 = \lambda_2[2, 1, 1]^T$ and the affine plane defined by $z = 1$. The plane $\lambda\ell$ spanned by \mathbf{x}_1 and \mathbf{x}_2 intersects the affine plane in the line described by ℓ .

In the Euclidean plane a point P can be described in terms of the coordinates $[x, y]$, which can be represented by a vector $\mathbf{p} = [x, y]^T$. This vector starts at the origin and ends in the point P .

In the homogeneous representation one additional dimension is introduced. To describe this it is convenient to use the orthogonal unit vectors \mathbf{e}_1 and \mathbf{e}_2 in the xy plane so that the vector $\mathbf{p} = [x, y]^T$ is written $\mathbf{p} = x\mathbf{e}_1 + y\mathbf{e}_2$. The third dimension is introduced by adding the unit vector \mathbf{e}_0 which is along the z direction so that $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_0$ are the orthogonal unit vectors of a right-handed coordinate frame with center at the origin. The next step is to represent the vector \mathbf{p} by the homogeneous vector

$$\mathbf{x} = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_0 \quad (108)$$

where $x_1 = x_3 x$ and $x_2 = x_3 y$. Then if $x_3 = 1$, this vector is

$$\mathbf{x} = x\mathbf{e}_1 + y\mathbf{e}_2 + \mathbf{e}_0 \quad (109)$$

Note that $\mathbf{x} - \mathbf{e}_0 = \mathbf{p}$, that is, the vector \mathbf{p} is recovered by subtracting \mathbf{e}_0 from the homogeneous representation \mathbf{x} . The geometric interpretation of this is based on the intersection of the vector \mathbf{x} and the affine plane, which is the plane defined by $x_3 = 1$. This plane is parallel to the $x_1 x_2$ plane, which is the same as the Euclidean xy plane.

Consider a plane through the origin in the homogeneous space with with normal vector $\ell = [a, b, c]^T$. If \mathbf{x}_1 and \mathbf{x}_2 are two distinct points on the plane, then the cross product of the homogeneous vectors \mathbf{x}_1 and \mathbf{x}_2 will be normal to the plane. This means that $\mathbf{x}_1 \times \mathbf{x}_2 = \lambda \ell$ for some nonzero λ .

The intersection of this plane and the affine plane will be a line with normal vector $\mathbf{n} = [a, b]^T$. The intersection between the vectors \mathbf{x}_1 and \mathbf{x}_2 will then be two points on this line.

Consider two planes through the origin of the homogeneous space with normal vectors ℓ_1 and ℓ_2 . Then the intersection of these planes will be a line through the origin. This line is represented by the homogeneous vector \mathbf{x} . The vector will be in both planes, and it follows that $\mathbf{x}^T \ell_1 = \mathbf{x}^T \ell_2 = 0$, and $\ell_1 \times \ell_2 = \mathbf{x}$. The intersection between each of the two planes and the affine plane will constitute a line, and intersection of the line \mathbf{x} with the affine plane will be a point that is on the intersection of the two lines in the affine plane.

4.10 Linear dependence of homogeneous and Euclidean points*

Consider three homogeneous points be $\mathbf{x} = [x_1, x_2, x_3]^T$, $\mathbf{y} = [y_1, y_2, y_3]^T$ and $\mathbf{z} = [z_1, z_2, z_3]^T$, and suppose that $\mathbf{z} = \alpha \mathbf{x} + \beta \mathbf{y}$. Then

$$z_1 = \alpha x_1 + \beta y_1 \quad (110)$$

$$z_2 = \alpha x_2 + \beta y_2 \quad (111)$$

$$z_3 = \alpha x_3 + \beta y_3 \quad (112)$$

The corresponding Euclidean points $\mathbf{p} = [p_1, p_2]^T = (1/x_3)[x_1, x_2]^T$, $\mathbf{q} = [q_1, q_2]^T = (1/y_3)[y_1, y_2]^T$ and $\mathbf{r} = [r_1, r_2]^T = (1/z_3)[z_1, z_2]^T$ will then satisfy

$$r_1 = (\alpha x_1 + \beta y_1)/z_3 = \alpha(x_3/z_3)p_1 + \beta(y_3/z_3)q_1 \quad (113)$$

$$r_2 = (\alpha x_2 + \beta y_2)/z_3 = \alpha(x_3/z_3)p_2 + \beta(y_3/z_3)q_2 \quad (114)$$

which gives $\mathbf{r} = \alpha(x_3/z_3)\mathbf{p} + \beta(y_3/z_3)\mathbf{q}$. It is concluded that if the homogeneous points \mathbf{x} , \mathbf{y} and \mathbf{z} are linearly dependent, then the corresponding Euclidean points \mathbf{p} , \mathbf{q} and \mathbf{r} will also be linearly dependent.

4.11 Points and lines in terms of determinants*

Consider three vectors $\mathbf{x} = [x_1, x_2, x_3]^T$, $\mathbf{y} = [y_1, y_2, y_3]^T$ and $\mathbf{z} = [z_1, z_2, z_3]^T$ in a three dimensional space. Then the triple scalar product of the three vectors is

$$V = \mathbf{x} \cdot (\mathbf{y} \times \mathbf{z}) = \mathbf{y} \cdot (\mathbf{z} \times \mathbf{x}) = \mathbf{z} \cdot (\mathbf{x} \times \mathbf{y}) \quad (115)$$

Consider the expression $\mathbf{z} \cdot (\mathbf{x} \times \mathbf{y})$. This is the volume of the parallelepiped defined by the three vectors. If the vector \mathbf{z} has a component in the positive direction of $\mathbf{x} \times \mathbf{y}$, then the volume V is positive, otherwise it is negative.

The triple scalar product can be evaluated as the determinant

$$\mathbf{x} \cdot (\mathbf{y} \times \mathbf{z}) = \begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{vmatrix} \quad (116)$$

This is easily verified by expanding the determinant with the first row as

$$\begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{vmatrix} = x_1 \begin{vmatrix} y_2 & y_3 \\ z_2 & z_3 \end{vmatrix} - x_2 \begin{vmatrix} y_1 & y_3 \\ z_1 & z_3 \end{vmatrix} + x_3 \begin{vmatrix} y_1 & y_2 \\ z_1 & z_2 \end{vmatrix} \quad (117)$$

If the three points \mathbf{x} , \mathbf{y} and \mathbf{z} are on the same plane, then the volume is obviously $V = 0$. The determinant is then zero, and the interpretation of this is that the three points are linearly dependent, that is, one of the points, say \mathbf{z} can be expressed as the linear combination $\mathbf{z} = \lambda \mathbf{x} + \mu \mathbf{y}$ of the other two.

In the homogeneous space this gives the following geometric insight. Consider two homogeneous vectors \mathbf{x}_1 and \mathbf{x}_2 . Then a plane with normal vector ℓ is defined by $\ell = \mathbf{x}_1 \times \mathbf{x}_2$, where \mathbf{x}_1 and \mathbf{x}_2 are in the plane. Then, if the vector \mathbf{x} is in the plane defined by \mathbf{x}_1 and \mathbf{x}_2 , the volume is $V = \mathbf{x} \cdot (\mathbf{x}_1 \times \mathbf{x}_2) = \mathbf{x} \cdot \ell = 0$. The intersection between the plane with normal vector ℓ and the affine plane is the line described by ℓ . The intersection between the vector \mathbf{x}_1 and the affine plane is a point on the line, and in the same way the intersection of the vector \mathbf{x}_2 and the affine plane is a point on the line. Since \mathbf{x} is in the plane defined by ℓ , the intersection between \mathbf{x} and the affine plane must be on the line. Therefore, $V = 0$ is exactly the condition for the point defined by \mathbf{x} to be on the line defined by ℓ . Note that this condition can also be written

$$V = \begin{vmatrix} x_1 & x_2 & x_3 \\ x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{vmatrix} = 0 \quad (118)$$

where the geometric interpretation is the the points are linearly dependent.

Example

The line ℓ is defined by the two points $\mathbf{x}_1 = [1, 0, 1]^T$ and $\mathbf{x}_2 = [2, 1, 1]^T$ on the line. The line is then $\ell = (\mathbf{x}_1)^\times \mathbf{x}_2 = [-1, 1, 1]^T$, which can also be written in the familiar form $y = x - 1$. The point $\mathbf{x} = [0, -1, 1]^T$ is on the same line, which is verified by calculation the determinant

$$\begin{vmatrix} \mathbf{x}^T \\ \mathbf{x}_1^T \\ \mathbf{x}_2^T \end{vmatrix} = \begin{vmatrix} 0 & -1 & 1 \\ 1 & 0 & 1 \\ 2 & 1 & 1 \end{vmatrix} = 0 \quad (119)$$

□

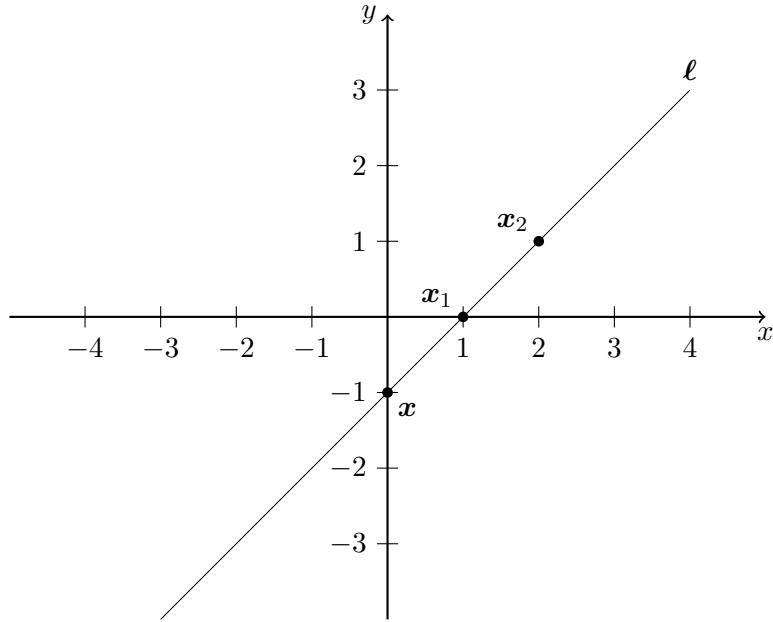


Figure 18: Three points on the same line

4.12 Point at infinity

A point $\mathbf{x} = [x_1, x_2, x_3]^T$ in homogeneous coordinates represents a point $\mathbf{p} = [x_1/x_3, x_2/x_3]^T$ when x_3 is nonzero. Suppose that x_1 and x_2 are constants, while x_3 tends to zero. Then

$$\lim_{x_3 \rightarrow 0} \mathbf{x} = \lim_{x_3 \rightarrow 0} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix} \quad (120)$$

A useful interpretation of this limiting action is obtained by writing it in the form

$$\lim_{x_3 \rightarrow 0} \mathbf{x} = \lim_{x_3 \rightarrow 0} x_3 \begin{bmatrix} x_1/x_3 \\ x_2/x_3 \\ 1 \end{bmatrix} \quad (121)$$

It is seen that this is be the homogeneous representation of the point

$$\lim_{x_3 \rightarrow 0} \mathbf{p} = \lim_{x_3 \rightarrow 0} \begin{bmatrix} x_1/x_3 \\ x_2/x_3 \end{bmatrix} \quad (122)$$

which will tend to infinity in the direction of $[x_1, x_2]^T$ when x_3 tends to zero. Therefore, a homogeneous point

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix}$$

is called a point at infinity in the direction of $[x_1, x_2]^T$. The point $[0, 0, 0]^T$ is undefined and does not represent any point.

Example

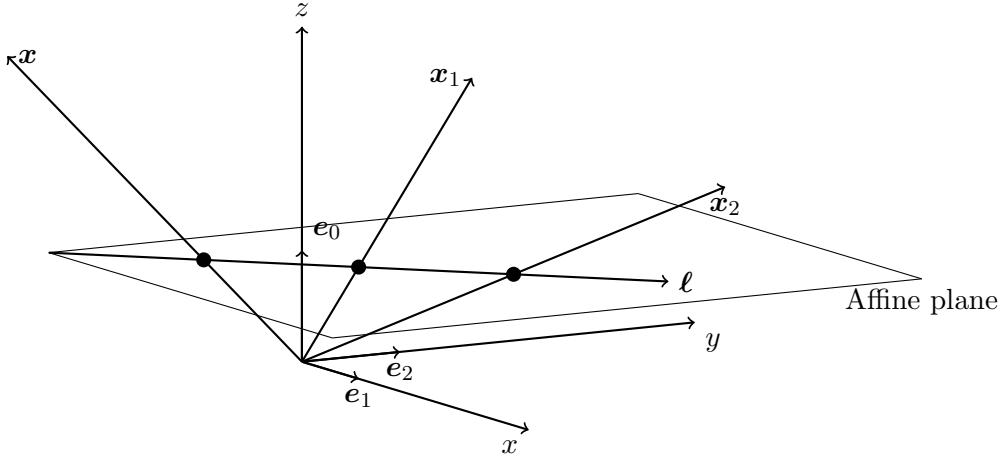


Figure 19: Two points are defined by the intersection of the vectors \mathbf{x}_1 and \mathbf{x}_2 and the affine plane defined by $z = 1$. The line ℓ is the intersection between the plane spanned by \mathbf{x}_1 and \mathbf{x}_2 and the affine plane. The vector \mathbf{x} , which is a linear combination of \mathbf{x}_1 and \mathbf{x}_2 , intersects the affine plane on the line ℓ .

Consider the homogeneous point $\mathbf{x} = [0, 1, \lambda]$, where the point will tend to a point $\mathbf{z} = [0, 1, 0]$ at infinity when $\lambda \rightarrow 0$. A sequence of vectors $\mathbf{x}_i = [0, 1, \lambda_i]$ is plotted in Figure 20 where $\lambda_1 = 1$, $\lambda_2 = 0.5$, $\lambda_3 = 0.25$ and $\lambda_4 = 0$. It is seen that the resulting point in 2D, which is the intersection between \mathbf{x}_i and the affine plane, tends to infinity in the direction of the y axis, and that the vector \mathbf{x}_4 , which is parallel to the affine plane, can be regarded to intersect the affine plane at infinity in the direction of the y axis.

□

4.13 Intersection of two parallel lines

Consider two parallel lines $\ell_1 = [a, b, c_1]^T$ and $\ell_2 = [a, b, c_2]^T$. The normal vector of the lines is then $\mathbf{n} = [a, b]^T$, and the common direction vector is $\mathbf{a} = [b, -a]^T$. The point of intersection is found to be

$$\mathbf{x} = \ell_1 \times \ell_2 = \begin{bmatrix} 0 & -c_1 & b \\ c_1 & 0 & -a \\ -b & a & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c_2 \end{bmatrix} = \begin{bmatrix} b(c_2 - c_1) \\ -a(c_2 - c_1) \\ 0 \end{bmatrix} = (c_2 - c_1) \begin{bmatrix} b \\ -a \\ 0 \end{bmatrix} \quad (123)$$

It is seen that the two points intersect at the point $\mathbf{x} = [b, -a, 0]^T$, which is a point at infinity in the direction of $\mathbf{a} = [b, -a]^T$, which is the direction vector of the lines.

4.14 Line at infinity

Consider the line

$$\ell_\infty = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

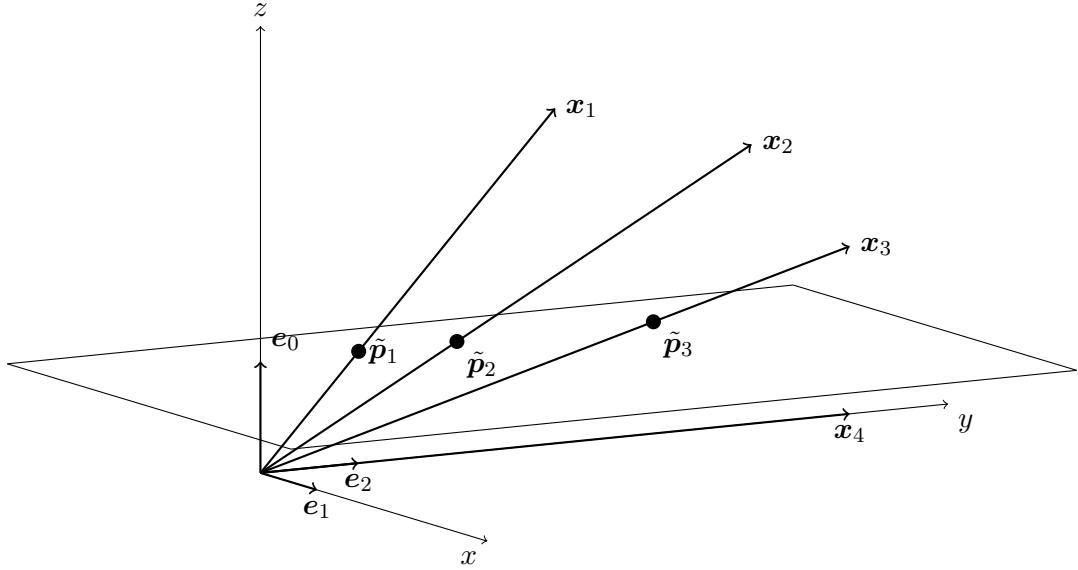


Figure 20: A sequence of vectors $\mathbf{x}_i = [0, 1, \lambda_i]^T$ where λ_i tends to zero. The resulting 2D points $\mathbf{p}_i = [0, 1/\lambda_i]^T$ are found from the corresponding homogeneous points $\tilde{\mathbf{p}}_i = [0, 1/\lambda_i, 1]^T$ at the intersection between the vectors and the affine plane. It is seen that the 2D points will move to infinity in the direction of the y axis.

Then any point $\mathbf{x}_\infty = [x, y, 0]^T$ at infinity will be on this line, which follows from $\mathbf{x}_\infty^T \ell_\infty = 0$. The line ℓ_∞ is therefore called a line at infinity.

A line $\ell = [a, b, c]^T$ has the normal vector $\mathbf{n} = [a, b]^T$, which means that the direction vector along the line is $\mathbf{a} = [b, -a]^T$. The line will intersect ℓ_∞ at the point

$$\mathbf{x} = \ell \times \ell_\infty = \begin{bmatrix} b \\ -a \\ 0 \end{bmatrix}$$

which is a point at infinity in the direction of the line. Any parallel line $\ell' = [a, b, c']^T$ will intersect ℓ_∞ at the same point $\mathbf{x} = [b, -a, 0]^T$. This means that two parallel lines will intersect the line at infinity at the same point, where the point of intersection is a point in infinity in the direction of the lines.

If two points at infinity are given as $\mathbf{x}_1 = [x_1, y_1, 0]^T$ and $\mathbf{x}_2 = [x_2, y_2, 0]^T$, then

$$\mathbf{x}_1 \times \mathbf{x}_2 = \begin{bmatrix} 0 & 0 & y_1 \\ 0 & 0 & -x_1 \\ -y_1 & x_1 & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ x_1 y_2 - y_1 x_2 \end{bmatrix} = (x_1 y_2 - y_1 x_2) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (124)$$

Since this is expressed in homogeneous coordinates, the scaling can be ignored, and it follows that $\mathbf{x}_1 \times \mathbf{x}_2 = \ell_\infty$. This means that the line passing through the two points \mathbf{x}_1 and \mathbf{x}_2 at infinity is the line at infinity ℓ_∞ .

4.15 Intersection of a line and a line at infinity

In this section the intersection of a line $\ell = [a, b, c]^T$ and the line at infinity $\ell_\infty = [0, 0, 1]^T$ will be studied. It is noted that the line ℓ has normal vector $\mathbf{n} = [a, b]^T$ and direction vector $\mathbf{a} = [b, -a]^T$.

The point of intersection is

$$\mathbf{z} = \ell \times \ell_\infty = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} b \\ -a \\ 0 \end{bmatrix} \quad (125)$$

It is seen that the point of intersection is a point at infinity in the direction of the line.

Example

Let a line be given by $\ell = [0, 1, -2]^T$, as shown in Figure 21. The points on the line satisfies $y = 2$. This is a line that is parallel with the x axis, and the direction vector is $\mathbf{a} = [1, 0]^T$. Then the intersection between this line and the line at infinity is at

$$\mathbf{z} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (126)$$

which is a point at infinity in the direction of the x axis. Note that the position of the intersection at infinity does not depend on the value of the offset coefficient c , which characterizes the distance from the origin to the line.

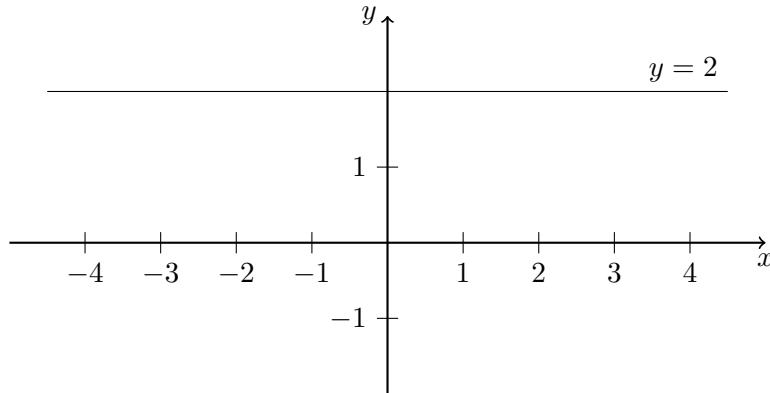


Figure 21: The line $y = 2$ will intersect the line at infinity at $\mathbf{z} = [1, 0, 0]^T$, which is the point at infinity in the direction of the x axis.

4.16 Fitting a line to a set of two points

Suppose that two separate points $\mathbf{x}_1 = [x_1, y_1, 1]^T$ and $\mathbf{x}_2 = [x_2, y_2, 1]^T$ are given, and the task is to find the line ℓ given in homogeneous coordinates. The points \mathbf{x}_i are on the line ℓ if $\mathbf{x}_i^T \ell = 0$. It has already been shown that the solution for the line is $\ell = \mathbf{x}_1 \times \mathbf{x}_2$. In this section we will present another solution based on the singular value decomposition. This is obviously a more complicated method for finding the line, and it is presented here as a simple

example to explain the use of the singular value decomposition to find nontrivial solutions to a system of equations in the form $\mathbf{A}\mathbf{n} = \mathbf{0}$, which is used to find the solution if there are more than two points to be fitted to the line. In the next section this will be extended to the fitting of a line to n noisy points, where the singular value decomposition gives a least-squares fit of the line to the points.

Define the matrix $\mathbf{A}_h = [\mathbf{x}_1, \mathbf{x}_2]^T \in \mathbb{R}^{2 \times 3}$. Then $\boldsymbol{\ell}$ must satisfy

$$\mathbf{A}_h \boldsymbol{\ell} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \end{bmatrix} \boldsymbol{\ell} = \mathbf{0} \quad (127)$$

The matrix \mathbf{A}_h will be of rank 2. This follows as there are only two rows in the matrix. The singular value decomposition of A is

$$\mathbf{A}_h = \sum_{i=1}^3 \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \quad \mathbf{u}_i \in \mathbb{R}^2, \quad \mathbf{v}_i \in \mathbb{R}^3 \quad (128)$$

where $\sigma_1 \geq \sigma_2 > 0$ and $\sigma_3 = 0$. The output vectors $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ are orthogonal unit vectors, and the input vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are orthogonal unit vectors, which means that

$$\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \quad (129)$$

The equation $\mathbf{A}_h \boldsymbol{\ell} = \mathbf{0}$ can then be written

$$\sigma_1 \mathbf{u}_1 (\mathbf{v}_1^T \boldsymbol{\ell}) + \sigma_2 \mathbf{u}_2 (\mathbf{v}_2^T \boldsymbol{\ell}) = 0 \quad (130)$$

Since σ_1 and σ_2 are nonzero, it follows that $\mathbf{v}_1^T \boldsymbol{\ell} = 0$ and $\mathbf{v}_2^T \boldsymbol{\ell} = 0$. From the orthogonality of the vectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ it follows that $\mathbf{v}_1^T \mathbf{v}_3 = 0$ and $\mathbf{v}_2^T \mathbf{v}_3 = 0$. The only nontrivial solution for $\boldsymbol{\ell}$ is therefore

$$\boldsymbol{\ell} = \lambda \mathbf{v}_3 \quad (131)$$

where $\lambda \neq 0$ is a scaling factor that can be selected to any nonzero value.

4.17 Fitting a line through the origin to two points with SVD

Consider two Euclidean points $\mathbf{p}_1 = [x_1, y_1]^T$ and $\mathbf{p}_2 = [x_2, y_2]^T$ with homogeneous representations $\mathbf{x}_1 = [x_1, y_1, 1]^T$ and $\mathbf{x}_2 = [x_2, y_2, 1]^T$. The line $\boldsymbol{\ell} = [\mathbf{n}^T, c]^T$ through the two points is given by

$$\boldsymbol{\ell} = \mathbf{x}_1^T \mathbf{x}_2 = \begin{bmatrix} 0 & -1 & y_1 \\ 1 & 0 & -x_1 \\ -y_1 & x_1 & 0 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} y_1 - y_2 \\ x_2 - x_1 \\ x_1 y_2 - x_2 y_1 \end{bmatrix} \quad (132)$$

For line fitting it turns out to be useful to describe the points by their distance to the centroid of the two points, which is the point

$$\bar{\mathbf{p}} = \frac{1}{2}(\mathbf{p}_1 + \mathbf{p}_2) = \begin{bmatrix} \frac{x_1 + x_2}{2} \\ \frac{y_1 + y_2}{2} \end{bmatrix} \quad (133)$$

with homogeneous representation $\bar{\mathbf{x}} = [\bar{\mathbf{p}}^T, 1]^T$. First it is noted that the centroid is on the line ℓ . This is verified by the calculation

$$\ell^T \bar{\mathbf{x}} = \mathbf{n}^T \bar{\mathbf{p}} + c = \frac{1}{2} \mathbf{n}^T (\mathbf{p}_1 + \mathbf{p}_2) + c = \frac{1}{2} (\mathbf{n}^T \mathbf{p}_1 + c + \mathbf{n}^T \mathbf{p}_2 + c) = \frac{1}{2} (\ell^T \mathbf{x}_1 + \ell^T \mathbf{x}_2) = 0 \quad (134)$$

Let $\bar{\mathbf{p}} = [\bar{x}, \bar{y}]^T$, and define $\delta \mathbf{p}_i = [\delta x_i, \delta y_i]^T$ by

$$\mathbf{p}_i = \bar{\mathbf{p}}_i + \delta \mathbf{p}_i \quad (135)$$

Then the line λ through the points $\delta \mathbf{p}_1$ and $\delta \mathbf{p}_2$ goes through the origin, which is confirmed by the calculation

$$\lambda = \delta \mathbf{p}_1^T \delta \mathbf{p}_2 = \begin{bmatrix} \delta y_1 - \delta y_2 \\ \delta x_2 - \delta x_1 \\ \delta x_1 \delta y_2 - \delta x_2 \delta y_1 \end{bmatrix} = \begin{bmatrix} y_1 - y_2 \\ x_2 - x_1 \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{n} \\ 0 \end{bmatrix} \quad (136)$$

which is verified by direct computation. This means that $\lambda = [\mathbf{n}^T, 0]^T$ is a line through the origin which has the same normal vector as ℓ .

Line regression by total least squares can be done based on the observation that $\delta \mathbf{p}_i$ is on $\lambda = [\mathbf{n}^T, 0]^T$ if and only if $\delta \mathbf{p}_i^T \mathbf{n} = 0$. It follows that the line can be found from

$$\mathbf{A} \mathbf{n} = \mathbf{0} \quad (137)$$

where

$$\mathbf{A} = \begin{bmatrix} \delta x_1 & \delta y_1 \\ \delta x_2 & \delta y_2 \end{bmatrix} \quad (138)$$

The solution is then

$$\mathbf{n} = \mathbf{v}_2 \quad (139)$$

where \mathbf{v}_2 is the last column of the matrix \mathbf{V} in the SVD $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$.

This result is derived as follows. \mathbf{A} will be of rank 1 since the line passes through the origin, which implies that $\delta \mathbf{p}_1$ and $\delta \mathbf{p}_2 = \gamma \mathbf{p}_1$ are linearly dependent. The singular value decomposition of \mathbf{A} is therefore

$$\mathbf{A} = \sum_{i=1}^2 \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \quad \mathbf{u}_i \in \mathbb{R}^2, \quad \mathbf{v}_i \in \mathbb{R}^2 \quad (140)$$

where $\sigma_1 > 0$ and $\sigma_2 = 0$, which follows since the number of nonzero singular values is equal to the rank of the matrix. The output vectors $\mathbf{u}_1, \mathbf{u}_2$ are orthogonal unit vectors, and the input vectors $\mathbf{v}_1, \mathbf{v}_2$ are orthogonal unit vectors, which means that

$$\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \quad (141)$$

The equation $\mathbf{A} \mathbf{n} = \mathbf{0}$ can then be written

$$\sigma_1 \mathbf{u}_1 (\mathbf{v}_1^T \ell) = 0 \quad (142)$$

Here σ_1 is nonzero, and it follows that $\mathbf{v}_1^T \ell = 0$. From the orthogonality of the vectors $\mathbf{v}_1, \mathbf{v}_2$ it follows that the only nontrivial solution of unit length for \mathbf{n} is

$$\mathbf{n} = \mathbf{v}_2 \quad (143)$$

4.18 Line regression gives a line through the centroid

In this section it is shown that regression of a line from a set of noisy points gives a line through the centroid of the points both with least squares and total least squares. This was shown by Adcock in 1878 [1] for total least squares, and is it shown here that the same applies for least squares.

In the total least squares problem a line $\ell = [a, b, c]^T$ is fitted to a set of n points $\mathbf{x}_i = [x_i, y_i, 1]^T$, $i = 1, \dots, n$ so that the sum of squared distances $d_i = \ell^T \mathbf{x}_i / |\mathbf{n}|$ from the points to the line is minimized. The cost function to be minimized is then

$$C_T = \sum_{i=1}^n \frac{(ax_i + by_i + c)^2}{a^2 + b^2} \quad (144)$$

It is seen that the optimal solution must satisfy

$$\frac{\partial C_T}{\partial c} = n \frac{2a \frac{1}{n} \sum_{i=1}^n x_i + 2b \frac{1}{n} \sum_{i=1}^n y_i + 2c}{a^2 + b^2} = 0 \quad (145)$$

and it follows that

$$a\bar{x} + b\bar{y} + c = 0 \quad (146)$$

where $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ and $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ are the coordinates of the centroid of the points. This shows that the centroid $\bar{\mathbf{x}} = [\bar{x}, \bar{y}, 1]^T$ is on the fitted line. Expressions for a and b can then be derived by first order optimality conditions as in [1]. Here this will instead be done with an SVD solution, which is the topic of the next section.

The least squares problem is to find a line $y = \alpha x + \beta$ so that the cost function

$$C_L = \sum_{i=1}^n (y_i - \alpha x_i - \beta)^2 \quad (147)$$

The optimal solution must satisfy

$$\frac{\partial C_L}{\partial \beta} = 2 \sum_{i=1}^n (y_i - \alpha x_i - \beta) = 2n \left(\frac{1}{n} \sum_{i=1}^n x_i - \alpha \frac{1}{n} \sum_{i=1}^n x_i - \beta \right) = 0 \quad (148)$$

which gives

$$\bar{y} = \alpha \bar{x} + \beta \quad (149)$$

This shows that the centroid is on the line.

4.19 Fitting a line through the origin to a set of noisy points

In this section it is shown how to compute the line $\ell = [\mathbf{n}^T, c]$ which minimizes the sum of squared distances from n homogeneous points $\mathbf{x}_i = [x_i, y_i, 1]^T$, $i = 1, \dots, n$ by using an SVD solution. As shown in the previous section, the line ℓ passes through the centroid $\bar{\mathbf{x}} = [\bar{x}, \bar{y}, 1]^T$ of the points, and it follows that

$$c = -(a\bar{x} + b\bar{y}) = -\mathbf{n}^T \bar{\mathbf{p}} \quad (150)$$

The line is transformed to the line $\boldsymbol{\lambda} = [\mathbf{n}^T, 0]^T$ by moving the centroid to the origin. This means that the points are moved to $\delta\mathbf{x}_i = [\delta\mathbf{p}_i^T, 1]^T = [\delta x_i, \delta y_i, 1]^T$ where

$$x_i = \bar{x} + \delta x_i \quad (151)$$

$$y_i = \bar{y} + \delta y_i \quad (152)$$

The points are on the line if $\mathbf{n}^T \delta\mathbf{p}_i = 0$. The distance from point i to the line $\boldsymbol{\lambda} = [\mathbf{n}^T, 0]^T$ is $d_i = \mathbf{n}^T \delta\mathbf{p}_i$ under the assumption that \mathbf{n} is a unit vector. The cost function to minimized is therefore

$$C = \sum_{i=1}^n d_i^2 = \sum_{i=1}^n (\mathbf{n}^T \delta\mathbf{p}_i)^2 \quad (153)$$

under the constraint $\mathbf{n}^T \mathbf{n} = 1$.

Define the matrix

$$\mathbf{A} = \begin{bmatrix} \delta\mathbf{p}_1^T \\ \delta\mathbf{p}_2^T \\ \vdots \\ \delta\mathbf{p}_n^T \end{bmatrix} \in \mathbb{R}^{n \times 2} \quad (154)$$

and define the vector $\boldsymbol{\delta} \in \mathbb{R}^n$ by

$$\mathbf{A}\mathbf{n} = \boldsymbol{\delta} \quad (155)$$

The minimization problem can then be formulated as the minimization of

$$C = \boldsymbol{\delta}^T \boldsymbol{\delta} \quad (156)$$

when $\mathbf{n}^T \mathbf{n} = 1$.

If all the points $\delta\mathbf{p}_i$ for $i = 1, \dots, n$ had been exactly on the line $\boldsymbol{\lambda}$, then any row i where $i \geq 2$ of the matrix \mathbf{A} would be a linear combination of the first row, and the rank of \mathbf{A} would be 1. In the noisy case there will be no such linear dependence, and the rank of the matrix \mathbf{A} will be 2. The singular value decomposition of \mathbf{A} is then

$$\mathbf{A} = \sum_{i=1}^2 \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \quad \mathbf{u}_i \in \mathbb{R}^n, \quad \mathbf{v}_i \in \mathbb{R}^2 \quad (157)$$

where $\sigma_1 \geq \sigma_2 > 0$. The equation $\mathbf{A}\mathbf{n} = \mathbf{0}$ can then be written

$$\sigma_1 \mathbf{u}_1 (\mathbf{v}_1^T \mathbf{n}) + \sigma_2 \mathbf{u}_2 (\mathbf{v}_2^T \mathbf{n}) = 0 \quad (158)$$

Since all the singular values are nonzero, the left hand side cannot be zero for nontrivial solutions. Therefore a least squares solution must be used. The least squares solution \mathbf{n} of unit length will be $\mathbf{n}_* = \mathbf{v}_2$ since this gives

$$\boldsymbol{\delta} = \mathbf{A}\mathbf{n}_* = \sigma_2 \mathbf{u}_2 \quad (159)$$

The square of the solution will be

$$\boldsymbol{\delta}^T \boldsymbol{\delta} = \|\mathbf{A}\mathbf{n}_*\|^2 = \sigma_2^2 \quad (160)$$

Any solution $\mathbf{n} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2$ of unit length, that is, with $c_1^2 + c_2^2 = 1$, will result in a square of the solution that is

$$\delta^T \delta = \|\mathbf{A}\mathbf{n}\|^2 = \sigma_1^2 c_1^2 + \sigma_2^2 c_2^2 \quad (161)$$

Since $\sigma_1 \geq \sigma_2 > 0$ it follows that

$$\sigma_1^2 c_1^2 + \sigma_2^2 c_2^2 \geq \sigma_2^2 (c_1^2 + c_2^2) = \sigma_2^2 \quad (162)$$

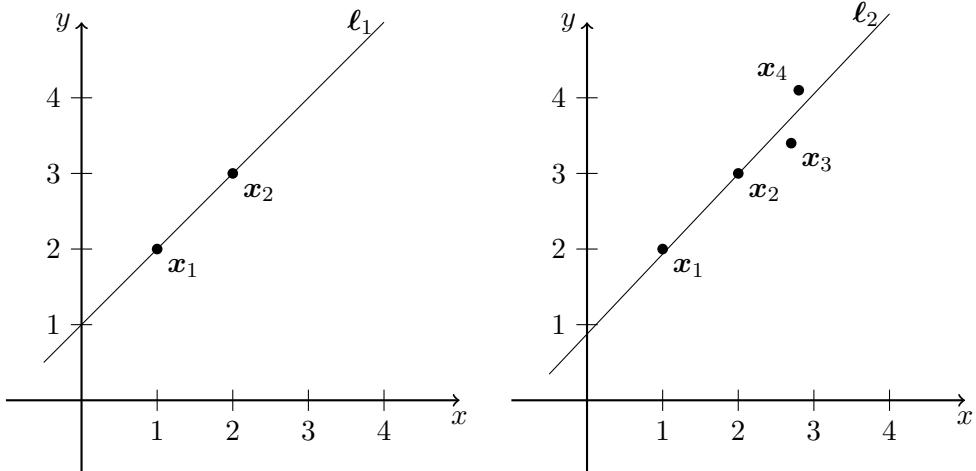
The least squares solution of unit length is therefore given by

$$\mathbf{n}_* = \mathbf{v}_2 \quad (163)$$

This means that the optimal line is $\boldsymbol{\lambda} = [\mathbf{v}_2^T, 0]^T$, which gives

$$\boldsymbol{\ell} = \begin{bmatrix} \mathbf{v}_2 \\ -\mathbf{v}_2^T \bar{\mathbf{p}} \end{bmatrix} \quad (164)$$

Example



(a) The line ℓ_1 calculated from the noise-free points x_1 and x_2 . The line intersects the two points.

(b) The line ℓ_2 calculated from the noise-free points x_1 and x_2 , and the noisy points x_3 and x_4 . The line gives a least-squares fit to the points.

Figure 22: Calculation of a line from 2 and 4 points.

The points $\mathbf{x}_1 = [1, 2, 1]^T$, $\mathbf{x}_2 = [2, 3, 1]^T$, $\mathbf{x}_3 = [2.7, 3.4, 1]^T$ and $\mathbf{x}_4 = [2.8, 4.1, 1]^T$ are given. The points \mathbf{x}_1 and \mathbf{x}_2 are on the line $y = x + 1$, which has the homogeneous representation $\boldsymbol{\ell} = [1, -1, 1]^T$. It is assumed that this is the true line, and that the points \mathbf{x}_1 and \mathbf{x}_2 are close to this line, but there is some deviation due to noise on these two points.

The solution from the two points \mathbf{x}_1 and \mathbf{x}_2 based on the singular value decomposition gives the exact solution $\boldsymbol{\ell}_1 = [1, -1, 1]^T$. The distance d_i from point \mathbf{x}_i to the line is then $d_1 = d_2 = 0$ up to machine precision. The result is shown in Figure 22a.

The solution from all four points including the noisy points \mathbf{x}_3 and \mathbf{x}_4 gives the solution $\mathbf{\ell}_2 = [-1.0588, 1.0, -0.875]^T$. The distance d_i from point \mathbf{x}_i to the line is then

$$d_1 = -0.0455 \quad (165)$$

$$d_2 = -0.00505 \quad (166)$$

$$d_3 = 0.229 \quad (167)$$

$$d_4 = -0.179 \quad (168)$$

The result is shown in Figure 22b.

```
# Script for finding a line with total least-squares
import numpy as np
import matplotlib.pyplot as plt
def distance_point_2_line(L,x):
    return np.dot(L,x)/np.linalg.norm(L[0:2])

# Input: Homogeneous points on the line y = x + 1 or L = (1,-1,1) with noise.
x1 = np.array([1,2,1]); x2 = np.array([2,3,1]) # Noise free
x3 = np.array([2.7,3.4,1]); x4 = np.array([2.8,4.1,1]) # With noise
X = np.block([[x1[0:2]], [x2[0:2]], [x3[0:2]], [x4[0:2]]]).T

# Line from 2 points with no noise
L2 = np.cross(x1, x2)
if L2[1] != 0:
    L2= L2/L2[1]
np.set_printoptions(precision=4)
#np.set_printoptions(formatter={'float': '{: 0.4f}'.format})
print('\n')
print('L2 = {}'.format(L2))
print('d1 = {:.4f}, d2 = {:.4f},'
      .format(distance_point_2_line(L2,x1), distance_point_2_line(L2,x2)))

# Line from 4 points with noise in points 3 and 4
x = X[0,:]; y = X[1,:]
xb = np.mean(x); yb = np.mean(y); pb = np.array([xb, yb])
delta_x = x - xb; delta_y = y - yb
A4 = np.vstack([delta_x, delta_y]).T
u4, s4, vt4 = np.linalg.svd(A4)
normal = vt4[1,:]
L4 = np.hstack((normal, - np.dot(pb,normal) ))
if L4[1] != 0:
    L4= L4/L4[1]
print('L4 = ', L4.T/L4[1])
d1 = distance_point_2_line(L4,x1)
d2 = distance_point_2_line(L4,x2)
d3 = distance_point_2_line(L4,x3)
d4 = distance_point_2_line(L4,x4)
```

```

print('d1 = {:.4f}, d2 = {:.4f}, d3 = {:.4f}, d4 = {:.4f}',
      .format(d1, d2, d3, d4))

# Plotting
x = np.arange(-1,7)
x_axis = np.block([[[-1,5], [0,0]]])
y_axis = np.block([[0,0], [-1,5]]);
plt.figure(1)
plt.figure(1).clear()
plt.plot(X[0,:], X[1,:], 'rd')
plt.plot(x, - (x*L2[0] + L2[2])/L2[1], '-g')
plt.plot(x, - (x*L4[0] + L4[2])/L4[1], '-b')
plt.plot(x_axis[0,:], x_axis[1,:], 'k', y_axis[0,:], y_axis[1,:], 'k')
plt.axis([-1, 5, -1, 5])
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.show()

```

□

4.20 Line regression with least squares versus total least squares

Line regression with least squares is a widely used method [45] where the task is to determine a line $y = ax + b$ from a set $\{(x_1, y_1), \dots, (x_n, y_n)\}$ of n samples. The underlying idea is that the values for x_i are accurate, and that there is uncertainty in the values of y_i . This will be the case if x_i is time and y_i is measured with a sensor that has considerable noise. The regression will then minimize the sum of the squared distances from the samples (x_i, y_i) in the y direction. The cost function to be minimized is then

$$C_{\text{LS}} = \sum_{i=1}^n (y_i - ax_i - b)^2 = (\mathbf{X}\boldsymbol{\alpha} - \mathbf{y})^T(\mathbf{X}\boldsymbol{\alpha} - \mathbf{y}) \quad (169)$$

where $\boldsymbol{\alpha} = [a, b]^T$, $\mathbf{y} = [y_1, \dots, y_n]^T$ and

$$\mathbf{X} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \quad (170)$$

and the solution is found from the normal equation to be

$$\boldsymbol{\alpha} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (171)$$

In contrast to this, total least squares [27] is used when x_i and y_i is measured with the same uncertainty. In this case the regression will minimize the sum of the squared distances from the samples (x_i, y_i) to the line along the normal to the line under the side constraint that

$\|\ell\| = 1$. This problem is solved by using a homogeneous representation $\ell = [a, b, c]^T$ of the line where the line be normalized so that the normal vector $\mathbf{n} = [a, b]^T$ is a unit vector. Then the distance from a sample (x_i, y_i) to the line ℓ along the normal to the line is $d_i = \mathbf{x}_i^T \ell$ where $\mathbf{x}_i = [x_i, y_i, 1]^T$ is the homogeneous representation of $\delta \mathbf{p} = [\delta x_i, \delta y_i]^T$. The cost function to be minimized is then

$$C_{\text{TLS}} = \sum_{i=1}^n (\delta \mathbf{p}_i^T \mathbf{n})^T (\delta \mathbf{p}_i^T \mathbf{n}) \quad (172)$$

with side constraint $\mathbf{n}^T \mathbf{n} = 1$. The solution is then found from the singular value decomposition $\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$ of

$$\mathbf{A} = \begin{bmatrix} \delta x_1 & \delta y_1 \\ \delta x_2 & \delta y_2 \\ \vdots & \vdots \\ \delta x_n & \delta y_n \end{bmatrix} \quad (173)$$

as

$$\ell = \begin{bmatrix} \mathbf{v}_2 \\ -\mathbf{v}_2^T \bar{\mathbf{p}} \end{bmatrix} \quad (174)$$

where \mathbf{v}_2 is the last column of \mathbf{V} .

Example

Consider the samples given by $\mathbf{x} = [1, 2, 3, 4]^T$ and $\mathbf{y} = [1, 1, 4, 4]^T$. Then least-squares regression gives

$$y = 1.2x - 0.5 \quad (175)$$

while total least squares regression gives the line

$$y = 1.39x - 0.97 \quad (176)$$

The results are shown in Figure 23.

□

```
# %%
# Script for regression by least squares and total least squares
import numpy as np
import matplotlib.pyplot as plt

# %%
x = np.array([1., 2., 3., 4.])
y = np.array([1., 1., 4., 4.])

# %%
# Least-squares regression
Xr = np.vstack([x, np.ones(x.shape)]).T
alpha = np.linalg.inv(Xr.T @ Xr) @ Xr.T @ y
print('alpha = {:.4f}, beta = {:.4f}'.format(alpha[0], alpha[1]))
```

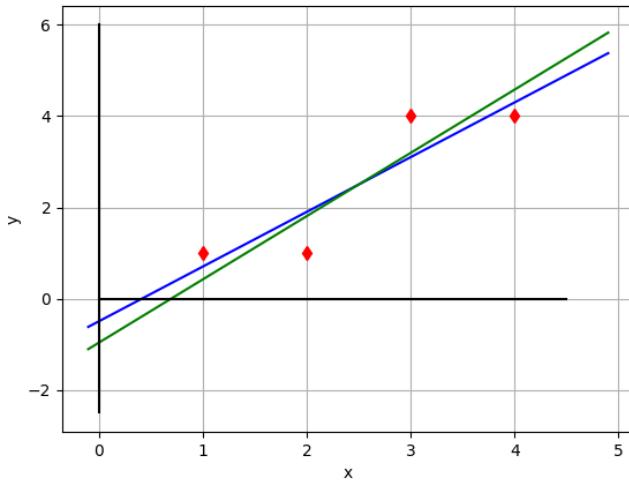


Figure 23: Regression using least squares (blue line) and total least squares (green line). The samples are shown as red diamonds.

```

# Total least squares regression
xb = np.mean(x); yb = np.mean(y); pb = np.array([xb, yb])
delta_x = x - xb; delta_y = y - yb
A = np.vstack([delta_x, delta_y]).T
u, s, vt = np.linalg.svd(A)
normal = vt[1,:]
L = np.hstack( (normal, - np.dot(pb,normal)) )
print( 'a = {:.4f}, b = {:.4f}, c = {:.4f}'.format(-L[0]/L[1], 1.0, -L[2]/L[1]) )

# %%
plt.figure(1)
plt.figure(1).clear()
plt.plot(x, y, 'rd')
x = np.arange(-0.1,5)
plt.plot(x, alpha[0]*x + alpha[1], '-b')
plt.plot(x, (-L[0]*x - L[2])/L[1], '-g')
x_axis = np.block([[0,4.5], [0,0]])
y_axis = np.block([[0,0], [-2.5,6]]);
plt.plot(x_axis[0,:], x_axis[1,:], 'k', y_axis[0,:], y_axis[1,:], 'k')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.show()

```

5 Homographies in two dimensional space

5.1 Introduction

Homographies play an important role in computer vision. Homographies are used to describe the transformation between different images. Such transformations may describe rigid body motion, scaling of size, stretching of objects and projective effect.

5.2 Homography

Consider a homogeneous description of a space of two dimensions. A homography is a linear and invertible transformation from a homogeneous vector \mathbf{x} to a homogeneous vector \mathbf{x}' given by

$$\mathbf{x}' = \mathbf{H}\mathbf{x} \quad (177)$$

The homography is described by its matrix \mathbf{H} , therefore this matrix is usually referred to as the homography. The matrix \mathbf{H} must be nonsingular for the transformation to be invertible. The homography can therefore be inverted as

$$\mathbf{x} = \mathbf{H}^{-1}\mathbf{x}' \quad (178)$$

where \mathbf{H}^{-1} is said to be the inverse homography.

Homographies offer an extension to the usual homogeneous transformation matrices

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(2) \quad (179)$$

that are used to describe rigid motion in the plane in terms of rotation and translation. In homographies the class of transformations are extended from $\mathbf{T} \in SE(2)$ to a general nonsingular homogeneous transformation matrix \mathbf{H} of dimension 3×3 , where the matrix can be written

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v_3 \end{bmatrix} \quad (180)$$

Here \mathbf{A} is a 2×2 submatrix related to rotation, scaling and stretching of objects, \mathbf{t} is related to translations, \mathbf{v} is related to perspective effects, and v_3 is a scaling factor.

Major references on homographies are [41] and [30].

5.3 Independent parameters of a homography in 2D

A homography $\mathbf{x}' = \mathbf{H}\mathbf{x}$ is a mapping from one homogeneous vector \mathbf{x} to another homogeneous vector \mathbf{x}' . A homogeneous vector \mathbf{x} is considered to be equivalent under scaling, that is, \mathbf{x} is considered to be the same vector as $\lambda\mathbf{x}$ for all scalars $\lambda \neq 0$, and this means that also a homography will be equivalent under scaling. This means that \mathbf{H} is considered to be the same homography as $\mu\mathbf{H}$ for all scalars $\mu \neq 0$. A general homography in will have 9 elements. Because of the scaling only 8 elements are independent. This means that 8 independent equations is needed to determine the elements of a general homography.

5.4 Homographies in two dimensions

Consider a planar homography \mathbf{H} from a point \mathbf{x} to a point $\mathbf{x}' = \mathbf{H}\mathbf{x}$ where the points are given by homogeneous coordinates in the plane. In terms of coordinates this mapping is written

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (181)$$

The homogeneous vectors \mathbf{x} and \mathbf{x}' can be scaled and still represent the same points in \mathbb{R}^2 , and the homogeneous matrix \mathbf{H} will then be scaled by a scalar. Therefore, the same mapping can be written

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mu \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (182)$$

for the nonzero constant $\mu = x_3/x'_3$. The two homographies \mathbf{H} and $\mu\mathbf{H}$ are considered to be equivalent transformations. This means that although \mathbf{H} has 9 entries, it has only 8 free parameters, as the matrix can be scaled.

5.5 Transformation of lines

A homography $\mathbf{x}' = \mathbf{H}\mathbf{x}$ with nonsingular \mathbf{H} maps lines to lines in the projective space \mathbb{P}^2 . This means that if three points \mathbf{x}_1 , \mathbf{x}_2 and \mathbf{x}_3 are on the same line ℓ , then the transformed points \mathbf{x}'_1 , \mathbf{x}'_2 and \mathbf{x}'_3 will be on the same transformed line ℓ' . The points \mathbf{x}_i are on the line ℓ if and only if $\ell^T \mathbf{x}_i = 0$. This gives $\ell^T \mathbf{H}^{-1} \mathbf{H} \mathbf{x}_i = 0$ so that $(\mathbf{H}^{-T} \ell)^T \mathbf{x}'_i = 0$. This means that the points \mathbf{x}'_i will be on the line $\ell' = \mathbf{H}^{-T} \ell$, which means that points are transformed according to

$$\mathbf{x}' = \mathbf{H}\mathbf{x} \quad (183)$$

while lines are transformed according to

$$\ell' = \mathbf{H}^{-T} \ell \quad (184)$$

A homography can be calculated from 4 line mappings in the same way as it can be calculated from 4 point mappings. The equations in the solution need some minor changes as the line mapping has the transformation \mathbf{H}^{-T} in place of \mathbf{H} .

5.6 Homographic mapping of pixel values

In computer vision homographies are used in 2D for the mapping from one image to another. Consider the homography $\mathbf{x}' = \mathbf{H}\mathbf{x}$. Let the image coordinates in the original image be given in homogeneous form by $\mathbf{x} = (u, v, 1)^T$ and let $\mathbf{x}' = (u', v', 1)^T$ be the corresponding image coordinates in the transformed image. Let the image intensity at \mathbf{x} be $f(\mathbf{x})$. Then the homography maps the intensity $f(\mathbf{x})$ at \mathbf{x} to the same intensity $g(\mathbf{x}')$ at \mathbf{x}' in the transformed image. This means that the two images are related by $g(\mathbf{x}') = f(\mathbf{x})$.

There is a certain complication here as the pixel coordinates $\mathbf{x} = (u, v, 1)^T$ are integers, while the transformed coordinates $\mathbf{x}' = (u', v', 1)^T$ will in general not be integers. A first attempt to solve this could be to round off (u', v') to the nearest pixel. This is not a good solution,

because two pixels can be mapped to the same pixel in the transformed image, and there may be undefined pixels in the transformed image if all the closest (u', v') are mapped to neighboring pixels.

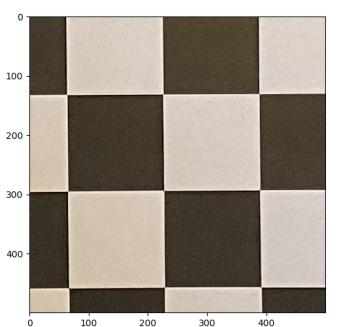
The recommended solution is to use the inverse mapping $\mathbf{x} = \mathbf{H}^{-1}\mathbf{x}'$ so that coordinates (u, v) are found for each pixel (u', v') in the transformed image. The resulting coordinates (u, v) will in general not be integers, so the original image must be interpolated to generate the function values $f(\mathbf{x})$.

Example

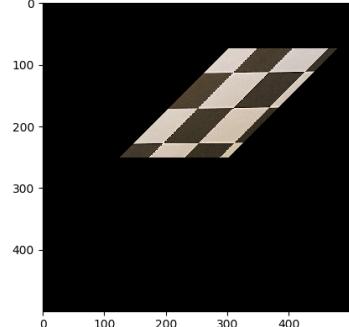
A checkerboard image is transformed with an affine homography

$$\mathbf{H} = \begin{bmatrix} s\mathbf{R}\mathbf{P} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (185)$$

The original image and the transformed image are shown in Figure 24.



(a) Checkerboard image.



(b) Transformed checkerboard image.

Figure 24: Transformation of image with homography

```

import numpy as np
import matplotlib.pyplot as plt
from timeit import default_timer as timer

def exp2(theta):
    return np.array([[np.cos(theta), -np.sin(theta)],
                   [np.sin(theta), np.cos(theta)]])

# Input image from file as a matrix. img.shape = (ny, nx, 3)
img0 = plt.imread('checkerboard.JPG')
img = img0[1500:2000, 2000:2500, :]    # Crop image array
ny = img.shape[0]; nx = img.shape[1]

plt.figure(1)
plt.figure(1).clear()
plt.imshow(img)

```

```

# Homography and its inverse
t = np.array([125,250]).reshape(2,1)
R = 0.25*exp2(-np.pi/4)
P = np.array([[2, 1],[0, 1]])
A = R @ P
H = np.block([[A, t], [0,0,1]])
Hinv = np.linalg.inv(H)

# Transformed image: Initialization
imgp = np.zeros((ny,nx,3), dtype=np.uint8)
# Index array for img
x = np.ones((3,nx,ny),dtype=int)
# Index array for imgp
xp = np.ones((3,nx,ny),dtype=int)
xp[0:2,:,:] = np.mgrid[0:nx,0:ny]

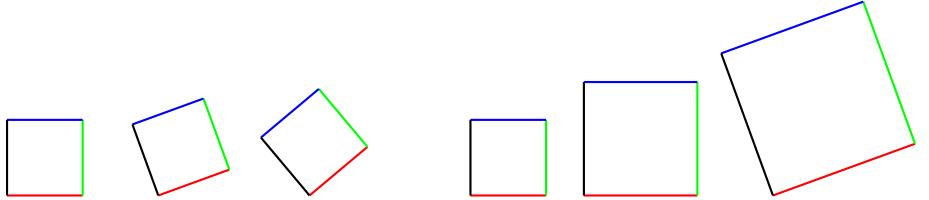
start = timer()
#Calculate x = Hinv @ xp
x = np.tensordot(Hinv, xp, [1,0])
x[0:3,:,:] /= x[2,:,:] # Scale homogeneous vector
x = np.floor(x).astype(int) # Ensure integer indices
print("Array operations", timer()-start)
start = timer()
for i in range(0,nx):
    for j in range(0,ny):
        # ix and iy are the indices in img corresponding to i and j in imgp
        ix = x[0,i,j]; iy = x[1,i,j]
        if 0 <= ix < nx and 0 <= iy < ny: # Index inside image domain
            imgp[j,i] = img[iy,ix]
print("Loop operations", timer()-start)

plt.figure(2)
plt.figure(2).clear()
plt.imshow(imgp)
plt.show()

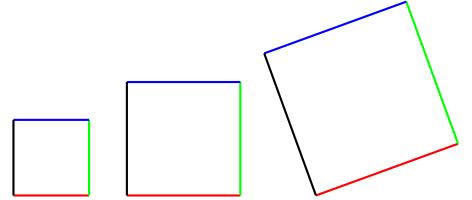
```

5.7 Types of homographies

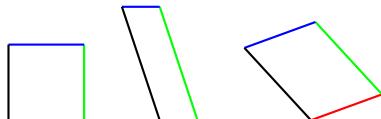
Homographies can be classified as Euclidean transformations, similarity transformations, affine transformations and projective transformations. A Euclidean transformation is a special case of a similarity transformation, which is a special case of an affine transformation, which again is a special case of a projective transformation.



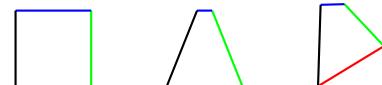
(a) Euclidean transformations: Rotation and translation of a rigid body



(b) Similarity transformations: Rotation, translation and scaling of a rigid body



(c) Affine transformations: Translation, rotation, stretching. Parallel lines remain parallel



(d) Projective transformations.

Figure 25: Examples of Euclidean, similarity, affine and projective transformations

5.8 Euclidean transformation

The first class of transformations to be presented is the Euclidean transformation

$$\mathbf{x}' = \mathbf{H}_e \mathbf{x} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x} \quad (186)$$

where \mathbf{t} is 2-dimensional translation vector and $\mathbf{R} \in O(2)$, where

$$O(2) = \{ \mathbf{R} \in \mathbb{R}^{2 \times 2} \mid \mathbf{R}\mathbf{R}^T = \mathbf{I} \text{ and } \det \mathbf{R} = \pm 1 \} \quad (187)$$

is the orthogonal group of order 2. The Euclidean transformation is a rigid motion when

$$\mathbf{R} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \in SO(2) \quad (188)$$

is a 2×2 rotation matrix where $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and $\det \mathbf{R} = 1$. Then $\mathbf{H}_e \in SE(2)$ is a 3×3 homogeneous transformation matrix.

The transformation is a rigid reflection when

$$\mathbf{R} = \begin{bmatrix} \cos \theta & \sin \theta \\ \sin \theta & -\cos \theta \end{bmatrix} \quad (189)$$

which is a 2×2 reflection matrix where $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and $\det \mathbf{R} = -1$.

A Euclidean transformation will have length and area as invariants, and in addition all the invariants of a similarity transformation.

□

5.9 Similarity transformation

The second class of transformations is the similarity transformation

$$\mathbf{x}' = \mathbf{H}_s \mathbf{x} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x} \quad (190)$$

where s is the scaling factor and $\mathbf{R} \in O(2)$ is a rotation matrix or a reflection matrix. A similarity transformation reduces to a Euclidean transformation when $s = 1$. The inverse of the similarity transformation is

$$\mathbf{H}_s^{-1} = \begin{bmatrix} (1/s)\mathbf{R}^T & -(1/s)\mathbf{R}^T \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (191)$$

which is also a similarity transformation. Similarity transforms will have ratio of lengths and angles as invariants, and in addition, all the invariants of an affine transformation.

Example

Consider a set of points $\mathbf{p}_i = [x_i, y_i]^T$, $i = 1, \dots, N$ in the plane, and let $\mathbf{x}_i = [\mathbf{p}_i^T, 1]^T$ be the corresponding homogeneous vector. The centroid of the points will be at

$$\mathbf{x}_c = \begin{bmatrix} \mathbf{p}_c \\ 1 \end{bmatrix}, \quad \mathbf{p}_c = \frac{1}{N} \sum_{i=1}^N \mathbf{p}_i \quad (192)$$

The average of the squared deviations of the points is

$$\bar{S} = \frac{1}{N} \sum_{i=1}^N (\mathbf{p}_i - \mathbf{p}_c)^2 \quad (193)$$

Then the transformation $\mathbf{x}'_i = \mathbf{N} \mathbf{x}_i$, $i = 1, \dots, N$ where

$$\mathbf{N} = \begin{bmatrix} s\mathbf{I} & -s\mathbf{p}_c \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (194)$$

will transform the set of points $\mathbf{x}_i = [\mathbf{p}_i^T, 1]^T$ to set of points $\mathbf{x}'_i = [\mathbf{p}'_i^T, 1]^T$ where

$$\mathbf{p}'_i = s(\mathbf{p}_i - \mathbf{p}_c)$$

It is seen that the Euclidean points are first centered about the origin by subtracting the mean \mathbf{p}_c , and then that result is scaled by s .

If the scaling factor is selected as

$$s = \sqrt{\frac{2}{\bar{S}}} \quad (195)$$

then the Euclidean points \mathbf{p}'_i , which are centered about the origin, will have an average squared distance of 2.

It is noted that the inverse transformation

$$\mathbf{N}^{-1} = \begin{bmatrix} s^{-1}\mathbf{I} & \mathbf{p}_c \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (196)$$

is a similarity transformation in the form of (190), while \mathbf{N} is a similarity transformation in the form of (191). In numerical computation a set of pixel points \mathbf{x}_i are often transformed to $\mathbf{x}'_i = \mathbf{N}\mathbf{x}_i$, and then computations are done with the normalized vectors \mathbf{x}'_i , which gives better numerical conditioning [30]. The results \mathbf{y}'_i are then transformed back to pixel coordinates with $\mathbf{y}_i = \mathbf{N}^{-1}\mathbf{y}'_i$.

```
# %%
import numpy as np

# %%
def normalize(X):
    n = len(X[0])
    Xc = np.mean(X[0:2, :], axis=1)
    dis = 0
    for i in range(n):
        dis = dis + (X[0:2, i] - Xc) @ (X[0:2, i] - Xc)
    sigma = np.sqrt(dis / n)
    s = np.sqrt(2) / sigma
    N = np.block([[s*np.identity(2), -s*Xc.reshape(2,1)], [0,0,1]])
    return N, Xc

# %%
XP = np.array([[20, 40, 1], [1320, 540, 1], [100, 400, 1],
               [300, 1400, 1]]).T
N, Xc = normalize(XP)
XN = N @ XP
print('XN = \n', XN)

# %%
# Check mean squared distance from origin
d = np.linalg.norm(XN[0:2, :], axis = 0)
dm = d@d/len(XN[0])
print('Average squared distance = ', dm)
```

□

5.10 Affine transformations

The third class of transformations is the affine transformation

$$\mathbf{x}' = \mathbf{H}_a \mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x} \quad (197)$$

where \mathbf{A} is any nonsingular 2×2 matrix. A special case is $\mathbf{A} = s\mathbf{R}$ where $\mathbf{R} \in O(2)$, which makes the affine transformation a similarity transformation.

The inverse transpose of the affine transformation, which is used in the transformation of lines,

is

$$\mathbf{H}_a^{-T} = \begin{bmatrix} \mathbf{A}^{-T} & \mathbf{0} \\ -\mathbf{t}^T \mathbf{A}^{-T} & 1 \end{bmatrix} \quad (198)$$

Affine transformations has the following invariants:

1. Collinear points, which are three or more points on the same line, are transformed to collinear points.
2. Parallel lines will be transformed two parallel lines.
3. The ratio of lengths for parallel lines is invariant
4. Convex sets are transformed to convex sets.
5. Centroids of vectors are invariant.

5.11 The effect of an affine transformation

The properties of the matrix \mathbf{A} in the general case can be investigated with the singular value description of the matrix \mathbf{A} . The singular value decomposition is

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T \quad (199)$$

where $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2)$ and $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2)$ are orthogonal 2×2 matrices and $\Sigma = \text{diag}(\sigma_1, \sigma_2)$ is a diagonal matrix with the singular values $\sigma_1 \geq \sigma_2 > 0$ where both singular values are positive because the \mathbf{A} is nonsingular. The orthogonal matrix \mathbf{V} can be regarded as the input matrix of the transformation, while Σ is the gain matrix, and \mathbf{U} is the output matrix. The orthogonality of \mathbf{U} and \mathbf{V} implies that $\mathbf{U}^{-1} = \mathbf{U}^T$ and $\mathbf{V}^{-1} = \mathbf{V}^T$. The singular value decomposition can be written in terms of the orthogonal vectors of the matrices \mathbf{U} and \mathbf{V} as

$$\mathbf{A} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T \quad (200)$$

The effect of the matrix \mathbf{A} in a transformation of a vector \mathbf{p} is then

$$\mathbf{A}\mathbf{p} = \sigma_1 (\mathbf{v}_1^T \mathbf{p}) \mathbf{u}_1 + \sigma_2 (\mathbf{v}_2^T \mathbf{p}) \mathbf{u}_2 \quad (201)$$

This means that the length of \mathbf{p} in the direction of the input vector \mathbf{v}_i is multiplied with the singular value σ_i and the result is used to scale the output vector \mathbf{u}_i .

More insight is gained from the polar decomposition of the transformation matrix \mathbf{A} which is derived from the singular value decomposition by using $\mathbf{V}^T \mathbf{V} = \mathbf{I}$. This leads to the expression

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T = \mathbf{U}\mathbf{V}^T \mathbf{V}\Sigma\mathbf{V}^T = (\mathbf{U}\mathbf{V}^T)(\mathbf{V}\Sigma\mathbf{V}^T) \quad (202)$$

Then the orthogonal matrix $\mathbf{R} = \mathbf{U}\mathbf{V}^T \in O(2)$ and the positive definite matrix $\mathbf{P} = \mathbf{V}\Sigma\mathbf{V}^T$ can be defined, and the polar decomposition is written in the form

$$\mathbf{A} = \mathbf{R}\mathbf{P} \quad (203)$$

The orthogonal matrix \mathbf{R} is either a rotation matrix, which is the case if $\det \mathbf{R} = 1$, or it can be a reflection matrix, which is the case if $\det \mathbf{R} = -1$. The positive definite matrix \mathbf{P} is a matrix that stretches the space with the factors σ_1 and σ_2 along two orthogonal directions defined

by the input orthogonal matrix \mathbf{V} . The orthogonal matrix \mathbf{V} is either a rotation matrix with determinant $+1$ or a reflection matrix with determinant -1 .

This shows that an affine transformation given by the matrix \mathbf{A} can be decomposed as a stretching with individual scale factors σ_i along orthogonal axes \mathbf{v}_i followed by a rotation or reflection \mathbf{R} .

The positive definite matrix \mathbf{P} can be written in terms of the input vectors and the singular values as

$$\mathbf{P} = \sigma_1 \mathbf{v}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{v}_2 \mathbf{v}_2^T \quad (204)$$

The effect of a transformation of a vector \mathbf{p} is then

$$\mathbf{Pp} = \sigma_1 (\mathbf{v}_1^T \mathbf{p}) \mathbf{v}_1 + \sigma_2 (\mathbf{v}_2^T \mathbf{p}) \mathbf{v}_2 \quad (205)$$

The geometric interpretation of this is that \mathbf{p} is scaled with the singular value σ_1 in the direction of \mathbf{v}_1 , and with σ_2 in the direction of \mathbf{v}_2 . This gives a stretching of the vector \mathbf{p} by a factor σ_i in the direction given by \mathbf{v}_i .

It is noted that the orthogonal matrix \mathbf{R} can be written in terms of the input and output vectors as

$$\mathbf{R} = \mathbf{U} \mathbf{V}^T = \mathbf{u}_1 \mathbf{v}_1^T + \mathbf{u}_2 \mathbf{v}_2^T \quad (206)$$

The affine transformation is

$$\mathbf{Ap} = \mathbf{R} \mathbf{P} \mathbf{p} = \sigma_1 (\mathbf{v}_1^T \mathbf{p}) \mathbf{u}_1 + \sigma_2 (\mathbf{v}_2^T \mathbf{p}) \mathbf{u}_2 \quad (207)$$

It is seen that the affine transformation gives a stretching of the vector \mathbf{p} by σ_i in the direction \mathbf{v}_i , and then the result is rotated or reflected by the matrix \mathbf{R} to the direction \mathbf{u}_i .

Example

A stretching of the x direction by a factor 1.5 is done with the affine transformation

$$\Sigma = \begin{bmatrix} 1.5 & 0 \\ 0 & 1 \end{bmatrix}$$

The unit vector in the x direction is written $\mathbf{e}_1 = [1, 0]^T$ and the unit vector in the y direction is written $\mathbf{e}_2 = [0, 1]^T$.

Let

$$\mathbf{V} = \begin{bmatrix} \cos(\pi/4) & -\sin(\pi/4) \\ \sin(\pi/4) & \cos(\pi/4) \end{bmatrix} \quad (208)$$

be a rotation matrix which represents a rotation $\pi/4$ in the xy plane. Then the same stretching along the vectors $\mathbf{v}_1 = \mathbf{V} \mathbf{e}_1$ and $\mathbf{v}_2 = \mathbf{V} \mathbf{e}_2$ gives the affine transformation

$$\mathbf{P} = \mathbf{V} \Sigma \mathbf{V}^T = \begin{bmatrix} 1.25 & 0.25 \\ 0.25 & 1.25 \end{bmatrix} \quad (209)$$

Finally, let \mathbf{R} be a rotation matrix which represents a rotation $\pi/6$ in the xy plane. Then the stretching in the directions \mathbf{v}_1 and \mathbf{v}_2 are rotated by \mathbf{R} according to

$$\mathbf{A} = \mathbf{R} \mathbf{P} = \mathbf{R} \mathbf{V} \Sigma \mathbf{V}^T = \begin{bmatrix} 0.9575 & -0.4085 \\ 0.8415 & 1.2075 \end{bmatrix} \quad (210)$$

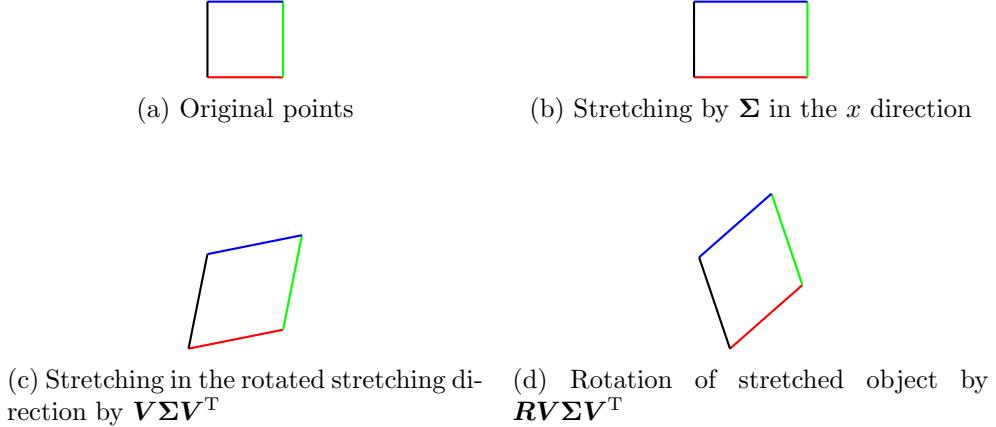


Figure 26: Polar decomposition of affine transformation

□

5.12 Affine transformation of points at infinity and the line at infinity

An affine transformation will transform a point $z_\infty = [\mathbf{a}^T, 0]^T$ at infinity according to

$$\mathbf{H}_a z_\infty = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{A}\mathbf{a} \\ 0 \end{bmatrix} = z'_\infty \quad (211)$$

where z'_∞ is a point at infinity in the direction $\mathbf{A}\mathbf{a}$.

It is also interesting to note that a line $\ell_\infty = [0, 0, 1]^T$ transforms according to

$$\mathbf{H}_a^{-T} \ell_\infty = \begin{bmatrix} \mathbf{A}^{-T} & \mathbf{0} \\ -\mathbf{t}^T \mathbf{A}^{-T} & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \ell_\infty \quad (212)$$

This means that the a line at infinity remains the line at infinity under an affine mapping.

5.13 Projective transformations

Finally, the fourth class of transformations is the projective transformation

$$\mathbf{x}' = \mathbf{H}_p \mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v_3 \end{bmatrix} \mathbf{x} \quad (213)$$

The invariants of a projective transformation include collinearity of points, intersection of lines, tangency, tangent discontinuities and cross ratios.

5.14 Transformation of lines with a projective homography

The inverse of the projective transformation is

$$\mathbf{H}_p^{-1} = \begin{bmatrix} \mathbf{B}^{-1} & -\mathbf{B}^{-1}\mathbf{t}/v_3 \\ -\mathbf{v}^T \mathbf{B}^{-1}/v_3 & \mathbf{v}^T \mathbf{B}^{-1}\mathbf{t}/v_3^2 + 1/v_3 \end{bmatrix} \quad (214)$$

where $\mathbf{B} = \mathbf{A} - \mathbf{t}\mathbf{v}^T/v_3$. This may be verified by direct computation. This expression will be derived in a later section where a decomposition of the projective transformation is presented. It is noted that the transposed inverse is

$$\mathbf{H}_p^{-T} = \begin{bmatrix} \mathbf{B}^{-T} & -\mathbf{B}^{-T}\mathbf{v}/v_3 \\ -\mathbf{t}^T\mathbf{B}^{-T}/v_3 & \mathbf{v}^T\mathbf{B}^{-1}\mathbf{t}/v_3^2 + 1/v_3 \end{bmatrix} \quad (215)$$

Matlab Example:

In this example a 4×4 projective transformation \mathbf{H}_p is inverted. The expressions are the same as in the 3×3 case.

```
% Input: Projective transformation
A = [1 0 0; 0 cos(pi/4) -sin(pi/4); 0 sin(pi/4) cos(pi/4)];
A(1,1) = A(1,1)+1; % A is not a rotation matrix
t = [1;2;0]; v = [1; 1; 0]; w = 2;
Hp = [A t; v' w]

B = A - t*v'/w; Binv = inv(B); % To simplify expressions
% Result: Inverse projective transformation
Hpinv = [Binv -Binv*t/w; -v'*Binv/w v'*Binv*t/(w*w) + 1/w]

Hpinv*Hp %Test: Identity matrix expected
```

□

5.15 Transformation of points at infinity and the line at infinity

In the case of the general projective transformation \mathbf{H}_p , the point at infinity is mapped to a homogeneous point

$$\mathbf{x}' = \mathbf{H}_p \mathbf{x}_\infty = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v_3 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{A}\mathbf{p} \\ \mathbf{v}^T\mathbf{p} \end{bmatrix} \quad (216)$$

which is not at infinity when $\mathbf{v}^T\mathbf{p} \neq 0$.

The line at infinity ℓ_∞ is transformed by the general projective mapping \mathbf{H}_p according to $\ell' = \mathbf{H}_p^{-T}\ell_\infty$, which gives

$$\ell' = \begin{bmatrix} \mathbf{B}^{-1} & -\mathbf{B}^{-1}\mathbf{t}/v_3 \\ -\mathbf{v}^T\mathbf{B}^{-1}/v_3 & \mathbf{v}^T\mathbf{B}^{-1}\mathbf{t}/v_3^2 + 1/v_3 \end{bmatrix}^T \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \frac{1}{v_3} \begin{bmatrix} -\mathbf{B}^{-T}\mathbf{v} \\ \mathbf{t}^T\mathbf{B}^{-T}\mathbf{v}/v_3 + 1 \end{bmatrix} \quad (217)$$

It is seen that this line is not necessarily at infinity, and that $-\mathbf{B}^{-T}\mathbf{v}$ is the normal vector to the line.

5.16 Decomposition of a projective transformation

A projective transformation

$$\mathbf{H}_p = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v_3 \end{bmatrix} \quad (218)$$

where \mathbf{A} is a nonsingular matrix can always be decomposed as

$$\mathbf{H}_p = \mathbf{H}_s \mathbf{H}_{as} \mathbf{H}_{ps} = \begin{bmatrix} s\mathbf{R} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_3 \end{bmatrix} \quad (219)$$

This is a sequence of three transformations starting with the projective transformation \mathbf{H}_{ps} , followed by the affine transformation \mathbf{H}_{as} , and finally the similarity transform \mathbf{H}_s , where the transformations are given by

$$\mathbf{H}_{ps} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_3 \end{bmatrix}, \quad \mathbf{H}_{as} = \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{H}_s = \begin{bmatrix} s\mathbf{R} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (220)$$

Here \mathbf{K} is upper triangular and $\det \mathbf{K} = 1$.

This decomposition gives the following expression for the total projective transform:

$$\mathbf{H}_p = \begin{bmatrix} s\mathbf{R}\mathbf{K} + \mathbf{r}\mathbf{v}^T & v_3\mathbf{r} \\ \mathbf{v}^T & v_3 \end{bmatrix} \quad (221)$$

It is seen that $\mathbf{A} = s\mathbf{R}\mathbf{K} + \mathbf{r}\mathbf{v}^T$ and $\mathbf{t} = v_3\mathbf{r}$.

5.17 Detailed expressions for the decomposition of projective homographies

The inverse of a projective mapping \mathbf{H}_p can be done by inverting the decomposition. Then the inverse can be found from

$$\mathbf{H}_p^{-1} = \mathbf{H}_{ps}^{-1} \mathbf{H}_{as}^{-1} \mathbf{H}_s^{-1} \quad (222)$$

This gives

$$\mathbf{H}_p^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{v}^T/v_3 & 1/v_3 \end{bmatrix} \begin{bmatrix} \mathbf{K}^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} (1/s)\mathbf{R}^T & -(1/s)\mathbf{R}^T\mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (223)$$

$$= \begin{bmatrix} \mathbf{K}^{-1}\mathbf{R}^T/s & -\mathbf{K}^{-1}\mathbf{R}^T\mathbf{r}/s \\ -\mathbf{v}^T\mathbf{K}^{-1}\mathbf{R}^T/(sv_3) & \mathbf{v}^T\mathbf{K}^{-1}\mathbf{R}^T\mathbf{r}/(sv_3) + 1/v_3 \end{bmatrix} \quad (224)$$

which can be written

$$\mathbf{H}_p^{-1} = \begin{bmatrix} \mathbf{B}^{-1} & -\mathbf{B}^{-1}\mathbf{t}/v_3 \\ -\mathbf{v}^T\mathbf{B}^{-1}/v_3 & \mathbf{v}^T\mathbf{B}^{-1}\mathbf{t}/v_3^2 + 1/v_3 \end{bmatrix} \quad (225)$$

where $\mathbf{B} = s\mathbf{R}\mathbf{K}$. It is seen that \mathbf{B} is nonsingular, and that $\mathbf{B}^{-1} = \mathbf{K}^{-1}\mathbf{R}^T/s$. The inverse can also be written in terms of \mathbf{A} , \mathbf{t} , \mathbf{v} and v_3 as

$$\mathbf{H}_p^{-1} = \begin{bmatrix} (\mathbf{A} - \mathbf{t}\mathbf{v}^T/v_3)^{-1} & -(\mathbf{A} - \mathbf{t}\mathbf{v}^T/v_3)^{-1}\mathbf{t}/v_3 \\ -\mathbf{v}^T(\mathbf{A} - \mathbf{t}\mathbf{v}^T/v_3)^{-1}/v_3 & \mathbf{v}^T(\mathbf{A} - \mathbf{t}\mathbf{v}^T/v_3)^{-1}\mathbf{t}/v_3^2 + 1/v_3 \end{bmatrix} \quad (226)$$

Remark: The decomposition can also be done in the reverse order as

$$\mathbf{H}_{pr} = \mathbf{H}_{ps} \mathbf{H}_{as} \mathbf{H}_s = \begin{bmatrix} s\mathbf{K}\mathbf{R} & \mathbf{K}\mathbf{r} \\ s\mathbf{v}^T\mathbf{K}\mathbf{R} & \mathbf{v}^T\mathbf{K}\mathbf{r} + v_3 \end{bmatrix} \quad (227)$$

□

5.18 A closer look at the projective property of a transformation

A special case of the projective transformation which is used in the decomposition of projective transformations is

$$\mathbf{H}_{ps} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ v_1 & v_2 & v_3 \end{bmatrix} \quad (228)$$

where $\mathbf{v} = [v_1, v_2]^T$. Note that the last row of the matrix can be interpreted as the transpose of the line

$$\ell_p = \begin{bmatrix} \mathbf{v} \\ v_3 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \quad (229)$$

which has normal vector \mathbf{v} , and the distance from the origin to the line in the direction of \mathbf{v} is $\delta_p = -v_3/\|\mathbf{v}\|$. It is noted that the distance from a point $\mathbf{x} = [\mathbf{p}^T, 1]^T$ to the line is

$$\delta_x = -\frac{\ell_p^T \mathbf{x}}{\|\mathbf{v}\|} = -\frac{\mathbf{v}^T \mathbf{p} + v_3}{\|\mathbf{v}\|} \quad (230)$$

The projective transformation \mathbf{H}_{ps} will transform a point at infinity $\mathbf{z}_\infty = [a_1, a_2, 0]^T = [\mathbf{a}^T, 0]^T$ to a point \mathbf{z}' which is not necessarily at infinity, which is seen from

$$\mathbf{z}' = \mathbf{H}_{ps} \mathbf{z}_\infty = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ 0 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \mathbf{v}^T \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{a} \\ \mathbf{v}^T \mathbf{a} \end{bmatrix} \quad (231)$$

Note that \mathbf{z}' will still be a point at infinity if $\mathbf{v}^T \mathbf{a} = 0$. The point \mathbf{z}' is then called a vanishing point. It is seen that for all points \mathbf{z}_∞ satisfying $\mathbf{v}^T \mathbf{a} \neq 0$ the resulting vanishing point \mathbf{z}' will correspond to a point

$$\mathbf{p}' = \frac{\mathbf{a}}{\mathbf{v}^T \mathbf{a}} = \begin{bmatrix} \frac{a_1}{\mathbf{v}^T \mathbf{a}} \\ \frac{a_2}{\mathbf{v}^T \mathbf{a}} \end{bmatrix}, \quad (232)$$

A point at infinity in the direction of the x axis will be given by $\mathbf{z}_x = [1, 0, 0]^T$, which has the vanishing point $\mathbf{z}'_x = [1, 0, v_1]^T$, which is the point $\mathbf{p}'_x = [1/v_1, 0]^T$ in the plane.

A point at infinity in the direction of the y axis will be given by $\mathbf{z}_y = [0, 1, 0]^T$, which has the vanishing point $\mathbf{z}'_y = [0, 1, v_2]^T$, which is the point $\mathbf{p}'_y = [0, 1/v_2]^T$ in the plane.

This means that two vanishing points have been found at \mathbf{z}'_x and \mathbf{z}'_y . Then the vanishing line is found to be

$$\ell' = \mathbf{z}'_x \times \mathbf{z}'_y = \begin{bmatrix} 0 & -v_1 & 0 \\ v_1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ v_2 \end{bmatrix} = \begin{bmatrix} -v_1 \\ -v_2 \\ 1 \end{bmatrix} = -\begin{bmatrix} \mathbf{v} \\ -1 \end{bmatrix} \quad (233)$$

The same result is found from $\ell' = \mathbf{H}_{ps}^{-T} \ell_\infty$.

If $\mathbf{v}^T \mathbf{a} = 0$, then $\mathbf{z}' = (\mathbf{a}^T, 0)^T$. Then also the transformed point is a point at infinity in the direction of the line ℓ' . The point \mathbf{z}' will still be on the line ℓ' .

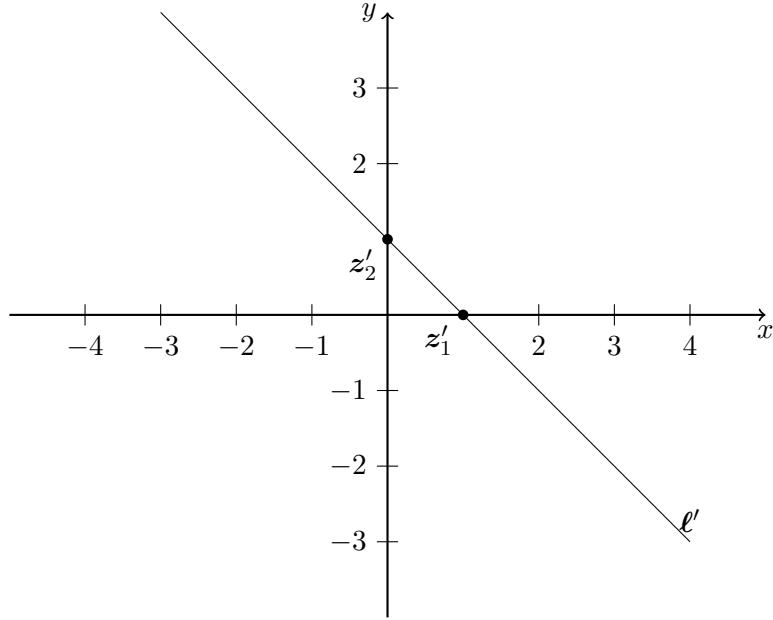


Figure 27: The projective homography \mathbf{H}_{p1} maps the two points \mathbf{z}_1 and \mathbf{z}_2 at infinity to the vanishing points $\mathbf{z}'_1 = \mathbf{H}_{p1}\mathbf{z}_1$ and $\mathbf{z}'_2 = \mathbf{H}_{p1}\mathbf{z}_2$. These vanishing points are on the vanishing line $\ell' = \mathbf{H}_{p1}^{-T}\ell_\infty$.

Example 1

Consider the perspective transformation $\mathbf{x}' = \mathbf{H}_{p1}\mathbf{x}$ where

$$\mathbf{H}_{p1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad (234)$$

Then the line at infinity is transformed to

$$\ell' = \mathbf{H}_{p1}^{-T}\ell_\infty = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \quad (235)$$

This is the line $y = -x + 1$ which intersects the y axis at $x = 1$ and the x axis at $y = 1$. The direction vector along ℓ' can be written $\mathbf{a} = (-1, 1)^T$.

Consider the point $\mathbf{z}_1 = (a, 0, 0)^T$ which is at infinity in the direction of the x axis. The perspective transformation of the point gives

$$\mathbf{z}'_1 = \mathbf{H}_{p1}\mathbf{z}_1 = a[1, 0, 1]^T \quad (236)$$

The transformed point is also on the x axis, but the point has been moved from infinity to a finite point, which is at $[1, 0]^T$ on the affine plane.

The point $\mathbf{z}_2 = (0, b, 0)^T$, which is at infinity in the direction along the y axis is transformed to

$$\mathbf{z}'_2 = \mathbf{H}_{p1}\mathbf{z}_2 = b[0, 1, 1]^T \quad (237)$$

which is a finite point on the y axis, which is at $[0, 1]^T$ on the affine plane..

A point at infinity which is not made finite by the transformation the the point $\mathbf{z}_3 = (\mathbf{a}^T, 0)^T = (-1, 1, 0)^T$, which is a point at infinity in the direction of \mathbf{a} , which is the direction vector of ℓ' . In this case

$$\mathbf{z}'_3 = \mathbf{H}_{p1}\mathbf{z}_3 = [-1, 1, 0]^T \quad (238)$$

which is still at infinity in the direction of \mathbf{a} . \square

Example 2

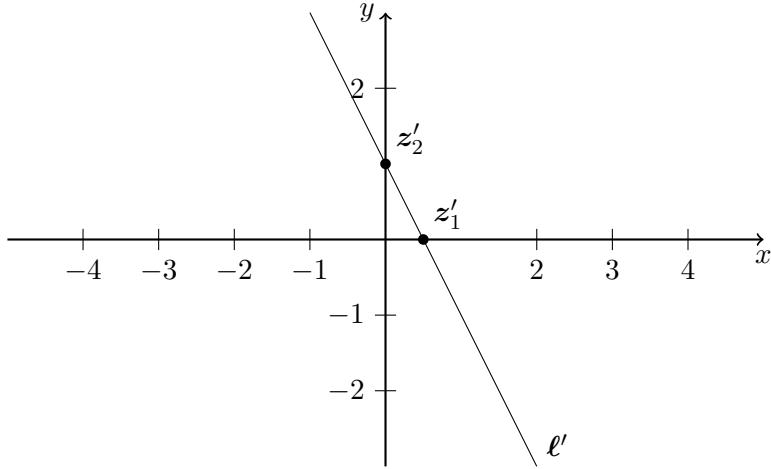


Figure 28: The projective homography \mathbf{H}_{p1} maps the two points \mathbf{z}_1 and \mathbf{z}_2 at infinity to the vanishing points $\mathbf{z}'_1 = \mathbf{H}_{p1}\mathbf{z}_1$ and $\mathbf{z}'_2 = \mathbf{H}_{p1}\mathbf{z}_2$. These vanishing points are on the vanishing line $\ell' = \mathbf{H}_{p1}^{-T}\ell_\infty$.

Next, consider the projective transformation $\mathbf{z}' = \mathbf{H}_{p2}\mathbf{z}$ where

$$\mathbf{H}_{p2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 1 & 2 \end{bmatrix} \quad (239)$$

Then the line at infinity is transformed to

$$\ell'_2 = \mathbf{H}_{p2}^{-T}\ell_\infty = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -0.5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -0.5 \\ 0.5 \end{bmatrix} \quad (240)$$

This is the line $y = -2x + 1$ which intersects the y axis at $x = 1$ and the x axis at $y = 2$. The direction vector along ℓ' can be written $\mathbf{a}_2 = (-1, 2)^T$. The points \mathbf{z}_1 and \mathbf{z}_2 are then transformed to

$$\mathbf{z}'_1 = \mathbf{H}_{p2}\mathbf{z}_1 = 2a(0.5, 0, 1)^T \quad (241)$$

$$\mathbf{z}'_2 = \mathbf{H}_{p2}\mathbf{z}_2 = b(0, 1, 1)^T \quad (242)$$

In this case the point $\mathbf{z}_4 = (-1, 2, 0)^T$ at infinity will remain unchanged at infinity under the perspective transformation \mathbf{H}_{p2} , which is verified by $\mathbf{z}'_4 = \mathbf{H}_{p2}\mathbf{z}_4 = \mathbf{z}_4$. The reason for this is that the point is in the direction of the vanishing line. \square

5.19 Projective transformations: Case 2

Consider the projective transformation $\mathbf{x}' = \mathbf{H}_p \mathbf{x}$ where the projective transformation is given by

$$\mathbf{H}_p = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{v}^T & v_3 \end{bmatrix} \quad (243)$$

where \mathbf{R} is a rotation matrix. Then a point $\mathbf{x} = (\mathbf{p}^T, 1)^T$ representing the Euclidean point $\mathbf{p} = (x, y)^T$ is transformed to the homogeneous point

$$\mathbf{x}' = \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = x'_3 \begin{bmatrix} \mathbf{p}' \\ 1 \end{bmatrix} \quad (244)$$

which represents the Euclidean point $\mathbf{p}' = (x', y')^T$. The transformation is

$$\mathbf{x}' = x'_3 \begin{bmatrix} \mathbf{p}' \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{v}^T & v_3 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}\mathbf{p} + \mathbf{t} \\ \mathbf{v}^T \mathbf{p} + v_3 \end{bmatrix} = (\mathbf{v}^T \mathbf{p} + v_3) \begin{bmatrix} \frac{\mathbf{R}\mathbf{p} + \mathbf{t}}{\mathbf{v}^T \mathbf{p} + v_3} \\ 1 \end{bmatrix} \quad (245)$$

The transformation of the Euclidean points is then

$$\mathbf{p}' = \frac{\mathbf{R}\mathbf{p} + \mathbf{t}}{\mathbf{v}^T \mathbf{p} + v_3} = \frac{\mathbf{R}\mathbf{p} + \mathbf{t}}{\ell_p^T \mathbf{x}} \quad (246)$$

where the line $\ell_p = [\mathbf{v}^T, v_3]^T$ is given as the transpose of the last row of \mathbf{H}_p . Here the appearance of the denominator gives a projection in the mapping from \mathbf{p} to \mathbf{p}' . It is noted that the denominator expression $\ell_p^T \mathbf{x}$ is proportional to the length from the point \mathbf{x} to the line ℓ_p .

It is seen that the projective transformation $\mathbf{x}' = \mathbf{H}_p \mathbf{x}$ in homogeneous coordinates is linear, while the resulting transformation from \mathbf{p} to \mathbf{p}' in Euclidean coordinates is nonlinear.

Example 1

Consider the projective transformation

$$\mathbf{H}_p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.25 & 0.25 & 1 \end{bmatrix} \quad (247)$$

A point $\mathbf{x} = [x_1, x_2, x_3]^T$ is then mapped to a point

$$\mathbf{x}' = \mathbf{H}_p \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ 0.25x_1 + 0.25x_2 + x_3 \end{bmatrix} = (0.25x_1 + 0.25x_2 + x_3) \begin{bmatrix} \frac{x_1}{0.25x_1 + 0.25x_2 + x_3} \\ \frac{x_2}{0.25x_1 + 0.25x_2 + x_3} \\ 1 \end{bmatrix} \quad (248)$$

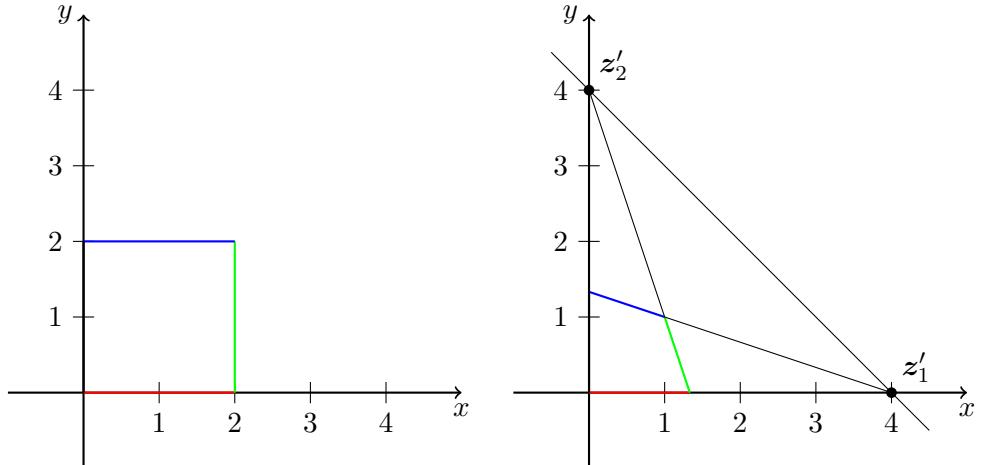
The vanishing point \mathbf{z}_1 of a point $\mathbf{x}_1 = [1, 0, 0]^T$ at infinity along the x axis is

$$\mathbf{z}_1 = \begin{bmatrix} 1 \\ 0 \\ 0.25 \end{bmatrix} = 0.25 \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix}$$

while the vanishing point \mathbf{z}_2 of a point $\mathbf{x}_2 = [10, 1, 0]^T$ at infinity along the x axis is

$$\mathbf{z}_2 = \begin{bmatrix} 0 \\ 1 \\ 0.25 \end{bmatrix} = 0.25 \begin{bmatrix} 0 \\ 4 \\ 1 \end{bmatrix}$$

The vanishing line will be at $\ell_v = \mathbf{z}_1 \times \mathbf{z}_2 = [-0.25, -0.25, 1]^T = 0.25[-1, -1, 4]^T$, which can be written $y = -x + 4$. This can be verified with $\mathbf{H}^T \ell_\infty = \ell$.



(a) Original square

(b) Transformed square. The vanishing points \mathbf{z}'_1 and \mathbf{z}'_2 are indicated as the intersection of the parallel lines. The vanishing line $y = -x + 4$ runs through the two vanishing points.

Figure 29: The projective transformation of a square.

□

Example 2

Consider the projective transformation

$$\mathbf{H}_p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.2 & 0.2 & 0.8 \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \ell_p^T & \end{bmatrix} \quad (249)$$

where the last row is the transpose of the line $\ell_p = [0.2, 0.2, 0.8]^T$, which can also be written in the form $y = -x - 4$ (Figure 30). The normal vector of ℓ_p is $\mathbf{v} = [v_1, v_2]^T = [0.2, 0.2]^T$.

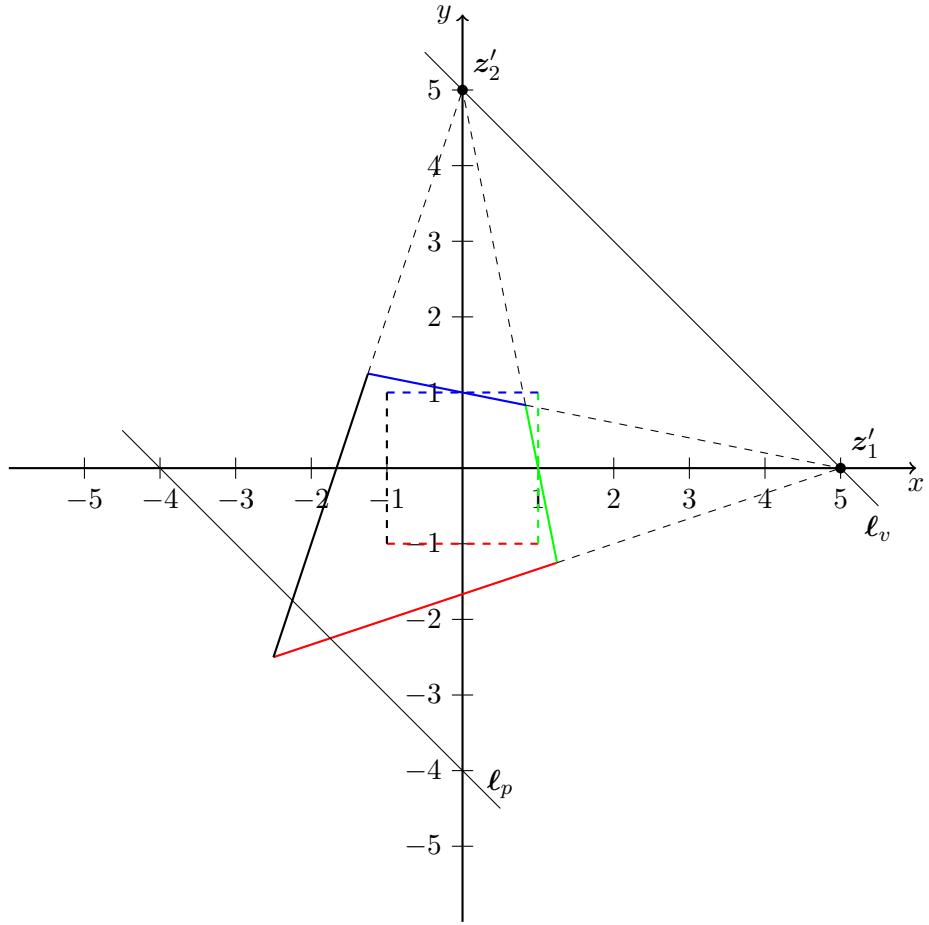


Figure 30: The projective transformation of a square. The line ℓ_p which defines the last row of the homography is shown. The vanishing line ℓ_v , which is parallel to ℓ_p is also shown. The part of the square that is closer to ℓ_p than the line $y = -x$ is made larger by the homography, while the part of the square that is on the other side of $y = -x$ is made smaller.

A homogeneous point $\mathbf{x} = [x, y, 1]^T$ is then mapped to a point

$$\mathbf{x}' = \mathbf{H}_p \mathbf{x} = \begin{bmatrix} x \\ y \\ 0.2x + 0.2y + 0.8 \end{bmatrix} = \begin{bmatrix} x \\ y \\ \ell_p^T \mathbf{x} \end{bmatrix} \quad (250)$$

The resulting point in the Euclidean plane will be

$$\mathbf{p}' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{x}{0.2x+0.2y+0.8} \\ \frac{y}{0.2x+0.2y+0.8} \end{bmatrix} \quad (251)$$

The distance from the point \mathbf{x} to the line ℓ_p is $\delta = -\ell_p^T \mathbf{x} / \|\mathbf{v}\|$, where $\|\mathbf{v}\| = \sqrt{0.08} = 0.2828$. It follows that

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \frac{1}{\ell_p^T x} \begin{bmatrix} x \\ y \end{bmatrix} \quad (252)$$

where $\ell_p^T \mathbf{x} = \|\mathbf{v}\| \delta$.

The line parallel to ℓ_p through the points $(0, 1)$ and $(1, 0)$ is $\ell_1 = [0.2, 0.2, -0.2]^T$, and is given by given by $0.2x + 0.2y = 0.2$. A point that is between ℓ_p and ℓ_1 will have $0.2x + 0.2y < 0.2$, which implies that the scaling factor satisfies $1/(\ell_p^T \mathbf{x}) > 1$. This scaling factor tends to infinity when the point approaches ℓ_p , and the transformation does not correspond to any Euclidean point \mathbf{p}' when $\ell_p^T \mathbf{x} = 0$, which occurs when \mathbf{x} is on ℓ_p . A point that is on the other side of ℓ_1 will have a scaling factor $1/(\ell_p^T \mathbf{x}) < 1$.

Moreover, when \mathbf{x} tends to a point at infinity $\mathbf{z}_\infty = [a_1, a_2, 0]^T$ in the direction of $\mathbf{a} = [a_1, a_2]^T$, the transformed point \mathbf{x}' will tend to a point on the vanishing line given by

$$\mathbf{z}'_v = \frac{1}{v_1 a_1 + v_2 a_2} \begin{bmatrix} a_1 \\ a_2 \\ v_1 a_1 + v_2 a_2 \end{bmatrix} \quad (253)$$

It is straightforward to verify \mathbf{z}'_v is on the vanishing line by evaluating $\ell_v^T \mathbf{z}'_v = 0$. It is seen that the vanishing point \mathbf{z}'_1 on the x axis and the vanishing point \mathbf{z}'_2 on the y axis are given by

$$\mathbf{z}'_1 = \frac{1}{v_1} \begin{bmatrix} 1 \\ 0 \\ v_1 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{z}'_2 = \frac{1}{v_2} \begin{bmatrix} 0 \\ 1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 5 \\ 1 \end{bmatrix} \quad (254)$$

The vanishing line will be at $\ell_v = \mathbf{z}'_1 \times \mathbf{z}'_2 = [-1, -1, 5]^T$, which can be written $y = -x + 5$. This is verified from $\mathbf{H}^T \ell_v = \ell_\infty$. It is interesting that in this example the line ℓ_p is parallel to the vanishing line ℓ_v .

It is noted that

$$\mathbf{H}_p^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -0.25 & -0.25 & 1.25 \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \ell_v^T & \end{bmatrix} \quad (255)$$

where the last row is a scaled version of the vanishing line ℓ_v .

□

5.20 The projective transformation of the camera model

The camera model has a projective effect that in certain cases can be described as a projective transformation from a plane to a plane. This is used in connection with calibration where a images of a checkerboard is used. Suppose that an image is taken of a planar object. The object frame o is fixed in the object so that the plane is in the xy plane of the o frame. This means that any point on the plane is described by the position vector $\mathbf{p}_o = (x_o, y_o)^T$ relative to the origin of the o frame. The transformation from the camera frame c to the object frame o is given by

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (256)$$

where $\mathbf{R} = \mathbf{R}_o^c$ and $\mathbf{t} = \mathbf{t}_{co}^c = (t_1, t_2, t_3)^T$. The position vector of a point in the o frame is $\mathbf{r}_o = (x_o, y_o, z_o)^T$. On the plane of the object the position vector is $\mathbf{r}_o = (x_o, y_o, 0)^T = (\mathbf{p}_o^T, 0)^T$. The geometric arrangement is shown in Figure 6.

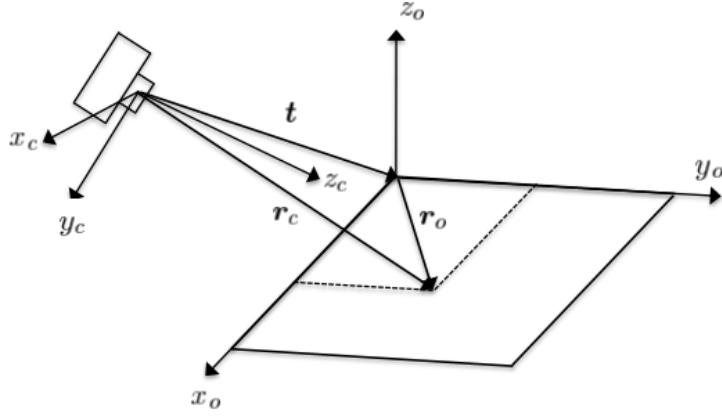


Figure 31: The geometric setup to study the perspective transformation of a camera where the point is in the xy plane of the object frame.

The position $\mathbf{r}_c = (x, y, z)^T$ of a point in the camera frame is given in homogeneous coordinates as

$$\tilde{\mathbf{r}}_c = \mathbf{T} \tilde{\mathbf{r}}_o \quad (257)$$

where $\tilde{\mathbf{r}}_c = (\mathbf{r}_c^T, 1)^T$ and $\tilde{\mathbf{r}}_o = (\mathbf{r}_o^T, 1)^T$ are homogeneous vectors. This is often written in the form

$$\mathbf{r}_c = [\mathbf{R} \quad \mathbf{t}] \tilde{\mathbf{r}}_o \quad (258)$$

In coordinate form this is

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \boldsymbol{\rho}_1^T & t_1 \\ \boldsymbol{\rho}_2^T & t_2 \\ \boldsymbol{\rho}_3^T & t_3 \end{bmatrix} \begin{bmatrix} \mathbf{r}_o \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\rho}_1^T \mathbf{r}_o + t_1 \\ \boldsymbol{\rho}_2^T \mathbf{r}_o + t_2 \\ \boldsymbol{\rho}_3^T \mathbf{r}_o + t_3 \end{bmatrix} \quad (259)$$

where $\boldsymbol{\rho}_i$ is row i of \mathbf{R} . To get a mapping from plane to plane, it is noticed that $z_o = 0$ on the object plane. The transformation can then be written

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & t_1 \\ R_{21} & R_{22} & t_2 \\ R_{31} & R_{32} & t_3 \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11}x_o + R_{12}y_o + t_1 \\ R_{21}x_o + R_{22}y_o + t_2 \\ R_{31}x_o + R_{32}y_o + t_3 \end{bmatrix} \quad (260)$$

The projection appears in the definition of the normalized homogeneous image coordinates $\tilde{\mathbf{s}} = (s_x, s_y, 1)^T$, which are given by

$$\tilde{\mathbf{s}} = \frac{1}{z} \mathbf{r}_c = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \quad (261)$$

It is seen from (260) that the normalized image coordinates are

$$s_x = \frac{R_{11}x_o + R_{12}y_o + t_1}{R_{31}x_o + R_{32}y_o + t_3} \quad (262)$$

$$s_y = \frac{R_{21}x_o + R_{22}y_o + t_2}{R_{31}x_o + R_{32}y_o + t_3} \quad (263)$$

where the projection is clearly seen.

To make the connection with homographies, this is written as the projective transformation

$$\mathbf{x}' = \mathbf{H}_p \mathbf{x} \quad (264)$$

where $\mathbf{x}' = z\tilde{\mathbf{s}} = (x, y, z)^T$, $\mathbf{x} = \tilde{\mathbf{p}}_o = (x_o, y_o, 1)^T$, and the homography is

$$\mathbf{H}_p = \begin{bmatrix} R_{11} & R_{12} & t_1 \\ R_{21} & R_{22} & t_2 \\ R_{31} & R_{32} & t_3 \end{bmatrix} \quad (265)$$

which is clearly a projective transformation of the form

$$\mathbf{H}_p = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix} \quad (266)$$

with

$$\mathbf{A} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} R_{31} \\ R_{32} \end{bmatrix}, \quad v_3 = t_3 \quad (267)$$

Example

Consider an image where the scene is a planar object where the object plane is the xy plane of the o frame. The transformation from the camera frame c to the object frame o is given by

$$\mathbf{T}_o^c = \begin{bmatrix} \mathbf{R}_y(\pi/4) & t_3 \mathbf{z} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (268)$$

where $t_3 = 2$ and $\mathbf{z} = (0, 0, 1)^T$ (Figure 32). The 4 points on the object plane, which is the xy plane of the o frame, have homogeneous position vectors $\tilde{\mathbf{r}}_{oi} = (\tilde{\mathbf{p}}_{oi}^T, 0, 1)^T$ where the coordinates $\tilde{\mathbf{p}}_{oi} = (x_{oi}, y_{oi})^T$ in the plane are given by

$$\tilde{\mathbf{p}}_{o1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \tilde{\mathbf{p}}_{o2} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \tilde{\mathbf{p}}_{o3} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \tilde{\mathbf{p}}_{o4} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad (269)$$

The square is shown in Figure 33. Then the positions $\mathbf{r}_{ci} = [\mathbf{R} \ \mathbf{t}] \tilde{\mathbf{r}}_{oi}$ of the points in the camera frame are

$$\mathbf{r}_{c1} = \begin{bmatrix} 0.7071 \\ 1 \\ 1.2929 \end{bmatrix}, \quad \mathbf{r}_{c2} = \begin{bmatrix} 0.7071 \\ -1 \\ 1.2929 \end{bmatrix}, \quad \mathbf{r}_{c3} = \begin{bmatrix} -0.7071 \\ -1 \\ 2.7071 \end{bmatrix}, \quad \mathbf{r}_{c4} = \begin{bmatrix} -0.7071 \\ 1 \\ 2.7071 \end{bmatrix} \quad (270)$$

The perspective is apparent in the normalized image coordinates (Figure 34), that are given by

$$\tilde{\mathbf{s}}_1 = \begin{bmatrix} 0.5469 \\ 0.7735 \\ 1 \end{bmatrix}, \quad \tilde{\mathbf{s}}_2 = \begin{bmatrix} 0.5469 \\ -0.7735 \\ 1 \end{bmatrix}, \quad \tilde{\mathbf{s}}_3 = \begin{bmatrix} -0.2612 \\ -0.3694 \\ 1 \end{bmatrix}, \quad \tilde{\mathbf{s}}_4 = \begin{bmatrix} -0.2612 \\ 0.3694 \\ 1 \end{bmatrix} \quad (271)$$

The Matlab code for the transformation:

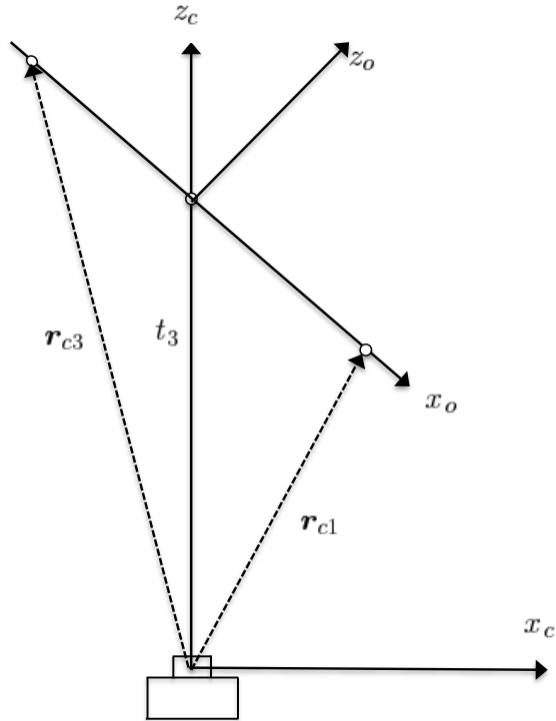


Figure 32: An image is taken of a scene where the object frame is translated $t_3 = 2$ m from the camera frame along the z_c axis and then rotated $\pi/4$ about the y axis. A square is defined by 4 corner points p_{o1}, \dots, p_{o4} in the xy plane of the object frame. Due to the rotation about the y axis a corner point with positive x_o coordinate like point 1 will be closer to the camera than a points with negative x_o coordinate like point 3. This is seen in the figure where the z_c coordinate of r_{c1} is less than the z_c coordinate of r_{c3} . This gives a perspective effect in the normalized image coordinates.

```
% Script for calculating the normalized image coordinates for a square
% in the xy plane of the object frame. The object frame is translated
% 2 m along the z axis of the camera frame and rotated about the y axis
% relative to the camera frame. The rotation about the y axis gives
% a perspective effect.

% Input: Points of a square in the xy plane of the object frame
pt1 = [1;1;1]; pt2 = [1;-1;1]; pt3 = [-1;-1;1]; pt4 = [-1;1;1];
% Input: Transformation from camera frame to object frame
R = Roty(pi/4); t = [0;0;2]; % Rotation of pi/4 about y, translation 2 along z_c

% Projective homogeneous transformation matrix
Hp = [R(1,1) R(1,2) t(1); R(2,1) R(2,2) t(2); R(3,1) R(3,2) t(3)];
rc1 = Hp*pt1; rc2 = Hp*pt2; rc3 = Hp*pt3; rc4 = Hp*pt4; % Camera coordinates
% Normalized image coordinates
```

```

st1 = rc1/rc1(3);  st2 = rc2/rc2(3);  st3 = rc3/rc3(3);  st4 = rc4/rc4(3);

figure(1); clf;
plot([pt1(1) pt2(1)], [pt1(2),pt2(2)], 'b', ...
      [pt2(1) pt3(1)], [pt2(2) pt3(2)], 'r', ...
      [pt3(1) pt4(1)], [pt3(2) pt4(2)], 'k', ...
      [pt4(1) pt1(1)], [pt4(2) pt1(2)], 'm');
grid; axis([-2 2 -2 2]);
figure(2); clf;
plot([st1(1) st2(1)], [st1(2),st2(2)], 'b', ...
      [st2(1) st3(1)], [st2(2) st3(2)], 'r', ...
      [st3(1) st4(1)], [st3(2) st4(2)], 'k', ...
      [st4(1) st1(1)], [st4(2) st1(2)], 'm');
grid; axis([-2 2 -2 2]);

```

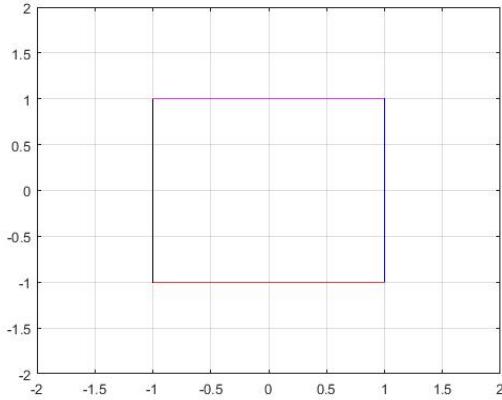


Figure 33: Original square in the xy plane of the object frame.

□

6 Computation of homographies in 2D

6.1 Computation of homographies from point mappings by DLT

Consider the homography $\mathbf{x}' = \mathbf{H}\mathbf{x}$ in 2D. The matrix $\mathbf{H} = \{h_{ij}\}$ of the transformation will have 8 independent elements due to the freedom of scaling. This means that 8 equations are needed to determine the elements of \mathbf{H} . A general method for calculating the elements of \mathbf{H} is based on rearranging a point mapping $\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i$ from a point \mathbf{x}_i to a point \mathbf{x}'_i into a form

$$\mathbf{A}_i \mathbf{h} = \mathbf{0} \quad (272)$$

where \mathbf{A}_i is given by the corresponding points \mathbf{x}_i and \mathbf{x}'_i , and

$$\mathbf{h} = (h_{11}, h_{21}, h_{31}, h_{12}, h_{22}, h_{32}, h_{13}, h_{23}, h_{33})^T$$

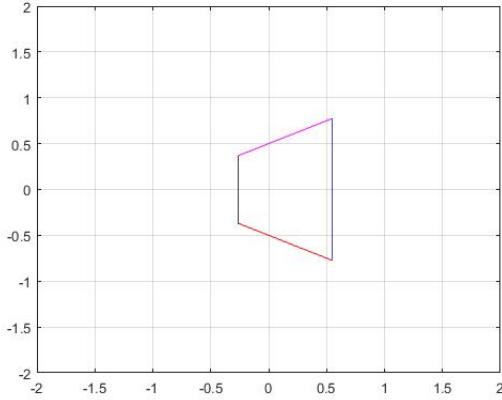


Figure 34: Square projected to the image plane and given in normalized image coordinates. Note that perspective affect due to the rotation of the object frame about the vertical x_c axis, where the distant points are on the left side of the projected square. It is clearly seen that the two lines that were originally horizontal now intersect at a point in infinity which is on the s_x axis.

is a column vector of dimension 9 containing all the elements of \mathbf{H} . This can be written

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix} \quad (273)$$

where \mathbf{h}_i is column vector i in

$$\mathbf{H} = [\mathbf{h}_1 \ \mathbf{h}_2 \ \mathbf{h}_3] \quad (274)$$

A point mapping in the plane involves the transformation of two coordinates, and it will provide two equations for the determination of the homography. This means that the matrix \mathbf{A}_i for one point mapping will give two independent equations. Therefore, to have 8 independent equations it is required to have at least 4 point mappings. In order for the 4 mappings to be independent the points \mathbf{x} are required to be in general position, which means that they are linearly independent. This means that no more than 2 points can be at the same line. With 4 independent point mappings there will be 4 independent matrices \mathbf{A}_1 , \mathbf{A}_2 , \mathbf{A}_3 and \mathbf{A}_4 that can be assembled into a matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \mathbf{A}_4 \end{bmatrix} \quad (275)$$

Then the homography \mathbf{H} can be found from

$$\mathbf{A}\mathbf{h} = \mathbf{0} \quad (276)$$

This type of equation is frequently seen in vision problems, and the solution technique is called direct linear transformation (DLT). It is clear that $\mathbf{h} = \mathbf{0}$ is a solution which is called the trivial solution. This solution is obviously not useful. Instead it is necessary to find a solution that is

in the nullspace of \mathbf{A} . This is typically done with the singular value decomposition. This will be described in the following for the two situations where there are 8 equations, which is the case when $\mathbf{A} \in \mathbb{R}^{8 \times 9}$, and when there are $n > 8$ equations, in which case $\mathbf{A} \in \mathbb{R}^{n \times 9}$.

6.2 Computation from 4 point mappings

It is possible to calculate a homography from 4 point mappings if the points are in general position, which means the points are sufficiently independent. In this case this will be satisfied if there are no more than 2 point on the same line. Then there will be 4 matrices \mathbf{A}_i in the matrix \mathbf{A} . If each matrix \mathbf{A}_i has two rows, then there will be 8 rows in \mathbf{A} and as a result, 8 independent equations in $\mathbf{A}\mathbf{h} = \mathbf{0}$. This means that the dimension of \mathbf{A} is 8×9 . The singular value decomposition of \mathbf{A} is then

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T, \quad \mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_8) \in \mathbb{R}^{8 \times 8}, \quad \mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_9) \in \mathbb{R}^{9 \times 9} \quad (277)$$

and Σ is a matrix of dimension 8×9 with the singular values σ_i on the diagonal and zeros elsewhere. This can also be written

$$\mathbf{A} = \sum_{i=1}^8 \sigma_i \mathbf{u}_i \mathbf{v}_i^T \quad (278)$$

where $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_8 > 0$. This gives

$$\mathbf{A}\mathbf{h} = \sum_{i=1}^8 \sigma_i \mathbf{u}_i \mathbf{v}_i^T \mathbf{h} = 0 \quad (279)$$

It is seen that a nonzero \mathbf{h} must satisfy $\mathbf{v}_i^T \mathbf{h} = 0$ for $i = 1, \dots, 8$. The solution is then found by noting that $\mathbf{v}_i^T \mathbf{v}_9 = 0$ for $i = 1, \dots, 8$. From this it is concluded that a nonzero solution for \mathbf{h} is given by $\mathbf{h} = \mu \mathbf{v}_9$ for some nonzero constant μ which can be selected freely. Then \mathbf{H} is found by extracting the columns from \mathbf{h} . Note that \mathbf{h} is determined with a free scaling, which is consistent with the fact that a homography does not change when the scaling is changed.

In the case where \mathbf{A}_i has three rows, where two are independent, there will be 12 rows in \mathbf{A} , and therefore there will be 12 equations in $\mathbf{A}\mathbf{h} = \mathbf{0}$, where 8 are independent. This means that the matrix \mathbf{A} has dimension 12×9 . The singular value decomposition of \mathbf{A} is then

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T, \quad \mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_{12}) \in \mathbb{R}^{12 \times 12}, \quad \mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_9) \in \mathbb{R}^{9 \times 9} \quad (280)$$

and Σ is a matrix of dimension 12×9 with the singular values σ_i on the diagonal and zeros elsewhere. This can also be written

$$\mathbf{A} = \sum_{i=1}^9 \sigma_i \mathbf{u}_i \mathbf{v}_i^T \quad (281)$$

where $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_8 > \sigma_9 = 0$. This gives

$$\mathbf{A}\mathbf{h} = \sum_{i=1}^8 \sigma_i \mathbf{u}_i \mathbf{v}_i^T \mathbf{h} + \sigma_9 \mathbf{u}_9 \mathbf{v}_9^T \mathbf{h} = 0 \quad (282)$$

Also in this case \mathbf{h} must satisfy $\mathbf{v}_i^T \mathbf{h} = 0$ for $i = 1, \dots, 8$ because the corresponding singular values are greater than zero. The solution is again $\mathbf{h} = \mu \mathbf{v}_9$ for some nonzero constant μ that can be selected.

6.3 Computation from more than 4 point mappings

In the case where there are $n_p > 4$ point mappings the problem of finding the homography will be overdetermined. In this case there will be n submatrices \mathbf{A}_i in the matrix \mathbf{A} , which will have dimension $n \times 9$ is where $n = 2n_p$ if there are two rows in \mathbf{A}_i , and $n = 3n_p$ if there are 3 rows in \mathbf{A}_i . The singular value decomposition is

$$\mathbf{A} = \sum_{i=1}^9 \sigma_i \mathbf{u}_i \mathbf{v}_i^T \quad (283)$$

where $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_8 > \sigma_9 = 0$. Also in this case the solution is $\mathbf{h} = \mu \mathbf{v}_9$ for some nonzero constant μ that can be selected. Note that this solution is the least-squares solution. This means that if there is measurement noise in the values of \mathbf{x}_i and \mathbf{x}'_i , then this solution is the optimal solution in the least-squares sense.

6.4 Conditions from point mappings: Method 1

The first method for computation of a homography \mathbf{H} from point mappings is based on the observation that the cross product of a vector with the same vector is the zero vector. This is used to get [67]

$$\mathbf{S}(\mathbf{x}') \mathbf{x}' = \mathbf{x}' \times \mathbf{x}' = \mathbf{0}$$

where $\mathbf{S}(\mathbf{x}')$ is the skew symmetric form of \mathbf{x}' . For the homography $\mathbf{x}' = \mathbf{H}\mathbf{x}$, this gives the equation

$$\mathbf{S}(\mathbf{x}') \mathbf{H}\mathbf{x} = \mathbf{0}$$

This is rearranged by writing \mathbf{H} in terms of its column vectors as $\mathbf{H} = [\mathbf{h}_1 \ \mathbf{h}_2 \ \mathbf{h}_3]$. Let $\mathbf{x} = (x_1, x_2, x_3)^T$. Then

$$\mathbf{S}(\mathbf{x}') \mathbf{H}\mathbf{x} = \mathbf{S}(\mathbf{x}') [\mathbf{h}_1 \ \mathbf{h}_2 \ \mathbf{h}_3] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (284)$$

$$= \mathbf{S}(\mathbf{x}') [\mathbf{h}_1 x_1 \ \mathbf{h}_2 x_2 \ \mathbf{h}_3 x_3] \quad (285)$$

$$= [\mathbf{S}(\mathbf{x}') x_1 \ \mathbf{S}(\mathbf{x}') x_2 \ \mathbf{S}(\mathbf{x}') x_3] \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix} \quad (286)$$

This leads to the expression

$$\mathbf{A}\mathbf{h} = \mathbf{0}$$

where

$$\mathbf{A} = [x_1 \mathbf{S}(\mathbf{x}') \ x_2 \mathbf{S}(\mathbf{x}') \ x_3 \mathbf{S}(\mathbf{x}')]$$

and

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix}.$$

This equation gives two independent equations for the components of \mathbf{H} . The homography has 8 degrees of freedom as it has 9 elements, and the scaling can be selected. Therefore at

least four point mappings $\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i$ must be used to determine \mathbf{H} . Then one equation

$$\mathbf{A}_i \mathbf{h} = \mathbf{0}$$

is found for each point mapping, where

$$\mathbf{A}_i = \begin{bmatrix} x_{1i} \mathbf{S}(\mathbf{x}'_i) & x_{2i} \mathbf{S}(\mathbf{x}'_i) & x_{3i} \mathbf{S}(\mathbf{x}'_i) \end{bmatrix}$$

All four equations are included in

$$\mathbf{A}\mathbf{h} = \mathbf{0}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \mathbf{A}_4 \end{bmatrix}$$

which is used to solve for \mathbf{H} . If there are $n > 4$ the matrix \mathbf{A} will be point mappings, then the matrix \mathbf{A} will be

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_n \end{bmatrix}$$

and the solution, which now is a least-squares solution, is found in the same way.

6.5 A simplified set of equations

A modified formulation of Method 1 is presented in [30], which gives an efficient set of equations. In this formulation the the rows \mathbf{h}^{jT} of the homography \mathbf{H} are used, and the homography is written

$$\mathbf{H}\mathbf{x} = \begin{bmatrix} \mathbf{h}^{1T} \\ \mathbf{h}^{2T} \\ \mathbf{h}^{3T} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{h}^{1T}\mathbf{x} \\ \mathbf{h}^{2T}\mathbf{x} \\ \mathbf{h}^{3T}\mathbf{x} \end{bmatrix} \quad (287)$$

Let $\mathbf{x}' = (x'_1, x'_2, x'_3)^T$. Then

$$\mathbf{S}(\mathbf{x}')\mathbf{H}\mathbf{x} = \begin{bmatrix} x'_2 \mathbf{h}^{3T}\mathbf{x} - x'_3 \mathbf{h}^{2T}\mathbf{x} \\ x'_3 \mathbf{h}^{1T}\mathbf{x} - x'_1 \mathbf{h}^{3T}\mathbf{x} \\ x'_1 \mathbf{h}^{2T}\mathbf{x} - x'_2 \mathbf{h}^{1T}\mathbf{x} \end{bmatrix} \quad (288)$$

and the condition $\mathbf{S}(\mathbf{x}')\mathbf{H}\mathbf{x} = \mathbf{0}$ can be written

$$\begin{bmatrix} \mathbf{0}^T & -x'_3 \mathbf{x}^T & x'_2 \mathbf{x}^T \\ x'_3 \mathbf{x}^T & \mathbf{0}^T & -x'_1 \mathbf{x}^T \\ -x'_2 \mathbf{x}^T & x'_1 \mathbf{x}^T & \mathbf{0}^T \end{bmatrix} \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix} = \mathbf{0} \quad (289)$$

This gives three equations, where only two are linearly independent. This means that any of the 3 equations can be obtained as a linear combination of the two other equations. The condition can therefore be written by removing, e.g., the third row, which gives

$$\begin{bmatrix} \mathbf{0}^T & -x'_3 \mathbf{x}^T & x'_2 \mathbf{x}^T \\ x'_3 \mathbf{x}^T & \mathbf{0}^T & -x'_1 \mathbf{x}^T \end{bmatrix} \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix} = \mathbf{0} \quad (290)$$

Then for each point mapping $\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i$ there are two conditions $\mathbf{A}_i\mathbf{h} = 0$ where

$$\mathbf{A}_i = \begin{bmatrix} \mathbf{0}^T & -x'_{3i}\mathbf{x}_i^T & x'_{2i}\mathbf{x}_i^T \\ x'_{3i}\mathbf{x}_i^T & \mathbf{0}^T & -x'_{1i}\mathbf{x}_i^T \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix} \quad (291)$$

A solution can be found for, e.g., four point mappings by finding the solution from

$$\mathbf{A}\mathbf{h} = \mathbf{0}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \mathbf{A}_4 \end{bmatrix} \quad (292)$$

6.6 Normalization

It is important to normalize the data sets $\{\mathbf{x}_i\}$ and $\{\mathbf{x}'_i\}$ to have good results from the computation of the homography. According to [30], normalization is not optional. In the case of pixel coordinates, the two first coordinates will be in the range 1–1000, while the third coordinate will be 1. This gives a difference in scale up to 1000. Moreover, quadratic terms are used to a large extent in typical \mathbf{A} matrices used in expressions like $\mathbf{A}\mathbf{h} = \mathbf{0}$, and then the scale difference of the elements may be up to 10^6 . This results in ill-conditioned equations and significant numeric sensitivity of the solutions, which is eliminated with proper scaling.

Note that the points $\mathbf{x}_i = [\mathbf{p}_i^T, 1]^T$ and $\mathbf{x}'_i = [\mathbf{p}'_i^T, 1]^T$ are homogeneous, and normalization is achieved by normalizing each of the data sets $\{\mathbf{p}_i\}$ and $\{\mathbf{p}'_i\}$ so that they have the centroid at the origin, and an average distance to the origin of $\sqrt{2}$. The third coordinate is not part of the normalization, as this coordinate is the scale of the homogeneous vector.

The normalization is done with the transformations $\mathbf{y}_i = \mathbf{N}\mathbf{x}_i$ and $\mathbf{y}'_i = \mathbf{N}'\mathbf{x}'_i$ so that the normalized vectors are given by $\mathbf{y}_i = [\mathbf{q}_i^T, 1]^T$ and $\mathbf{y}'_i = [\mathbf{q}'_i^T, 1]^T$, where each of the data sets $\{\mathbf{q}_i\}$ and $\{\mathbf{q}'_i\}$ has the centroid at the origin, and an average distance to the origin of $\sqrt{2}$. The normalized homography \mathbf{G} is then computed from

$$\mathbf{y}'_i = \mathbf{G}\mathbf{y}_i \quad (293)$$

Insertion of the transformations gives

$$\mathbf{N}'\mathbf{x}'_i = \mathbf{G}\mathbf{N}\mathbf{x}_i \quad (294)$$

This means that the homography \mathbf{H} can be recovered from the normalized homography \mathbf{G} using

$$\mathbf{H} = (\mathbf{N}')^{-1}\mathbf{G}\mathbf{N} \quad (295)$$

6.7 Conditions from point mappings: Method 2

The second solution is found from

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = x'_3 \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (296)$$

where $x' = x'_1/x'_3$ and $y' = x'_2/x'_3$. Insertion of the equation of the last row, which is

$$x'_3 = h_{31}x + h_{32}y + h_{33} \quad (297)$$

into the equations of the first and second rows gives

$$(h_{31}x + h_{32}y + h_{33})x' = h_{11}x + h_{12}y + h_{13} \quad (298)$$

$$(h_{31}x + h_{32}y + h_{33})y' = h_{21}x + h_{22}y + h_{23} \quad (299)$$

$$(300)$$

which leads to

$$xh_{11} + yh_{12} + h_{13} - xx'h_{31} - yx'h_{32} - x'h_{33} = 0 \quad (301)$$

$$xh_{21} + yh_{22} + h_{23} - xy'h_{31} - yy'h_{32} - y'h_{33} = 0 \quad (302)$$

which are 2 linear equations that can be used to determine the elements of \mathbf{H} . Moreover, the two equations give conditions for the Euclidean points (x, y) and (x', y') to satisfy the homography. The rows \mathbf{h}^{jT} of

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}^{1T} \\ \mathbf{h}^{2T} \\ \mathbf{h}^{3T} \end{bmatrix} \quad (303)$$

are stacked in a column vector

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix}.$$

Then the equations for one point mapping $\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i$ can be written

$$\mathbf{A}_i \mathbf{h} = \mathbf{0} \quad (304)$$

where

$$\mathbf{A}_i = \begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i x'_i & -y_i x'_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i y'_i & -y_i y'_i & -y'_i \end{bmatrix}$$

To find \mathbf{H} , which has 8 unknowns, it is necessary to have 4 point mappings $\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i$, $i_1, 2, 3, 4$. Each point mapping gives two equations, and \mathbf{H} can be found from

$$\mathbf{A}\mathbf{h} = \mathbf{0}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \mathbf{A}_4 \end{bmatrix} \in \mathbb{R}^{8 \times 9} \quad (305)$$

If there are $n > 4$ point mappings, the matrix \mathbf{A} will have n submatrices \mathbf{A}_i . The least-squares solution is found in the same way from a singular value decomposition.

Comment 1

It is noted that the conditions (301) and (302) can be written

$$\mathbf{h}^{1T} \mathbf{x} - x' \mathbf{h}^{3T} \mathbf{x} = 0 \quad (306)$$

$$\mathbf{h}^{2T} \mathbf{x} - y' \mathbf{h}^{3T} \mathbf{x} = 0 \quad (307)$$

Comparison with the cross product

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{h}^{1T} \mathbf{x} \\ \mathbf{h}^{2T} \mathbf{x} \\ \mathbf{h}^{3T} \mathbf{x} \end{bmatrix} = \begin{bmatrix} y' \mathbf{h}^{3T} \mathbf{x} - \mathbf{h}^{2T} \mathbf{x} \\ \mathbf{h}^{1T} \mathbf{x} - x' \mathbf{h}^{3T} \mathbf{x} \\ x' \mathbf{h}^{2T} \mathbf{x} - y' \mathbf{h}^{1T} \mathbf{x} \end{bmatrix} \quad (308)$$

shows that the left hand side of the first equation is the second term of the cross product, while the left hand side of the second equation is minus the first term of the cross product. The cross product is obviously the zero vector, since

$$\begin{bmatrix} \mathbf{h}^{1T} \mathbf{x} \\ \mathbf{h}^{2T} \mathbf{x} \\ \mathbf{h}^{3T} \mathbf{x} \end{bmatrix} = x'_3 \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad (309)$$

□

Comment 2

If the homogeneous vectors are not scaled, then

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (310)$$

Then the last equation

$$x'_3 = h_{31}x_1 + h_{32}x_2 + h_{33}x_3 \quad (311)$$

can be used in the first and second equations to get

$$(h_{31}x_1 + h_{32}x_2 + h_{33}x_3)x'_1 = x'_3(h_{11}x_1 + h_{12}x_2 + h_{13}x_3) \quad (312)$$

$$(h_{31}x_1 + h_{32}x_2 + h_{33}x_3)x'_2 = x'_3(h_{21}x_1 + h_{22}x_2 + h_{23}x_3) \quad (313)$$

$$(314)$$

This leads to the alternative formulation

$$x_1x'_3h_{11} + x_2x'_3h_{12} + x_3x'_3h_{13} - x_1x'_1h_{31} - x_2x'_1h_{32} - x_3x'_1h_{33} = 0 \quad (315)$$

$$x_1x'_3h_{21} + x_2x'_3h_{22} + x_3x'_3h_{23} - x_1x'_2h_{31} - x_2x'_2h_{32} - x_3x'_2h_{33} = 0 \quad (316)$$

or $\mathbf{A}\mathbf{h} = \mathbf{0}$ with

$$\mathbf{A} = \begin{bmatrix} x_1x'_3 & x_2x'_3 & x_3x'_3 & 0 & 0 & 0 & -x_1x'_1 & -x_2x'_1 & -x_3x'_1 \\ 0 & 0 & 0 & x_1x'_3 & x_2x'_3 & x_3x'_3 & -x_1x'_2 & -x_2x'_2 & -x_3x'_2 \end{bmatrix} \quad (317)$$

This can be written

$$x'_3\mathbf{h}^{1T}\mathbf{x} - x'_1\mathbf{h}^{3T}\mathbf{x} = 0 \quad (318)$$

$$x'_3\mathbf{h}^{2T}\mathbf{x} - x'_2\mathbf{h}^{3T}\mathbf{x} = 0 \quad (319)$$

□

Example 1

Python script for DLT with 4 points

```

% Calculation of homography in Python
import numpy as np

def Skew(r):
    S = np.array([[ 0, -r[2], r[1]], [r[2], 0, -r[0]], [-r[1], r[0], 0]])
    return S

# Input: Points
x1 = np.array([0,0,1])
x2 = np.array([1,0,1])
x3 = np.array([1,1,1])
x4 = np.array([0,1,1])
H = np.array([[1, 0, 0], [0, 1, 0], [-0.2487, -1, 1.2806]])
H = H/H[2,2]

xp1 = H @ x1
xp2 = H @ x2
xp3 = H @ x3
xp4 = H @ x4

# Scaling of transformed points
xp1 = xp1/xp1[2]
xp2 = xp2/xp2[2]
xp3 = xp3/xp3[2]
xp4 = xp4/xp4[2]

# Method 1:
A1 = np.hstack([x1[0]*Skew(xp1), x1[1]*Skew(xp1), x1[2]*Skew(xp1)])
A2 = np.hstack([x2[0]*Skew(xp2), x2[1]*Skew(xp2), x2[2]*Skew(xp2)])
A3 = np.hstack([x3[0]*Skew(xp3), x3[1]*Skew(xp3), x3[2]*Skew(xp3)])
A4 = np.hstack([x4[0]*Skew(xp4), x4[1]*Skew(xp4), x4[2]*Skew(xp4)])

A = np.vstack((A1, A2, A3, A4))
u,s,v = np.linalg.svd(A)
h = v[8]/v[8,8]
Ha = np.block([[h[0:3]], [h[3:6]], [h[6:9]]]).T # Result: Homography

# Method 2:
zzz = np.zeros(3)
B = np.block([[x1, zzz, -x1*xp1[0]], [zzz, x1, -x1*xp1[1]],
              [x2, zzz, -x2*xp2[0]], [zzz, x2, -x2*xp2[1]],
              [x3, zzz, -x3*xp3[0]], [zzz, x3, -x3*xp3[1]],
              [x4, zzz, -x4*xp4[0]], [zzz, x4, -x4*xp4[1]]])
ub,sb,vb = np.linalg.svd(B)
hb = vb[8]/vb[8,8]
Hb = np.block([[hb[0:3]], [hb[3:6]], [hb[6:9]]]) # Result: Homography

```

```

np.set_printoptions(formatter={'float': '{: 0.4f}'.format})
print('Result Method 1:')
print(Ha)
print('Result Method 2:')
print(Hb)

MATLAB script

% Calculation of homography in MATLAB
skewm =@(u) [0 -u(3) u(2); u(3) 0 -u(1);-u(2) u(1) 0];
% Input: Points
x1 = [0;0;1]; x2 = [1;0;1]; x3 = [1;1;1]; x4 = [0;1;1];
H = [1 0 0; 0 1 0; -0.2487 -1 1.2806]/1.2806 % Input: Unknown homography
xp1 = H*x1; xp2 = H*x2; xp3 = H*x3; xp4 = H*x4;
% Scaling of transformed points
xp1 = xp1/xp1(3); xp2 = xp2/xp2(3); xp3 = xp3/xp3(3); xp4 = xp4/xp4(3);

% Method 1:
A1 = [x1(1)*skewm(xp1) x1(2)*skewm(xp1) x1(3)*skewm(xp1)];
A2 = [x2(1)*skewm(xp2) x2(2)*skewm(xp2) x2(3)*skewm(xp2)];
A3 = [x3(1)*skewm(xp3) x3(2)*skewm(xp3) x3(3)*skewm(xp3)];
A4 = [x4(1)*skewm(xp4) x4(2)*skewm(xp4) x4(3)*skewm(xp4)];
A = [A1; A2; A3; A4];
[u,s,v] = svd(A);
h = v(:,9); h = h/v(9,9);
Ha = [h(1:3) h(4:6) h(7:9)] % Result: Homography

% Method 2:
zzz = [0;0;0];
B1 = [x1', zzz', -x1'*xp1(1); zzz', x1', -x1'*xp1(2)];
B2 = [x2', zzz', -x2'*xp2(1); zzz', x2', -x2'*xp2(2)];
B3 = [x3', zzz', -x3'*xp3(1); zzz', x3', -x3'*xp3(2)];
B4 = [x4', zzz', -x4'*xp4(1); zzz', x4', -x4'*xp4(2)];
B = [B1; B2; B3; B4];
[ub,sb,vb] = svd(B);
hb = vb(:,9); hb = hb/vb(9,9);
Hb = [hb(1:3) hb(4:6) hb(7:9)] % Result: Homography

```

Example 2

Here is a Python script where the code is vectorized by computing the matrix A with vector expressions. Method 2 is used.

```

import numpy as np
from scipy.optimize import least_squares
from numpy import random

def Skew(r):

```

```

S = np.array([[ 0, -r[2], r[1]], [r[2], 0, -r[0]], [-r[1], r[0], 0]])
return S

# Input: Points
x = np.block([[1,1,1], [1,0,1], [0,1,1], [0,0,1],
              [3,2,1], [0.5,0.6,1]]).T
H = np.array([[1500, 0, 640], [0, 1500, 512], [0, 0, 1]])
xp = H @ x
xp = xp/xp[2,:]

nx = x.shape[1]
zzz = np.zeros((nx,3))
A = np.block([[x.T, zzz, -x.T*xp[0].reshape(nx,1)],
              [zzz, x.T, -x.T*xp[1].reshape(nx,1) ]])
ua, sa, vat = np.linalg.svd(A)
ha = vat[8]/vat[8,8]
Ha = np.block([[ha[0:3]], [ha[3:6]], [ha[6:9]]])

np.set_printoptions(formatter={'float': '{: 0.3f}'.format})
print('Ha = \n {}'.format(Ha))

```

6.8 Pose estimation with PnP using 4 points in a plane

In this section the estimation of the displacement between the camera frame and the object frame is discussed for the case when there are 4 point in the object which are in the same plane. The solution is based on identifying 4 points in the object, where the points are in the same plane of the object frame. Then the position of the points can be given by two coordinates in the plane of the object frame. In this case it is possible to define a homography from the point coordinates the object frame to the normalized image coordinates, and the homography can then be determined from the 4 point mappings. This technique is described in textbooks like [30, 67].

A plane π is defined by an object frame o so that the origin of o is in the plane, and the z_o axis is normal to the plane. The pose of frame o relative to the camera frame c is given by

$$\mathbf{T}_o^c = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (320)$$

where $\mathbf{R} = \mathbf{R}_o^c$ and $\mathbf{t} = \mathbf{t}_{co}^c$. Four points that are fixed in the object has been identified in the image. It is assumed that the 4 points $\mathbf{r}_{o,1}^o, \mathbf{r}_{o,2}^o, \mathbf{r}_{o,3}^o, \mathbf{r}_{o,4}^o$ are all in the plane π , and have homogeneous positions $\tilde{\mathbf{r}}_{o,i}^o = (x_i, y_i, 0, 1)^T$, $i = 1, 2, 3, 4$. This is shown in Figure 85.

The normalized image coordinates of the 4 points are i are

$$\lambda_i \tilde{\mathbf{s}}_i = \mathbf{\Pi} \tilde{\mathbf{r}}_{c,i}^c = \mathbf{\Pi} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \tilde{\mathbf{r}}_{o,i}^o$$

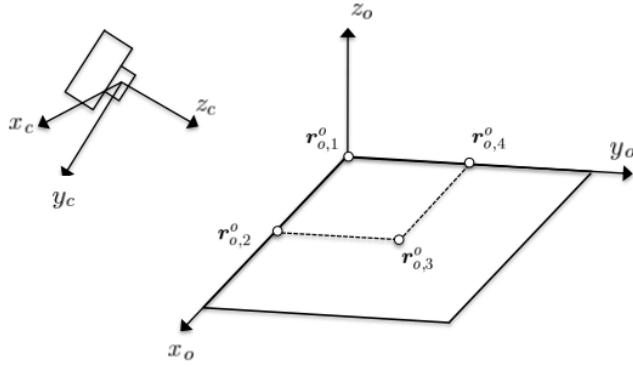


Figure 35: Four points in the xy plane of the object frame.

where λ_i is the depth coordinate and

$$\mathbf{\Pi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

This is written

$$\lambda_i \tilde{\mathbf{s}}_i = [\mathbf{R} \quad \mathbf{t}] \tilde{\mathbf{r}}_{o,i}^o$$

Because the z coordinate of $\tilde{\mathbf{r}}_{o,i}^o$ is zero, this can be reformulated and simplified as the homography

$$\tilde{\mathbf{s}}_i = \mathbf{H} \tilde{\mathbf{x}}_i$$

where

$$\mathbf{H} = [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}]$$

and

$$\tilde{\mathbf{x}}_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

The depth coordinate is set to unity due to the freedom of scaling in the homography.

There are 4 point mappings, which gives 8 conditions on the planar homography \mathbf{H} . This means that \mathbf{H} can be found using standard techniques as presented in previous sections. The with \mathbf{H} known, it is possible to compute the displacement \mathbf{T}_o^c . The columns of $\mathbf{H} = [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \mathbf{h}_3]$ then stacked in the column vector

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix}$$

The conditions are written

$$\mathbf{A}\mathbf{h} = \mathbf{0}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \mathbf{A}_4 \end{bmatrix}$$

and \mathbf{A}_i is found using point mapping i . and find \mathbf{h} by the singular value decomposition. As usual

$$\mathbf{h} = \mathbf{v}_9$$

where the singular value decomposition

$$\mathbf{A} = \sum_{i=1}^9 \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \quad \mathbf{u}_i \in \mathbb{R}^{12}, \quad \mathbf{v}_i \in \mathbb{R}^9$$

is used.

The column vectors of \mathbf{H} are then found from

$$\mathbf{r}_1 = k \mathbf{h}_1 \quad (321)$$

$$\mathbf{r}_2 = k \mathbf{h}_2 \quad (322)$$

$$\mathbf{t} = k \mathbf{h}_3 \quad (323)$$

where the scaling is determined from

$$k = \frac{\text{sgn}(\mathbf{v}_9(9))}{\|\mathbf{h}_1\|}$$

where the sign is selected so that the z component of the translation \mathbf{t} from the camera frame to object frame frame is positiv, which meant that the object frame is in front of the camera.

The final column vector of the rotation matrix is found from

$$\mathbf{r}_3 = \mathbf{r}_1^\times \mathbf{r}_2$$

Python script

```
import numpy as np
def skewm(r):
    return np.array([[0, -r[2], r[1]], [r[2], 0, -r[0]], [-r[1], r[0], 0]])
def expso3(u):
    S = skewm(u)
    return np.eye(3) + np.sinc(np.linalg.norm(u)/np.pi)*S \
        + 0.5*(np.sinc(np.linalg.norm(u)/(2*np.pi)))**2 * S @ S

# Input: Homogeneous points in object frame
x1 = np.array([0, 0, 0, 1])
x2 = np.array([0.1, 0, 0, 1])
x3 = np.array([0.1, 0.1, 0, 1])
x4 = np.array([0, 0.1, 0, 1])

Rco = expso3(np.pi/4*np.array([0,1,0]))
ococ = np.array([0.1, 0, 0.5])
Tco = np.eye(4)
Tco[0:3,0:3] = Rco
```

```

Tco[0:3,3] = ococ[0:3]

# Homogeneous points in camera frame
xp1 = Tco @ x1
xp2 = Tco @ x2
xp3 = Tco @ x3
xp4 = Tco @ x4

# Sensor readings: Homogeneous normalized image coordinates
xp1 = xp1[0:3]/xp1[2]
xp2 = xp2[0:3]/xp2[2]
xp3 = xp3[0:3]/xp3[2]
xp4 = xp4[0:3]/xp4[2]

A1 = np.hstack([x1[0]*skewm(xp1), x1[1]*skewm(xp1), skewm(xp1)]);
A2 = np.hstack([x2[0]*skewm(xp2), x2[1]*skewm(xp2), skewm(xp2)]);
A3 = np.hstack([x3[0]*skewm(xp3), x3[1]*skewm(xp3), skewm(xp3)]);
A4 = np.hstack([x4[0]*skewm(xp4), x4[1]*skewm(xp4), skewm(xp4)]);
A = np.vstack((A1, A2, A3, A4))
u,s,v = np.linalg.svd(A)
h = v[8]
scale = np.sign(h[8])*np.linalg.norm(h[0:3])
r1 = h[0:3]/scale
r2 = h[3:6]/scale
r3 = np.cross(r1,r2)
t = h[6:9]/scale
H = np.identity(4)
H[0:3,0] = r1; H[0:3,1] = r2; H[0:3,2] = r3; H[0:3,3] = t # Result: Homography

np.set_printoptions(formatter={'float': '{: 0.4f}'.format})
print ('Estimated pose:')
print(H)
print ('Actual pose:')
print(Tco)

```

MATLAB script

```

% Input: Homogeneous points in object frame
tro1o =[0 0 0 1]';tro2o =[0.1 0 0 1]';tro3o =[0.1 0.1 0 1]';tro4o=[0 0.1 0 1]';

% Input: Unkown homogeneous transformation matrix from c to o
skewm =@(u) [0 -u(3) u(2); u(3) 0 -u(1);-u(2) u(1) 0];
expso3 =@(u) eye(3)+sinc(norm(u)/pi)*skewm(u) ...
+0.5*(sinc(norm(u)/(2*pi)))^2*(skewm(u))^2;

Rco = expso3(pi/4*[0;1;0]); ococ = [0.1 0 0.5]'; Tco = [Rco ococ; 0 0 0 1]

```

```

% Homogeneous points in camera frame
trc1c = Tco*tro1o; trc2c = Tco*tro2o; trc3c = Tco*tro3o; trc4c = Tco*tro4o;
% Projection matrix
P = [1 0 0 0; 0 1 0 0; 0 0 1 0];
%Sensor readings: Homogeneous normalized image coordinates
ts1 = P*trc1c/trc1c(3); ts2 = P*trc2c/trc2c(3);
ts3 = P*trc3c/trc3c(3); ts4 = P*trc4c/trc4c(3);

%Planar homography using sensor readings and object data
A1 = [tro1o(1)*Skew(ts1) tro1o(2)*Skew(ts1) Skew(ts1)];
A2 = [tro2o(1)*Skew(ts2) tro2o(2)*Skew(ts2) Skew(ts2)];
A3 = [tro3o(1)*Skew(ts3) tro3o(2)*Skew(ts3) Skew(ts3)];
A4 = [tro4o(1)*Skew(ts4) tro4o(2)*Skew(ts4) Skew(ts4)];
A = [A1; A2; A3; A4];
[U,S,V] = svd(A);

%Nullspace solution
h1 = V(1:3,9);
h2 = V(4:6,9);
h3 = V(7:9,9);
% Normalization to unit column vectors and sign change for positive depth
scaleH = sign(h3(3))*norm(h1);
h1 = h1/scaleH; h2 = h2/scaleH; h3 = h3/scaleH;
RR = [h1,h2,cross(h1, h2)];
[Ur,~,Vr] = svd(RR); R = Ur*diag([1,1,det(Ur*Vr')])*Vr';

TcoCalc = [R h3;0 0 0 1] % Result: Calculated Tco

```

□

7 Optimization in the computation of homographies

7.1 Introduction

The accuracy of the DLT solution of a homography $\mathbf{x}' = \mathbf{H}\mathbf{x}$ can be improved with iterative optimization with the DLT solution as the starting point. In this section such iterative optimization schemes are presented. Also the selection of a suitable error function is important, and this will be discussed in detail.

7.2 The algebraic error of a DLT problem

In the case where there are $n > 4$ point mappings, the solution is still found from the the direct linear transformation with

$$\mathbf{A}\mathbf{h} = \mathbf{0}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_n \end{bmatrix} \quad (324)$$

where

$$\mathbf{A}_i = \begin{bmatrix} x_{1i}x'_{3i} & x_{2i}x'_{3i} & x_{3i}x'_{3i} & 0 & 0 & 0 & -x_{1i}x'_{1i} & -x_{2i}x'_{1i} & -x_{3i}x'_{1i} \\ 0 & 0 & 0 & x_{1i}x'_{3i} & x_{2i}x'_{3i} & x_{3i}x'_{3i} & -x_{1i}x'_{2i} & -x_{2i}x'_{2i} & -x_{3i}x'_{2i} \end{bmatrix} \quad (325)$$

In this case the solution $\hat{\mathbf{h}} = \gamma \mathbf{v}_9$ is a least squares solution, which means that there will be an error vector

$$\boldsymbol{\epsilon} = \mathbf{A}\hat{\mathbf{h}} \quad (326)$$

which is called the algebraic error. The magnitude

$$d_{\text{alg}} = |\boldsymbol{\epsilon}| \quad (327)$$

of the algebraic error is called the algebraic distance of the DLT problem. Note that the algebraic error depends on the formulation of the DLT problem, and that there is not necessarily a clear geometric interpretation of this error.

To investigate the algebraic error of the DLT problem further, the homography corresponding to the least squares solution $\hat{\mathbf{h}}$ is denoted $\hat{\mathbf{H}}$, and the resulting homogeneous vector in the second image is written $\hat{\mathbf{x}}'_i = \hat{\mathbf{H}}\mathbf{x}_i$. Then

$$\hat{\mathbf{x}}'_i = \begin{bmatrix} \hat{x}'_{1i} \\ \hat{x}'_{2i} \\ \hat{x}'_{3i} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{h}}^{1T} \\ \hat{\mathbf{h}}^{2T} \\ \hat{\mathbf{h}}^{3T} \end{bmatrix} \mathbf{x}_i = \begin{bmatrix} \hat{\mathbf{h}}^{1T} \mathbf{x}_i \\ \hat{\mathbf{h}}^{2T} \mathbf{x}_i \\ \hat{\mathbf{h}}^{3T} \mathbf{x}_i \end{bmatrix} \quad (328)$$

It is seen from (318) and (319) that the algebraic error for point mapping i can be written

$$\boldsymbol{\epsilon}_i = \mathbf{A}_i \hat{\mathbf{h}} = \begin{bmatrix} x'_{3i} \hat{\mathbf{h}}^{1T} \mathbf{x}_i - x'_{1i} \hat{\mathbf{h}}^{3T} \mathbf{x}_i \\ x'_{3i} \hat{\mathbf{h}}^{2T} \mathbf{x}_i - x'_{2i} \hat{\mathbf{h}}^{3T} \mathbf{x}_i \end{bmatrix} \quad (329)$$

which in combination with (328) gives

$$\boldsymbol{\epsilon}_i = \begin{bmatrix} x'_{3i} \hat{x}'_{1i} - x'_{1i} \hat{x}'_{3i} \\ x'_{3i} \hat{x}'_{2i} - x'_{2i} \hat{x}'_{3i} \end{bmatrix} \quad (330)$$

The distance $d_{\text{alg}}(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)$ of the algebraic error is therefore given by

$$d_{\text{alg}}(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 = \boldsymbol{\epsilon}_i^T \boldsymbol{\epsilon}_i = (x'_{2i} \hat{x}'_{3i} - x'_{3i} \hat{x}'_{2i})^2 + (x'_{3i} \hat{x}'_{1i} - x'_{1i} \hat{x}'_{3i})^2 \quad (331)$$

7.3 Transfer error in the second image

Consider the point mapping $(\mathbf{x}_i, \mathbf{x}'_i)$ which is supposed to satisfy $\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i$, and suppose that a homography matrix $\hat{\mathbf{H}}$ has been computed from n point observations. Then the difference between the observed point \mathbf{x}'_i and the estimated point $\hat{\mathbf{x}}'_i = \hat{\mathbf{H}}\mathbf{x}_i$ is described with the transfer error in the second image. The transfer error vector $\hat{\mathbf{p}}'_i - \mathbf{p}'_i$ is the Euclidean vector from the Euclidean point \mathbf{p}'_i represented by \mathbf{x}'_i and the Euclidean point $\hat{\mathbf{p}}'_i$ which is represented by $\hat{\mathbf{x}}'_i = \hat{\mathbf{H}}\mathbf{x}_i$. This gives

$$\hat{\mathbf{p}}'_i - \mathbf{p}'_i = \frac{1}{\hat{x}'_{3i}} \begin{bmatrix} \hat{x}'_{1i} \\ \hat{x}'_{2i} \end{bmatrix} - \frac{1}{x'_{3i}} \begin{bmatrix} x'_{1i} \\ x'_{2i} \end{bmatrix} = \frac{1}{x'_{3i} \hat{x}'_{3i}} \begin{bmatrix} \hat{x}'_{1i} x'_{3i} - x'_{1i} \hat{x}'_{3i} \\ \hat{x}'_{2i} x'_{3i} - x'_{2i} \hat{x}'_{3i} \end{bmatrix} \quad (332)$$

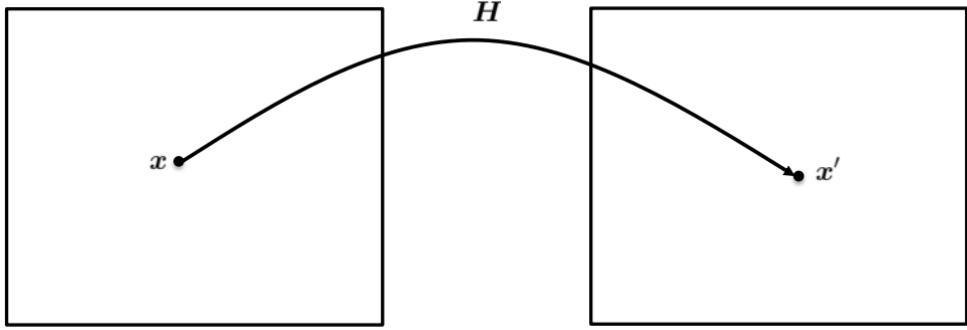


Figure 36: The homography $\mathbf{x}' = \mathbf{H}\mathbf{x}$ as a transformation from the first image to the second image.

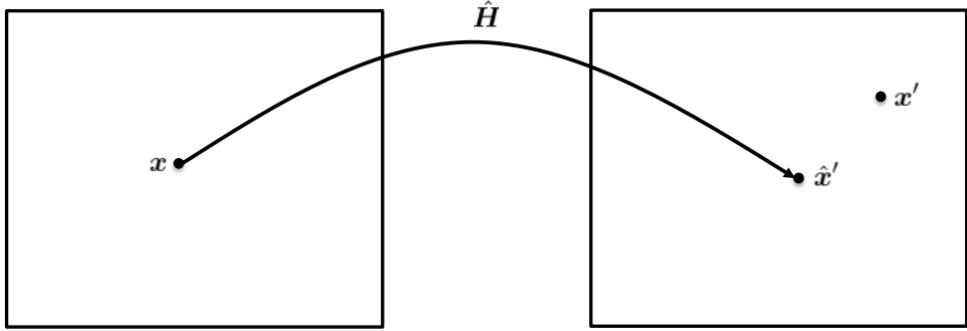


Figure 37: The homography $\hat{\mathbf{x}}' = \hat{\mathbf{H}}\mathbf{x}$ which gives a transfer error in the second image.

It is seen that the transfer error in the second image is related to the algebraic error in (330) by

$$\hat{\mathbf{p}}'_i - \mathbf{p}'_i = \frac{\epsilon_i}{x'_{3i}\hat{x}'_{3i}} \quad (333)$$

The distance $d(\mathbf{x}', \hat{\mathbf{H}}\mathbf{x}_i)$ of the transfer error is defined by

$$d_t(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 = \frac{(\hat{x}'_{1i}x'_{3i} - x'_{1i}\hat{x}'_{3i})^2 + (\hat{x}'_{2i}x'_{3i} - x'_{2i}\hat{x}'_{3i})^2}{(x'_{3i}\hat{x}'_{3i})^2} \quad (334)$$

It is seen that the distance of the transfer error is related to the distance of the algebraic error according to

$$d_t(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 = \frac{d_{\text{alg}}(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2}{(x'_{3i}\hat{x}'_{3i})^2} \quad (335)$$

7.4 Transfer error in the first image

Again, consider the point observation $(\mathbf{x}_i, \mathbf{x}'_i)$ which is supposed to satisfy $\mathbf{x}_i = \mathbf{H}^{-1}\mathbf{x}'_i$, and suppose that a homography matrix $\hat{\mathbf{H}}$ has been computed from n point observations. Then a transfer error can be defined also in the first image by considering the error between \mathbf{x}_i and

the estimated point $\hat{\mathbf{x}}_i = \hat{\mathbf{H}}^{-1}\mathbf{x}'_i$. Note that $\hat{\mathbf{x}}_i$ is the point that would correspond to \mathbf{x}'_i if the homography was given by $\hat{\mathbf{H}}$. The transfer error in the first image is then given by

$$d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 = (\hat{\mathbf{p}}_i - \mathbf{p}_i)^T (\hat{\mathbf{p}}_i - \mathbf{p}_i) \quad (336)$$

where $\hat{\mathbf{p}}_i$ is the Euclidean point represented by $\hat{\mathbf{x}}_i$.

7.5 Minimization of the transfer error in the second image

The transfer error in the second image can be minimized as a nonlinear least-squares problem by iteration where the transfer error is minimized with respect to the elements of the homography $\hat{\mathbf{H}}$. To formulate the minimization problem it is assumed that n point correspondences $(\mathbf{x}_i, \mathbf{x}'_i)$ have been used to compute a DLT solution \mathbf{H}_0 . It is assumed that \mathbf{x}_i is scaled so that $x_{3i} = 1$. Then optimization of the transfer error is used to find the solution $\hat{\mathbf{H}}$ which minimizes the transfer error in the second image.

The transfer error in the second image for point correspondence i is then found by computing

$$\hat{\mathbf{x}}'_i = \hat{\mathbf{H}}\mathbf{x}_i \quad (337)$$

and the corresponding Euclidean point $\hat{\mathbf{p}}'_i$ from

$$\begin{bmatrix} \hat{\mathbf{p}}'_i \\ 1 \end{bmatrix} = \frac{1}{\hat{x}'_{3i}} \hat{\mathbf{x}}'_i \quad (338)$$

where \hat{x}'_{3i} is the third coordinate of $\hat{\mathbf{x}}'_i$. The residual for point correspondence i is then

$$\mathbf{r}_i(\hat{\mathbf{h}}) = \mathbf{p}'_i - \hat{\mathbf{p}}'_i(\hat{\mathbf{h}}) \quad (339)$$

The objective function to be minimized is then

$$f(\hat{\mathbf{h}}) = \sum_{i=1}^n \mathbf{r}_i^T \mathbf{r}_i \quad (340)$$

which can be written

$$f(\hat{\mathbf{h}}) = \mathbf{r}^T \mathbf{r} \quad (341)$$

where $\mathbf{r} = [\mathbf{r}_1^T, \dots, \mathbf{r}_n^T]^T$.

The transfer error in the second image was minimized in Bouguet's calibration toolbox¹. In that case the homography was a mapping from $\mathbf{x} = [x, y, 1]^T$, where (x, y) are the coordinates of a point in a checker-board calibration plane, and $\mathbf{x}' = [u, v, 1]^T$ where (u, v) are the pixel coordinates. In that case the accuracy of the coordinates in the calibration plane will be very accurate, and there will only be an error in \mathbf{x}' . For a point mapping $(\mathbf{x}_i, \mathbf{x}'_i)$ there will be a transfer error due to the difference between \mathbf{x}'_i and $\hat{\mathbf{x}}'_i = \hat{\mathbf{H}}\mathbf{x}_i$. The homography was found by first using a DLT to find the initial solution. Then the solution for the homography was refined by minimizing the transfer error with an iterative solution.

Example

¹http://www.vision.caltech.edu/bouguetj/calib_doc/

```

import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import least_squares

def skewm(r):
    return np.array([[0,-r[2],r[1]], [r[2],0,-r[0]], [-r[1],r[0],0]])

def expso3(u):
    S = skewm(u); un = np.linalg.norm(u)
    return np.identity(3) + np.sinc(un/np.pi)*S \
        + 0.5*(np.sinc(un/(2*np.pi)))**2 * S@S

def residual_transfer_error(hh):
    Hh = hh.reshape(3,3)
    xph = Hh @ x
    ph = xph[0:2]/xph[2,:]
    residual = p - ph
    return residual.flatten()

K = np.array([[1500, 0, 640], [0, 1500, 512], [0, 0, 1]])
ex = np.array([1,0,0]); ez = np.array([0,0,1])
t = np.array([0,0,2]).reshape(3,1)
R = expso3(100/180*np.pi*ex)
T = np.block([[R, t], [0,0,0,1]])
Ha = K @ np.block([R[:,0:2], t])

# Points in object frame
roh = np.array([[0,0,0,1], [1,0,0,1], [1,1,0,1], [0,1,0,1],
                [2,-1,0,1], [1.5,1,0,1], [-1,3,0,1], [-0.2,1,0,1]]).T
nx = roh.shape[1]
x = np.block([[roh[0:2,:]], [np.ones(nx)]])
# Transformation from object plane to pixel coordinates
xp = Ha @ x
xp = xp/xp[2,:]
xp = np.floor(xp+0.5) # Truncates to closest integer pixel position
xp += np.array([[0,0,0], [0,0,0], [1,0,0], [0,2,0],
                [3,-3,0], [-1,1,0], [-1,0,0], [2,0,0]]).T # Noise
p = xp[0:2]

# DLT solution with vectorized A
zzz = np.zeros((nx,3))
A = np.block([[x.T, zzz, -x.T*p[0].reshape(nx,1)],
              [zzz, x.T, -x.T*p[1].reshape(nx,1)]])
u, s, vt = np.linalg.svd(A)
h = vt[8]/vt[8,8]
Hdlt = np.block([[h[0:3]], [h[3:6]], [h[6:9]]])

```

```

# Optimization of transfer error with DLT solution as initial value
H0 = Hdlt.copy()
h0 = H0.flatten()
r0 = residual_transfer_error(h0); d_DLT = np.dot(r0,r0)
res_1 = least_squares(residual_transfer_error, h0)

# Printing
np.set_printoptions(formatter={'float': '{: 0.9f}'.format})
hopt = res_1.x
r_opt = residual_transfer_error(hopt); d_opt = np.dot(r_opt,r_opt)
Hopt = hopt.reshape(3,3); Hopt = Hopt/Hopt[2,2]*Ha[2,2]
print('\n optimality = ', res_1.optimality)
print('\n Hopt = {} \n Hdlt = {} \n Ha = {} \n'
      '\n d_DLT = {} \n d_opt = {}',
      .format(Hopt/2, Hdlt, Ha/2, d_DLT, d_opt))

```

7.6 The reprojection error

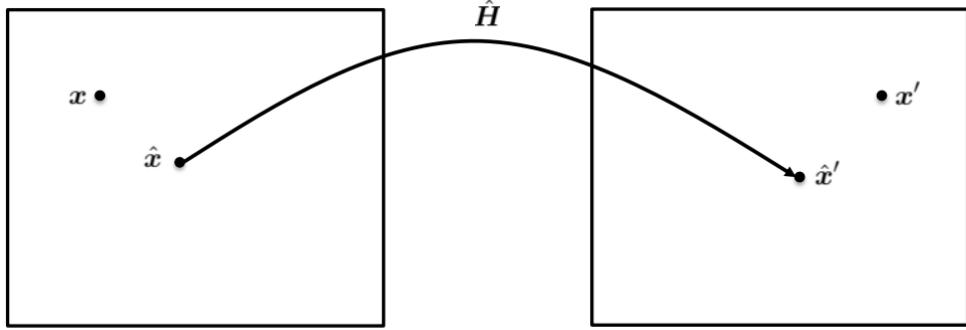


Figure 38: The homography $\hat{\mathbf{H}}\hat{\mathbf{x}}$ which gives an error in the both images that defines the reprojection error.

The idea of the reprojection error is to define a pair of points $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}'_i$ so that the estimated homography $\hat{\mathbf{H}}$ is an exact transformation between the two points, which means that

$$\hat{\mathbf{x}}'_i = \hat{\mathbf{H}}\hat{\mathbf{x}}_i \quad (342)$$

Then there will be a geometric error $d(\mathbf{x}_i, \hat{\mathbf{x}}_i)$ in the first image, and a geometric error $d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)$ in the second image, and the reprojection error d_{ri} for correspondence i is defined as the sum of the geometric error in the first image plus the geometric error in the second image. This gives

$$d_{ri}^2 = d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 \quad (343)$$

$$= (\mathbf{p}_i - \hat{\mathbf{p}}_i)^T (\mathbf{p}_i - \hat{\mathbf{p}}_i) + (\mathbf{p}'_i - \hat{\mathbf{p}}'_i)^T (\mathbf{p}'_i - \hat{\mathbf{p}}'_i) \quad (344)$$

$$= (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (x'_i - \hat{x}'_i)^2 + (y'_i - \hat{y}'_i)^2 \quad (345)$$

Let $\hat{\mathbf{x}}_i = [x_i, y_i, 1]^T$ and $\hat{\mathbf{x}}_i = [\hat{x}_{1i}, \hat{x}_{2i}, \hat{x}_{3i}]^T = \hat{x}_{3i}[\hat{x}_i, \hat{y}_i, 1]^T$. In terms of the coordinates the homography (342) is written

$$\hat{x}'_{1i} = \hat{h}_{11}\hat{x}_i + \hat{h}_{12}\hat{y}_i + \hat{h}_{13} \quad (346)$$

$$\hat{x}'_{2i} = \hat{h}_{21}\hat{x}_i + \hat{h}_{22}\hat{y}_i + \hat{h}_{23} \quad (347)$$

$$\hat{x}'_{3i} = \hat{h}_{31}\hat{x}_i + \hat{h}_{32}\hat{y}_i + \hat{h}_{33} \quad (348)$$

Insertion of $\hat{x}_{1i} = \hat{x}_{3i}\hat{x}'_i$ and $\hat{x}_{2i} = \hat{x}_{3i}\hat{y}'_i$ in the two first equations where the expression of \hat{x}'_{3i} is inserted from the third equation gives

$$(\hat{h}_{31}\hat{x}_i + \hat{h}_{32}\hat{y}_i + \hat{h}_{33})\hat{x}'_i = \hat{h}_{11}\hat{x}_i + \hat{h}_{12}\hat{y}_i + \hat{h}_{13} \quad (349)$$

$$(\hat{h}_{31}\hat{x}_i + \hat{h}_{32}\hat{y}_i + \hat{h}_{33})\hat{y}'_i = \hat{h}_{21}\hat{x}_i + \hat{h}_{22}\hat{y}_i + \hat{h}_{23} \quad (350)$$

$$(351)$$

It is noted that this leads to the following 2 linear equations for the elements of $\hat{\mathbf{H}}$:

$$\hat{x}_i\hat{h}_{11} + \hat{y}_i\hat{h}_{12} + \hat{h}_{13} - \hat{x}_i\hat{x}'_i\hat{h}_{31} - \hat{y}_i\hat{x}'_i\hat{h}_{32} - \hat{x}'_i\hat{h}_{33} = 0 \quad (352)$$

$$\hat{x}_i\hat{h}_{21} + \hat{y}_i\hat{h}_{22} + \hat{h}_{23} - \hat{x}_i\hat{y}'_i\hat{h}_{31} - \hat{y}_i\hat{y}'_i\hat{h}_{32} - \hat{y}'_i\hat{h}_{33} = 0 \quad (353)$$

Moreover, it is seen that \hat{x}'_i and \hat{y}'_i can be expressed in terms of \hat{x}_i , \hat{y}_i and \hat{h}_{kl} as

$$\hat{x}'_i = \frac{\hat{h}_{11}\hat{x}_i + \hat{h}_{12}\hat{y}_i + \hat{h}_{13}}{\hat{h}_{31}\hat{x}_i + \hat{h}_{32}\hat{y}_i + \hat{h}_{33}} \quad (354)$$

$$\hat{y}'_i = \frac{\hat{h}_{21}\hat{x}_i + \hat{h}_{22}\hat{y}_i + \hat{h}_{23}}{\hat{h}_{31}\hat{x}_i + \hat{h}_{32}\hat{y}_i + \hat{h}_{33}} \quad (355)$$

This means that the reprojection error for correspondence i can be expressed in terms of \hat{x}_i , \hat{y}_i and \hat{h}_{kl} as

$$\begin{aligned} d_{ri}^2 &= (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \\ &+ \left(x'_i - \frac{\hat{h}_{11}\hat{x}_i + \hat{h}_{12}\hat{y}_i + \hat{h}_{13}}{\hat{h}_{31}\hat{x}_i + \hat{h}_{32}\hat{y}_i + \hat{h}_{33}} \right)^2 + \left(y'_i - \frac{\hat{h}_{21}\hat{x}_i + \hat{h}_{22}\hat{y}_i + \hat{h}_{23}}{\hat{h}_{31}\hat{x}_i + \hat{h}_{32}\hat{y}_i + \hat{h}_{33}} \right)^2 \end{aligned} \quad (356)$$

7.7 The Sampson error in Euclidean 2D

The Sampson error is useful in the calculation of an approximation of the reprojection error of the DLT problem [30]. The Sampson error was introduced in [60] for efficient calculation of the geometric distance in the plane from a point to a conic curve like an ellipse.

Suppose that $\mathbf{p}_0 = [x_0, y_0]^T$ is a Euclidean point on a curve in the plane, where the curve is implicitly determined by the scalar function $g(\mathbf{p}_0) = 0$. Consider a point $\mathbf{p} = [x, y]^T$ which is close to the curve, and suppose that \mathbf{p}_0 is the closest point on the curve. Let $\mathbf{p} = \mathbf{p}_0 + \delta\mathbf{p}$, where $\delta\mathbf{p}$ is the distance between the points (Figure 39). The first order approximation of the distance $\delta\mathbf{p}$ between the points is found from

$$g(\mathbf{p}) = g(\mathbf{p}_0) + \nabla^T g(\mathbf{p})\delta\mathbf{p} = \nabla^T g(\mathbf{p})\delta\mathbf{p} \quad (357)$$

where it is used that $g(\mathbf{p}_0) = \mathbf{0}$, and $\nabla g = [\partial g / \partial x, \partial g / \partial y]^T$ is the gradient of $g(\mathbf{p})$. It follows that the magnitude of the distance $\delta\mathbf{p}$ must satisfy

$$|\delta\mathbf{p}|^2 = \frac{g(\mathbf{p})^2}{|\nabla g(\mathbf{p})|^2} \quad (358)$$

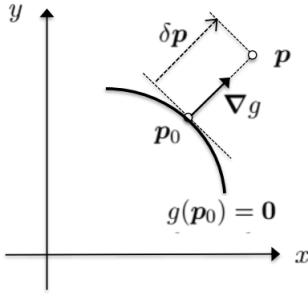


Figure 39: The Sampson error in 2D is found as a first order approximation to the distance from a surface $g(\mathbf{p}) = 0$ to a point \mathbf{p} . The distance is approximated in terms of the gradient ∇g .

7.8 Sampson's error in higher dimension problems

Consider the case where a 4-dimensional Euclidean problem is treated. The coordinates of the space are (x, y, x', y') , which is represented by the vector

$$\mathbf{z} = [x, y, x', y']^T \quad (359)$$

It is seen that the coordinates are the Euclidean coordinates of the homogeneous vectors $\mathbf{x} = [x, y, 1]^T$ and $\mathbf{x}' = [x', y', 1]^T$. Suppose that a 2-dimensional surface is defined by the two equations $g_1(\mathbf{z}_0) = 0$ and $g_2(\mathbf{z}_0) = 0$ where \mathbf{z}_0 is a point on the surface. This means that \mathbf{z}_0 is on the surface whenever

$$\mathbf{g}(\mathbf{z}_0) = \begin{bmatrix} g_1(\mathbf{z}_0) \\ g_2(\mathbf{z}_0) \end{bmatrix} = \mathbf{0} \quad (360)$$

Then, if a point \mathbf{z} is close to the surface, and \mathbf{z}_0 is the point on the surface that is closest to \mathbf{z} , the first order approximation of $\mathbf{g}(\mathbf{z})$ is given by the Taylor series expansion

$$\mathbf{g}(\mathbf{z}) = \mathbf{g}(\mathbf{z}_0) + \mathbf{J}(\mathbf{z})(\mathbf{z} - \mathbf{z}_0) \quad (361)$$

Here $\mathbf{g}(\mathbf{z})$ is assumed to be known, $\mathbf{g}(\mathbf{z}_0) = \mathbf{0}$, $\mathbf{J}(\mathbf{z})$ is the Jacobian defined by

$$\mathbf{J} = \begin{bmatrix} \frac{\partial g_1}{\partial x} & \frac{\partial g_1}{\partial y} & \frac{\partial g_1}{\partial x'} & \frac{\partial g_1}{\partial y'} \\ \frac{\partial g_2}{\partial x} & \frac{\partial g_2}{\partial y} & \frac{\partial g_2}{\partial x'} & \frac{\partial g_2}{\partial y'} \end{bmatrix} \quad (362)$$

and the Euclidean distance

$$\mathbf{r}(\mathbf{z}) = \mathbf{z} - \mathbf{z}_0 \quad (363)$$

from the point \mathbf{z} to the surface is to be determined. The Euclidean distance $\mathbf{r}(\mathbf{z})$ can be found from

$$\mathbf{g}(\mathbf{z}) = \mathbf{J}(\mathbf{z})\mathbf{r}(\mathbf{z}) \quad (364)$$

The least squares solution for the distance $\mathbf{r}(\mathbf{z})$ is given by

$$\mathbf{r}(\mathbf{z}) = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{g}(\mathbf{z}) \quad (365)$$

which is called Sampson's error.

This means that the square of the Euclidean distance between the point \mathbf{z} and \mathbf{z}_0 can be approximated in terms of Sampson's error as

$$d_r^2 = (\mathbf{z} - \mathbf{z}_0)^T (\mathbf{z} - \mathbf{z}_0) \quad (366)$$

$$= \mathbf{g}(\mathbf{z})^T (\mathbf{J} \mathbf{J}^T)^{-1} \mathbf{J} \mathbf{J}^T (\mathbf{J} \mathbf{J}^T)^{-1} \mathbf{g}(\mathbf{z}) \quad (367)$$

$$= \mathbf{g}(\mathbf{z})^T (\mathbf{J} \mathbf{J}^T)^{-1} \mathbf{g}(\mathbf{z}) \quad (368)$$

7.9 Sampson's error for the reprojection error

The reprojection error can be described in a form which makes it possible to use Sampson's error. This is done in 4-dimensional Euclidean space by introducing the vector $\hat{\mathbf{z}} = [\hat{x}, \hat{y}, \hat{x}', \hat{y}']^T$ [30].

The Euclidean points (\hat{x}, \hat{y}) and (\hat{x}', \hat{y}') are consistent with the homography $\hat{\mathbf{H}}$ if $\hat{\mathbf{z}} = [\hat{x}, \hat{y}, \hat{x}', \hat{y}']^T$ is on the surface defined by the two equations

$$g_x(\hat{\mathbf{z}}) = \hat{x}\hat{h}_{11} + \hat{y}\hat{h}_{12} + \hat{h}_{13} - \hat{x}\hat{x}'\hat{h}_{31} - \hat{y}\hat{x}'\hat{h}_{32} - \hat{x}'\hat{h}_{33} = 0 \quad (369)$$

$$g_y(\hat{\mathbf{z}}) = \hat{x}\hat{h}_{21} + \hat{y}\hat{h}_{22} + \hat{h}_{23} - \hat{x}\hat{y}'\hat{h}_{31} - \hat{y}\hat{y}'\hat{h}_{32} - \hat{y}'\hat{h}_{33} = 0 \quad (370)$$

which follows from (301) and (302). This means that we have a surface defined by

$$\mathbf{g}(\hat{\mathbf{z}}) = \begin{bmatrix} g_x(\hat{\mathbf{z}}) \\ g_y(\hat{\mathbf{z}}) \end{bmatrix} = \mathbf{0} \quad (371)$$

The reprojection error, which is found from the distance from the point \mathbf{z} to the closest point $\hat{\mathbf{z}}$ on the surface $\mathbf{g}(\hat{\mathbf{z}}) = \mathbf{0}$ can then be approximated with Sampson's error.

Suppose that the point $\mathbf{z} = [x, y, x', y']^T$ be close to the surface, and suppose that $\hat{\mathbf{z}}$ is the point on the surface that is closest to \mathbf{z} . Then the Sampson error is found from

$$\mathbf{g}(\mathbf{z}) = \mathbf{g}(\hat{\mathbf{z}}) + \mathbf{J}(\mathbf{z})(\mathbf{z} - \hat{\mathbf{z}}) = \mathbf{J}(\mathbf{z})(\mathbf{z} - \hat{\mathbf{z}}) \quad (372)$$

where $\mathbf{g}(\mathbf{z}) = [g_x(\mathbf{z}), g_y(\mathbf{z})]^T$ and $\mathbf{J}(\mathbf{z})$ is the Jacobian. It follows that

$$\mathbf{g}(\mathbf{z}) = \mathbf{J}(\mathbf{z})(\mathbf{z} - \hat{\mathbf{z}}) \quad (373)$$

since $\mathbf{g}(\hat{\mathbf{z}}) = \mathbf{0}$. The least squares solution is then

$$\mathbf{z} - \hat{\mathbf{z}} = \mathbf{J}^T (\mathbf{J} \mathbf{J}^T)^{-1} \mathbf{g}(\mathbf{z}) \quad (374)$$

In a minimization of the reprojection error, which is the geometric distance from \mathbf{z} to $\hat{\mathbf{z}}$, the square of the reprojection error is approximated with Sampson's error as

$$d_{sr}^2 = \mathbf{r}^T \mathbf{r} \quad (375)$$

where

$$\mathbf{r} = \mathbf{J}^T (\mathbf{J} \mathbf{J}^T)^{-1} \mathbf{g}(\mathbf{z}) \quad (376)$$

is the residual found from Sampson's error.

7.10 Jacobian

The function $\mathbf{g}(\mathbf{z})$ as given by (369) and (370) can be written

$$\mathbf{g}(\mathbf{z}) = \begin{bmatrix} g_x(\mathbf{z}) \\ g_y(\mathbf{z}) \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{h}}^{1\text{T}} \mathbf{x} - x' \hat{\mathbf{h}}^{3\text{T}} \mathbf{x} \\ \hat{\mathbf{h}}^{2\text{T}} \mathbf{x} - y' \hat{\mathbf{h}}^{3\text{T}} \mathbf{x} \end{bmatrix} \quad (377)$$

It is seen that the terms are the terms of the algebraic error vector in (290) where \mathbf{x}' is normalized so that $x'_3 = 1$, and with a change of ordering of the components and a sign change. The Jacobian is given by

$$\mathbf{J}(\mathbf{z}) = \frac{\partial \mathbf{g}(\mathbf{z})}{\partial \mathbf{z}} = \begin{bmatrix} \frac{\partial g_x}{\partial x} & \frac{\partial g_x}{\partial y} & \frac{\partial g_x}{\partial x'} & \frac{\partial g_x}{\partial y'} \\ \frac{\partial g_y}{\partial x} & \frac{\partial g_y}{\partial y} & \frac{\partial g_y}{\partial x'} & \frac{\partial g_y}{\partial y'} \end{bmatrix} \quad (378)$$

The Jacobian is then found to be

$$\mathbf{J}(\mathbf{z}) = \begin{bmatrix} \hat{h}_{11} - x' \hat{h}_{31} & \hat{h}_{12} - x' \hat{h}_{32} & -\hat{\mathbf{h}}^{3\text{T}} \mathbf{x} & 0 \\ \hat{h}_{21} - y' \hat{h}_{31} & \hat{h}_{22} - y' \hat{h}_{32} & 0 & -\hat{\mathbf{h}}^{3\text{T}} \mathbf{x} \end{bmatrix} \quad (379)$$

7.11 Optimization of the homography with Sampson's approximation of the reprojection error

In this section a set of N point mappings \mathbf{x}'_i and \mathbf{x}_i are given, and a homography \mathbf{H} has been computed using DLT. The task is to calculate the homography $\hat{\mathbf{H}}$ which gives the minimum reprojection error. A direct minimization of the reprojection error as given by (356) involves the determination of both the elements of the homography matrix $\hat{\mathbf{H}}$ and the point pairs $(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i)$ for $i = 1, \dots, n$.

If instead Sampson's approximation of the error is used, the reprojection error is only minimized with respect to the homography \mathbf{H} , which is a much simpler problem. Sampson's approximation of the reprojection error for each point mapping is then given by

$$d_{sr,i}^2 = \mathbf{r}_i^T \mathbf{r}_i \quad (380)$$

where the residual is

$$\mathbf{r}_i(\mathbf{z}_i) = \mathbf{J}(\mathbf{z}_i)^T (\mathbf{J}(\mathbf{z}_i) \mathbf{J}(\mathbf{z}_i)^T)^{-1} \mathbf{g}(\mathbf{z}_i) \quad (381)$$

It is noted that

$$\mathbf{r}_i(\mathbf{z}_i)^T \mathbf{r}_i(\mathbf{z}_i) = \mathbf{g}(\mathbf{z}_i)^T (\mathbf{J}(\mathbf{z}_i) \mathbf{J}(\mathbf{z}_i)^T)^{-1} \mathbf{g}(\mathbf{z}_i) \quad (382)$$

The homography is then found by minimizing

$$d_{sr}^2 = \sum_{i=1}^N d_{sr,i}^2 = \sum_{i=1}^N \mathbf{r}_i(\mathbf{z}_i)^T \mathbf{r}_i(\mathbf{z}_i) \quad (383)$$

with respect to the elements h_{ij} . The minimization problem is then

$$\min_{\hat{\mathbf{H}}} d_{sr}^2 = \sum_{i=1}^N \mathbf{r}_i(\mathbf{z}_i)^T \mathbf{r}_i(\mathbf{z}_i) \quad (384)$$

where z_i are the given point correspondences.

Example

Script for calculation of a homography by minimization of the reprojection error:

```

import numpy as np
from scipy.optimize import least_squares

def residual_Sampson(h):
    # Calculate residual array
    H = h.reshape(3,3)
    J = Sampson_jacobian(H)
    E = algebraic_error(H)
    N = x.shape[1]    # x is a global variable
    res = np.zeros((4, N))
    for i in range(0, N):  # i iterates over point correspondences
        Ei = np.block([[E[i]], [E[i+N]]])
        Ji = np.block([[J[i]], [J[i+N]]])
        Mi = np.linalg.inv(Ji@Ji.T)
        res[:,i] = (Ji.T @ Mi @ Ei).reshape(4,)
    return res.flatten()

def Sampson_jacobian(H):
    h3 = H[2]; hv3 = h3.reshape(3,1)
    N = x.shape[1]  # x and xp are global variables
    one_v = np.ones((N,1))
    J = np.block([[H[0,0]*one_v - xp[0].reshape(N,1)*H[2,0] ,
                  H[0,1]*one_v - xp[0].reshape(N,1)*H[2,1] ,
                  - x.T@hv3, np.zeros((N,1))],
                  [H[1,0]*one_v - xp[1].reshape(N,1)*H[2,0] ,
                  H[1,1]*one_v - xp[1].reshape(N,1)*H[2,1] ,
                  np.zeros((N,1)), - x.T@hv3]])
    return J

def algebraic_error(H):
    h1 = H[0]; h2 = H[1]; h3 = H[2]
    hv3 = h3.reshape(3,1)
    N = x.shape[1]  # x and xp are global variables
    eps = np.block([[ (x.T@h1).reshape(N,1) - xp[0].reshape(N,1)*x.T@hv3] ,
                   [ (x.T@h2).reshape(N,1) - xp[1].reshape(N,1)*x.T@hv3] ])
    return eps

# True homography
H0 = np.array([1., 0., 1., 0, 2., 1, 1, 1, 2.]).reshape(3,3)
#H0 = np.array([1500., 0., 640., 0, 1500., 512, 0, 0, 1.]).reshape(3,3)
x = np.array([[0.,0.,1.], [1.,0.,1.], [1.,1.,1.], [0.,1.,1.],
              [2.,-1.,1.], [-1.,3.,1.], [4.,1.,1.]]).T

```

```

# Point correspondences
xp = H0 @ x
xp = xp/xp[2,:] # Normalization of xp

# Initial homography
h0_Sampson = np.array([1, 0., 0., 0, 1., 0, 0, 0, 1.])
# Optimization of H
res_1 = least_squares(residual_Sampson, h0_Sampson)

np.set_printoptions(formatter={'float': '{: 0.6f}'.format})
H = res_1.x.reshape(3,3); H = H/H[2,2]*H0[2,2]
print('\n Calculated homography: H = \n', H)
print('\n cost = ', res_1.cost)
print('\n optimality = ', res_1.optimality)
print('\n True homography: H0 = \n', H0)

```

7.12 Optimization of the reprojection error with Sampson's error by iteration

In section the case that is considered is when a homography \mathbf{H} between two images is given, and two matching points \mathbf{x}'_m and \mathbf{x}_m have been found by noisy measurement in the two images [17]. Note that in this case the point \mathbf{x}'_m has been measured, and it has not been computed as $\mathbf{H}\mathbf{x}_m$. The task is then to find the pair of points \mathbf{x}' and \mathbf{x} that gives the minimum geometric error for the given homography. Let $\mathbf{z}_m = [\mathbf{x}_m^T, \mathbf{x}'_m^T]^T$. Then, if \mathbf{z}_0 is the point on the surface $\mathbf{g}(\mathbf{z}_0)$ that is closest to \mathbf{z}_m , then

$$\mathbf{g}(\mathbf{z}_m) = \mathbf{J}(\mathbf{z})(\mathbf{z}_m - \mathbf{z}_0) \quad (385)$$

Then the least-squares solution for \mathbf{z}_0 which minimizes the geometric error for the given homography \mathbf{H} is

$$\mathbf{z}_0^* = \mathbf{z}_m - \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}\mathbf{g}(\mathbf{z}) \quad (386)$$

In computations, an iterative scheme

$$\mathbf{z}(k+1) = \mathbf{z}(k) - \mathbf{J}(\mathbf{z}(k))^T[\mathbf{J}(\mathbf{z}(k))\mathbf{J}(\mathbf{z}(k))^T]^{-1}\mathbf{g}(\mathbf{z}(k)) \quad (387)$$

can be used if a general optimization function is not available. Here $\mathbf{z}(k)$ is the vector at iteration k .

7.13 The gold standard

In the book of Hartley and Zisserman [30] the most accurate method for solving a particular problem is termed the gold standard of the problem. In the case of the computation of a homography from $n > 4$ point pairs $(\mathbf{x}_i, \mathbf{x}'_i)$ the gold standard involves the minimization of the reprojection error with respect to $\hat{\mathbf{H}}$ and $\hat{\mathbf{x}}_i$, $i = 1, \dots, n$. This is an optimization problem in $2n + 9$ variables, as there are 2 variables in $\hat{\mathbf{x}}_i$ and 9 variables in $\hat{\mathbf{H}}$. The initial estimate of $\hat{\mathbf{H}}$ is calculated by a DLT solution of (291) using the singular value decomposition.

Since $\hat{\mathbf{x}}'_i = \hat{\mathbf{H}}\hat{\mathbf{x}}_i$ the minimization problem can be written

$$\min_{\hat{\mathbf{H}}, \hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_n} d_r^2 = \sum_{i=1}^n d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{H}}\hat{\mathbf{x}}_i)^2 \quad (388)$$

where

$$d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 = \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|^2 \quad (389)$$

$$d(\mathbf{x}'_i, \hat{\mathbf{H}}\hat{\mathbf{x}}_i)^2 = \|\mathbf{p}'_i - \hat{\mathbf{p}}'_i\|^2 \quad (390)$$

are squared Euclidean distances. The Euclidean points are given by

$$\mathbf{x}_i = \begin{bmatrix} \mathbf{p}_i \\ 1 \end{bmatrix}, \quad \mathbf{x}'_i = \begin{bmatrix} \mathbf{p}'_i \\ 1 \end{bmatrix} \quad (391)$$

which are given point correspondences, and

$$\hat{\mathbf{x}}_i = \begin{bmatrix} \hat{\mathbf{p}}_i \\ 1 \end{bmatrix}, \quad \hat{\mathbf{H}}\hat{\mathbf{x}}_i = \begin{bmatrix} \hat{\mathbf{p}}'_i \\ 1 \end{bmatrix} \quad (392)$$

which are the point correspondences $(\hat{\mathbf{x}}_i, \hat{\mathbf{H}}\hat{\mathbf{x}}_i)$ to be found by minimization of the reprojection error.

The minimization can be expressed in terms of residuals as

$$\min_{\hat{\mathbf{H}}, \hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_n} d_r^2 = \sum_{i=1}^n \mathbf{r}_i^T \mathbf{r}_i \quad (393)$$

where the residuals are given by

$$\mathbf{r}_i = \begin{bmatrix} \mathbf{p}_i - \hat{\mathbf{p}}_i \\ \mathbf{p}'_i - \hat{\mathbf{p}}'_i \end{bmatrix} \quad (394)$$

This formulation is used in the Python optimization function `scipy.optimize.least_squares`.

Example

This is a Python script where the reprojection error is minimized as in the gold standard. No DLT is run in this example, instead the homography is initialized to a matrix close to the true matrix.

```
import numpy as np
from scipy.optimize import least_squares

# Function which computes the vector of residuals
def fun_reprojection(z):
    # Calculate residual vector
    nz = z.shape[0]; nx = int((nz - 9)/2);
    hhat = z[0:9]; xhat = z[9:nz].reshape(2,nx)
    Hhat = hhat.reshape(3,3)
    xphat_h = Hhat @ np.block([[xhat], [np.ones(nx)]]));
    xphat_h = xphat_h / xphat_h[2,:]; xphat = xphat_h[0:2,:]
```

```

xx = xa - xhat; xxhat = xap - xphat
return np.block([xx.flatten(), xxhat.flatten()])

# True homography
ha = np.array([1500., 0., 640., 0, 1500., 512, 0, 0, 1.])
Ha = ha.reshape(3,3)
xa_h = np.array([[0.,0.,1.], [1.,0.,1.], [1.,1.,1.], [0.,1.,1.],
                 [2,-1, 1], [-1,3,1], [4,1,1]]).T
# Point correspondences
xap_h = Ha @ xa_h
xap_h = xap_h/xap_h[2,:] # Normalization of xp
xa = xa_h[0:2,:]; xap = xap_h[0:2,:] # Euclidean vectors

# Initial values for homography and point
h0 = np.array([1400., 0., 620., 0, 1550., 502, 0, 0, 1.]) # Initial homography
x0 = xa # Initial Euclidean point
z0_reprojection = np.block([h0, x0.T.flatten()]) # Initial value vector.

# Optimization of H
res_1 = least_squares(fun_reprojection, z0_reprojection)

np.set_printoptions(formatter={'float': '{: 0.4f}'.format})
z = res_1.x
H = z[0:9].reshape(3,3); H = H/H[2,2]*Ha[2,2] # Optimized and scaled homography
print('\n Optimized homography H = \n', H)
xh = z[9:z.shape[0]].reshape(xa.shape[0], xa.shape[1])
xh_h = np.block([[xh], [np.ones(xh.shape[1])]])
xph_h = H @ xh_h; xph_h = xph_h/xph_h[2,:]; xph = xph_h[0:2,:]
print('\n Points from optimization: xh =\n', xh)
print('\n Transformed points from optimization: xhp =\n', xph)
print('\n optimality = ', res_1.optimality)
print('\n\n Actual points: x =\n', xa)
print('\n\n Actual transformed points: xap =\n', xap)
print('\n Actual homography: Ha = \n', Ha)

```

7.14 Invariance of the geometric error

Consider a set of n point correspondences $(\mathbf{x}_i, \mathbf{x}'_i)$ where the points \mathbf{x}_i in the first image are given in a coordinate frame a . In the same way the points \mathbf{x}'_i in the second image are given in a coordinate frame a' . The homography of the transformation is termed \mathbf{H}_a and $\mathbf{x}'_i = \mathbf{H}_a \mathbf{x}_i$. Suppose that the same point correspondences are given with respect to a frame b in the first image, and b' in the second image. Let $\mathbf{T} = \mathbf{T}_b^a$ be the homogeneous transformation matrix from frame a to frame a' , and $\mathbf{T}' = \mathbf{T}_{b'}^{a'}$. The point correspondences will then be $(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}')$ where $\tilde{\mathbf{x}}_i = \mathbf{T} \mathbf{x}_i$ and $\tilde{\mathbf{x}}'_i = \mathbf{T}' \mathbf{x}'_i$. The homography will then be $\tilde{\mathbf{H}}$ so that $\tilde{\mathbf{x}}'_i = \tilde{\mathbf{H}} \tilde{\mathbf{x}}_i$. Then

$$\tilde{\mathbf{x}}'_i = \mathbf{T}' \mathbf{H} \mathbf{x}_i = \mathbf{T}' \mathbf{H} \mathbf{T}^{-1} \tilde{\mathbf{x}}_i \quad (395)$$

which means that the homography with respect to the representation in the frames b and b' will be

$$\tilde{\mathbf{H}} = \mathbf{T}' \mathbf{H} \mathbf{T}^{-1} \quad (396)$$

The transformation between the frames involves translation and rotation of the homogeneous vectors, and in addition, a possible scaling. The geometric distance between Euclidean points will not change when their homogeneous representations undergo the same transformation, rotation and scaling. Therefore the geometric distance is unchanged when there is a change of reference frame. This can also be expressed as follows: The geometric error in the second image is

$$d(\tilde{\mathbf{x}}'_i, \tilde{\mathbf{H}} \tilde{\mathbf{x}}_i) = d(\mathbf{T}' \mathbf{x}'_i, \mathbf{T}' \mathbf{H} \mathbf{T}^{-1} \mathbf{T} \mathbf{x}_i) = d(\mathbf{T}' \mathbf{x}'_i, \mathbf{T}' \mathbf{H} \mathbf{x}_i) = d(\mathbf{x}'_i, \mathbf{H} \mathbf{x}_i) \quad (397)$$

In the same way it is found that the geometric in the first image is

$$d(\tilde{\mathbf{x}}_i, \tilde{\mathbf{H}}^{-1} \tilde{\mathbf{x}}'_i) = d(\mathbf{T} \mathbf{x}_i, \mathbf{T} \mathbf{H}^{-1} \mathbf{x}'_i) = d(\mathbf{x}_i, \mathbf{H}^{-1} \mathbf{x}'_i) \quad (398)$$

In both cases it is used that the Euclidean point represented by a homogeneous vector does not change when the homogeneous vector is scaled. It is seen that the geometric error is invariant to a Euclidean transformation of the points \mathbf{x}_i and \mathbf{x}'_i .

7.15 The DLT method is not invariant to Euclidean transformations

The algebraic error vector of the DLT solution is given by the two first elements of the vector $\boldsymbol{\epsilon}_i = (\mathbf{x}'_i)^\times \mathbf{H} \mathbf{x}_i$. If the point correspondences are given in the transformed homogeneous vectors $(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}')$, then the algebraic error will be $\tilde{\boldsymbol{\epsilon}}_i = (\tilde{\mathbf{x}}'_i)^\times \tilde{\mathbf{H}} \tilde{\mathbf{x}}_i$. The resulting algebraic error will be scaled depending on \mathbf{T} and \mathbf{T}' .

8 Points and planes in 3D geometry

8.1 Points in 3D

A point in 3-dimensional Euclidean space can be represented by the coordinate vector $\mathbf{p} = [x, y, z]^T \in \mathbb{R}^3$, where x, y and z are the coordinates of the point. The homogeneous representation of the Euclidean point \mathbf{p} is given by the vector

$$\mathbf{x} = [x_1, x_2, x_3, x_4]^T \in \mathbb{P}^3 \quad (399)$$

where \mathbb{P}^3 is the 3-dimensional projective space. The homogeneous vector \mathbf{x} which represents the Euclidean point

$$\mathbf{p} = \frac{1}{x_4} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathbb{R}^3. \quad (400)$$

The homogeneous vector can therefore be written

$$\mathbf{x} = x_4 \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} \quad (401)$$

This vector represents the same Euclidean point \mathbf{p} for all $x_4 \neq 0$. Therefore, all homogeneous vectors \mathbf{x} with different scaling factor x_4 are considered to be equivalent.

8.2 Normalization of homogeneous vectors in the camera model

The position of a point $\mathbf{r}_{cp}^c = [x, y, z]^T$ in the scene can be represented by the homogeneous point $\tilde{\mathbf{r}}_{cp}^c = [x, y, z, 1]^T$. A homogeneous point

$$\tilde{\mathbf{r}}_{cp}^c = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (402)$$

in the scene gives the normalized image vector

$$\tilde{\mathbf{s}} = \frac{1}{z} \mathbf{\Pi} \tilde{\mathbf{r}}_{cp}^c = \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \quad (403)$$

If the homogeneous point in the scene is given in a scaled format as

$$\tilde{\mathbf{r}} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = x_4 \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (404)$$

the normalized image vector can be calculated from

$$\tilde{\mathbf{s}} = \frac{1}{x_3} \mathbf{\Pi} \tilde{\mathbf{r}} = \frac{1}{x_3} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_1/x_3 \\ x_2/x_3 \\ 1 \\ x_4/x_3 \end{bmatrix} = \begin{bmatrix} x/z \\ y/z \\ 1 \\ x_4/x_3 \end{bmatrix} \quad (405)$$

8.3 Point at infinity

Consider the homogeneous point defined by the limit

$$\lim_{x_4 \rightarrow 0} \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 0 \end{bmatrix} \quad (406)$$

Since the scaling factor is arbitrary, this homogeneous point will correspond to the Euclidean point

$$\mathbf{p} = \lim_{x_4 \rightarrow 0} x_4 \begin{bmatrix} x_1/x_4 \\ x_2/x_4 \\ x_3/x_4 \end{bmatrix} \quad (407)$$

which will tend to infinity in the direction $[x_1, x_2, x_3]^T$ when x_4 tends to zero. This means that a homogeneous point

$$\mathbf{z} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 0 \end{bmatrix} \quad (408)$$

is a point at infinity in the direction $[x_1, x_2, x_3]^T$.

The point $[0, 0, 0, 0]^T$ is undefined, and is excluded from the homogeneous representations.

8.4 Planes in 3D geometry

A plane π is the set of all Euclidean points $\mathbf{p} = [x, y, z]^T \in \mathbb{R}^3$ that satisfy

$$ax + by + cz + d = 0 \quad (409)$$

The geometric interpretation is that the plane has the normal $\mathbf{n} = [a, b, c]^T$, and that the distance from the origin to the plane in the direction of \mathbf{n} is $-d/|\mathbf{n}|$, which is seen from $\mathbf{n}^T \mathbf{p} = -d$. The plane π can be described by the homogeneous coordinates

$$\boldsymbol{\pi} = [a, b, c, d]^T. \quad (410)$$

Then the equation of the plane can be written

$$\boldsymbol{\pi}^T \mathbf{x} = 0 \quad (411)$$

where $\mathbf{x} = [x_1, x_2, x_3, x_4]^T$ is the homogeneous representation of a Euclidean point.

8.5 Distance from a point to a plane

Consider the plane $\pi = [a, b, c, d]^T$ where $\mathbf{n} = [a, b, c]^T$ is the normal vector of the plane. Let $\mathbf{x} = [\mathbf{p}^T, 1]^T$ be the homogeneous vector to the point $\mathbf{p} = [x, y, z]^T$ in the plane π . The distance δ from the origin to the plane is then the length of a vector from the origin to the plane in the direction of the normal vector \mathbf{n} , which gives the distance

$$\delta = \frac{\mathbf{n}^T \mathbf{p}}{|\mathbf{n}|} = -\frac{d}{|\mathbf{n}|} \quad (412)$$

Consider a plane $\pi_1 = [a, b, c, d_1]^T$, which has the same normal vector as the plane π , and therefore is parallel to π . Let \mathbf{p}_1 be a point in the plane π_1 , which means that $\mathbf{n}^T \mathbf{p}_1 = -d_1$. The plane π_1 will be at a distance

$$\delta_1 = \frac{\mathbf{n}^T \mathbf{p}_1}{|\mathbf{n}|} = -\frac{d_1}{|\mathbf{n}|} \quad (413)$$

from the origin.

This means that a point $\mathbf{x}_1 = [\mathbf{p}_1, 1]^T$ on π_1 will be a distance $\delta = (d - d_1)/|\mathbf{n}|$ from π . To find an expression for the distance δ_π , it is observed that

$$\boldsymbol{\pi}^T \mathbf{x}_1 = \mathbf{n}^T \mathbf{p}_1 + d = d - d_1 \quad (414)$$

This leads to the expression

$$\delta_1 = \frac{\boldsymbol{\pi}^T \mathbf{x}_1}{|\mathbf{n}|} \quad (415)$$

for the distance δ_1 from a plane π to a point $\mathbf{x}_1 = [\mathbf{p}_1^T, 1]^T$.

From geometric arguments, the distance can also be found from

$$\delta = \frac{\mathbf{n}^T (\mathbf{p}_1 - \mathbf{p})}{|\mathbf{n}|} \quad (416)$$

where $\mathbf{x} = [\mathbf{p}^T, 1]^T$ is an arbitrary point on the plane. \square

Example: Consider the plane $\boldsymbol{\pi} = [0, 1, 0, -1]^T$ which is the plane $y = 1$ with unit normal vector $\mathbf{n} = [0, 1, 0]^T$ in the y direction. Then the point $\mathbf{x}_1 = [1, 3, 5, 1]^T$ is at a distance

$$\delta = \boldsymbol{\pi}^T \mathbf{x}_1 = [0, 1, 0, -1][1, 3, 5, 1]^T = 3 - 1 = 2$$

from the plane in the direction of \mathbf{n} . The point $\mathbf{p} = [0, 1, 0]^T$ is on the plane. It is verified that

$$\delta = \mathbf{n}^T (\mathbf{p}_1 - \mathbf{p}) = [0, 1, 0]^T [1, 2, 5] = 2$$

which is the distance to \mathbf{x}_1 in the direction of \mathbf{n} . \square

8.6 Plane at infinity

Any point at infinity $\mathbf{x}_\infty = [x_1, x_2, x_3, 0]^T$ will be on the plane

$$\boldsymbol{\pi}_\infty = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

because $\mathbf{x}_\infty^T \boldsymbol{\pi}_\infty = 0$. The plane $\boldsymbol{\pi}_\infty$ is therefore called the plane at infinity.

8.7 Determining a plane from 3 points

Consider the three known points $\mathbf{x}_1 = [\mathbf{p}_1^T, 1]^T$, $\mathbf{x}_2 = [\mathbf{p}_2^T, 1]^T$ and $\mathbf{x}_3 = [\mathbf{p}_3^T, 1]^T$ where $\mathbf{p}_i = [x_i, y_i, z_i]^T$ is the Euclidean coordinate vector corresponding to the homogeneous vector \mathbf{x}_i . The three points are on an unknown plane $\boldsymbol{\pi}$. Then the plane can be found from the points.

A geometric approach is to define the direction vectors $\mathbf{a}_1 = \mathbf{p}_1 - \mathbf{p}_3$ and $\mathbf{a}_2 = \mathbf{p}_2 - \mathbf{p}_3$, which are vectors in the plane. Then

$$\mathbf{n} = \mathbf{a}_1 \times \mathbf{a}_2 = (\mathbf{p}_1 - \mathbf{p}_3) \times (\mathbf{p}_2 - \mathbf{p}_3) \quad (417)$$

is the normal vector of the plane with magnitude $|\mathbf{n}| = |\mathbf{a}_1 \times \mathbf{a}_2|$. The distance δ to the plane from the origin can then be found to satisfy

$$\delta |\mathbf{n}| = \mathbf{p}_3 \cdot \mathbf{n} = \mathbf{p}_3 \cdot [(\mathbf{p}_1 - \mathbf{p}_3) \times (\mathbf{p}_2 - \mathbf{p}_3)] \quad (418)$$

$$= \mathbf{p}_3 \cdot (\mathbf{p}_1 \times \mathbf{p}_2 - \mathbf{p}_1 \times \mathbf{p}_3 - \mathbf{p}_3 \times \mathbf{p}_2 + \mathbf{p}_3 \times \mathbf{p}_3) \quad (419)$$

$$= \mathbf{p}_3 \cdot (\mathbf{p}_1 \times \mathbf{p}_2) \quad (420)$$

The plane is then given by

$$\boldsymbol{\pi} = \begin{bmatrix} (\mathbf{p}_1 - \mathbf{p}_3) \times (\mathbf{p}_2 - \mathbf{p}_3) \\ -\mathbf{p}_3 \cdot (\mathbf{p}_1 \times \mathbf{p}_2) \end{bmatrix} \quad (421)$$

Example

A plane is given in terms of the three points

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (422)$$

The normal vector of the plane is then

$$\mathbf{n} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

The last parameter is

$$d = - \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = -1$$

The resulting plane is

$$\boldsymbol{\pi} = [1, 1, 1, -1]^T$$

It is straightforward to verify that

$$\mathbf{x}_1^T \boldsymbol{\pi} = \mathbf{x}_2^T \boldsymbol{\pi} = \mathbf{x}_3^T \boldsymbol{\pi} = 0$$

which shows that the three points are on the plane. \square

Example

```

x1 = [1;0;0]; x2 = [0;1;0]; x3 = [0;0;1]; % Input: Points on the plane
n = cross((x1-x3),(x2-x3)); % Normal vector
d = -dot(x3,(cross(x1,x2)));
Pi = [n;d] % Result: Plane

dot(Pi,[x1;1]) % Test if x1 is on plane
dot(Pi,[x2;1]) % Test if x2 is on plane
dot(Pi,[x3;1]) % Test if x3 is on plane

```

8.8 Fitting a plane to a set of points

Suppose that n points $\mathbf{x}_i = [x_i, y_i, z_i, 1]^T$ on an unknown plane $\boldsymbol{\pi}$ are given in homogeneous coordinates. The points are on a plane $\boldsymbol{\pi}$ if $\mathbf{x}_i^T \boldsymbol{\pi} = 0$. Define the matrix $\mathbf{A} = [\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_n]^T \in \mathbb{R}^{n \times 4}$. Then the vector $\boldsymbol{\pi}$ of the plane must satisfy

$$\mathbf{A} \boldsymbol{\pi} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} \boldsymbol{\pi} = \mathbf{0} \quad (423)$$

The singular value decomposition of A is

$$A = \sum_{i=1}^4 \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \quad \mathbf{u}_i \in \mathbb{R}^n, \quad \mathbf{v}_i \in \mathbb{R}^4 \quad (424)$$

where $\sigma_1 \geq \sigma_2 \geq \sigma_3 > 0$ and $\sigma_4 = 0$. This means that π must be orthogonal to \mathbf{v}_1 , \mathbf{v}_2 and \mathbf{v}_3 , and that the only nontrivial solutions for π is along \mathbf{v}_4 . Then the equation for the plane is

$$\pi = k \mathbf{v}_4 \quad (425)$$

for some constant k . If there is noise so that the points are approximately on the plane, then this solution is the least squares solution.

9 Points, lines and planes in 3D using Plücker coordinates

9.1 Introduction

We have seen that points and lines can be represented by homogeneous vectors in 2D, while points and planes can be represented by homogeneous vectors in 3D. What is missing in the description that has been presented so far is lines in 3D. To describe lines in 3D we will introduce Plücker coordinates. This is a powerful tool for the description of points, lines and planes in 3D, and lead to a wide range of results related to the intersection of lines, the intersection of planes, the intersection of a line and a plane, and on how to calculate a line from two points on the line, and how to calculate a plane from three points on the plane, or from a line in the plane in combination with a point on the plane. The results in this section rely to a great extent on the [54]. The basic geometric description underlying the Plücker coordinates of a line is based on the unit direction vector of the line and its moment about a reference point, which is treated in [44]. Another important reference on projective geometry is [61] where an alternative presentation is given based on determinants. Computational techniques are discussed in [65].

9.2 Degrees of freedom for a line in 3D

A line in 3D has 4 degrees of freedom, which means that it can be described in terms of 4 parameters. This is explained in [30] by considering a line and its intersection points with two planes, e.g., the xy and the xz plane. Then, the line can be uniquely determined from the two points of intersection whenever the line is not contained in either of these two planes (Figure 40). Each of the intersection points is given by two parameters, and it can be concluded that the line is determined with 4 parameters. While this example can be used to explain that a line has 4 degrees of freedom, it is not an efficient way of describing a line, as the description will be undefined if the line is in one of the planes, so some other description should be used. In the following, a description with Plücker coordinates will be presented. It is useful to first to consider the geometric background for this description.

9.3 Description of a line with a point on the line and the direction vector

A line can be described by the Euclidean positions \mathbf{p} and \mathbf{q} of two points on the line, which is a 6-parameter description (Figure 41). Then the position of any point on the line can be given

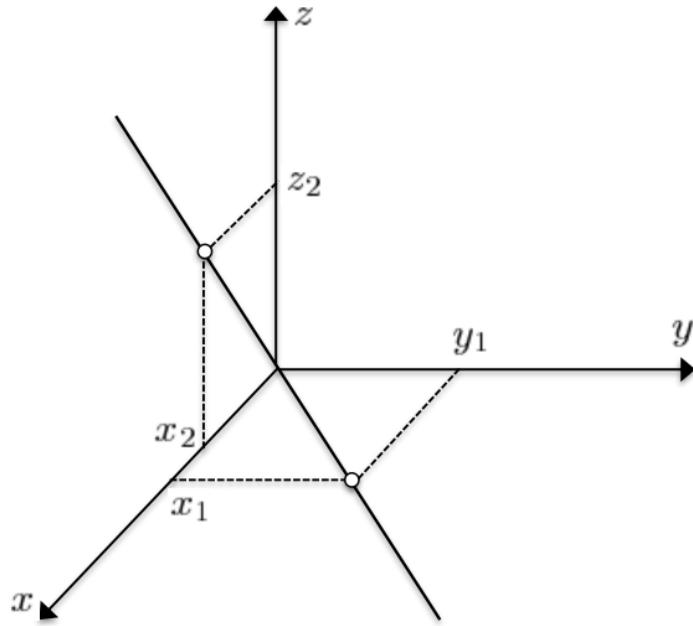


Figure 40: A line defined by the intersection (x_1, y_1) with the xy plane and the intersection (x_2, z_2) with the xz plane, which gives 4 parameters.

as

$$\mathbf{r} = \lambda \mathbf{p} + (1 - \lambda) \mathbf{q} \quad (426)$$

where λ is a real scalar. A direction vector \mathbf{a} of the line can be defined as

$$\mathbf{a} = \mathbf{p} - \mathbf{q} \quad (427)$$

when \mathbf{p} and \mathbf{q} are two points on the line. Then the equation for any point on the line can be written

$$\mathbf{r} = \mathbf{q} + \lambda(\mathbf{p} - \mathbf{q}) = \mathbf{q} + \lambda\mathbf{a} \quad (428)$$

This shows that a line can be described by a point \mathbf{q} on the line and a direction vector \mathbf{a} . This description has 6 parameters, with 3 parameters for \mathbf{p} and 3 parameters for $\lambda\mathbf{a}$.

9.4 Description of a line with a the direction vector and the moment

A more efficient description of a line is the pair of vectors (\mathbf{a}, \mathbf{m}) where \mathbf{a} is the direction vector and \mathbf{m} is the moment of the direction vector about the origin of a reference frame, which is given by

$$\mathbf{m} = \mathbf{q} \times \mathbf{a}$$

where \mathbf{q} is the vector from the origin of a reference frame to an arbitrary point on the line. It follows that the moment vector \mathbf{m} is perpendicular to the direction vector \mathbf{a} , and that $\mathbf{a} \times \mathbf{m} = \mathbf{0}$.

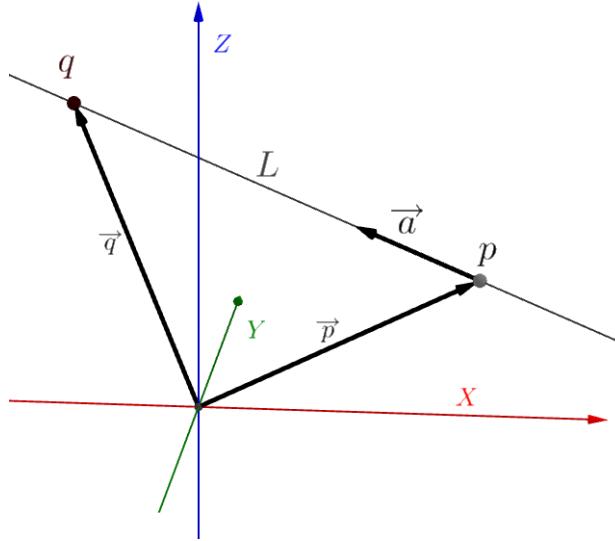


Figure 41: A line L through the points at p and q with direction vector a .

The point on the line which is closest to the origin of the reference frame is given by

$$\mathbf{q}^* = \frac{\mathbf{a} \times \mathbf{m}}{\mathbf{a} \cdot \mathbf{a}} \quad (429)$$

This is verified with the calculation

$$\mathbf{q}^* = \frac{\mathbf{a} \times \mathbf{m}}{\mathbf{a} \cdot \mathbf{a}} = \frac{\mathbf{a} \times (\mathbf{q} \times \mathbf{a})}{\mathbf{a} \cdot \mathbf{a}} = \frac{(\mathbf{a} \cdot \mathbf{a})\mathbf{q} - \mathbf{a} \cdot \mathbf{q}\mathbf{a}}{\mathbf{a} \cdot \mathbf{a}} = \mathbf{q} - \frac{(\mathbf{a} \cdot \mathbf{q})}{\mathbf{a} \cdot \mathbf{a}}\mathbf{a} \quad (430)$$

where \mathbf{q} is an arbitrary point on the line. This shows that \mathbf{q}^* is a point on the line, and from

$$\mathbf{a} \cdot \mathbf{q}^* = \frac{\mathbf{a} \cdot (\mathbf{a} \times \mathbf{m})}{\mathbf{a} \cdot \mathbf{a}} = \mathbf{0} \quad (431)$$

it follows that the vector \mathbf{q}^* is perpendicular to the line. It can therefore be concluded that \mathbf{q}^* is the point on the line that is closest to the origin (Figure 42).

It is noted that two points \mathbf{q} and $\mathbf{r} = \mathbf{q} + \lambda\mathbf{a}$ on the line gives the same moment since

$$\mathbf{r} \times \mathbf{a} = (\mathbf{q} + \lambda\mathbf{a}) \times \mathbf{a} = \mathbf{q} \times \mathbf{a} + \lambda\mathbf{a} \times \mathbf{a} = \mathbf{q} \times \mathbf{a} = \mathbf{m}$$

It is noted that the moment can also be found from the vector cross product of two points \mathbf{q} and $\mathbf{p} = \mathbf{q} + \mathbf{a}$ on the line, which is seen from

$$\mathbf{q} \times \mathbf{p} = \mathbf{q} \times (\mathbf{p} - \mathbf{q} + \mathbf{q}) = \mathbf{q} \times (\mathbf{a} + \mathbf{q}) = \mathbf{m}$$

Moreover, if a point \mathbf{r} satisfies $\mathbf{r} \times \mathbf{a} = \mathbf{m}$, then

$$\mathbf{q}^* = \frac{\mathbf{a} \times \mathbf{m}}{\mathbf{a} \cdot \mathbf{a}} = \frac{\mathbf{a} \times (\mathbf{r} \times \mathbf{a})}{\mathbf{a} \cdot \mathbf{a}} = \frac{(\mathbf{a} \cdot \mathbf{a})\mathbf{r} - (\mathbf{a} \cdot \mathbf{r})\mathbf{a}}{\mathbf{a} \cdot \mathbf{a}} = \mathbf{r} - \mu\mathbf{a}$$

where $\mu = (\mathbf{a} \cdot \mathbf{r})/(\mathbf{a} \cdot \mathbf{a})$. This shows that if $\mathbf{r} \times \mathbf{a} = \mathbf{m}$, then $\mathbf{r} = \mathbf{q}^* + \mu\mathbf{a}$, which is a point on the line.

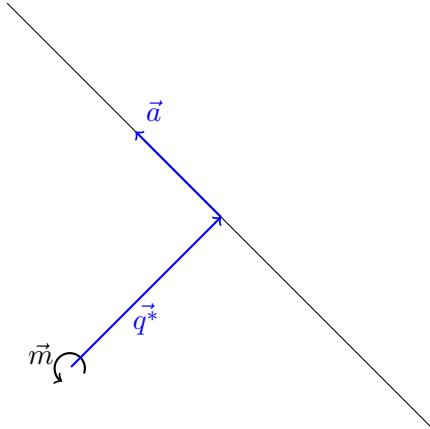


Figure 42: The direction vector \mathbf{a} and moment $\mathbf{m} = \mathbf{q}^* \times \mathbf{a}$ of a line.

9.5 Lines with unit direction vector

A line in Plücker coordinates will often be scaled so that the direction vector is a unit vector, which means that $\|\mathbf{a}\| = 1$. Then the point on the line that is closest to the origin is

$$\mathbf{q}^* = \mathbf{a} \times \mathbf{m} \quad (432)$$

and the moment can be expressed as

$$\mathbf{m} = \mathbf{q}^* \times \mathbf{a} \quad (433)$$

where the three vectors are orthogonal. In addition, if $|\mathbf{a}| = 1$ then $|\mathbf{m}| = |\mathbf{q}^*|$ as shown in Figure 43.

It has then been established that a line can be described in terms of the pair of vectors (\mathbf{a}, \mathbf{m}) where \mathbf{a} is the direction vector, and \mathbf{m} is the moment of \mathbf{a} . This is a description in terms of 6 parameters with the two conditions $|\mathbf{a}| = 1$ and $\mathbf{a} \times \mathbf{m} = \mathbf{0}$, which is in agreement with the fact that the line has 4 degrees of freedom.

9.6 Change of reference point for a line in 3D

Suppose that a line is given by the unit direction vector \mathbf{a} and the moment $\mathbf{m}_{/i} = \mathbf{p}_i \times \mathbf{a}$, where \mathbf{p}_i is the vector from the origin of frame i to a point on the line. The moment of the line with respect to the origin of frame j will be $\mathbf{m}_{/j} = \mathbf{p}_j \times \mathbf{a}$, where \mathbf{p}_j is the vector from the origin of frame j to the same point on the line. This vector is given by $\mathbf{p}_j = \mathbf{p}_{ji} + \mathbf{p}_i$, where \mathbf{p}_{ji} is the vector from the origin of frame j to the origin of frame i . This means that the moment with respect to frame j is $\mathbf{m}_{/j} = \mathbf{p}_j \times \mathbf{a} = (\mathbf{p}_i + \mathbf{p}_{ji}) \times \mathbf{a}$, and it follows that

$$\mathbf{m}_{/j} = \mathbf{m}_{/i} + \mathbf{p}_{ji} \times \mathbf{a} \quad (434)$$

To sum up, the line can be represented with reference to frame i with \mathbf{a} and $\mathbf{m}_{/i}$, or it can be represented with reference to frame j with \mathbf{a} and $\mathbf{m}_{/j} = \mathbf{m}_{/i} + \mathbf{p}_{ji} \times \mathbf{a}$. It is noted that the direction vector is the same in the two cases, while the moment will change according to (434).

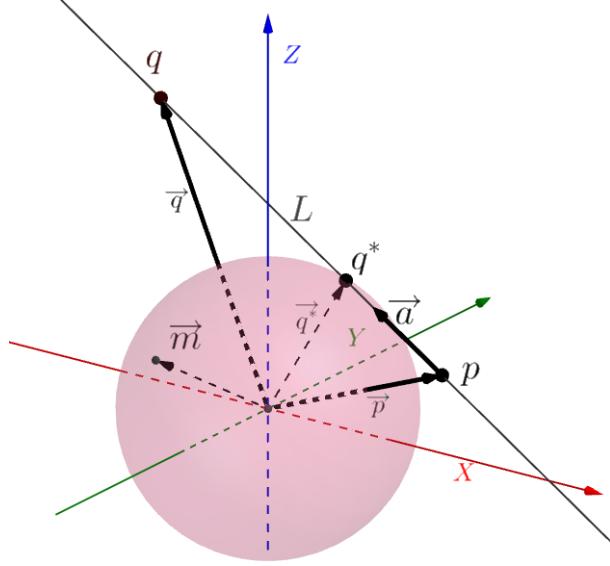


Figure 43: A line L with unit direction vector \mathbf{a} and moment $\mathbf{m} = \mathbf{p} \times \mathbf{a}$. The radius of the sphere is equal to the magnitude of \mathbf{m} . Note that the line is tangent to this sphere at \mathbf{q}^* , which is the closest point to the origin on the line.

9.7 Plücker coordinates for a line

In this section the Plücker coordinates of a line will be presented, and it will be shown that this is equivalent to the description in terms of the direction vector and the moment of the direction vector.

Lines in \mathbb{R}^3 can be given in a homogeneous description in \mathbb{P}^3 by Plücker coordinates. A line through the two homogeneous points

$$\tilde{\mathbf{x}} = [x_1, x_2, x_3, x_4]^T = [\mathbf{x}^T, x_4]^T \quad (435)$$

$$\tilde{\mathbf{y}} = [y_1, y_2, y_3, y_4]^T = [\mathbf{y}^T, y_4]^T \quad (436)$$

is considered. The line is then given in terms of the Plücker coordinates as

$$\mathbf{L} = (l_{41}, l_{42}, l_{43}, l_{23}, l_{31}, l_{12}) \quad (437)$$

where

$$l_{ij} = x_i y_j - x_j y_i \quad (438)$$

are the Plücker coordinates of the line. Since the points $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ in the definition are homogeneous, it follows that the Plücker coordinates are homogeneous, which means that the line is unchanged when the coordinates are scaled by the same factor.

It is noted that the Plücker coordinates are the determinants of the 2×2 submatrices of the matrix

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{bmatrix} \quad (439)$$

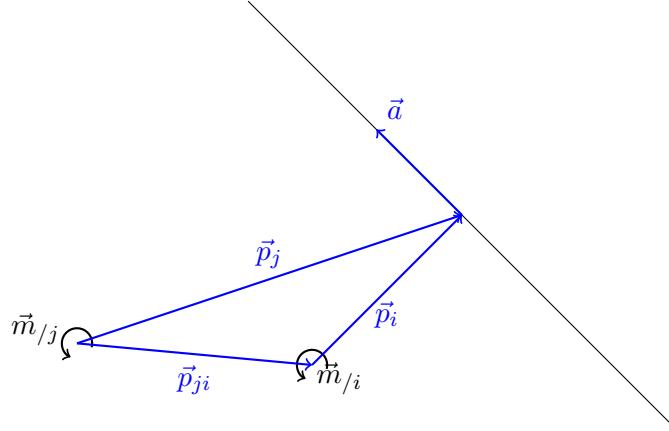


Figure 44: The moments \mathbf{m}_i and \mathbf{m}_j of a line for two different reference points.

It is convenient to define the two homogeneous coordinate vectors

$$\mathbf{l} = [l_{41}, l_{42}, l_{43}]^T \quad (440)$$

$$\mathbf{l}' = [l_{23}, l_{31}, l_{12}]^T \quad (441)$$

and write the line as

$$\mathbf{L} = (\mathbf{l}, \mathbf{l}') \quad (442)$$

From (438) it follows that

$$\mathbf{l} \cdot \mathbf{l}' = 0 \quad (443)$$

This is verified by the direct calculation

$$\begin{aligned} \mathbf{l} \cdot \mathbf{l}' &= l_{41}l_{23} + l_{42}l_{31} + l_{43}l_{12} \\ &= (x_4y_1 - x_1y_4)(x_2y_3 - x_3y_2) + (x_4y_2 - x_2y_4)(x_3y_1 - x_1y_3) \\ &\quad + (x_4y_3 - x_3y_4)(x_1y_2 - x_2y_1) \end{aligned}$$

which sums up to zero.

Useful expressions for the vectors \mathbf{l} and \mathbf{l}' is obtained by noting that

$$\mathbf{l} = \begin{bmatrix} l_{41} \\ l_{42} \\ l_{43} \end{bmatrix} = \begin{bmatrix} x_4y_1 - x_1y_4 \\ x_4y_2 - x_2y_4 \\ x_4y_3 - x_3y_4 \end{bmatrix} = x_4 \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} - y_4 \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = x_4\mathbf{y} - y_4\mathbf{x} \quad (444)$$

and

$$\mathbf{l}' = \begin{bmatrix} l_{23} \\ l_{31} \\ l_{12} \end{bmatrix} = \begin{bmatrix} x_2y_3 - x_3y_2 \\ x_3y_1 - x_1y_3 \\ x_1y_2 - x_2y_1 \end{bmatrix} = \mathbf{x} \times \mathbf{y} \quad (445)$$

This shows that the line can be written in terms of the points (\mathbf{x}, x_4) and (\mathbf{y}, y_4) as

$$(\mathbf{l}, \mathbf{l}') = (x_4\mathbf{y} - y_4\mathbf{x}, \mathbf{x} \times \mathbf{y}) \quad (446)$$

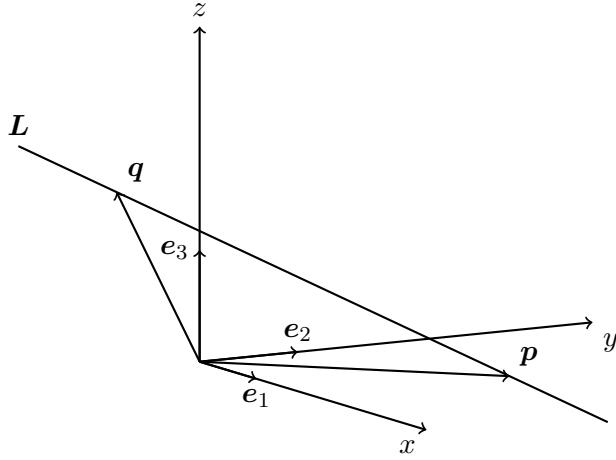


Figure 45: A line \mathbf{L} through the two Euclidean points $\mathbf{p} = \mathbf{x}/x_4$ and $\mathbf{q} = \mathbf{y}/y_4$.

9.8 Geometric interpretation of the Plücker coordinates

Consider a line defined by two points represented by the homogenous vectors $\tilde{\mathbf{x}} = (\mathbf{x}, x_4)$ and $\tilde{\mathbf{y}} = (\mathbf{y}, y_4)$ on the line. These homogeneous points represents the Euclidean points $\mathbf{q} = \mathbf{x}/x_4$ and $\mathbf{p} = \mathbf{y}/y_4$. Then the direction vector of the line is $\mathbf{a} = \mathbf{p} - \mathbf{q}$ while $\mathbf{m} = \mathbf{q} \times \mathbf{a}$ is called the moment of the line. The direction vector of the line is therefore

$$\mathbf{a} = \frac{\mathbf{y}}{y_4} - \frac{\mathbf{x}}{x_4} = \frac{x_4\mathbf{y} - y_4\mathbf{x}}{x_4y_4} \quad (447)$$

while the moment is

$$\mathbf{m} = \left(\frac{\mathbf{x}}{x_4} \right) \times \left(\frac{x_4\mathbf{y} - y_4\mathbf{x}}{x_4y_4} \right) = \frac{\mathbf{x} \times \mathbf{y}}{x_4y_4} \quad (448)$$

From these two equations and (446) it is seen that the Plücker coordinates of the line can be written in terms of the direction vector \mathbf{a} and the moment \mathbf{m} as

$$(\mathbf{l}, \mathbf{l}') = \frac{1}{x_4y_4} (\mathbf{a}, \mathbf{m}) \quad (449)$$

The Plücker coordinates are homogeneous, which means that the scaling can be selected arbitrarily, and therefore this can be scaled to

$$(\mathbf{l}, \mathbf{l}') = (\mathbf{a}, \mathbf{m}) \quad (450)$$

This shows that the Plücker coordinates of a line describes the direction vector and the moment of the line.

The point on the line that is closest to the origin has the position $\mathbf{q} = \mathbf{a} \times \mathbf{m} / \|\mathbf{a}\|^2 = \mathbf{l} \times \mathbf{l}' / \|\mathbf{l}\|^2$.

Example

Consider the points $\tilde{\mathbf{x}}_1 = [0, 0, 0, 1]^T$ and $\tilde{\mathbf{y}}_1 = [1, 0, 0, 1]^T$. The line through the two points is

$$(\mathbf{l}_1, \mathbf{l}'_1) = ([1, 0, 0]^T, [0, 0, 0]^T)$$

This is a line through the origin along the x axis.

Consider the points $\tilde{\mathbf{x}}_2 = [1, 0, 0, 1]^T$ and $\tilde{\mathbf{y}}_2 = [0, 1, 0, 1]^T$. The line through the two points is

$$(\mathbf{l}_2, \mathbf{l}'_2) = ([-1, 1, 0]^T, [0, 0, 1]^T)$$

This is a line with direction vector $\mathbf{l}_2 = [-1, 1, 0]^T$. Then $\mathbf{l}_2^T \mathbf{l}'_2 = [1, 1, 0]^T$, and it follows that the line is through the point $\mathbf{q}_2 = \frac{1}{2}\sqrt{2}[1, 1, 0]^T$. \square

9.9 Invariance of Plücker coordinates to selection of points on the line

The Plücker coordinates of the line do not depend on which points on the line that are used. To verify this the points $\tilde{\mathbf{z}} = \lambda_1 \tilde{\mathbf{x}} + \mu_1 \tilde{\mathbf{y}}$ and $\tilde{\mathbf{w}} = \lambda_2 \tilde{\mathbf{x}} + \mu_2 \tilde{\mathbf{y}}$ are considered. The points $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ represent the Euclidean points $\mathbf{p} = \mathbf{x}/x_4$ and $\mathbf{q} = \mathbf{y}/y_4$. It follows that $\tilde{\mathbf{z}}$ represents the Euclidean point

$$\frac{\lambda_1 \mathbf{x} + \mu_1 \mathbf{y}}{\lambda_1 x_4 + \mu_1 y_4} = \frac{\lambda_1 x_4 \mathbf{p} + \mu_1 y_4 \mathbf{q}}{\lambda_1 x_4 + \mu_1 y_4} = \frac{\lambda_1 x_4}{\lambda_1 x_4 + \mu_1 y_4} \mathbf{p} + \frac{\mu_1 y_4}{\lambda_1 x_4 + \mu_1 y_4} \mathbf{q} \quad (451)$$

which clearly is a linear combination of the two point \mathbf{p} and \mathbf{q} on the line. In the same way it is found that $\tilde{\mathbf{w}}$ represents the Euclidean point

$$\frac{\lambda_2 \mathbf{x} + \mu_2 \mathbf{y}}{\lambda_2 x_4 + \mu_2 y_4} = \frac{\lambda_2 x_4}{\lambda_2 x_4 + \mu_2 y_4} \mathbf{p} + \frac{\mu_2 y_4}{\lambda_2 x_4 + \mu_2 y_4} \mathbf{q} \quad (452)$$

which is also a linear combination of the two Euclidean points \mathbf{p} and \mathbf{q} . This shows that $\tilde{\mathbf{z}}$ and $\tilde{\mathbf{w}}$ represents two Euclidean points on the line. The first Plücker coordinate of a line through $\tilde{\mathbf{z}}$ and $\tilde{\mathbf{w}}$ is

$$(\lambda_1 x_4 + \mu_1 y_4)(\lambda_2 x_1 + \mu_2 y_1) - (\lambda_1 x_1 + \mu_1 y_1)(\lambda_2 x_4 + \mu_2 y_4) \quad (453)$$

$$= (\lambda_1 \mu_2 - \lambda_2 \mu_1)(x_4 y_1 - x_1 y_4) \quad (454)$$

$$= (\lambda_1 \mu_2 - \lambda_2 \mu_1) l_{41} \quad (455)$$

In the same way the remaining Plücker coordinates are found to be $(\lambda_1 \mu_2 - \lambda_2 \mu_1) l_{ij}$, and it is seen that the Plücker coordinates are the same as the coordinates found from $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$, except for the scaling factor $\lambda_1 \mu_2 - \lambda_2 \mu_1$. Since the Plücker coordinates are homogeneous, the line is unchanged under scaling, which means that the line found from the two points $\lambda_1 \tilde{\mathbf{x}} + \mu_1 \tilde{\mathbf{y}}$ and $\lambda_2 \tilde{\mathbf{x}} + \mu_2 \tilde{\mathbf{y}}$ is the same as the line found from $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$.

9.10 Condition for a point to be on a line

Consider the point

$$\tilde{\mathbf{z}} = [z_1, z_2, z_3, z_4]^T \quad (456)$$

which is also written (\mathbf{z}, z_4) , and the line $(\mathbf{l}, \mathbf{l}')$, which is defined by the two points (\mathbf{x}, x_4) and (\mathbf{y}, y_4) on the line. Consider the matrix

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{bmatrix} \quad (457)$$

If the point \tilde{z} is on the line, then the points \tilde{x} , \tilde{y} and \tilde{z} are linearly dependent as $\tilde{z} = \lambda\tilde{x} + (1 - \lambda)\tilde{y}$ for some $\lambda \in \mathbb{R}$, and it follows that the matrix is of rank 2. This implies that all 3×3 submatrices will have rank 2, which again implies that the determinant of any 3×3 sub-matrix of will be zero.

There are 4 sub-matrices, which are obtained by removing one of the columns of the matrix. The condition that the determinants of these 4 sub-matrices are zero, gives the four equations

$$z_1l_{23} + z_2l_{31} + z_3l_{12} = 0 \quad (458)$$

$$-z_4l_{23} + z_2l_{43} - z_3l_{42} = 0 \quad (459)$$

$$-z_4l_{31} + z_3l_{41} - z_1l_{43} = 0 \quad (460)$$

$$-z_4l_{12} + z_1l_{42} - z_2l_{41} = 0 \quad (461)$$

The first equation can be rewritten as $\mathbf{z} \cdot \mathbf{l}' = 0$, while the next three equations gives $-z_4\mathbf{l}' + \mathbf{z} \times \mathbf{l} = 0$. This means that the condition for the point $\tilde{z} = (z, z_4)$ to be on the line $(\mathbf{l}, \mathbf{l}')$ is

$$\mathbf{z} \cdot \mathbf{l}' = 0, \quad -z_4\mathbf{l}' + \mathbf{z} \times \mathbf{l} = 0 \quad (462)$$

9.11 The line defined by a finite point and a point at infinity

Consider the finite point $\tilde{x} = (\mathbf{x}, x_4)$ and the point $\tilde{\mathbf{a}} = (\mathbf{a}, 0)$ at infinity in the direction of \mathbf{a} . The line through these points is given by

$$(\mathbf{l}, \mathbf{l}') = (x_4\mathbf{a}, \mathbf{x} \times \mathbf{a}) \quad (463)$$

This is a line through the finite point \tilde{x} with direction vector \mathbf{a} . It is verified that the point \tilde{x} is on the line since $\mathbf{x} \cdot (\mathbf{x} \times \mathbf{a}) = 0$ and $-x_4\mathbf{x} \times \mathbf{a} + \mathbf{x} \times (x_4\mathbf{a}) = 0$. Moreover, $\tilde{\mathbf{a}}$ is on the line since $\mathbf{a} \cdot (\mathbf{x} \times \mathbf{a}) = 0$ and $\mathbf{a} \times (x_4\mathbf{a}) = 0$.

9.12 A line defined by two points at infinity

Consider the homogeneous point $\tilde{\mathbf{a}} = (\mathbf{a}, 0)$ at infinity in the direction of \mathbf{a} and the homogeneous point $\tilde{\mathbf{b}} = (\mathbf{b}, 0)$ at infinity in the direction of \mathbf{b} . The line through these points is given by

$$(\mathbf{l}, \mathbf{l}') = (\mathbf{0}, \mathbf{a} \times \mathbf{b}) \quad (464)$$

This is a line at infinity with moment $\mathbf{a} \times \mathbf{b}$ which is orthogonal to the direction vectors \mathbf{a} and \mathbf{b} of the points at infinity. This is illustrated in Figure 46.

Note that in 3D there are different lines $(\mathbf{0}, \mathbf{l}') = (\mathbf{0}, \mathbf{a} \times \mathbf{b})$ at infinity which depends on the moment \mathbf{l}' , while in 2D there is only one line $\ell = [0, 0, 1]^T$ at infinity.

9.13 Condition for a point to be at a line at infinity

Consider the line $(\mathbf{0}, \mathbf{l}')$ at infinity and a point $(\mathbf{a}, 0)$ at infinity. Then the general condition (462) for a point to be on a line can be used to find the condition for the point $(\mathbf{a}, 0)$ to be on $(\mathbf{0}, \mathbf{l}')$, which is

$$\mathbf{a} \cdot \mathbf{l}' = 0 \quad (465)$$

which means that a homogeneous point $(\mathbf{a}, 0)$ at infinity in the direction \mathbf{a} will be on the line $(\mathbf{0}, \mathbf{l}')$ at infinity only if the direction \mathbf{a} is orthogonal to the moment \mathbf{l}' of the line at infinity.

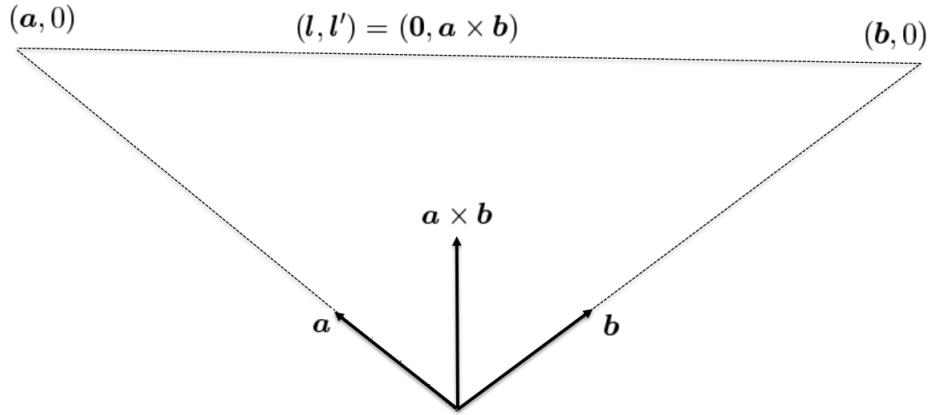


Figure 46: Two vanishing points which defines a line at infinity.

9.14 The Plücker coordinates of a plane

A plane can be defined by three homogeneous points $\tilde{\mathbf{x}} = (\mathbf{x}, x_4)$, $\tilde{\mathbf{y}} = (\mathbf{y}, y_4)$ and $\tilde{\mathbf{z}} = (\mathbf{z}, z_4)$ in the plane. To find an expression for the plane it is used that if $\tilde{\mathbf{w}} = (\mathbf{w}, w_4)$ is a homogeneous point on the plane, when the homogeneous points $\tilde{\mathbf{x}}$, $\tilde{\mathbf{y}}$, $\tilde{\mathbf{z}}$ and $\tilde{\mathbf{w}}$ must be linearly dependent, since the corresponding Euclidean points must be linearly dependent. This implies that the determinant formed by the four homogeneous points is zero, which is written

$$\begin{vmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ w_1 & w_2 & w_3 & w_4 \end{vmatrix} = 0 \quad (466)$$

This determinant can be expanded in terms of its last row as

$$\begin{vmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ w_1 & w_2 & w_3 & w_4 \end{vmatrix} = D_1 w_1 + D_2 w_2 + D_3 w_3 + D_4 w_4 \quad (467)$$

where the coefficients are given by the 3×3 determinants

$$D_1 = \begin{vmatrix} x_2 & x_3 & x_4 \\ y_2 & y_3 & y_4 \\ z_2 & z_3 & z_4 \end{vmatrix} = -[-z_4 l_{23} + (z_2 l_{43} - z_3 l_{42})] \quad (468)$$

$$D_2 = \begin{vmatrix} x_1 & x_3 & x_4 \\ y_1 & y_3 & y_4 \\ z_1 & z_3 & z_4 \end{vmatrix} = -[-z_4 l_{31} + (z_3 l_{41} - z_1 l_{43})] \quad (469)$$

$$D_3 = \begin{vmatrix} x_1 & x_2 & x_4 \\ y_1 & y_2 & y_4 \\ z_1 & z_2 & z_4 \end{vmatrix} = -[-z_4 l_{12} + (z_1 l_{42} - z_2 l_{41})] \quad (470)$$

$$D_4 = \begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{vmatrix} = -[z_1 l_{23} + z_2 l_{31} + z_3 l_{12}] \quad (471)$$

Define the coefficients

$$u_1 = -D_1 = -z_4 l_{23} + (z_2 l_{43} - z_3 l_{42}) \quad (472)$$

$$u_2 = -D_2 = -z_4 l_{31} + (z_3 l_{41} - z_1 l_{43}) \quad (473)$$

$$u_3 = -D_3 = -z_4 l_{12} + (z_1 l_{42} - z_2 l_{41}) \quad (474)$$

$$u_4 = -D_4 = z_1 l_{23} + z_2 l_{31} + z_3 l_{12} \quad (475)$$

where $l_{ij} = x_i y_j - x_j y_i$ are the Plücker coordinates of the line through $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$. This it follows that

$$u_1 w_1 + u_2 w_2 + u_3 w_3 + u_4 w_4 = 0 \quad (476)$$

form any point $\tilde{\mathbf{w}}$ on the plane. This means that the plane is given by

$$\tilde{\mathbf{u}} = [u_1, u_2, u_3, u_4]^T = (\mathbf{u}, u_4) \quad (477)$$

where $\mathbf{u} = [u_1, u_2, u_3]^T$ is the normal vector of the plane. It is seen from (472)–(474) that $\mathbf{u} = -z_4 \mathbf{l}' + \mathbf{z} \times \mathbf{l}$, and from (475) that $u_4 = \mathbf{z} \cdot \mathbf{l}'$. This means that the plane can be written

$$\tilde{\mathbf{u}} = (\mathbf{u}, u_4) = (-z_4 \mathbf{l}' + \mathbf{z} \times \mathbf{l}, \mathbf{z} \cdot \mathbf{l}') \quad (478)$$

This description of a plane in Plücker coordinates is often referred to as a dual representation.

9.15 The Plücker coordinates of a plane from the normal vector and a point in the plane

Consider a plane with Plücker coordinates $\tilde{\mathbf{u}} = (\mathbf{u}, u_4)$. Suppose that the normal vector \mathbf{u} and a point $\tilde{\mathbf{x}} = (\mathbf{x}, 1)$ in the plane is given. Then the Plücker coordinates are given by

$$(\mathbf{u}, u_4) = (\mathbf{u}, -\mathbf{u} \cdot \mathbf{x}) \quad (479)$$

This is shown by considering the dual representation

$$\tilde{\mathbf{u}} = (-z_4 \mathbf{l}' + \mathbf{z} \times \mathbf{l}, \mathbf{z} \cdot \mathbf{l}') \quad (480)$$

of a plane defined by an arbitrary line $\mathbf{L} = (\mathbf{l}, \mathbf{l}')$ in the plane and an arbitrary point $\tilde{\mathbf{z}} = (\mathbf{z}, z_4)$ in the plane. Suppose that the direction vector of the line is \mathbf{a} , and that the line passes through the Euclidean point \mathbf{x} . Then

$$\mathbf{l} = \mathbf{a}, \quad \mathbf{l}' = \mathbf{x} \times \mathbf{a} \quad (481)$$

The homogeneous point $\tilde{\mathbf{z}}$ represents the Euclidean point $\mathbf{r} = \mathbf{z}/z_4$ which is in the plane. Let $\mathbf{b} = \mathbf{r} - \mathbf{x}$ be the vector from \mathbf{x} to \mathbf{r} . It follows that the vectors \mathbf{a} and \mathbf{b} are in the plane. It is noted that

$$\mathbf{z} = z_4 \mathbf{r} = z_4(\mathbf{x} + \mathbf{b}) \quad (482)$$

Then the vector part of $\tilde{\mathbf{u}}$ is

$$\mathbf{u} = -z_4 \mathbf{l}' + \mathbf{z} \times \mathbf{l} = -z_4(\mathbf{x} \times \mathbf{a}) + z_4(\mathbf{x} + \mathbf{b}) \times \mathbf{a} = z_4 \mathbf{b} \times \mathbf{a} \quad (483)$$

which is in agreement to the fact that \mathbf{u} is normal to the plane. The scalar part is

$$u_4 = \mathbf{z} \cdot \mathbf{l}' = z_4(\mathbf{x} + \mathbf{b}) \cdot (\mathbf{x} \times \mathbf{a}) = z_4 \mathbf{b} \cdot (\mathbf{x} \times \mathbf{a}) = -\mathbf{x} \cdot (z_4 \mathbf{b} \times \mathbf{a}) = -\mathbf{x} \cdot \mathbf{u} \quad (484)$$

which verifies (479) since the point \mathbf{x} in the plane is arbitrary.

It is noted that the distance from the origin to the plane in the direction of the normal vector \mathbf{u} is given by

$$\delta = \frac{\mathbf{x} \cdot \mathbf{u}}{\|\mathbf{u}\|} \quad (485)$$

This means that $u_4 = -\delta \|\mathbf{u}\|$. This means that $\tilde{\mathbf{u}} = (\mathbf{u}, u_4)$ is the usual vector representation

$$\tilde{\mathbf{u}} = (\mathbf{u}, -\delta \|\mathbf{u}\|) \quad (486)$$

of a plane with normal vector \mathbf{u} at a distance $\delta = -u_4/\|\mathbf{u}\|$ from the origin.

Example 1

A plane is defined by the 3 points $\tilde{\mathbf{x}} = (\mathbf{x}, x_4) = [1, 0, 0, 1]^T$, $\tilde{\mathbf{y}} = (\mathbf{y}, y_4) = [1, 0, 1, 1]^T$ and $\tilde{\mathbf{z}} = (\mathbf{z}, z_4) = [1, 1, 1, 1]^T$. The line through $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{z}}$ is given by

$$(\mathbf{l}, \mathbf{l}') = (y_4 \mathbf{z} - z_4 \mathbf{y}, \mathbf{y} \times \mathbf{z}) \quad (487)$$

which gives $\mathbf{l} = [0, 1, 0]^T$ and $\mathbf{l}' = [-1, 0, 1]^T$. This is a line with direction vector in the y direction passing through the point $\mathbf{q} = \mathbf{l} \times \mathbf{l}' = [1, 0, 1]^T$. Then the plane is found in dual form to be

$$\tilde{\mathbf{u}} = (-x_4 \mathbf{l}' + \mathbf{x} \times \mathbf{l}, \mathbf{x} \cdot \mathbf{l}') = [1, 0, 0, -1]^T \quad (488)$$

This is a plane with normal vector in the x direction, which is at a distance $\delta = 1$ from the origin in the direction of the normal vector.

```
# Script for computing a plane form 3 points
import numpy as np
```

```
x = np.array([1,0,0]); x4 = 1
y = np.array([1,0,1]); y4 = 1
z = np.array([1,1,1]); z4 = 1
```

```

L = y4*z - z4*y
Lp = np.cross(y,z)
u = -x4*Lp + np.cross(x,L)
u4 = np.dot(x,Lp)
print('l = {}, lp = {}'.format(L,Lp))
print('u = {}, u4 = {}'.format(u,u4))

```

□

Example 2

A plane is defined by the 3 points $\tilde{\mathbf{x}} = [1, 1, 0, 1]^T$, $\tilde{\mathbf{y}} = [1, 0, 1, 1]^T$ and $\tilde{\mathbf{z}} = [1, 0, 0, 1]^T$. Let the line be defined by $(\mathbf{l}, \mathbf{l}') = (y_4 \mathbf{z} - z_4 \mathbf{y}, \mathbf{y} \times \mathbf{z})$, which gives $\mathbf{l} = [0, 0, -1]^T$ and $\mathbf{l}' = [0, 1, 0]^T$. The plane is then given by

$$\tilde{\mathbf{u}} = (-x_4 \mathbf{l}' + \mathbf{x} \times \mathbf{l}, \mathbf{x} \cdot \mathbf{l}') = [-1, 0, 0, 1]^T \quad (489)$$

This is a plane with normal vector $\mathbf{u} = [0, 0, 1]^T$, and distance $\delta = 1$ from the origin in the direction of \mathbf{u} . □

9.16 Calculation of points on a plane

In this section a parametric equation for the points in a plane $\tilde{\mathbf{u}} = (\mathbf{u}, u_4)$. This is done by finding two perpendicular vectors \mathbf{a}_1 and \mathbf{a}_2 in the plane. It is recalled that \mathbf{u} is the normal vector to the plane, and it follows that $(\mathbf{a}_1, \mathbf{a}_2, \mathbf{u})$ will be a set of orthogonal vectors.

Let the Euclidean point on the plane that is closest to the origin is denoted by \mathbf{x} . The vector from the origin to the point \mathbf{x} is then of length $\delta = -u_4/\|\mathbf{u}\|$ along the normal vector \mathbf{u} , which follows from

$$0 = \tilde{\mathbf{x}} \cdot \tilde{\mathbf{u}} = \mathbf{x}^T \mathbf{u} + u_4 = \mathbf{x} \cdot \mathbf{u} - \delta \|\mathbf{u}\| = 0 \quad (490)$$

It follows that the point on the plane that is closest to the origin is at

$$\mathbf{x} = -\frac{u_4}{\|\mathbf{u}\|^2} \mathbf{u} = \delta \mathbf{u} \quad (491)$$

This point has the homogeneous form $\tilde{\mathbf{x}} = (\mathbf{x}, 1)$.

Other points on the plane can be found by finding two perpendicular vectors \mathbf{a}_1 and \mathbf{a}_2 that are in the plane, which is the case whenever $\mathbf{u} \cdot \mathbf{a}_1 = 0$, $\mathbf{u} \cdot \mathbf{a}_2 = 0$ and $\mathbf{a}_1 \cdot \mathbf{a}_2 = 0$. Suppose that \mathbf{u} is a unit vector. An algorithm to find the vectors in the plane can then be based on finding a vector \mathbf{a}_1 that is perpendicular to \mathbf{u} and then calculating $\mathbf{a}_2 = \mathbf{u} \times \mathbf{a}_1$. This will give an orthogonal set of unit vectors $(\mathbf{a}_1, \mathbf{a}_2, \mathbf{u})$. The vector \mathbf{a}_1 is found by first finding a vector \mathbf{v} which is not parallel with \mathbf{u} . This can be done by finding the index $i \in \{1, 2, 3\}$ so that u_i is the coordinate of $\mathbf{u} = [u_1, u_2, u_3]^T$ with the smallest absolute value. Then the vector \mathbf{v} is selected as $\mathbf{v} = \mathbf{e}_i$, where \mathbf{e}_i is a unit vector in direction i . This ensures that the vector \mathbf{v} is not parallel to \mathbf{u} . Then the component of \mathbf{v} which is orthogonal to \mathbf{u} is found as

$$\mathbf{w} = \mathbf{v} - \frac{\mathbf{v}^T \mathbf{u}}{\|\mathbf{u}\|^2} \mathbf{u} \quad (492)$$

The vector \mathbf{w} is orthogonal to \mathbf{u} , and the vectors \mathbf{u} , \mathbf{a}_1 and \mathbf{a}_2 will then be a set of orthogonal unit vectors by setting

$$\mathbf{a}_1 = \mathbf{w}/\|\mathbf{w}\|, \quad \mathbf{a}_2 = \mathbf{u} \times \mathbf{a}_1 \quad (493)$$

Any point \mathbf{z} in the plane will then be given by

$$\mathbf{z} = \mathbf{x} + \alpha_1 \mathbf{a}_1 + \alpha_2 \mathbf{a}_2 \quad (494)$$

9.17 More on the geometric interpretation of the Plücker coordinates of a plane

Consider the plane

$$(\mathbf{u}, u_4) = (-z_4 \mathbf{l}' + \mathbf{z} \times \mathbf{l}, \mathbf{z} \cdot \mathbf{l}') \quad (495)$$

which contains the line

$$(\mathbf{l}, \mathbf{l}') = (x_4 \mathbf{y} - y_4 \mathbf{x}, \mathbf{x} \times \mathbf{y}) \quad (496)$$

defined by the two homogeneous points $\tilde{\mathbf{x}} = (\mathbf{x}, x_4)$, $\tilde{\mathbf{y}} = (\mathbf{y}, y_4)$ according to (446) and the homogeneous point $\tilde{\mathbf{z}} = (\mathbf{z}, z_4)$.

It is assumed that the problem is scaled so that \mathbf{l} is a unit vector, and $z_4 = 1$. It is recalled that $\mathbf{l}' = \mathbf{q} \times \mathbf{l}$ and $\mathbf{q} = \mathbf{l} \times \mathbf{l}'$, where \mathbf{q} is the vector to the point on the line that is closest to the origin.

Consider the calculation

$$\mathbf{l} \times \mathbf{u} = -\mathbf{l} \times \mathbf{l}' + \mathbf{l} \times (\mathbf{z} \times \mathbf{l}) = -\mathbf{q} + \mathbf{w} \quad (497)$$

where

$$\mathbf{w} = \mathbf{l} \times (\mathbf{z} \times \mathbf{l}) = \mathbf{z} - (\mathbf{l} \cdot \mathbf{z})\mathbf{l} \quad (498)$$

is the component of \mathbf{z} which is orthogonal to \mathbf{l} . This means that

$$\mathbf{d} = \mathbf{l} \times \mathbf{u} \quad (499)$$

is the vector from the line $(\mathbf{l}, \mathbf{l}')$ to the point \mathbf{w} , which has the same distance from the line as the point \mathbf{z} . It follows that the distance from the line $(\mathbf{l}, \mathbf{l}')$ to the point \mathbf{z} is

$$\delta = |\mathbf{d}| = |\mathbf{u}| \quad (500)$$

where it is used that \mathbf{l} is a unit vector.

Moreover, it is interesting to note that with this scaling, $|\mathbf{u}|$ is the area of the parallelogram spanned by \mathbf{l} and $\tilde{\mathbf{z}}$ as shown in Figure 48.

Finally, it is noted that this is not valid if $(\mathbf{l}, \mathbf{l}')$ is a line at infinity, because then $\mathbf{l} = \mathbf{0}$.

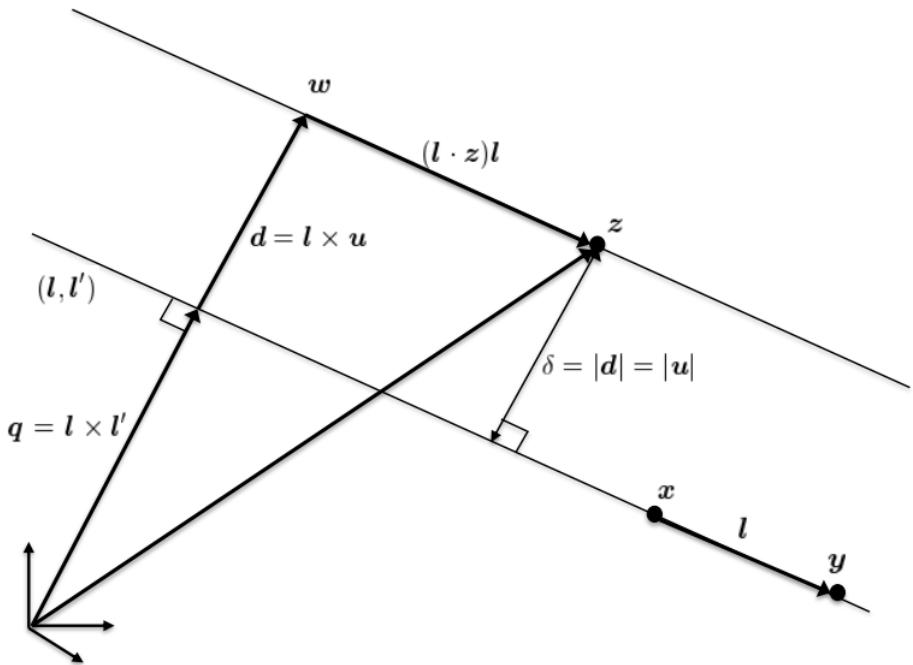


Figure 47: The distance δ from a line $(\mathbf{l}, \mathbf{l}')$ to a point $(\mathbf{z}, 1)$ can be found from the expression for the plane defined by the line and the point. If the line is scaled so that the direction vector the line is a unit vector, and $z_4 = 1$, then the distance δ is found to be the magnitude of $\mathbf{u} = -\mathbf{l}' + \mathbf{z} \times \mathbf{l}$.

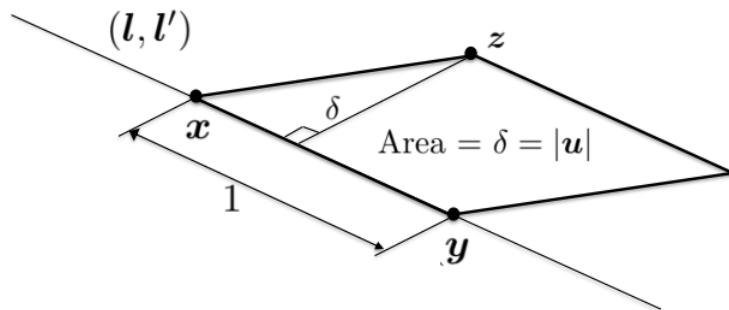


Figure 48: The area of the parallelogram formed by the three Euclidean points \mathbf{x} , \mathbf{y} and \mathbf{z} corresponding to the homogeneous points $(\mathbf{x}, 1)$, $(\mathbf{y}, 1)$ and $(\mathbf{z}, 1)$ us equal to the magnitude of the direction vector \mathbf{u} of the plane defined by the three points.

9.18 A plane defined by a finite point and a line at infinity

The line $(\mathbf{l}, \mathbf{l}') = (\mathbf{0}, \mathbf{a} \times \mathbf{b})$ is a line at infinity which passes through the two points $\tilde{\mathbf{a}} = (\mathbf{a}, 0)$ and $\tilde{\mathbf{b}} = (\mathbf{b}, 0)$. It is assumed that $\mathbf{l}' = \mathbf{a} \times \mathbf{b}$ is a unit vector. The plane $\tilde{\mathbf{u}} = (\mathbf{u}, u_4)$ spanned by this line and the finite point $\tilde{\mathbf{x}} = (\mathbf{x}, x_4)$ is

$$\begin{aligned}\tilde{\mathbf{u}} &= (-x_4 \mathbf{a} \times \mathbf{b}, \mathbf{x} \cdot (\mathbf{a} \times \mathbf{b})) \\ &= -x_4 \left(\mathbf{a} \times \mathbf{b}, -\frac{\mathbf{x}}{x_4} \cdot (\mathbf{a} \times \mathbf{b}) \right)\end{aligned}\quad (501)$$

It is seen that the normal vector of the plane is $\mathbf{l}' = \mathbf{a} \times \mathbf{b}$. This means that the normal vector of the plane is the moment of the line at infinity for all $\tilde{\mathbf{x}}$. If $\tilde{\mathbf{x}}$ is at the origin, then $\mathbf{u} = (\mathbf{l}', 0)$.

9.19 A plane defined by a three points at infinity

Let $\tilde{\mathbf{a}} = (\mathbf{a}, 0)$, $\tilde{\mathbf{b}} = (\mathbf{b}, 0)$ and $\tilde{\mathbf{c}} = (\mathbf{c}, 0)$ be 3 points at infinity. Then

$$(\mathbf{l}, \mathbf{l}') = (\mathbf{0}, \mathbf{a} \times \mathbf{b}) \quad (502)$$

is the line at infinity which passes through the two points $\tilde{\mathbf{a}} = (\mathbf{a}, 0)$ and $\tilde{\mathbf{b}} = (\mathbf{b}, 0)$. The plane spanned by this line and the point $\tilde{\mathbf{c}} = (\mathbf{c}, 0)$ is

$$\tilde{\mathbf{u}}_\infty = (\mathbf{u}, u_4) = (c_4 \mathbf{l}', \mathbf{c} \cdot (\mathbf{a} \times \mathbf{b})) = \mathbf{c} \cdot (\mathbf{a} \times \mathbf{b}) (\mathbf{0}, 1) \quad (503)$$

Due to the homogeneous nature of the vector this can be scaled to

$$\tilde{\mathbf{u}}_\infty = (\mathbf{0}, 1) \quad (504)$$

which is known as the plane at infinity.

9.20 A line determined by the intersection of two planes

It is interesting to note that a plane $\tilde{\mathbf{u}} = (\mathbf{u}, u_4)$ is represented by a homogeneous vector of the same form as the homogeneous representation $\tilde{\mathbf{x}} = (\mathbf{x}, x_4)$ of a point. In the terminology of projective geometry a plane is said to be the dual of a point.

Consider the line

$$\mathbf{L} = (\mathbf{l}, \mathbf{l}') \quad (505)$$

Then the dual of the line is defined as

$$\mathbf{L}^* = (\mathbf{l}^*, \mathbf{l}'^*) = (\mathbf{l}', \mathbf{l}) \quad (506)$$

which is also referred to as a dual line. It is noted that the dual of a line is obtained by interchanging the order of the two vectors \mathbf{l} and \mathbf{l}' of the Plücker coordinates.

The Plücker coordinates of a line can be expressed in terms of the homogeneous coordinates of two points (\mathbf{x}, x_4) and (\mathbf{y}, y_4) on the line as $(\mathbf{l}, \mathbf{l}') = (x_4 \mathbf{y} - y_4 \mathbf{x}, \mathbf{x} \times \mathbf{y})$. It turns out that the Plücker coordinates of a dual line can be determined in the same way by in terms of the homogeneous coordinates of two planes (\mathbf{u}, u_4) and (\mathbf{v}, v_4) that intersect at the line. The expression for the resulting dual line is

$$\mathbf{L}^* = (u_4 \mathbf{v} - v_4 \mathbf{u}, \mathbf{u} \times \mathbf{v}) \quad (507)$$

Note that the expression has the same form as the for a line determined by two points. This means that the line at the intersection of the two planes is given by

$$\mathbf{L} = (\mathbf{u} \times \mathbf{v}, u_4 \mathbf{v} - v_4 \mathbf{u}) \quad (508)$$

A derivation of this result is presented in the next section.

9.21 Derivation of the expression for a line as the intersection of two planes

This derivation of (508) is based on [61]. It is recalled that the Plücker coordinates of a line through the points $\tilde{\mathbf{x}} = (\mathbf{x}, x_4)$ and $\tilde{\mathbf{y}} = (\mathbf{y}, y_4)$ are given by $l_{ij} = x_i y_j - x_j y_i$. In analogy with this, the dual Plücker coordinates defined by the two planes $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ are given by

$$p_{ij} = u_i v_j - u_j v_i \quad (509)$$

The dual Plücker coordinates are then written in the form $(\mathbf{p}, \mathbf{p}')$ where

$$\mathbf{p} = (p_{41}, p_{42}, p_{43}) \quad (510)$$

$$\mathbf{p}' = (p_{23}, p_{31}, p_{12}) \quad (511)$$

In the following it will be shown that these dual Plücker coordinates represents the dual line at the intersection of the two planes $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$. The first step is to find the relation between the dual Plücker coordinates $(\mathbf{p}, \mathbf{p}')$ and the Plücker coordinates $(\mathbf{l}, \mathbf{l}')$ of the line of intersection by considering two points (\mathbf{x}, x_4) and (\mathbf{y}, y_4) on the line $(\mathbf{l}, \mathbf{l}')$. These two points must be on both planes. This means that the point (\mathbf{x}, x_4) satisfies the equations

$$u_1 x_1 + u_2 x_2 + u_3 x_3 + u_4 x_4 = 0 \quad (512)$$

$$v_1 x_1 + v_2 x_2 + v_3 x_3 + v_4 x_4 = 0 \quad (513)$$

while (\mathbf{y}, y_4) satisfies

$$u_1 y_1 + u_2 y_2 + u_3 y_3 + u_4 y_4 = 0 \quad (514)$$

$$v_1 y_1 + v_2 y_2 + v_3 y_3 + v_4 y_4 = 0 \quad (515)$$

Elimination of x_4 from (512,513) and elimination of y_4 from (514,515) gives

$$p_{41} x_1 + p_{42} x_2 + p_{43} x_3 = 0 \quad (516)$$

$$p_{41} y_1 + p_{42} y_2 + p_{43} y_3 = 0 \quad (517)$$

The following results are then found by elimination of p_{41} , and elimination of p_{42}

$$p_{42} l_{12} = p_{43} l_{31} \quad (518)$$

$$p_{41} l_{12} = p_{43} l_{23} \quad (519)$$

From this it is concluded that

$$\frac{p_{41}}{l_{23}} = \frac{p_{42}}{l_{31}} = \frac{p_{43}}{l_{12}} \quad (520)$$

A similar procedure starting with the elimination of u_4 from (512,514) and elimination of v_4 from (513,515) will, due to the symmetry of the equations, lead to

$$l_{42}p_{12} = l_{43}p_{31} \quad (521)$$

$$l_{41}p_{12} = l_{43}p_{23} \quad (522)$$

which gives

$$\frac{p_{23}}{l_{41}} = \frac{p_{31}}{l_{42}} = \frac{p_{12}}{l_{43}} \quad (523)$$

Moreover, the elimination of u_1 from (512,514) and elimination of v_1 from (513,515) leads to

$$l_{12}p_{42} = l_{31}p_{43} \quad (524)$$

$$l_{41}p_{42} = l_{31}p_{23} \quad (525)$$

and it follows that a scaling factor $\gamma \neq 0$ can be defined so that

$$\gamma = \frac{p_{41}}{l_{23}} = \frac{p_{42}}{l_{31}} = \frac{p_{43}}{l_{12}} = \frac{p_{23}}{l_{41}} = \frac{p_{31}}{l_{42}} = \frac{p_{12}}{l_{43}} \quad (526)$$

This means that $p_{41} = \gamma l_{23}$, $p_{42} = \gamma l_{31}$, ... and it follows that the dual Plücker coordinates $(\mathbf{p}, \mathbf{p}')$ of the line are related to the Plücker coordinates $(\mathbf{l}, \mathbf{l}')$ by

$$(l_{41}, l_{42}, l_{43}, l_{23}, l_{31}, l_{12}) = (p_{23}, p_{31}, p_{12}, p_{41}, p_{42}, p_{43}) \quad (527)$$

It is seen that

$$\mathbf{l} = \mathbf{p}', \quad \mathbf{l}' = \mathbf{p} \quad (528)$$

This means that the dual Plücker coordinates $(\mathbf{p}, \mathbf{p}')$ of the line of intersection of two planes correspond to the Plücker coordinates $(\mathbf{l}, \mathbf{l}')$ of the line where the order of the vectors as swapped according to (528).

9.22 The line at the intersection of a finite plane and the plane at infinity

Consider the plane $\tilde{\mathbf{u}} = (\mathbf{u}, u_4)$ where \mathbf{u} is a unit vector. According to the general expression (508) for the line at the intersection between two lines, the intersection of $\tilde{\mathbf{u}} = (\mathbf{u}, u_4)$ with the plane $\mathbf{u}_\infty = (\mathbf{0}, 1)$ at infinity is at the line

$$\mathbf{L} = (\mathbf{0}, \mathbf{u}) \quad (529)$$

where the scaling by -1 has been left out since the line is homogeneous. This is a line at infinity where the moment is equal to the normal vector of the finite plane $\tilde{\mathbf{u}}$. It is seen that the intersection between the plane $\tilde{\mathbf{u}}$ and the plane at infinity is independent of the parameter u_4 , which determines the distance from the origin to the plane. This means that the intersection between the planes $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{u}}_\infty$ is the same as the intersection between the plane $(\mathbf{u}, 0)$ and $\tilde{\mathbf{u}}_\infty$, where $(\mathbf{u}, 0)$ is a plane through the origin which is parallel to $\tilde{\mathbf{u}}$.

9.23 A point as the intersection of a line and a plane

The dual of the result (478) where a plane is found from a line and a point in the plane gives the intersection point of a dual line $(\mathbf{l}^*, \mathbf{l}'')$ and a plane (\mathbf{u}, u_4) as

$$(\mathbf{x}, x_4) = (-u_4 \mathbf{l}'' + \mathbf{u} \times \mathbf{l}^*, \mathbf{u} \cdot \mathbf{l}'') \quad (530)$$

Insertion of $\mathbf{l} = \mathbf{l}''$ and $\mathbf{l}' = \mathbf{l}^*$ gives the point of intersection as

$$(\mathbf{x}, x_4) = (-u_4 \mathbf{l} + \mathbf{u} \times \mathbf{l}', \mathbf{u} \cdot \mathbf{l}) \quad (531)$$

9.24 A point as the intersection of a line and the plane at infinity

The intersection point of a line $(\mathbf{l}, \mathbf{l}')$ and a plane $(\mathbf{0}, 1)$ at infinity is

$$(\mathbf{x}, x_4) = (\mathbf{l}, 0) \quad (532)$$

This is a point at infinity in the direction of \mathbf{l} , which is the direction vector of the line. It is noted that all parallel lines intersect the plane at infinity in the same point at infinity.

9.25 Condition for a line to be in a plane

The dual result of the condition for a point to be on a line is the condition for a dual line $(\mathbf{l}^*, \mathbf{l}'^*)$ to be in a plane (\mathbf{u}, u_4) , which is

$$\mathbf{u} \cdot \mathbf{l}^* = 0, \quad -u_4 \mathbf{l}'^* + \mathbf{u} \times \mathbf{l}^* = 0 \quad (533)$$

The corresponding condition for the line $(\mathbf{l}, \mathbf{l}')$ to be in the plane (\mathbf{u}, u_4) is therefore

$$\mathbf{u} \cdot \mathbf{l} = 0, \quad -u_4 \mathbf{l} + \mathbf{u} \times \mathbf{l}' = 0 \quad (534)$$

The geometric interpretation of the first condition $\mathbf{u} \cdot \mathbf{l} = 0$ is that the line is normal to the plane normal, and therefore parallel with the plane.

If the line is at infinity, then the line is given by $(\mathbf{0}, \mathbf{l}')$. Then condition for the line to be in the plane is $\mathbf{u} \times \mathbf{l}' = 0$. This is in agreement for the expression for line at the intersection of the plane (\mathbf{u}, u_4) and the plane $(\mathbf{0}, 1)$ at infinity, which is the line $(\mathbf{0}, \mathbf{u})$, which is a line at infinity.

Example

```
% Script for computing the intersection between a line and a plane
L = [0;0;1]; Lp = cross([1;1;0],L);
u = [1;1;1]; u4 = sqrt(3);

x = -u4*L + cross(u,Lp); x4 = dot(u,L);
if x4==0 & x'*x == 0
    fprintf('\nThe line is in the plane \n\n');
else if x4==0
    fprintf('\nThe line is parallel with the plane \n');
    fprintf('Intersection in point at infinity');
    q = [x;0]
else
    % Euclidean point of intersection
    q = x/x4

    % Test (zeros expected)
    Test_u = dot(x,u) + x4*u4
    Test_Ls = dot(x,Lp)
    Test_Lv = -x4*Lp + cross(x,L)
end
end
```

□

9.26 Plane spanned by two intersecting lines

The plane defined by two intersecting lines $(\mathbf{l}_1, \mathbf{l}'_1)$ and $(\mathbf{l}_2, \mathbf{l}'_2)$ can be found in the same way as the plane defined by a line and point by using one of the lines and a point on the other line. The line $(\mathbf{l}_1, \mathbf{l}'_1)$ has direction vector \mathbf{l}_1 , which means that the point $(\mathbf{l}_1, 0)$ at infinity will be on the line. In the same way, the point $(\mathbf{l}_2, 0)$ at infinity will be on the line $(\mathbf{l}_2, \mathbf{l}'_2)$. This gives two alternative expressions for the plane as

$$(\mathbf{u}, u_4) = (\mathbf{l}_1 \times \mathbf{l}_2, \mathbf{l}_1 \cdot \mathbf{l}'_2) = (\mathbf{l}_1 \times \mathbf{l}_2, -\mathbf{l}_2 \cdot \mathbf{l}'_1) \quad (535)$$

This is in agreement with the condition $\mathbf{l}_1 \cdot \mathbf{l}'_2 = -\mathbf{l}_2 \cdot \mathbf{l}'_1$ as given by (562) for two lines $(\mathbf{l}_1, \mathbf{l}'_1)$ and $(\mathbf{l}_2, \mathbf{l}'_2)$ to intersect.

9.27 Point as intersection of three planes

A plane can be described by 3 homogeneous points on the plane. The dual version of this is that a point can be described as the intersection of 3 planes $\tilde{\mathbf{u}} = (\mathbf{u}, u_4)$, $\tilde{\mathbf{v}} = (\mathbf{v}, v_4)$ and $\tilde{\mathbf{w}} = (\mathbf{w}, w_4)$. This is done by finding the line

$$\mathbf{L} = (\mathbf{u} \times \mathbf{v}, u_4 \mathbf{v} - v_4 \mathbf{u}) \quad (536)$$

at the intersection of $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$. The intersection with the plane $\tilde{\mathbf{w}}$ is then given by (531), which gives

$$\tilde{\mathbf{x}} = (-w_4 \mathbf{l} + \mathbf{w} \times \mathbf{l}', \mathbf{w} \cdot \mathbf{l}) \quad (537)$$

Example

Three planes are given by $\tilde{\mathbf{u}} = ([1, 0, 0]^T, 0)$, $\tilde{\mathbf{v}} = ([0, 1, 0]^T, 0)$ and $\tilde{\mathbf{w}} = ([0, 0, 1]^T, 0)$. The line of intersection between $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ is given by $(\mathbf{l}', \mathbf{l}) = (\mathbf{u} \times \mathbf{v}, u_4 \mathbf{v} - v_4 \mathbf{u})$, which gives $\mathbf{l} = [0, 0, 1]^T$ and $\mathbf{l}' = [0, 0, 0]^T$. The point of intersection is then

$$\tilde{\mathbf{x}} = (-w_4 \mathbf{l} + \mathbf{w} \times \mathbf{l}', \mathbf{w} \cdot \mathbf{l}) = ([0, 0, 0]^T, 1) \quad (538)$$

which is the origin. This makes sense, since the three planes are the yz plane, the xz plane and the xy plane, which all contain the origin.

```
% Script for computing the intersection point of three planes
u = [1;0;0]; u4 = 1;
v = [0;1;0]; v4 = 2;
w = [0;0;1]; w4 = 2;

L = cross(u,v); Lp = u4*v - v4*u;
x = -w4*L + cross(w,Lp); x4 = dot(w,L);
% Point of intersection
q = x/x4
```

```
% Test is the point are in all the planes (zeros expected)
Test_u = dot(x,u) + x4*u4
Test_v = dot(x,v) + x4*v4
Test_w = dot(x,w) + x4*w4
```

□

9.28 The distance from a line to a point

The distance from a point to a line is easily calculated in 2D using homogeneous vector representations for point and line. In 3D the distance from a point to a plane is straightforward to calculate with homogeneous vectors for point and plane. When it comes to the distance from a point to a line in 3D the approach requires some more work.

Consider a point (\mathbf{x}, x_4) and a line $(\mathbf{l}, \mathbf{l}')$ in 3D, where \mathbf{l} is a unit vector. Let the point $(\mathbf{q}, 1)$ be the homogeneous point on the line that is closest to the origin, which means that

$$\mathbf{q} = \frac{\mathbf{l} \times \mathbf{l}'}{\mathbf{l} \cdot \mathbf{l}} \quad (539)$$

and $\mathbf{l}' = \mathbf{q} \times \mathbf{l}$.

Then the Euclidean vector from the Euclidean point represented by $(\mathbf{q}, 1)$ to the Euclidean point represented by (\mathbf{x}, x_4) is

$$\mathbf{d} = \frac{\mathbf{x}}{x_4} - \mathbf{q} = \frac{\mathbf{x} - x_4 \mathbf{q}}{x_4} \quad (540)$$

The component of \mathbf{d} that is orthogonal to the line is

$$\mathbf{d}_\perp = \frac{(\mathbf{l} \times \mathbf{d}) \times \mathbf{l}}{\mathbf{l} \cdot \mathbf{l}} = \left(\frac{\mathbf{l} \times (\mathbf{x} - x_4 \mathbf{q})}{x_4 \mathbf{l} \cdot \mathbf{l}} \right) \times \mathbf{l} = \frac{(\mathbf{l} \times \mathbf{x}) \times \mathbf{l} + x_4 \mathbf{l}' \times \mathbf{l}}{x_4 \mathbf{l} \cdot \mathbf{l}} \quad (541)$$

It follows that the shortest vector from the line to the point is (\mathbf{x}, x_4) is

$$\mathbf{d}_\perp = \frac{\mathbf{l} \times (-x_4 \mathbf{l}' + \mathbf{x} \times \mathbf{l})}{x_4 \mathbf{l} \cdot \mathbf{l}} \quad (542)$$

The vectors \mathbf{l}' and $\mathbf{x} \times \mathbf{l}$ are orthogonal to the vector \mathbf{l} , and it follows that the distance is

$$|\mathbf{d}_\perp| = \frac{|-x_4 \mathbf{l}' + \mathbf{x} \times \mathbf{l}|}{|x_4| |\mathbf{l}|} \quad (543)$$

Example

```
% Script for computing the distance from a line (L,Lp) to a point (x,x4)
L = [0;1;0]; Lp = cross([1;0;1],L);
q = cross(L,Lp)
x = 2*[0;1;2]; x4 = 2;

dperp = (cross(L,(-x4*Lp + cross(x,L))))/(x4*dot(L,L))
distance = norm(dperp)
```

□

9.29 Plücker matrices

An alternative formulation is based on Plücker matrices [30]. Then a line through two points $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ is given by the Plücker matrix

$$\mathbf{P} = \tilde{\mathbf{x}}\tilde{\mathbf{y}}^T - \tilde{\mathbf{y}}\tilde{\mathbf{x}}^T \quad (544)$$

It follows that the elements of the matrix are given by

$$P_{ij} = x_i y_j - x_j y_i = l_{ij} \quad (545)$$

where l_{ij} are the Plücker coordinates of the line. It is seen that $P_{ii} = 0$ and that $P_{ij} = -P_{ji}$, which means that the Plücker matrix \mathbf{P} is skew symmetric. Moreover, it is straightforward to verify that the line has Plücker coordinates given by the representation

$$\mathbf{L} = (\mathbf{l}, \mathbf{l}') = (x_4 \mathbf{y} - y_4 \mathbf{x}, \mathbf{x} \times \mathbf{y}) = (l_{41}, l_{42}, l_{43}, l_{23}, l_{31}, l_{12}) \quad (546)$$

Straightforward calculation of the elements of \mathbf{P} gives

$$\mathbf{P} = \begin{bmatrix} 0 & l_{12} & -l_{31} & -l_{41} \\ -l_{12} & 0 & l_{23} & -l_{42} \\ l_{31} & -l_{23} & 0 & -l_{43} \\ l_{41} & l_{42} & l_{43} & 0 \end{bmatrix} \quad (547)$$

This can be written in terms of the vectors \mathbf{l} and \mathbf{l}' as

$$\mathbf{P} = \begin{bmatrix} -(\mathbf{l}')^\times & -\mathbf{l} \\ \mathbf{l}^T & 0 \end{bmatrix} \quad (548)$$

It is noted that the expression $-(\mathbf{l}')^\times$ for the upper 3×3 sub-matrix is consistent with the general result $(\mathbf{x}^\times \mathbf{y})^\times = \mathbf{y} \mathbf{x}^T - \mathbf{x} \mathbf{y}^T$.

If the line \mathbf{L} and the plane $\tilde{\mathbf{u}} = (\mathbf{u}, u_4)$ are given, then

$$\mathbf{P} \tilde{\mathbf{u}} = \begin{bmatrix} -(\mathbf{l}')^\times & -\mathbf{l} \\ \mathbf{l}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ u_4 \end{bmatrix} = \begin{bmatrix} -u_4 \mathbf{l} + \mathbf{u}^\times \mathbf{l}' \\ \mathbf{u}^T \mathbf{l} \end{bmatrix} \quad (549)$$

It is seen that the point of intersection $\tilde{z} = (z, z_4)$ between the line \mathbf{L} and the plane $\tilde{\mathbf{u}}$ is given by

$$\mathbf{P} \tilde{\mathbf{u}} = \tilde{z} \quad (550)$$

9.30 The intersection of a line and the plane at infinity

Consider the line $\mathbf{L} = (\mathbf{l}, \mathbf{l}')$ and the corresponding Plücker matrix \mathbf{P} given by (548). Then the intersection of the line and the plane $\pi_\infty = [0, 0, 0, 1]^T$ at infinity is found to be

$$\mathbf{P} \pi_\infty = \begin{bmatrix} -(\mathbf{l}')^\times & -\mathbf{l} \\ \mathbf{l}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} = - \begin{bmatrix} \mathbf{l} \\ 0 \end{bmatrix} \quad (551)$$

This is the point at infinity in the direction of \mathbf{l} .

A dual interpretation is that $\mathbf{P}\pi_\infty = -[\mathbf{l}^T, 0]^T$ is a plane through the origin with normal vector \mathbf{l} . The intersection of this plane and the line must be at the point on the line that is closest to the origin. These geometric arguments are in agreement with the calculation

$$\mathbf{P}\mathbf{P}\pi_\infty = \begin{bmatrix} -(\mathbf{l}')^\times & -\mathbf{l} \\ \mathbf{l}^T & 0 \end{bmatrix} \begin{bmatrix} -\mathbf{l} \\ 0 \end{bmatrix} = -\begin{bmatrix} \mathbf{l}^\times \mathbf{l}' \\ \mathbf{l} \cdot \mathbf{l} \end{bmatrix}$$

It has been shown that the Euclidean point on the line that is closest to the origin is $\mathbf{q} = \mathbf{l}^\times \mathbf{l}' / (\mathbf{l} \cdot \mathbf{l})$, which is represented by the homogeneous point

$$\mathbf{P}\mathbf{P}\pi_\infty = -\frac{1}{\mathbf{l} \cdot \mathbf{l}} \begin{bmatrix} \mathbf{q} \\ 1 \end{bmatrix} \quad (552)$$

9.31 Dual Plücker matrices

The dual Plücker matrix \mathbf{P}^* of the line at the intersection of two planes $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{v}}$ is given by

$$\mathbf{P}^* = \tilde{\mathbf{u}} \tilde{\mathbf{v}}^T - \tilde{\mathbf{v}} \tilde{\mathbf{u}}^T \quad (553)$$

The elements l_{ij}^* of the dual Plücker matrix correspond to the elements of the dual line $\mathbf{L}^* = (\mathbf{l}', \mathbf{l})$. The corresponding dual Plücker matrix is

$$\mathbf{P}^* = \begin{bmatrix} -\mathbf{l}^\times & -\mathbf{l}' \\ \mathbf{l}'^T & 0 \end{bmatrix} \quad (554)$$

The plane $\tilde{\mathbf{w}} = (\mathbf{w}, w_4)$ spanned by the line and the point $\tilde{\mathbf{z}} = (\mathbf{z}, z_4)$ can then be found as

$$\tilde{\mathbf{w}} = \mathbf{P}^* \tilde{\mathbf{z}} = \begin{bmatrix} -\mathbf{l}^\times & -\mathbf{l}' \\ \mathbf{l}'^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ z_4 \end{bmatrix} = \begin{bmatrix} -z_4 \mathbf{l}' + \mathbf{z}^\times \mathbf{l} \\ \mathbf{z}^T \mathbf{l}' \end{bmatrix} \quad (555)$$

10 Intersecting lines and non-intersecting lines

10.1 Condition for two lines to intersect

Consider the lines $(\mathbf{l}, \mathbf{l}')$ and $(\mathbf{m}, \mathbf{m}')$, where the vectors are given as usual in agreement with (444) and (445) by

$$\mathbf{l} = \begin{bmatrix} l_{41} \\ l_{42} \\ l_{43} \end{bmatrix}, \quad \mathbf{l}' = \begin{bmatrix} l_{23} \\ l_{31} \\ l_{12} \end{bmatrix} \quad (556)$$

$$\mathbf{m} = \begin{bmatrix} m_{41} \\ m_{42} \\ m_{43} \end{bmatrix}, \quad \mathbf{m}' = \begin{bmatrix} m_{23} \\ m_{31} \\ m_{12} \end{bmatrix} \quad (557)$$

Let $\tilde{\mathbf{x}} = (\mathbf{x}, x_4)$ and $\tilde{\mathbf{y}} = (\mathbf{y}, y_4)$ be two points on $(\mathbf{l}, \mathbf{l}')$ so that

$$l_{ij} = x_i y_j - x_j y_i \quad (558)$$

and let $\tilde{\mathbf{z}} = (\mathbf{z}, z_4)$ and $\tilde{\mathbf{w}} = (\mathbf{w}, w_4)$ be two points on $(\mathbf{m}, \mathbf{m}')$ so that

$$m_{ij} = z_i w_j - z_j w_i \quad (559)$$

If the lines intersect, then all points on the two lines must be in the plane that contains the two lines. This means that the 4 points must be linearly dependent, and it follows that

$$\begin{vmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ w_1 & w_2 & w_3 & w_4 \end{vmatrix} = 0 \quad (560)$$

This determinant can be evaluated by Laplace expansion in combination with (558) and (559), which gives [61]

$$l_{41}m_{23} + l_{42}m_{31} + l_{43}m_{12} + l_{23}m_{41} + l_{31}m_{42} + l_{12}m_{43} = 0 \quad (561)$$

It is seen that this can be written $\mathbf{l} \cdot \mathbf{m}' + \mathbf{m} \cdot \mathbf{l}' = 0$. It is concluded that the two lines $(\mathbf{l}, \mathbf{l}')$ and $(\mathbf{m}, \mathbf{m}')$ will be in the same plane, and will therefore intersect if and only if

$$\mathbf{l} \cdot \mathbf{m}' + \mathbf{m} \cdot \mathbf{l}' = 0 \quad (562)$$

10.2 Point as intersection of two lines: Geometric method

Suppose that the two lines $(\mathbf{l}_1, \mathbf{l}'_1)$ and $(\mathbf{l}_2, \mathbf{l}'_2)$ are known to be intersecting. An algorithm which computes the point of intersection between the two lines can then be developed from geometric arguments with the method of Shoemake [65]. The first step of the algorithm is to find a plane (\mathbf{v}, v_4) which contains $(\mathbf{l}_1, \mathbf{l}'_1)$. Then in the second step the intersection of (\mathbf{v}, v_4) and $(\mathbf{l}_2, \mathbf{l}'_2)$ is found. This requires that (\mathbf{v}, v_4) is not the common plane of the two lines. This is ensured by letting the plane be spanned by the line $(\mathbf{l}_1, \mathbf{l}'_1)$ and the point $(\mathbf{n}, 0)$ at infinity, where $\mathbf{n} = \mathbf{l}_1 \times \mathbf{l}_2$. Then the plane is $(\mathbf{v}, v_4) = (\mathbf{n} \times \mathbf{l}_1, \mathbf{n} \cdot \mathbf{l}'_1)$.

The algorithm for finding the point of intersection is then

$$\mathbf{n} = \mathbf{l}_1 \times \mathbf{l}_2 \quad (563)$$

$$(\mathbf{v}, v_4) = (\mathbf{n} \times \mathbf{l}_1, \mathbf{n} \cdot \mathbf{l}'_1) \quad (564)$$

$$(\mathbf{x}, x_4) = (-v_4 \mathbf{l}_2 + \mathbf{v} \times \mathbf{l}'_2, \mathbf{v} \cdot \mathbf{l}_2), \quad (565)$$

The solution is robust in the sense that it gives an approximate solution if the two lines do not intersect. The approximate solution is the intersection of the second line and the plane, which means that the calculated intersection point is a point at the second line. It is noted that this method cannot be used if one of the lines are at infinity.

Example

Consider the lines $(\mathbf{l}_1, \mathbf{l}'_1)$ and $(\mathbf{l}_2, \mathbf{l}'_2)$ where $\mathbf{l}_1 = [1, 0, 0]^T$ and $\mathbf{l}_2 = \frac{1}{\sqrt{3}}[1, 1, 1]^T$, and where $(\mathbf{l}_1, \mathbf{l}'_1)$ passes through the point $\mathbf{p}_1 = [1, 1, 3]^T$ and $(\mathbf{l}_2, \mathbf{l}'_2)$ passes through $\mathbf{p}_2 = \mathbf{p}_1 + \alpha[0, 1, 0]^T$. If $\alpha = 0$, then both lines pass through \mathbf{p}_1 , which is then the point of intersection. The method of Shoemake then gives the intersection at \mathbf{p}_i . The lines, the plane and the point of intersection is shown in Figure 49.

If $\alpha = 0.4$, then the two lines will not intersect. The method of Shoemake will then find the point of intersection between the line $(\mathbf{l}_2, \mathbf{l}'_2)$ and the plane (\mathbf{v}, v_4) as shown in Figure 49.

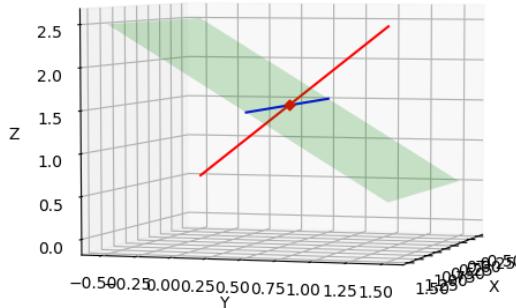


Figure 49: The line (l_1, l'_1) (blue) and the line (l_2, l'_2) (red) and the point of intersection (red diamond). The plane (v, v_4) is shown in transparent green.

```

# The method of Shoemake for computing the intersection point of two intersecting lines,
# a least squares solution for the geometric error
# and a least squares method for the algebraic error
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
from matplotlib import colors as mcolors

def skewm(r):
    return np.array([[0,-r[2],r[1]], [r[2],0,-r[0]], [-r[1],r[0],0]])

def cc(arg):
    return mcolors.to_rgba(arg, alpha=0.6)

def plot_points(X, **kwargs):
    if 'col' in kwargs:
        if 'mark' in kwargs:
            ax.scatter3D(X[0,:], X[1,:], X[2,:], c= kwargs['col'],
                         marker= kwargs['mark'])
        else:
            ax.scatter3D(X[0,:], X[1,:], X[2,:], c= kwargs['col'])
    else:
        ax.scatter3D(X[0,:], X[1,:], X[2,:])
    return

def plot_line(x, y, **kwargs):
    alpha = np.linspace(0, 1, 100)
    L = np.multiply(alpha, (x - y).reshape(3,1)) + y.reshape(3,1)
    if 'style' in kwargs:

```

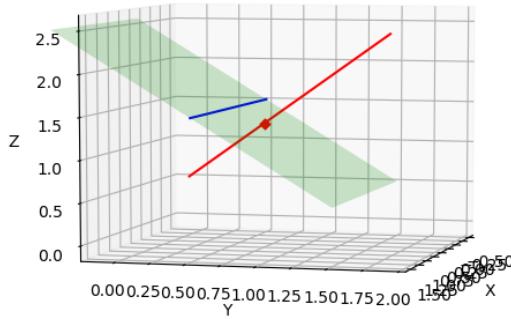


Figure 50: The line (l_1, l'_1) (blue) and the line (l_2, l'_2) (red) and the point of intersection (red diamond) between the line (l_2, l'_2) and the plane (v, v_4) .

```

    ax.plot(L[0,:], L[1,:], L[2,:], kwargs['style'])
else:
    ax.plot(L[0,:], L[1,:], L[2,:])
return
def plot_square(x, y, z, col, fill):
    verts = [list(zip(x, y, z))]
    poly = Poly3DCollection(verts, facecolors=cc(col))
    poly.set_alpha(fill)
    ax.add_collection3d(poly)
    return
def vectors_in_plane(u):
    un = np.linalg.norm(np.abs(u))
    im = np.argmin(u)
    E = np.array([[1,0,0], [0,1,0], [0,0,1]])
    e = E[:,im]
    a = e - u * np.dot(e,u)/un**2
    a = a/np.linalg.norm(a)
    b = np.cross(u, a)/un
    return a, b

pt = np.array([2, 2, 6, 4])
p1 = pt[0:3]/pt[3]
p2 = p1 - 1.0*np.array([0,1,0])

# Lines
L1 = np.array([1, 0, 0]); L1p = np.cross(p1, L1)
L2 = np.array([1, 1, 1]); L2p = np.cross(p2, L2)

```

```

# Common normal
Ln = np.cross(L1,L2); Lnp = np.cross(L1,L2p) + np.cross(L1p,L2)

if (np.dot(L1,L2p) + np.dot(L2,L1p)) == 0:
    print('Lines intersect')
# Shoemake's plane through L1
n1 = np.cross(L1, Ln)
v1 = np.cross(n1, L1); v14 = np.dot(n1,L1p)
# Intersection of L1 and Ln
x1 = -v14*Ln + np.cross(v1,Lnp); x14 = np.dot(v1,Ln);
y1 = x1/x14
# Shoemake's plane through L2
n2 = np.cross(L2, Ln)
v2 = np.cross(n2, L2); v24 = np.dot(n2,L2p)
# Intersection of L2 and Ln
x2 = -v24*Ln + np.cross(v2,Lnp); x24 = np.dot(v2,Ln);
y2 = x2/x24

# Shoemake's planes for L1 and L2
n = np.cross(L1, L2)
v = np.cross(n, L1); v4 = np.dot(n,L1p)
w = np.cross(n, L2); w4 = np.dot(n,L2p)
# Point of intersection between v and (L2, L2p)
x = -v4*L2 + np.cross(v,L2p); x4 = np.dot(v,L2);
y = x/x4
# Point of intersection between w and (L1, L1p)
xw = -w4*L1 + np.cross(w,L1p); xw4 = np.dot(w,L1);
yw = xw/xw4

# Planes for plotting of Shoemake's planes for L1 and L2
a1, a2 = vectors_in_plane(v)
qv = - v4 * v / np.linalg.norm(v)**2
qv = y
X_plane = np.block([[qv - a1 - a2], [qv - a1 + a2],
                     [qv + a1 + a2], [qv + a1 - a2]]).T
aw1, aw2 = vectors_in_plane(w)
qw = - w4 * w / np.linalg.norm(w)**2
#qw = p1
Xw_plane = np.block([[yw - aw1 - aw2], [yw - aw1 + aw2],
                     [yw + aw1 + aw2], [yw + aw1 - aw2]]).T

# Linear algebra solution for least squares geometric error
L1_norm = np.linalg.norm(L1)
L2_norm = np.linalg.norm(L2)
A = np.block([[skewm(L1/L1_norm)], [skewm(L2/L2_norm)]]);
b = - np.block([L1p/L1_norm, L2p/L2_norm]).reshape(6,1);

```

```

# z = np.linalg.inv(A.T @ A) @ A.T @ b
z, residuals, rank, s = np.linalg.lstsq(A, b, rcond=-1)

# Linear algebra solution for least squares algebraic error
A_svd = np.block([[[-skewm(L1/L1_norm), -(L1p/L1_norm).reshape(3,1)],
                   [L1p/L1_norm, 0],
                   [-skewm(L2/L2_norm), -(L2p/L2_norm).reshape(3,1)],
                   [L2p/L2_norm, 0]]]);
U,S,Vt = np.linalg.svd(A_svd);
zt = Vt[3,:]; zz = zt[0:3]/zt[3]

fig = plt.figure(1)
fig.clear()
ax = fig.gca(projection='3d')
plot_square(X_plane[0,:], X_plane[1,:], X_plane[2,:], 'b', 0.2)
plot_square(Xw_plane[0,:], Xw_plane[1,:], Xw_plane[2,:], 'g', 0.2)
plot_line(p1 - L1, p1 + L1, style='--b')
plot_line(p2 - L2, p2 + L2, style='--g')
plot_line(y1, y2, style='--g')
plot_points(y.reshape(3,1), col='r', mark='D')
plot_points(y1.reshape(3,1), col='b', mark='D')
plot_points(z.reshape(3,1), col='c', mark='D')
plot_points(zz.reshape(3,1), col='m', mark='D')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.show()

```

□

10.3 Least-squares geometric error for the intersection point of lines

Consider the case where a set of lines are given, and the point of intersection is to be found. It is assumed that there is some uncertainty in parameters of the lines, which means that the lines do not necessarily intersect.

First it is noted that the condition (593) for a point to be on a line that is not at infinity can be simplified to

$$\mathbf{C}_i \begin{bmatrix} \mathbf{x} \\ x_4 \end{bmatrix} = \mathbf{0} \quad (566)$$

where the 3×4 matrix

$$\mathbf{C}_i = \begin{bmatrix} \mathbf{l}_i^\times & \mathbf{l}'_i \end{bmatrix} \quad (567)$$

will be of rank 2 since \mathbf{l}_i^\times is of rank 2. This means that the intersection point of two lines $(\mathbf{l}_1, \mathbf{l}'_1)$ and $(\mathbf{l}_2, \mathbf{l}'_2)$ can be found from

$$\mathbf{C} \begin{bmatrix} \mathbf{x} \\ x_4 \end{bmatrix} = \mathbf{0}, \quad \mathbf{C} = \begin{bmatrix} \mathbf{C}_1 \\ \mathbf{C}_2 \end{bmatrix} \quad (568)$$

where the 6×4 matrix \mathbf{C} will be of rank 3 if the lines intersect, since the intersection on the line imposes the condition (562) on the two lines.

It is recalled that the distance from the line $(\mathbf{l}_i, \mathbf{l}'_i)$ with unit direction vector \mathbf{l}_i to a homogeneous point $\tilde{\mathbf{x}} = (\mathbf{x}, 1)$ is $\delta = \|\mathbf{u}\|$, where

$$\mathbf{u} = -\mathbf{l}'_i + \mathbf{x} \times \mathbf{l}_i = -\mathbf{C}_i \tilde{\mathbf{x}} \quad (569)$$

is the normal vector of the plane defined by the line and the point. This means that the magnitude of

$$\boldsymbol{\epsilon}_i = \mathbf{C}_i \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \quad (570)$$

is equal to the magnitude of the distance from the line $(\mathbf{l}_i, \mathbf{l}'_i)$ to the point $\tilde{\mathbf{x}} = (\mathbf{x}, 1)$ under the condition that $\|\mathbf{l}_i\| = 1$.

It is noted that

$$\boldsymbol{\epsilon}_i = \mathbf{C}_i \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} = \mathbf{A}_i \mathbf{x} - \mathbf{b}_i \quad (571)$$

where

$$\mathbf{A}_i = \mathbf{l}_i^\times, \quad \mathbf{b}_i = -\mathbf{l}'_i \quad (572)$$

It follows that if a set of lines $(\mathbf{l}_i, \mathbf{l}'_i)$, $i = 1, \dots, N$ with unit direction vectors \mathbf{l}_i is given, then the point $\tilde{\mathbf{x}} = (\mathbf{x}, 1)$ that is geometrically closest to the lines in the least-squares sense, is the point that minimizes

$$\sum_{i=1}^N \boldsymbol{\epsilon}_i^T \boldsymbol{\epsilon}_i = \sum_{i=1}^N \|\mathbf{A}_i \mathbf{x} - \mathbf{b}_i\|^2 \quad (573)$$

This can be rewritten with

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_N \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_N \end{bmatrix} \quad (574)$$

which gives the minimization problem

$$\sum_{i=1}^N \boldsymbol{\epsilon}_i^T \boldsymbol{\epsilon}_i = \|\mathbf{A} \mathbf{x} - \mathbf{b}\|^2 \quad (575)$$

The solution is then the least-squares solution of

$$\mathbf{A} \mathbf{x} = \mathbf{b} \quad (576)$$

which is found from the normal equations $\mathbf{A}^T \mathbf{A} \mathbf{x}^* = \mathbf{A}^T \mathbf{b}$ as

$$\mathbf{x}^* = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} \quad (577)$$

It is noted that this method cannot be used for lines at infinity or points at infinity.

Example

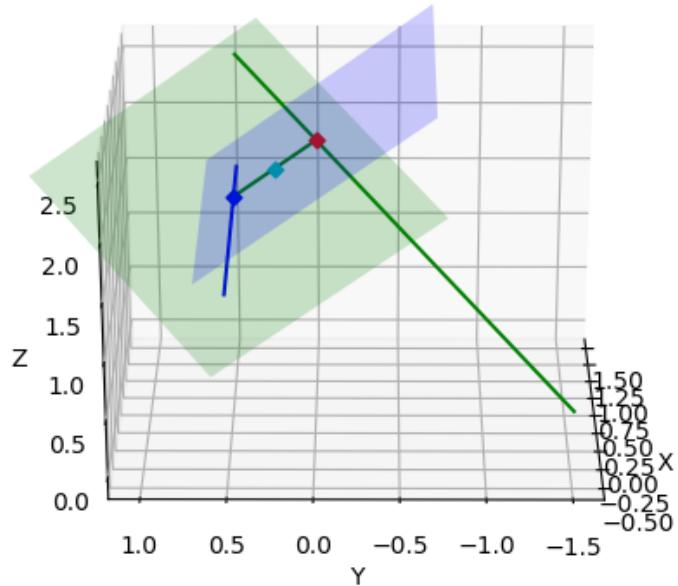


Figure 51: The line (l_1, l'_1) (blue) and the line (l_2, l'_2) (green), the plane through (l_1, l'_1) (blue), the plane through (l_2, l'_2) (green) and the points of intersection according to Shoemake between the lines and the planes (red and blue diamond). The minimization of the least-squares geometric error gives the point in the middle of the common normal between the two points from Shoemake's method.

```
% Four lines that are close to intersecting at q = (0,0,1)
q = [0;0;1];
L1 = [1;0;0]; L1 = L1/norm(L1); q1 = q + [0;0;0.1]; L1p = cross(q1,L1);
L2 = [0;1;0]; L2 = L2/norm(L2); q2 = q + [0.1;0;0]; L2p = cross(q2,L2);
L3 = [1;0;0]; L3 = L3/norm(L3); q3 = q + [0;0;-0.1]; L3p = cross(q3,L3);
L4 = [0;1;0]; L4 = L4/norm(L4); q4 = q + [-0.2;0;0]; L4p = cross(q4,L4);

% Calculation of closest point to the four lines
skewm =@(u) [0 -u(3) u(2); u(3) 0 -u(1);-u(2) u(1) 0];
A = [skewm(L1); skewm(L2); skewm(L3); skewm(L4)];
b = -[L1p;L2p;L3p;L4p];
z = inv(A'*A)*A'*b

% Distance from lines to point
A*z - b
```

□

10.4 The common normal of two lines

Consider two non-parallel lines $(\mathbf{l}_1, \mathbf{l}'_1)$ and $(\mathbf{l}_2, \mathbf{l}'_2)$. According to screw theory for lines [44] the two lines will have a common normal $(\mathbf{l}_n, \mathbf{l}'_n)$, which is a line that intersect both lines at a right angle. The distance between the two lines along the common normal is the shortest distance between the two lines. In the presentation of [44] an expression for the common normal is presented where it is used that lines $(\mathbf{l}_1, \mathbf{l}'_1)$ and $(\mathbf{l}_2, \mathbf{l}'_2)$ are screws. The theory of screws will not be introduced in this note, but it is noted that the cross product of the two lines treated as screws is

$$(\mathbf{l}_1, \mathbf{l}'_1) \times (\mathbf{l}_2, \mathbf{l}'_2) = (\mathbf{l}_1 \times \mathbf{l}_2, \mathbf{l}_1 \times \mathbf{l}'_2 + \mathbf{l}'_1 \times \mathbf{l}_2) \quad (578)$$

The common normal $(\mathbf{l}_n, \mathbf{l}'_n)$ is given by this cross product as

$$(\mathbf{l}_n, \mathbf{l}'_n) = (\mathbf{l}_1 \times \mathbf{l}_2, \mathbf{l}_1 \times \mathbf{l}'_2 + \mathbf{l}'_1 \times \mathbf{l}_2) \quad (579)$$

It is seen that the direction vector of the common normal is $\mathbf{l}_n = \mathbf{l}_1 \times \mathbf{l}_2$, which is obviously normal to the direction vector of both lines.

It is verified that $(\mathbf{l}_n, \mathbf{l}'_n)$ intersects $(\mathbf{l}_1, \mathbf{l}'_1)$ by showing that the condition (562) holds, which is done with the calculation

$$\mathbf{l}_1 \cdot \mathbf{l}'_n + \mathbf{l}_n \cdot \mathbf{l}'_1 = \mathbf{l}_1 \cdot (\mathbf{l}_1 \times \mathbf{l}'_2 + \mathbf{l}'_1 \times \mathbf{l}_2) + (\mathbf{l}_1 \times \mathbf{l}_2) \cdot \mathbf{l}'_1 \quad (580)$$

$$= \mathbf{l}_1 \cdot (\mathbf{l}'_1 \times \mathbf{l}_2) + (\mathbf{l}_1 \times \mathbf{l}_2) \cdot \mathbf{l}'_1 \quad (581)$$

$$= \mathbf{l}_1 \cdot (\mathbf{l}'_1 \times \mathbf{l}_2) - \mathbf{l}'_1 \cdot (\mathbf{l}_2 \times \mathbf{l}_1) \quad (582)$$

$$= 0 \quad (583)$$

where the cyclic property $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = \mathbf{b} \cdot (\mathbf{c} \times \mathbf{a})$ of the scalar triple vector product is used. In the same way it can be shown that $(\mathbf{l}_n, \mathbf{l}'_n)$ intersects $(\mathbf{l}_2, \mathbf{l}'_2)$. The points of intersection between the common normal and the two lines can be found with the method of Shoemake.

10.5 The distance between two nonparallel lines

The displacement from line $(\mathbf{l}_1, \mathbf{l}'_1)$ to line $(\mathbf{l}_2, \mathbf{l}'_2)$ can then be described as a translation d along the common normal, and then a rotation θ about the common normal (Figure 52). This means that the line $(\mathbf{l}_2, \mathbf{l}'_2)$ will be in a plane π_1 which is perpendicular to the common normal, while $(\mathbf{l}_2, \mathbf{l}'_2)$ will be in a plane π_2 which is perpendicular to the common normal. The distance between the planes π_1 and π_2 , which is also the distance between the two lines along the common normal, is then found as

$$d = \mathbf{l}_n \cdot (\mathbf{x}_2 - \mathbf{x}_1) \quad (584)$$

where \mathbf{x}_1 is an arbitrary point on $(\mathbf{l}_1, \mathbf{l}'_1)$, and \mathbf{x}_2 is an arbitrary point on $(\mathbf{l}_2, \mathbf{l}'_2)$. The solution is undefined when the lines are parallel.

Example

Consider the lines $(\mathbf{l}_1, \mathbf{l}'_1) = (\mathbf{e}_x, \mathbf{0})$ and $(\mathbf{l}_2, \mathbf{l}'_2) = (\mathbf{e}_y, [-1, 0, 1]^T)$ where $\mathbf{e}_x = [1, 0, 0]^T$ is the unit vector along the x axis, and $\mathbf{e}_y = [0, 1, 0]^T$ is the unit vector along the y axis. The first line

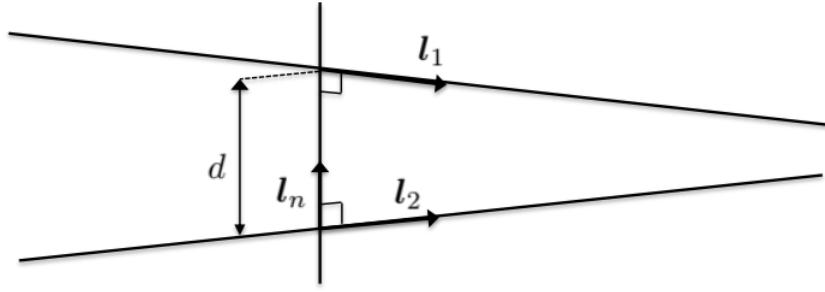


Figure 52: The distance d between two lines is the translation between the lines along the common normal.

passes through the origin, while the second line passes through the point $\mathbf{q}_2 = \mathbf{l}_2 \times \mathbf{l}'_2 = [1, 0, 1]^T$. The common normal is

$$(\mathbf{l}_n, \mathbf{l}'_n) = (\mathbf{e}_x \times \mathbf{e}_y, \mathbf{e}_x \times [-1, 0, 1]^T) = (\mathbf{e}_z, -\mathbf{e}_y) \quad (585)$$

where \mathbf{e}_z is the unit vector along the z axis. The common normal passes through the point $\mathbf{q}_n = \mathbf{e}_z \times (-\mathbf{e}_y) = \mathbf{e}_y \times \mathbf{e}_z = \mathbf{e}_x$.

10.6 The midpoint of the common normal of two nonparallel lines

Consider the two lines $(\mathbf{l}_1, \mathbf{l}'_1)$ and $(\mathbf{l}_2, \mathbf{l}'_2)$ with common normal

$$(\mathbf{l}_n, \mathbf{l}'_n) = (\mathbf{l}_1 \times \mathbf{l}_2, \mathbf{l}_1 \times \mathbf{l}'_2 + \mathbf{l}'_1 \times \mathbf{l}_2) \quad (586)$$

The first line and the common normal intersect, and according to (535) define the plane

$$(\mathbf{u}, u_4) = (\mathbf{l}_1 \times \mathbf{l}_n, \mathbf{l}_1 \cdot \mathbf{l}'_n) = (\mathbf{l}_1 \times \mathbf{l}_n, -\mathbf{l}_n \cdot \mathbf{l}'_1) \quad (587)$$

The second line and the common normal will also be intersecting, and defines the plane

$$(\mathbf{v}, v_4) = (\mathbf{l}_2 \times \mathbf{l}_n, \mathbf{l}_2 \cdot \mathbf{l}'_n) = (\mathbf{l}_2 \times \mathbf{l}_n, -\mathbf{l}_n \cdot \mathbf{l}'_2) \quad (588)$$

According to (531) the common normal will then intersect the first line at the intersection of the plane (\mathbf{v}, v_4) and the line $(\mathbf{l}_1, \mathbf{l}'_1)$, which is at

$$(\mathbf{x}, x_4) = (-v_4 \mathbf{l}_1 + \mathbf{v} \times \mathbf{l}'_1, \mathbf{v} \cdot \mathbf{l}_1) \quad (589)$$

In the same way it is found that the common normal will intersect the second line at the intersection of the plane (\mathbf{u}, u_4) and the line $(\mathbf{l}_2, \mathbf{l}'_2)$, which is at

$$(\mathbf{y}, y_4) = (-u_4 \mathbf{l}_2 + \mathbf{u} \times \mathbf{l}'_2, \mathbf{u} \cdot \mathbf{l}_2) \quad (590)$$

The midpoint of the two lines is then found as the Euclidean point

$$\mathbf{q}_m = \frac{1}{2} \left(\frac{\mathbf{x}}{x_4} + \frac{\mathbf{y}}{y_4} \right) = \frac{y_4 \mathbf{x} + x_4 \mathbf{y}}{2x_4 y_4} \quad (591)$$

as shown in Figure 53.

Example

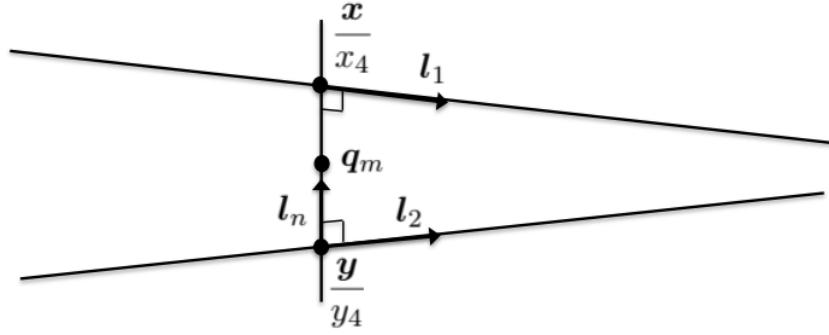


Figure 53: The distance d between two lines is the translation between the lines along the common normal.

```
% Script which calculates the distance and the midpoint between two lines
L1 = [1;0;0]; L1p = [0;0;0]; % Line 1
L2 = [0;1;0]; L2p = [-1;0;1]; % Line 2
% q1 = cross(L1,L1p);
% q2 = cross(L2,L2p);

% Common normal
Ln = cross(L1,L2); Lnp = cross(L1,L2p) + cross(L1p,L2);
% qn = cross(Ln,Lnp);

u = cross(L1,Ln); u4 = dot(L1,Lnp); % Plane with line 1 and common normal
v = cross(L2,Ln); v4 = dot(L2,Lnp); % Plane with line 2 and common normal
% Intersection of lines and common normal
x = -v4*L1 + cross(v,L1p); x4 = dot(v,L1);
y = -u4*L2 + cross(u,L2p); y4 = dot(u,L2);

% Distance
d = norm((y4*x-x4*y)/(x4*y4))

% Midpoint
qm = (y4*x+x4*y)/(2*x4*y4)
```

□

10.7 Point of intersection between a set of lines by minimization of algebraic error *

The condition for a point $\tilde{x} = (\mathbf{x}, x_4)$ to be on a line (l_i, l'_i) is given by (462), which gives

$$\mathbf{x} \cdot l'_i = 0, \quad -x_4 l'_i + \mathbf{x} \times l_i = \mathbf{0} \quad (592)$$

This condition can be written in matrix form as

$$\mathbf{A}_i \begin{bmatrix} \mathbf{x} \\ x_4 \end{bmatrix} = \mathbf{0} \quad (593)$$

where

$$\mathbf{A}_i = \begin{bmatrix} -\mathbf{l}_i^\times & -\mathbf{l}'_i \\ (\mathbf{l}'_i)^T & 0 \end{bmatrix} \quad (594)$$

is known as the dual Plücker matrix.

First it is assumed that the line is finite, which means that \mathbf{l} will be the direction vector of the line. This means that \mathbf{l}_i^\times will be of rank 2. The matrix \mathbf{A}_i will also be of rank 2 since the solutions $\tilde{\mathbf{x}}$ have one degree of freedom along the line, and one degree of freedom since $\tilde{\mathbf{x}}$ is homogeneous. This means that \mathbf{A}_i is of rank 2 independently from the value of \mathbf{l}'_i .

Then it is assumed that the line is at infinity, then $\mathbf{l}_i = \mathbf{0}$, and \mathbf{l}'_i will be a nonzero vector. In this case the matrix \mathbf{A}_i will still be of rank 2.

Then the condition for the point to be on the two lines $(\mathbf{l}_1, \mathbf{l}'_1)$ and $(\mathbf{l}_2, \mathbf{l}'_2)$ is given by

$$\mathbf{A} \begin{bmatrix} \mathbf{x} \\ x_4 \end{bmatrix} = \mathbf{0}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix} \quad (595)$$

where the matrix \mathbf{A} will be of rank 3 if the two lines intersect. The solution is then found by using the SVD $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. The solution is $\tilde{\mathbf{x}} = \gamma \mathbf{v}_4$, where \mathbf{v}_4 is the last column of the \mathbf{V} matrix. This method can also be used for lines at infinity and for parallel lines. It is straightforward to extend this method to N lines with

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_N \end{bmatrix} \quad (596)$$

This method will minimize an algebraic error.

Example 1

In the example of two lines this method gives a point which is not on the common normal as shown in Figure 54.

Example 2

```
% Two lines intersecting at q
L1 = [1;0;0]; q1 = [0;1;0]; L1p = cross(q1,L1);
L2 = [0;1;0]; q2 = [0;1;0]; L2p = cross(q2,L2);

% Lines at infinity
% L1 = [0;0;0]; L2 = [0;0;0]; L1p = [1;0;0]; L2p = [0;1;0];
% Parallel lines
% L1 = [1;0;0]; q1 = [0;1;0]; L1p = cross(q1,L1);
% L2 = [1;0;0]; q2 = [0;2;0]; L2p = cross(q2,L2);

skewm = @(u) [0 -u(3) u(2); u(3) 0 -u(1); -u(2) u(1) 0];
A = [-skewm(L1) -L1p; L1p' 0; -skewm(L2) -L2p; L2p' 0];
[U,S,V] = svd(A);
```

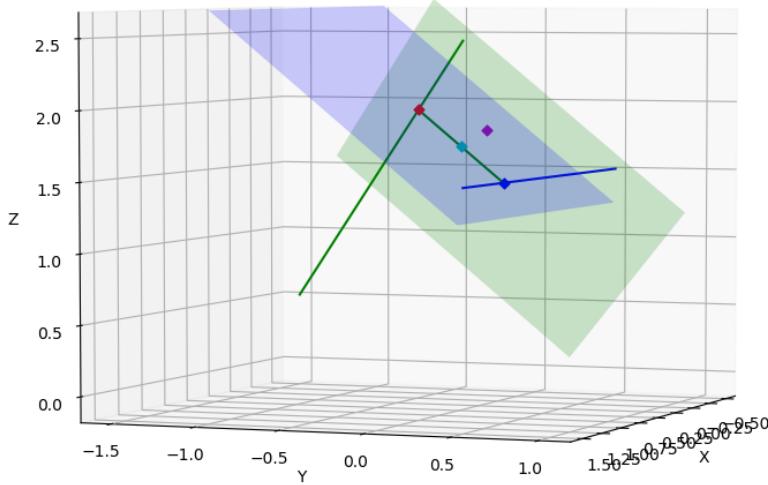


Figure 54: The line (l_1, l'_1) (blue) and the line (l_2, l'_2) (green), the plane through (l_1, l'_1) (blue), the plane through (l_2, l'_2) (green) and the points of intersection according to Shoemake between the lines and the planes (red and blue diamond). The minimization of the least-squares geometric error gives the point (cyan) in the middle of the common normal between the two points from Shoemake's method. The least-squares minimization of the algebraic error gives a point (magenta) which is not on the common normal.

```

xt = v(:,4);

if xt(4) ~=0
    xt = xt/xt(4); % Finite point is scaled
end
% Point of intersection
xt

```

□

11 Images of geometric objects

11.1 The image of a line

In this section an expression for the image of a 3D line

$$\mathbf{L} = (\mathbf{a}, \mathbf{m}) \quad (597)$$

will be found, where \mathbf{a} is the direction vector and \mathbf{m} is the moment. Suppose that the homogeneous points $\tilde{\mathbf{x}} = (\mathbf{x}, 1)$ and $\tilde{\mathbf{y}} = (\mathbf{y}, 1)$ are on the line, where $\mathbf{x} = [x_1, x_2, x_3]^T$ and $\mathbf{y} = [y_1, y_2, y_3]^T$. Then the Plücker coordinates of the line are given by

$$\mathbf{L} = (\mathbf{y} - \mathbf{x}, \mathbf{x} \times \mathbf{y}) \quad (598)$$

The points $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$ correspond to the normalized image coordinate vectors

$$\mathbf{s}_x = \frac{1}{x_3} \mathbf{x} \quad (599)$$

$$\mathbf{s}_y = \frac{1}{y_3} \mathbf{y} \quad (600)$$

The line \mathbf{L} is imaged to the line ℓ in the normalized image plane. Then \mathbf{s}_x and \mathbf{s}_y will be on the line ℓ , and it follows that

$$\ell = \mathbf{s}_x \times \mathbf{s}_y = \frac{1}{x_3 y_3} \mathbf{x} \times \mathbf{y} \quad (601)$$

Since the line is homogeneous and $\mathbf{m} = \mathbf{x} \times \mathbf{y}$ the image can be written

$$\ell = \mathbf{m} \quad (602)$$

This means that the normalized image of the line $\mathbf{L} = (\mathbf{a}, \mathbf{m})$ is $\ell = \mathbf{m}$.

11.2 The plane defined by a 3D line and the camera center

Consider a 3D line in the scene given in the camera frame in Plücker coordinates as

$$\mathbf{L} = (\mathbf{a}, \mathbf{m}) \quad (603)$$

This line and the origin of the camera frame $\tilde{\mathbf{z}} = (\mathbf{z}, z_4) = (\mathbf{0}, 1)$ of the defines a plane

$$\tilde{\mathbf{u}} = (\mathbf{u}, u_4) = (-z_4 \mathbf{m} + \mathbf{z} \times \mathbf{a}, \mathbf{z} \cdot \mathbf{m}) \quad (604)$$

which after scaling with -1 gives

$$\tilde{\mathbf{u}} = (\mathbf{m}, 0) \quad (605)$$

The image of the line in the normalized image plane is according to (602) given by $\ell = \mathbf{m}$, which can be interpreted as the normal vector to the plane $\tilde{\mathbf{u}}$.

11.3 The image of a line defined by two planes

Suppose that the line \mathbf{L} is the intersection of the planes $\tilde{\mathbf{u}} = (\mathbf{u}, u_4)$ and $\tilde{\mathbf{v}} = (\mathbf{v}, v_4)$. Then the line is given in terms of the planes as

$$\mathbf{L} = (\mathbf{u} \times \mathbf{v}, u_4 \mathbf{v} - v_4 \mathbf{u}) \quad (606)$$

This gives the alternative expression

$$\ell = u_4 \mathbf{v} - v_4 \mathbf{u} \quad (607)$$

for the normalized image of the line.

11.4 Triangulation of a point in a known plane

Consider the 3D point $\tilde{\mathbf{x}} = (\mathbf{x}, 1)$ in a known plane $\tilde{\mathbf{u}} = (\mathbf{u}, u_4)$, which implies $\tilde{\mathbf{x}} \cdot \tilde{\mathbf{u}} = 0$, or, equivalently, $\mathbf{x} \cdot \mathbf{u} = -u_4$. The normalized image position is given by $\mathbf{s} = \mathbf{x}/x_3$. Then $\mathbf{s} \cdot \mathbf{u} = -u_4/x_3$, which gives $x_3 = -u_4/(\mathbf{s} \cdot \mathbf{u})$, and it follows that

$$\mathbf{x} = x_3 \mathbf{s} = \frac{-u_4}{\mathbf{s} \cdot \mathbf{u}} \mathbf{s} \quad (608)$$

This method of determining the 3D Euclidean vector \mathbf{x} is called triangulation.

12 The geometry of a laser line scanner

12.1 The geometry of the laser plane

Consider a laser scanner system which consist of a camera and a laser scanner. The laser scanner has a a mirror which rotates about an axis, and the laser beam is swept over a plane which is the laser plane. The projection of the laser plane on an object in the scene is detected with a camera which is offset from the laser scanner. If the object is a plane, then the projected laser plane will be the intersection between the laser plane and the object plane, which is a line. The geometry of this type of laser scanner can be described in the same way as a stereo camera arrangement were the second camera is replaced with a laser scanner. The camera frame is denoted frame c , and the scanner frame is denoted frame s . The laser frame is fixed in the laser.

The displacement from camera frame c to scanner frame s is given by

$$\mathbf{T}_s^c = \begin{bmatrix} \mathbf{R}_s^c & \mathbf{t}_{cs}^c \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (609)$$

The notation is simplified by writing $\mathbf{T}_s^c = \mathbf{T}$, $\mathbf{R}_s^c = \mathbf{R}$ and $\mathbf{t} = \mathbf{t}_{cs}^c$.

The normal vector of the laser plane is is given in the laser frame as \mathbf{u}^s , and in the camera frame as $\mathbf{u} = \mathbf{R}_s^c \mathbf{u}^s$. If the laser plane is, e.g., the yz plane of the laser frame, then the normal vector is along the x axis of the laser frame s , and $\mathbf{u} = \mathbf{r}_1$.

The laser plane is given in Plücker coordinates in the coordinates of c and referenced to frame c by (479) as

$$\tilde{\mathbf{u}} = (\mathbf{u}, u_4) = (\mathbf{u}, -\mathbf{t} \cdot \mathbf{u}) \quad (610)$$

since \mathbf{u} is the normal vector, and \mathbf{t} is a point in the plane in coordinate frame c .

12.2 The image of a point in the laser plane

Consider the homogeneous point $\tilde{\mathbf{x}} = (\mathbf{x}, 1)$ where $\mathbf{x} = [x_1, x_2, x_3]^T$ which is given in the coordinates of c . Suppose that $\tilde{\mathbf{x}}$ is in the laser plane in the laser plane $\tilde{\mathbf{u}} = (\mathbf{u}, u_4)$. Then, since $\tilde{\mathbf{x}}^T \tilde{\mathbf{u}} = 0$, it follows that $u_4 = -\mathbf{x} \cdot \mathbf{u}$. The point $\tilde{\mathbf{x}}$ will be imaged to the point

$$\mathbf{s} = \frac{\mathbf{x}}{x_3} \quad (611)$$

in the normalized image plane. This means that $\mathbf{x} = x_3 \mathbf{s}$. An expression for x_3 is found from

$$\mathbf{s} \cdot \mathbf{u} = \frac{\mathbf{x} \cdot \mathbf{u}}{x_3} = -\frac{u_4}{x_3} \quad (612)$$

which gives $x_3 = -u_4/(\mathbf{s} \cdot \mathbf{u})$. This means that the Euclidean point x can be found from

$$\mathbf{x} = -\left(\frac{u_4}{\mathbf{s} \cdot \mathbf{u}}\right) \mathbf{s} \quad (613)$$

12.3 The image of a line at the intersection of a plane and the laser plane

Next, consider an object plane in the scene given by $\tilde{\mathbf{v}} = (\mathbf{v}, v_4)$ in the coordinates of c and referenced to frame c . Then the line of intersection between the laser plane $\tilde{\mathbf{u}}$ and the object plane $\tilde{\mathbf{v}}$ is given in Plücker coordinates as

$$\mathbf{L} = (\mathbf{a}, \mathbf{m}) \quad (614)$$

where the direction vector \mathbf{a} and the moment vector \mathbf{m} are given by (508), which gives

$$\mathbf{a} = \mathbf{u} \times \mathbf{v} \quad (615)$$

$$\mathbf{m} = u_4 \mathbf{v} - v_4 \mathbf{u} \quad (616)$$

The image of the line in the normalized image plane is given by (602) as the 2D line

$$\ell = \mathbf{m} = u_4 \mathbf{v} - v_4 \mathbf{u} \quad (617)$$

12.4 Reconstruction of a 3D line from a laser scanner system

Suppose that a unknown 3D line $\mathbf{L} = (\mathbf{a}, \mathbf{m})$ is imaged to the 2D line ℓ by a laser scanner. The question is how the 3D line \mathbf{L} can be computed from the 2D line ℓ . It is recalled from (611) that the image of a point $\tilde{\mathbf{x}} = (\mathbf{x}, 1)$ on the laser plane is $\mathbf{s} = \mathbf{x}/x_3$. Then the line \mathbf{L} can be found by determining two image points \mathbf{s} and \mathbf{r} on the line ℓ . According to (613) these points correspond to the two Euclidean points

$$\mathbf{x} = -\left(\frac{u_4}{\mathbf{s} \cdot \mathbf{u}}\right) \mathbf{s}, \quad \mathbf{y} = -\left(\frac{u_4}{\mathbf{r} \cdot \mathbf{u}}\right) \mathbf{r} \quad (618)$$

on the line \mathbf{L} . This means that the two homogeneous points $\tilde{\mathbf{x}} = (\mathbf{x}, 1)$ and $\tilde{\mathbf{y}} = (\mathbf{y}, 1)$ will be on the line \mathbf{L} , and it follows from (446) that the 3D line is given by

$$\mathbf{L} = (\mathbf{y} - \mathbf{x}, \mathbf{x} \times \mathbf{y}) \quad (619)$$

12.5 Laser scanner image of plane with weld groove

Consider a setup where a plane with a weld groove is imaged with a laser scanner system. A laser line will be imaged by the camera at the line which is the intersection of the plane and the laser plane, or the planes of the groove and the laser plane. The system geometry is shown in Figure 55. The resulting image is shown in Figure 56.

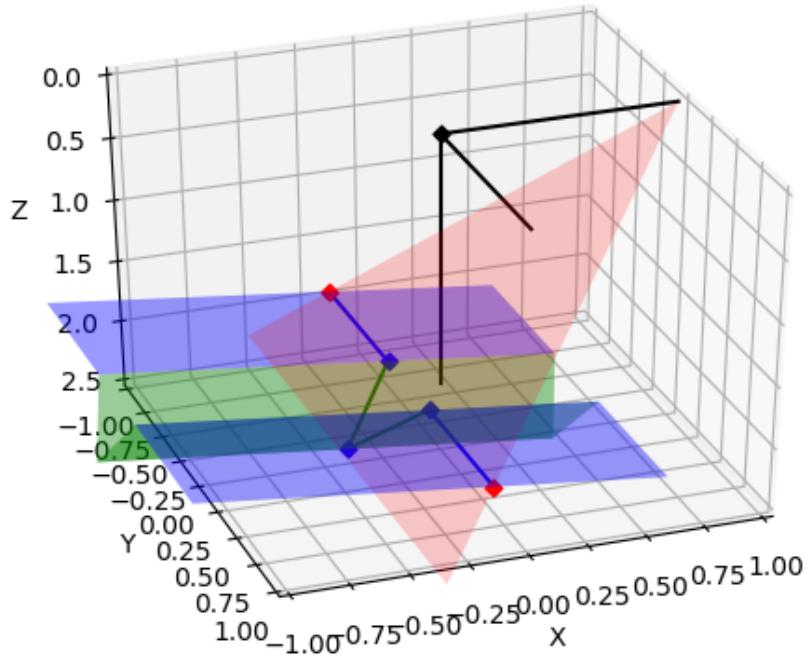


Figure 55: Geometry of laser scanner system where the laser plane (red) projects a line on the plane (blue) and the weld groove (green). The camera frame is shown with black axes with z_c pointing downwards.

```
% Script for plotting a laser scanner for detection of a weld groove

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
from matplotlib import colors as mcolors

def skewm(r):
    return np.array([[0,-r[2],r[1]], [r[2],0,-r[0]], [-r[1],r[0],0]])
def vex3(uh):
    return np.array([uh[2,1], uh[0,2], uh[1,0]])
def expso3(u):
    S = skewm(u); un = np.linalg.norm(u)
    return np.identity(3) + np.sinc(un/np.pi)*S \
        + 0.5*(np.sinc(un/(2*np.pi)))**2 * S@S
def R_and_t_to_T(R,t):
    return np.block([[R, t.reshape(3,1)], [0,0,0,1]])
```

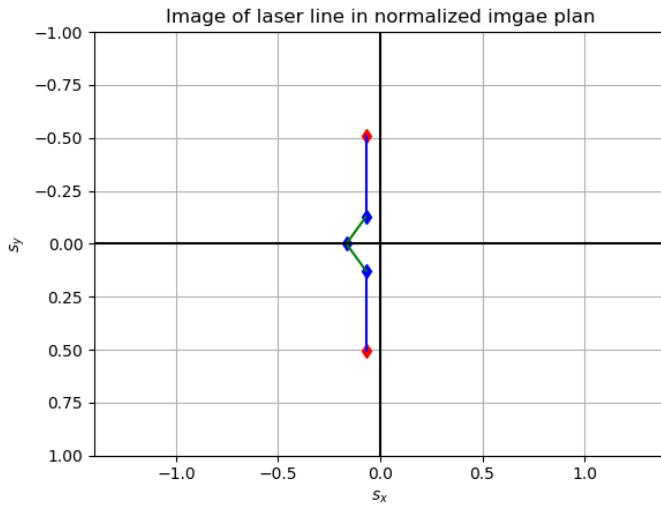


Figure 56: Resulting image in the normalized image plane. The image of the weld groove (green) is seen as a deviation from the straight line (blue) which is the image of the plane.

```

def cc(arg):
    return mcolors.to_rgba(arg, alpha=0.6)
def plot_points(X, **kwargs):
    if 'col' in kwargs:
        if 'mark' in kwargs:
            ax.scatter3D(X[0,:], X[1,:], X[2,:], c= kwargs['col'],
                          marker= kwargs['mark'])
        else:
            ax.scatter3D(X[0,:], X[1,:], X[2,:], c= kwargs['col'])
    else:
        ax.scatter3D(X[0,:], X[1,:], X[2,:])
    return
def plot_line(x, y, **kwargs):
    alpha = np.linspace(0, 1, 100)
    L = np.multiply(alpha, (x - y).reshape(3,1)) + y.reshape(3,1)
    if 'style' in kwargs:
        ax.plot(L[0,:], L[1,:], L[2,:], kwargs['style'])
    else:
        ax.plot(L[0,:], L[1,:], L[2,:])
    return
def plot_square(x, y, z, col, fill):
    verts = [list(zip(x, y, z))]
    poly = Poly3DCollection(verts, facecolors=cc(col))
    poly.set_alpha(fill)
    ax.add_collection3d(poly)
    return

```

```

def vectors_in_plane(u):
    un = np.linalg.norm(np.abs(u))
    im = np.argmin(u)
    E = np.array([[1,0,0], [0,1,0], [0,0,1]])
    e = E[:,0]
    a = e - u * np.dot(e,u)/un**2; a = a/np.linalg.norm(a)
    b = np.cross(u, a)
    return a, b
def plot_triangle(x, y, z, col, fill):
    verts = [list((x, y, z))]
    poly = Poly3DCollection(verts, facecolors=cc(col))
    poly.set_alpha(fill)
    ax.add_collection3d(poly)
    return

ex = np.array([1,0,0]); ey = np.array([0,1,0]); ez = np.array([0,0,1]);
zzz = np.zeros(3)
# Frame 1: Camera frame. Frame 2: Laser frame
R01 = expso3((0/180)*np.pi*ey); R02 = expso3((-30/180)*np.pi*ey);
t01 = -0.0*ex; t02 = ex;
# laser scanner
T01 = R_and_t_to_T(R01,t01); T02 = R_and_t_to_T(R02,t02)
T12 = np.linalg.inv(T01) @ T02
R12 = T12[0:3,0:3]; t12 = T12[0:3,3]
# Laser plane is in the yz plane of frame 2
u = R12[0:3,0]; u4 = - np.dot(u,t12)

# object plane
v = np.array([-0.2,0.,1]); v = v/np.linalg.norm(v); v4 = - np.dot(2.*ez,v)
avx, avy = vectors_in_plane(v)
qv = - v4 * v / np.linalg.norm(v)**2 # Point on v closest to origin
X_plane = np.block([[qv - avx - avy], [qv - avx + avy],
                     [qv + avx + avy], [qv + avx - avy]]).T

# Intersection of laser plane and object plane
La1 = np.cross(u,v); La1_n = np.linalg.norm(La1);
La1 = La1/La1_n; La1p = (u4*v - v4*u)/La1_n
qa1 = np.cross(La1,La1p) # point on line closest to origin
xa1 = qa1 - La1; ya1 = qa1 + La1 # End points of line for plotting

# Side lines and bottom line for the groove
dg = 0.25 # Size
pag = qv*(1 + 2*dg/np.linalg.norm(qv)) # Bottom point
pag1 = qv - dg * avy; pag2 = qv + dg * avy # Side points
Lag = avx; Lag1 = Lag; Lag2 = Lag
Lagp = np.cross(pag, Lag);

```

```

Lag1p = np.cross(pag1, Lag1);
Lag2p = np.cross(pag2, Lag2);
# Intersection of groove lines and laser plane
xag = (-u4*Lag + np.cross(u,Lagp)) / np.dot(u,Lag)
xag1 = (-u4*Lag1 + np.cross(u,Lag1p)) / np.dot(u,Lag1)
xag2 = (-u4*Lag2 + np.cross(u,Lag2p)) / np.dot(u,Lag2)

# Groove planes for plotting
X1_plane = np.block([[qv - avx - avy], [qv + avx - avy],
                     [pag1 + avx], [pag1 - avx]]).T
X2_plane = np.block([[qv - avx + avy], [qv + avx + avy],
                     [pag2 + avx], [pag2 - avx]]).T
Xg1_plane = np.block([[pag2 + avx], [pag2 - avx],
                     [pag - avx], [pag + avx]]).T
Xg2_plane = np.block([[pag1 + avx], [pag1 - avx],
                     [pag - avx], [pag + avx]]).T

# Points in normalized image plane
s1 = xa1/xa1[2]; r1 = ya1/ya1[2] # End points of line for plotting
sg = xag/xag[2]; sg1 = xag1/xag1[2]; sg2 = xag2/xag2[2] # Groove points

# Reconstructed points
x1 = - u4/(np.dot(s1,u))*s1; y1 = - u4/(np.dot(r1,u))*r1
xg = - u4/(np.dot(sg,u))*sg;
xg1 = - u4/(np.dot(sg1,u))*sg1; xg2 = - u4/(np.dot(sg2,u))*sg2;
L1 = y1 - x1; L1_n = np.linalg.norm(L1)
L1 = L1/L1_n; L1p = np.cross(x1,y1)/L1_n
# print('Point on line? \n', np.dot(y1,L1p), -L1p + np.cross(y1,L1), '\n\n')

fig = plt.figure(1)
fig.clear()
ax = fig.gca(projection='3d')
plot_square(X1_plane[0,:], X1_plane[1,:], X1_plane[2,:], 'b', 0.4)
plot_square(X2_plane[0,:], X2_plane[1,:], X2_plane[2,:], 'b', 0.4)
plot_square(Xg1_plane[0,:], Xg1_plane[1,:], Xg1_plane[2,:], 'g', 0.4)
plot_square(Xg2_plane[0,:], Xg2_plane[1,:], Xg2_plane[2,:], 'g', 0.4)
plot_triangle(t12, x1+0.25*(x1-t12), y1+0.25*(y1-t12), 'r', 0.2)
plot_line(x1, xg2, style ='-b')
plot_line(y1, xg1, style ='-b')
plot_line(xg1, xg, style=' -g')
plot_line(xg2, xg, style=' -g')
plot_line(zzz, 2*ez, style=' -k')
plot_line(zzz, ex, style=' -k')
plot_line(zzz, ey, style=' -k')
plot_points(zzz.reshape(3,1), col='k', mark='D')
plot_points(x1.reshape(3,1), col='r', mark='D')

```

```

plot_points(y1.reshape(3,1), col='r', mark='D')
plot_points(xg.reshape(3,1), col='b', mark='D')
plot_points(xg1.reshape(3,1), col='b', mark='D')
plot_points(xg2.reshape(3,1), col='b', mark='D')

ax.set_xlim(-1, 1)
ax.set_ylim(-1, 1)
ax.set_zlim(0, 2.5)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.show()

x_axis = np.block([[[-1.4,1.4], [0,0]]])
y_axis = np.block([[0,0], [-1,1]]);
alpha = np.linspace(0, 1, 100)
s_alpha = s1.reshape(3,1) + alpha*(sg2-s1).reshape(3,1)
r_alpha = r1.reshape(3,1) + alpha*(sg1-r1).reshape(3,1)
sg1_alpha = sg1.reshape(3,1) + alpha*(sg-sg1).reshape(3,1)
sg2_alpha = sg2.reshape(3,1) + alpha*(sg-sg2).reshape(3,1)

plt.figure(2)
plt.figure(2).clear()
plt.plot(s1[0], s1[1], 'rd')
plt.plot(r1[0], r1[1], 'rd')
plt.plot(sg[0], sg[1], 'bd')
plt.plot(sg1[0], sg1[1], 'bd')
plt.plot(sg2[0], sg2[1], 'bd')
plt.plot(s_alpha[0], s_alpha[1], 'b')
plt.plot(r_alpha[0], r_alpha[1], 'b')
plt.plot(sg1_alpha[0], sg1_alpha[1], 'g')
plt.plot(sg2_alpha[0], sg2_alpha[1], 'g')
plt.title('Image of laser line in normalized imgae plan')
plt.plot(x_axis[0,:], x_axis[1,:], 'k', y_axis[0,:], y_axis[1,:], 'k')
plt.axis([-1.4, 1.4, 1, -1])
plt.xlabel('$s_x$')
plt.ylabel('$s_y$')
plt.grid()
plt.show()

```

13 Homographies in 3D*

13.1 Transformation of points

A homography \mathbf{H} in 3-dimensional projective space is a nonsingular 4×4 matrix. The transformation is

$$\mathbf{x}' = \mathbf{H}\mathbf{x} \quad (620)$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = x_4 \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \quad \mathbf{x}' = \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \\ x'_4 \end{bmatrix} = x'_4 \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \quad (621)$$

are the homogeneous vectors. The homogenous vector \mathbf{x} has the scaling factor x_4 , and corresponds to the point $\mathbf{p} = (x, y, z)^T$ in Euclidean space for all scaling factors $x_4 \neq 0$. In the same way the homogeneous vector \mathbf{x}' with scaling factor x'_4 corresponds to the point $\mathbf{p}' = (x', y', z')^T$ in Euclidean space for all scaling factors $x'_4 \neq 0$.

A homography transforms one homogeneous vector to another homogeneous vector. This means that the homographies \mathbf{H} and $\alpha\mathbf{H}$ where $\alpha \neq 0$ will be equivalent transformations in the sense that the resulting vectors $\mathbf{x}' = \mathbf{H}\mathbf{x}$ and $\mathbf{x}'_\alpha = \alpha\mathbf{H}\mathbf{x}$ will represent the same Euclidean point. The matrix \mathbf{H} of a homography will have 16 elements. Because an arbitrary scaling of the matrix will give an equivalent homography, it follows that there are 15 independent elements, as one of the elements can be scaled, e.g., to unity.

13.2 Transformation of planes

A plane is described by a vector $\boldsymbol{\pi} = [a, b, c, d]^T$ in the homogeneous description of 3-dimensional space. A point \mathbf{x} is on the plane $\boldsymbol{\pi}$ if $\mathbf{x}^T \boldsymbol{\pi} = 0$. In a transformation where points are transformed by $\mathbf{x}' = \mathbf{H}\mathbf{x}$, the transformed point will be on the transformed plane $\boldsymbol{\pi}'$. This means that $\mathbf{x}'^T \boldsymbol{\pi}' = 0$, and it follows that $\mathbf{x}^T \mathbf{H}^T \boldsymbol{\pi}' = 0$. This gives $\mathbf{H}^T \boldsymbol{\pi}' = \boldsymbol{\pi}$, and it is concluded that the plane is transformed as

$$\boldsymbol{\pi}' = \mathbf{H}^{-T} \boldsymbol{\pi} \quad (622)$$

13.3 Types of homographies

As in 2D, homographies in 3D can be classified as Euclidean, similarity, affine or projective.

A Euclidean homography is written

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (623)$$

where $\mathbf{R} \in O(3)$ is a orthogonal matrix, and $\mathbf{t} \in \mathbb{R}^3$ is a translation vector. Here $O(3) = \{\mathbf{R} : \mathbf{R}^T \mathbf{R} = \mathbf{I}\}$ is the orthogonal group, which includes the rotation matrices $\mathbf{R} \in SO(3)$ that satisfies $\det \mathbf{R} = 1$, and the reflection matrices that satisfy $\det \mathbf{R} = -1$.

A similarity homography is written

$$\mathbf{H} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (624)$$

where $\mathbf{R} \in O(3)$ is a orthogonal matrix, $s \in \mathbb{R}$ is a scaling factor, and $\mathbf{t} \in \mathbb{R}^3$ is a translation vector.

An affine homography is of the form

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (625)$$

where $\mathbf{A} \in \mathbb{R}^{3 \times 3}$ is a nonsingular matrix. It is seen that a similarity homography is a special case of an affine homography.

A projective homography, which is the most general form, is written

$$\mathbf{H}_p = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v_4 \end{bmatrix} \quad (626)$$

where $\mathbf{A} \in \mathbb{R}^{3 \times 3}$ is a nonsingular matrix.

13.4 Affine transformations

Affine transformations are given by

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (627)$$

or

$$\mathbf{x}' = \mathbf{H}_a \mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x} \quad (628)$$

where \mathbf{A} is any nonsingular 3×3 matrix.

The singular value description of the matrix \mathbf{A} is

$$\mathbf{A} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \sigma_3 \mathbf{u}_3 \mathbf{v}_3^T \quad (629)$$

where $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3]$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$ are orthogonal 3×3 matrices and $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \sigma_3)$ is a diagonal matrix with the singular values $\sigma_1 \geq \sigma_2 \geq \sigma_3 > 0$ where the singular values are positive because the \mathbf{A} is nonsingular.

As in 2-dimensional geometry, matrix \mathbf{A} can be analyzed in terms of its polar decomposition

$$\mathbf{A} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T = (\mathbf{U} \mathbf{V}^T) (\mathbf{V} \boldsymbol{\Sigma} \mathbf{V}^T) = \mathbf{R} \mathbf{P} \quad (630)$$

where $\mathbf{R} = \mathbf{U} \mathbf{V}^T \in O(3)$ is an orthogonal matrix, and $\mathbf{P} = \mathbf{V} \boldsymbol{\Sigma} \mathbf{V}^T$ is a positive definite matrix. The orthogonal matrix \mathbf{R} is a rotation matrix when $\det \mathbf{R} = 1$, or a reflection matrix when $\det \mathbf{R} = -1$. The positive definite matrix \mathbf{P} is a matrix that stretches the space with the factors σ_1, σ_2 and σ_3 along 3 orthogonal directions defined by the input orthogonal matrix \mathbf{V} . The orthogonal matrix \mathbf{V} is either a rotation matrix with determinant +1 or a reflection matrix with determinant -1.

This shows that an affine transformation given by the matrix \mathbf{A} can be decomposed as a stretching with individual scale factors σ_1, σ_2 and σ_3 along the orthogonal axes $\mathbf{v}_1, \mathbf{v}_2$ and \mathbf{v}_3 , followed by a rotation or reflection \mathbf{R} .

13.5 Affine transformation of points at infinity and planes at infinity

The inverse of the affine transformation is

$$\mathbf{H}_a^{-1} = \begin{bmatrix} \mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (631)$$

Affine transformations has as invariants parallelism, the ratio of areas, ratio of lengths for parallel lines, ratio of lengths and midpoints on a line, and centroids of vectors.

An affine transformation will transform a point $\mathbf{x}_\infty = (\mathbf{r}, 0)$ at infinity according to

$$\mathbf{H}_a \mathbf{x}_\infty = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{A}\mathbf{r} \\ 0 \end{bmatrix} = \mathbf{x}'_\infty \quad (632)$$

where \mathbf{x}'_∞ is a point at infinity in the direction $\mathbf{A}\mathbf{r}$. This means that an affine transformation will map a point at infinity to a point at infinity.

The plane at infinity $\boldsymbol{\pi}_\infty = [0, 0, 0, 1]^T$ transforms according to

$$\mathbf{H}_a^{-T} \boldsymbol{\pi}_\infty = \begin{bmatrix} \mathbf{A}^{-T} & \mathbf{0} \\ -\mathbf{t}^T \mathbf{A}^{-T} & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \boldsymbol{\pi}_\infty \quad (633)$$

This means that the the plane at infinity remains the plane at infinity under an affine mapping.

13.6 Projective transformations

Finally, the fourth class of transformations is the projective transformation

$$\mathbf{x}' = \mathbf{H}_p \mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v_4 \end{bmatrix} \mathbf{x} \quad (634)$$

The inverse of the projective transformation is

$$\mathbf{H}_p^{-1} = \begin{bmatrix} \mathbf{B}^{-1} & -\mathbf{B}^{-1}\mathbf{t}/v_4 \\ -\mathbf{v}^T \mathbf{B}^{-1}/v_4 & \mathbf{v}^T \mathbf{B}^{-1}\mathbf{t}/v_4^2 + 1/v_4 \end{bmatrix} \quad (635)$$

where $\mathbf{B} = \mathbf{A} - \mathbf{t}\mathbf{v}^T/v_4$ which is verified by direct computation.

The invariants of a projective transformation include collinearity of points, intersection of lines, tangency, tangent discontinuities and cross ratios.

In the case of the general projective transformation \mathbf{H}_p , the point at infinity is mapped to a homogeneous point

$$\mathbf{x}' = \mathbf{H}_p \mathbf{x}_\infty = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v_4 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{A}\mathbf{p} \\ \mathbf{v}^T \mathbf{p} \end{bmatrix} \quad (636)$$

which is not at infinity when $\mathbf{v}^T \mathbf{p} \neq 0$.

Example

The plane at infinity π_∞ is transformed by the general projective mapping \mathbf{H}_p according to $\pi' = \mathbf{H}_p^{-T} \pi_\infty$. If

$$\mathbf{H}_p = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_4 \end{bmatrix}, \quad (637)$$

then the transformation of the plane at infinity is

$$\pi' = \begin{bmatrix} \mathbf{I} & -\mathbf{v}/v_4 \\ \mathbf{0}^T & 1/v_4 \end{bmatrix}^T \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \frac{1}{v_4} \begin{bmatrix} -\mathbf{v} \\ 1 \end{bmatrix} \quad (638)$$

It is seen that this line is not necessarily at infinity, and that \mathbf{v} is the normal vector to the plane. \square

13.7 Decomposition of a projective homography

A general projective homography \mathbf{H}_p

$$\mathbf{H}_p = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v_4 \end{bmatrix} \quad (639)$$

can always be decomposed as

$$\mathbf{H}_p = \mathbf{H}_s \mathbf{H}_{as} \mathbf{H}_{ps} = \begin{bmatrix} s\mathbf{R} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_4 \end{bmatrix} \quad (640)$$

which is a sequence of three transformations starting with the projective transformation

$$\mathbf{H}_{ps} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_4 \end{bmatrix}, \quad (641)$$

followed by the affine transformation

$$\mathbf{H}_{as} = \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (642)$$

where \mathbf{K} is upper diagonal with $\det \mathbf{K} = 1$, and the similarity transformation

$$\mathbf{H}_s = \begin{bmatrix} s\mathbf{R} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (643)$$

13.8 Affine reconstruction

Suppose that a 3D point set has been transformed with the projective transformation

$$\mathbf{x}' = \mathbf{H}_p \mathbf{x} \quad (644)$$

where \mathbf{H}_p is unknown. The plane at infinity π_∞ is transformed by the projective transformation to the plane

$$\pi' = \mathbf{H}_p^{-T} \pi_\infty \quad (645)$$

Suppose that the plane at infinity is identified in the transformed image as $\boldsymbol{\pi}' = [a, b, c, d]^T$. This can be done, e.g., by identifying 3 points at infinity in the transformed image and then calculating the plane at infinity from these 3 points. Then the transformation

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ a & b & c & d \end{bmatrix} \quad (646)$$

will make the composite transformation $\mathbf{x}'' = \mathbf{H}\mathbf{H}_p\mathbf{x}$ affine.

This is verified by noting that the transformation of the plane is $\boldsymbol{\pi}'' = \mathbf{H}^{-T}\boldsymbol{\pi}'$, which is equivalent to $\boldsymbol{\pi}' = \mathbf{H}^T\boldsymbol{\pi}''$. Then $\boldsymbol{\pi}'' = \boldsymbol{\pi}_\infty$ will correspond to $\boldsymbol{\pi}'$, which is seen from

$$\mathbf{H}^T\boldsymbol{\pi}'' = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & d \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \boldsymbol{\pi}' \quad (647)$$

This means that $\mathbf{H}\mathbf{H}_p\boldsymbol{\pi}_\infty = \boldsymbol{\pi}_\infty$ which means that the composite transformation $\mathbf{H}\mathbf{H}_p$ transforms the plane at infinity to the plane at infinity, and it can be concluded that $\mathbf{H}\mathbf{H}_p$ is an affine transformation. On background of this, the transformation \mathbf{H} is called an affine reconstruction.

13.9 Projection in terms of a homography

The projective homography

$$\mathbf{H}_{ps} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_4 \end{bmatrix}, \quad (648)$$

is characterized by the vector \mathbf{v} and the scalar v_4 . Let the plane $\boldsymbol{\pi}$ be defined by

$$\boldsymbol{\pi} = \begin{bmatrix} \mathbf{v} \\ v_4 \end{bmatrix} \quad (649)$$

This plane has normal vector \mathbf{v} , and is located at a distance $\delta_\pi = -v_4/|\mathbf{v}|$ from the origin. A homogeneous point $\mathbf{x} = (\mathbf{p}^T, 1)^T$ will be at a distance

$$\delta(\mathbf{p}) = \frac{\boldsymbol{\pi}^T \mathbf{x}}{|\mathbf{v}|} = \frac{\mathbf{v}^T \mathbf{p} + v_4}{|\mathbf{v}|} \quad (650)$$

from the plane $\boldsymbol{\pi}$.

The projective homography gives

$$\mathbf{x}' = \mathbf{H}_{ps}\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{v}^T \mathbf{p} + v_4 \end{bmatrix} = (\mathbf{v}^T \mathbf{p} + v_4) \begin{bmatrix} \frac{\mathbf{p}}{\mathbf{v}^T \mathbf{p} + v_4} \\ 1 \end{bmatrix} \quad (651)$$

This means that \mathbf{x}' corresponds to the point

$$\mathbf{p}' = \frac{\mathbf{p}}{\delta(\mathbf{p})|\mathbf{v}|} \quad (652)$$

which in coordinates gives

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \frac{1}{\delta(\mathbf{p})|\mathbf{v}|} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (653)$$

which shows that the effect of the transformation is that the coordinates of \mathbf{p} are divided by $\delta(\mathbf{p})|\mathbf{v}|$, which is the distance $\delta(\mathbf{p})$ from the point to the plane multiplied with the magnitude of the normal vector \mathbf{v} .

13.10 Orthographic projections

An orthographic projection of a homogeneous point \mathbf{x} to a plane is done by projecting the point orthogonally to the plane. This transformation from a 3D point to a 2D point in a plane is not invertible, which means that the orthogonal projection is not a homography.

Consider a plane $\boldsymbol{\pi} = (\mathbf{n}^T, 0)^T$ through the origin with unit normal vector \mathbf{n} and a homogeneous point $\mathbf{x} = (\mathbf{p}^T, 1)^T$. Then the component of the position vector \mathbf{p} along the unit normal is $\mathbf{p}^{\parallel} = (\mathbf{n}^T \mathbf{p})\mathbf{n}$, and the projection of the point into the plane is

$$\mathbf{p}_{\boldsymbol{\pi}} = \mathbf{p} - \mathbf{p}^{\parallel} = \mathbf{p} - \mathbf{n}(\mathbf{n}^T \mathbf{p}) = (\mathbf{I} - \mathbf{n}\mathbf{n}^T)\mathbf{p} \quad (654)$$

In homogeneous coordinates this is written

$$\mathbf{x}_{\boldsymbol{\pi}} = \mathbf{x} - \boldsymbol{\pi}(\boldsymbol{\pi}^T \mathbf{x}) = (\mathbf{I} - \boldsymbol{\pi}\boldsymbol{\pi}^T)\mathbf{x} = \mathbf{N}\mathbf{x} \quad (655)$$

where the orthographic transformation matrix to the plane $\boldsymbol{\pi}$ through the origin is given by 4×4 matrix

$$\mathbf{N} = \mathbf{I} - \boldsymbol{\pi}\boldsymbol{\pi}^T \quad (656)$$

Example:

The orthographic projection of the point $\mathbf{x} = (x, y, z, 1)^T$ to the plane $z = 0$, with homogeneous description $\boldsymbol{\pi} = (0, 0, 1, 0)^T$ is done with the transformation

$$\begin{bmatrix} x' \\ y' \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (657)$$

It is seen that $x' = x$ and $y' = y$. Note that this transformation is not a homography as the matrix of the transformation is singular. This orthographic projection can also be written with the result as a homogeneous vector $(x', y', 1)^T$ in the plane as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (658)$$

□

Example: A special type of orthographic projection is used in the camera model where a homogeneous point $(x, y, z, 1)^T$ is projected to the nonhomogeneous part $(x, y, z)^T$ by

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (659)$$

where $x' = x$, $y' = y$ and $z' = z$. In the camera model, the resulting point $(x', y', z')^T$ is then treated as a homogeneous point in the plane, and the point is scaled according to

$$\begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} = \frac{1}{z'} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{bmatrix} \quad (660)$$

This is the normalized image coordinates in the camera model. \square

13.11 The camera projection*

Consider the projective homography $\mathbf{x}' = \mathbf{H}\mathbf{x}$ where

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z+w \end{bmatrix} = (z+w) \begin{bmatrix} \frac{x}{z+w} \\ \frac{y}{z+w} \\ \frac{z}{z+w} \\ 1 \end{bmatrix} \quad (661)$$

An orthographic projection can be used to map this to the xy plane using

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = (z+w) \begin{bmatrix} \frac{x}{z+w} \\ \frac{y}{z+w} \\ 1 \end{bmatrix} \quad (662)$$

To get the usual camera projection a slight modification must be done to the transformation, which makes it singular, and therefore not a homography:

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z \end{bmatrix} = z \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ \frac{z}{z} \\ 1 \end{bmatrix} \quad (663)$$

An orthographic projection can be used to map this to the xy plane using

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = z \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ \frac{z}{z} \\ 1 \end{bmatrix} \quad (664)$$

This is the camera projection.

13.12 The absolute conic

The absolute conic \mathbf{C}_∞ is a conic on the plane π_∞ at infinity. The conic is given by

$$\mathbf{x}_\infty^T \mathbf{C}_\infty \mathbf{x}_\infty = 0, \quad x_4 = 0 \quad (665)$$

where $\mathbf{x}_\infty = (x_1, x_2, x_3, 0)^T$ is a point at infinity and

$$\mathbf{C}_\infty = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \quad (666)$$

where \mathbf{I} is the identity matrix. It is noted that the points on the absolute conic satisfies

$$x_1^2 + x_2^2 + x_3^2 = 0 \quad (667)$$

$$x_4 = 0 \quad (668)$$

which means that all points on the absolute conic are imaginary points at infinity.

Example

Let $\mathbf{a}_1, \mathbf{a}_2$ and \mathbf{a}_3 be orthogonal unit vectors along the axes of a right-hand coordinate frame. These vectors can be regarded as direction vectors. Let the homogeneous form of the direction vector \mathbf{a}_i be $\mathbf{z}_i = [\mathbf{a}_i^T, 0]^T$, which is a point at infinity in the direction of \mathbf{a}_i . Then $\mathbf{a}_i^T \mathbf{a}_j = \mathbf{z}_i^T \mathbf{C}_\infty \mathbf{z}_j$. This means that

$$\mathbf{z}_i^T \mathbf{C}_\infty \mathbf{z}_j = \mathbf{a}_i^T \mathbf{a}_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \quad (669)$$

14 Quadrics*

14.1 Introduction

A quadric is given by a symmetric matrix $\mathbf{Q} \in \mathbb{R}^{4 \times 4}$, and the points \mathbf{x} on the quadric satisfies the quadratic equation

$$\mathbf{x}^T \mathbf{Q} \mathbf{x} = 0 \quad (670)$$

A quadric can also be written $g(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x} = 0$ where

$$g(\mathbf{x}) = \mathbf{a}(\mathbf{x})^T \mathbf{q} \quad (671)$$

where \mathbf{q} is the vector of the coefficients of \mathbf{Q} . The gradient is given by

$$\frac{\partial}{\partial \mathbf{x}} = \nabla g(\mathbf{x}) = (\nabla \mathbf{a}(\mathbf{x}))^T \mathbf{q} \quad (672)$$

14.2 Dual quadric

A plane π is tangent to the quadric at a point \mathbf{x} on the quadric if $\pi = \mathbf{Q} \mathbf{x}$, which is seen from $\mathbf{x}^T \pi = \mathbf{x}^T \mathbf{Q} \mathbf{x} = 0$. The dual quadric \mathbf{Q}^* gives the condition

$$\pi^T \mathbf{Q}^* \pi = 0 \quad (673)$$

for the planes that are tangent to the quadric \mathbf{Q} .

If the quadric \mathbf{Q} is nonsingular, then the point where $\boldsymbol{\pi}$ is tangent is given by $\mathbf{x} = \mathbf{Q}^{-1}\boldsymbol{\pi}$. This is a point on the quadric, and therefore

$$0 = \mathbf{x}^T \mathbf{Q} \mathbf{x} = \boldsymbol{\pi}^T \mathbf{Q}^{-T} \mathbf{Q} \mathbf{Q}^{-1} \boldsymbol{\pi} = \boldsymbol{\pi}^T \mathbf{Q}^{-1} \boldsymbol{\pi} \quad (674)$$

where it is used that \mathbf{Q} is symmetric. This means that if the conic matrix \mathbf{Q} is nonsingular, then $\mathbf{Q}^* = \mathbf{Q}^{-1}$.

14.3 Transformation of quadrics

Consider the homography $\mathbf{x}' = \mathbf{H}\mathbf{x}$, and suppose that \mathbf{x} is on the quadric $\mathbf{x}^T \mathbf{Q} \mathbf{x} = 0$. Then the transformed point \mathbf{x}' will be on the transformed quadric $\mathbf{x}'^T \mathbf{Q}' \mathbf{x}' = 0$ where

$$\mathbf{Q}' = \mathbf{H}^{-T} \mathbf{Q} \mathbf{H}^{-1} \quad (675)$$

This is seen from

$$\mathbf{x}'^T (\mathbf{H}^{-T} \mathbf{Q} \mathbf{H}^{-1}) \mathbf{x}' = (\mathbf{x}^T \mathbf{H}^T) (\mathbf{H}^{-T} \mathbf{Q} \mathbf{H}^{-1}) (\mathbf{H}\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x} = 0 \quad (676)$$

Suppose that the plane $\boldsymbol{\pi}$ is tangent to the quadric \mathbf{Q} , which means that $\boldsymbol{\pi}^T \mathbf{Q}^* \boldsymbol{\pi} = 0$, where \mathbf{Q}^* is the dual quadric. Then the transformed plane $\boldsymbol{\pi}' = \mathbf{H}^{-T} \boldsymbol{\pi}$ will be on the transformed dual quadric

$$\mathbf{Q}^* = \mathbf{H} \mathbf{Q}^* \mathbf{H}^T \quad (677)$$

This is seen from

$$\boldsymbol{\pi}'^T (\mathbf{H} \mathbf{Q} \mathbf{H}^T) \boldsymbol{\pi}' = (\boldsymbol{\pi}^T \mathbf{H}^{-1}) (\mathbf{H} \mathbf{Q} \mathbf{H}^T) (\mathbf{H}^{-T} \mathbf{x}) = \boldsymbol{\pi}^T \mathbf{Q}^* \boldsymbol{\pi} = 0 \quad (678)$$

15 Recovery of affine and metric properties

A homography $\mathbf{x}' = \mathbf{H}_p \mathbf{x}$ in 2D can be calculated from 4 point mappings, which was demonstrated in previous sections. A transformed image \mathbf{x}' can then be fully recovered as $\mathbf{x} = \mathbf{H}_p^{-1} \mathbf{x}'$. It has been shown that a projective homography can be decomposed into a similarity transform, an affine transform and a projective transform. In the case of full recovery, then the recovery includes the similarity, affine and projective effects of the homography. In the case where point mappings are not available, it is still possible to achieve partial recovery of the original image. One possibility is to identify the transformed line at infinity $\ell' = \mathbf{H}_p^{-T} \ell_\infty$ and then use this to find a transformation $\mathbf{x}'' = \mathbf{H}\mathbf{x}'$ that will recover the affine properties of the original image \mathbf{x} . The recovery of the affine properties means that lines that are parallel in the original image \mathbf{x} will be parallel in the recovered image \mathbf{x}'' , and that the line at infinity is transformed back to infinity. Another possibility is to use additional data like lines that intersect at a right angle in the original image, and use this to recover metric properties in the image in addition to recovering the affine properties. This can be done based on the concept of conic sections and the two circular points, which are complex points at infinity, as presented in [30]. In this section a simpler derivation is used based on line transformations and the transformation of the scalar product of the normal vectors of lines that intersect at a right angle. Conics and circular points are presented in a later section for completeness, but this is not required to

understand the method. The fact that metric recovery is based on finding a transformation $\mathbf{x}'' = \mathbf{H}\mathbf{x}'$ so that lines that intersect at a right angle in \mathbf{x} will intersect at a right angle in \mathbf{x}'' means that the similarity part of \mathbf{H}_p cannot be found, because lines that intersect at a right angle will still intersect at a right angle when the lines have been transformed by a similarity transform, which is a rotation, translation and a scaling. The recovered image will therefore differ from the original image by some similarity transform.

15.1 Recovering affine properties using the line at infinity

Suppose the points \mathbf{x} of an image are transformed with a projective transformation $\mathbf{x}' = \mathbf{H}_p\mathbf{x}$, and that the line ℓ_∞ at infinity is transformed to a line $\ell' = \mathbf{H}_p^{-T}\ell_\infty$ that is not at infinity. The question to be addressed in this section is how find a transform $\mathbf{x}'' = \mathbf{H}\mathbf{x}'$ so that the composite transform $\mathbf{x}'' = \mathbf{H}\mathbf{H}_p\mathbf{x}$ is affine, that is, how can the homography \mathbf{H} be found so that $\mathbf{H}\mathbf{H}_p$ is affine. This will be done by finding a homography that will transform the line ℓ' back to a line at infinity.

It is clear that if the transformation \mathbf{H}_p is known, then the transformation can be inverted, and the original image can be recovered with $\mathbf{x} = \mathbf{H}_p^{-1}\mathbf{x}'$, and the line ℓ' is transformed back again as $\ell_\infty = \mathbf{H}_p^T\ell'$.

Consider the case where \mathbf{H}_p is unknown. Suppose that the vanishing line ℓ' is found in the transformed image as $\ell' = [a, b, c]^T$. Then it is possible to recover the affine properties of the original image with a transformation that will transform the line ℓ' to the line at infinity. This can be done with the projective transformation

$$\mathbf{x}'' = \mathbf{H}\mathbf{x}' \quad (679)$$

where

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a & b & c \end{bmatrix} \quad (680)$$

This is verified by noting that the line transformation is $\ell'' = \mathbf{H}^{-T}\ell'$, which is equivalent to $\ell'' = \mathbf{H}^T\ell''$. If $\ell'' = \ell_\infty = [0, 0, 1]^T$, then it follows that

$$\mathbf{H}^T\ell'' = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \ell' \quad (681)$$

which shows that \mathbf{H} will transform ℓ' back to the line at infinity.

The line at infinity can be found in the transformed image if two vanishing points \mathbf{z}'_1 and \mathbf{z}'_2 at infinity can be determined in the transformed image. Then the vanishing line is found to be

$$\ell'_v = \mathbf{z}'_1 \times \mathbf{z}'_2 \quad (682)$$

One method for determining a point at infinity is based on the result that two parallel lines intersect at a point of infinity in the direction of the lines. This means that the point at infinity in the transformed image can be found as the intersection of two parallel lines in the transformed image.

To be specific, consider two lines $\ell_1 = (\mathbf{n}_1, c_1)$ and $\ell_2 = (\mathbf{n}_1, c_2)$ that are parallel with direction vector \mathbf{a}_1 that is normal to \mathbf{n}_1 . These lines intersect at the point $\mathbf{z}_1 = (\mathbf{a}_1, 0)$ at infinity. The lines are transformed to the lines

$$\ell'_1 = \mathbf{H}_p^{-T} \ell_1, \quad \ell'_2 = \mathbf{H}_p^{-T} \ell_2 \quad (683)$$

Suppose that ℓ'_1 and ℓ'_2 can be determined in the transformed image by some image processing technique. Then the intersection of the two lines will be

$$\mathbf{z}'_1 = \ell'_1 \times \ell'_2 \quad (684)$$

It is straightforward to verify that this point is $\mathbf{z}'_1 = \mathbf{H}_p \mathbf{z}_1$, which means that the transformed point at infinity has been found in the transformed image.

Suppose that a second pair of lines ℓ'_3 and ℓ'_4 can be determined, where ℓ'_3 and ℓ'_4 correspond to the parallel lines $\ell_3 = (\mathbf{n}_2, c_3)$ and $\ell_4 = (\mathbf{n}_2, c_4)$, which intersect at the point $\mathbf{z}_2 = (\mathbf{a}_2, 0)$ at infinity, where $\mathbf{a}_2^T \mathbf{n}_2 = 0$. Then a second vanishing point can be calculated to be at

$$\mathbf{z}'_2 = \ell'_3 \times \ell'_4 \quad (685)$$

and the vanishing line can then be found as the line through \mathbf{z}'_1 and \mathbf{z}'_2 given by $\ell'_v = \mathbf{z}'_1 \times \mathbf{z}'_2$. This is shown in Figure 57.

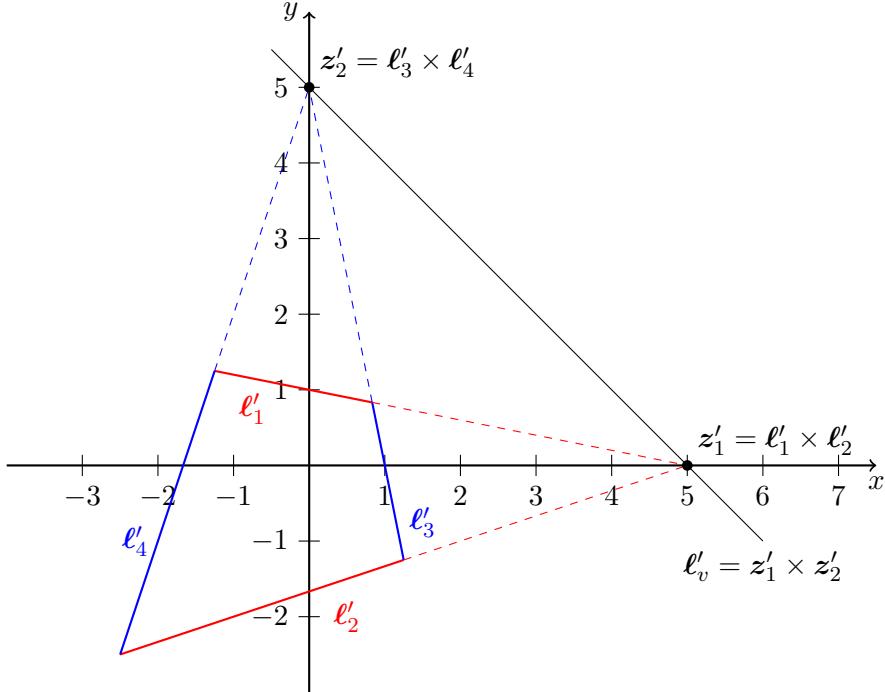


Figure 57: The vanishing ℓ'_v line is found from the two vanishing points \mathbf{z}'_1 and \mathbf{z}'_2 . Each vanishing point is found as the intersection of two parallel lines.

Example 2

Suppose that the two points $\mathbf{x}'_1 = [0, 1, 1]^T$ and $\mathbf{x}'_2 = [1, 1, 1]^T$ have been identified as points at infinity. Then the line at infinity is at

$$\ell' = \mathbf{x}'_1 \times \mathbf{x}'_2 = [0, 1, -1]^T \quad (686)$$

which is the line $y = 1$. The transformation \mathbf{H} that will recover the affine properties is then obtained by letting the vanishing line be the last row of the homography, which gives

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \quad (687)$$

It is seen from

$$\mathbf{x}_1'' = \mathbf{H}\mathbf{x}_1' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad (688)$$

$$\mathbf{x}_2'' = \mathbf{H}\mathbf{x}_2' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad (689)$$

that this transforms the two points at the line to points at infinity, where the direction of each point is not changed by the transformation. \square

Example

```
% Input: Transformed points corresponding to corners of a square
x1 = [0;0.3;1]; x2 = [0.2;-1;1]; x3 = [1;-0.3;1]; x4 = [0.5;0.4;1];
% x1 = [0;1;1]; x2 = [0;-1;1]; x3 = [1;-0.5;1]; x4 = [0.5;0.75;1];
figure(1);clf;
plot([x1(1) x2(1)], [x1(2),x2(2)], 'b', ...
      [x2(1) x3(1)], [x2(2) x3(2)], 'r', ...
      [x3(1) x4(1)], [x3(2) x4(2)], 'k', ...
      [x4(1) x1(1)], [x4(2) x1(2)], 'm');
grid
axis([-0.5 2.5 -1.5 2.5]); hold on;
% Calculate points at infinity from parallel lines.
% Parallel L1&L2 intersect at xi1, Parallel L3&L4 intersect at xi2
L1 = cross(x1,x4); L2 = cross(x2,x3); xi1 = cross(L1,L2); xi1 = xi1/xi1(3);
L3 = cross(x2,x1); L4 = cross(x3,x4); xi2 = cross(L3,L4); xi2 = xi2/xi2(3);
plot(xi1(1),xi1(2), 'db', xi2(1),xi2(2), 'dk')
plot([x1(1) xi2(1)], [x1(2) xi2(2)], ':k', ...
      [x4(1) xi2(1)], [x4(2) xi2(2)], ':k', ...
      [x3(1) xi1(1)], [x3(2) xi1(2)], ':b', ...
      [x4(1) xi1(1)], [x4(2) xi1(2)], ':b', ...
      [xi1(1) xi2(1)], [xi1(2) xi2(2)], '--k')
hold off;
% Calculate line at infinity
Li = cross(xi1,xi2); Li = Li/(-Li(2));
% Result: Projective transform that moves Li to infinity
H = [1 0 0; 0 1 0; Li'];
% Affine recovery of points
x = H*x1; xa1 = x/x(3);
```

```

x = H*x2; xa2 = x/x(3);
x = H*x3; xa3 = x/x(3);
x = H*x4; xa4 = x/x(3);

figure(2);clf;
plot([xa1(1) xa2(1)], [xa1(2),xa2(2)], 'b', ...
[xa2(1) xa3(1)], [xa2(2) xa3(2)], 'r', ...
[xa3(1) xa4(1)], [xa3(2) xa4(2)], 'k', ...
[xa4(1) xa1(1)], [xa4(2) xa1(2)], 'm');
grid
axis([-0.5 2.5 -1.5 2.5]); hold on;

```

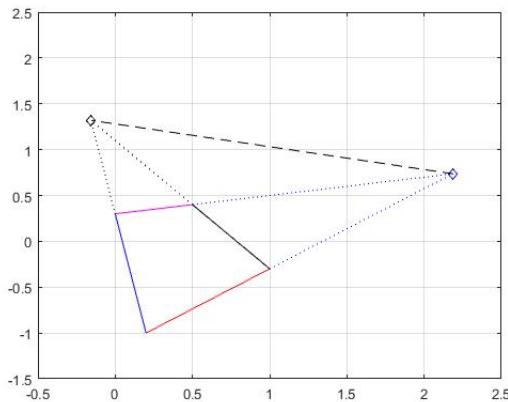


Figure 58: Projected square. Note the points at infinity marked by blue diamonds which are found as the intersection of the projected parallel lines. Also note the projected line at infinity that goes through the two points at infinity.

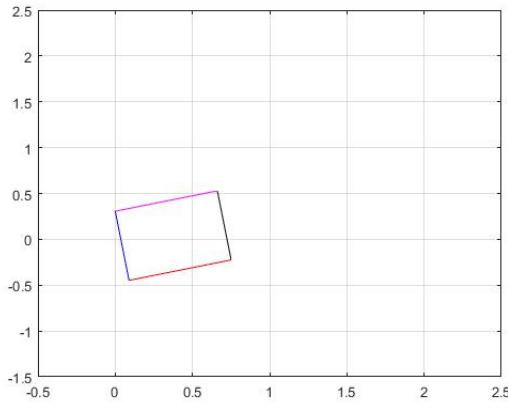


Figure 59: Square after affine recovery. Parallel lines are parallel, and points at infinity and the line at infinity has been moved to infinity.

□

15.2 Recovery of metric properties

Consider a projective homography $\mathbf{x}' = \mathbf{H}_p \mathbf{x}$ where the homography is described as an affine transformation \mathbf{H}_{as} followed by a projective transformation \mathbf{H}_{ps} , where

$$\mathbf{H}_{ps} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_3 \end{bmatrix}, \quad \mathbf{H}_{as} = \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (690)$$

The homography is then written

$$\mathbf{H}_p = \mathbf{H}_{ps} \mathbf{H}_{as} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_3 \end{bmatrix} \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{v}^T \mathbf{K} & v_3 \end{bmatrix} \quad (691)$$

It is noted that in the general case, the decomposition should also include a similarity transformation \mathbf{H}_s . Then the transformation would be $\mathbf{H}_{ps} \mathbf{H}_{as} \mathbf{H}_s$. However, the method that is presented in this section will not be able to find the similarity component of the homography. Therefore, the similarity part is left out in the development.

The lines $\boldsymbol{\ell} = (\mathbf{n}^T, c)^T$ and $\boldsymbol{\lambda} = (\mathbf{m}^T, d)^T$ are transformed to the lines $\boldsymbol{\ell}' = (\mathbf{n}'^T, c')^T$ and $\boldsymbol{\lambda}' = (\mathbf{m}'^T, d')^T$ according to $\boldsymbol{\ell}' = \mathbf{H}_p^{-T} \boldsymbol{\ell}$ and $\boldsymbol{\lambda}' = \mathbf{H}_p^{-T} \boldsymbol{\lambda}$.

The scalar product of \mathbf{n} and \mathbf{m} can be written in terms of the lines $\boldsymbol{\ell}$ and $\boldsymbol{\lambda}$ as

$$\mathbf{n}^T \mathbf{m} = \boldsymbol{\ell}^T \mathbf{C}_\infty \boldsymbol{\lambda} \quad (692)$$

where

$$\mathbf{C}_\infty = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \quad (693)$$

The scalar product can also be described by the transformed $\boldsymbol{\ell}'$ and $\boldsymbol{\lambda}'$ lines by inserting $\boldsymbol{\ell} = \mathbf{H}_p^T \boldsymbol{\ell}'$ and $\boldsymbol{\lambda} = \mathbf{H}_p^T \boldsymbol{\lambda}'$. This gives

$$\mathbf{n}^T \mathbf{m} = \boldsymbol{\ell}'^T \mathbf{C}'_\infty \boldsymbol{\lambda}' \quad (694)$$

where the coefficient matrix for the transformed lines is

$$\mathbf{C}'_\infty = \mathbf{H}_p \mathbf{C}_\infty \mathbf{H}_p^T \quad (695)$$

It is noted that the coefficient matrix can be described by the camera parameter matrix and the the perspective vector \mathbf{v} as

$$\mathbf{C}'_\infty = \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{v}^T \mathbf{K} & v_3 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{K}^T & \mathbf{K}^T \mathbf{v} \\ \mathbf{0}^T & v_3 \end{bmatrix} = \begin{bmatrix} \mathbf{K} \mathbf{K}^T & \mathbf{K} \mathbf{K}^T \mathbf{v} \\ \mathbf{v}^T \mathbf{K} \mathbf{K}^T & \mathbf{v}^T \mathbf{K} \mathbf{K}^T \mathbf{v} \end{bmatrix} \quad (696)$$

To find the matrix \mathbf{C}_∞ the lines $\boldsymbol{\ell}$ and $\boldsymbol{\lambda}$ are selected so that they intersect at a right angle. Then the normal vectors are orthogonal, which gives $\mathbf{n}^T \mathbf{m} = 0$. This gives the equation

$$\boldsymbol{\ell}'^T \mathbf{C}'_\infty \boldsymbol{\lambda}' = 0 \quad (697)$$

which gives one condition on the coefficients of \mathbf{C}'_∞ if the lines $\boldsymbol{\ell}'$ and $\boldsymbol{\lambda}'$ can be identified in the transformed image. For computation the coefficient matrix is written in the standard for conic coefficient matrices:

$$\mathbf{C}'_\infty = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix} \quad (698)$$

while the lines are written

$$\boldsymbol{\ell}' = (L_1, L_2, L_3)^T, \quad \boldsymbol{\lambda}' = (\lambda_1, \lambda_2, \lambda_3)^T \quad (699)$$

Then the quadratic form can be reformulated as

$$\boldsymbol{\ell}'^T \mathbf{C}'_\infty \boldsymbol{\lambda}' = \mathbf{A} \mathbf{c} = 0 \quad (700)$$

where $\mathbf{c} = [a, b, c, d, e, f]^T$ is a vector of the coefficients of \mathbf{C}'_∞ , and ctor

$$\mathbf{A} = \begin{bmatrix} L_1 \lambda_1 & \frac{L_1 \lambda_2 + L_2 \lambda_1}{2} & L_2 \lambda_2 & \frac{L_1 \lambda_3 + L_3 \lambda_1}{2} & \frac{L_2 \lambda_3 + L_3 \lambda_2}{2} & L_3 \lambda_3 \end{bmatrix} \quad (701)$$

The coefficient matrix \mathbf{C}'_∞ has 6 parameters. Due to the free scaling only 5 are independent. This means that the matrix can be determined by 5 orthogonal line pairs. Each line pair i will give a matrix \mathbf{A}_i , $i = 1, \dots, 5$ that can be assembled in a 5×6 matrix \mathbf{A} to give

$$\begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_5 \end{bmatrix} \mathbf{c} = \mathbf{A} \mathbf{c} = \mathbf{0} \quad (702)$$

The solution is found by singular value decomposition $\mathbf{A} = \mathbf{U}_A \boldsymbol{\Sigma}_A \mathbf{V}_A^T$ as

$$\mathbf{c} = \alpha \mathbf{v}_6 \quad (703)$$

where α is a nonzero constant, and \mathbf{v}_6 is the last column in \mathbf{V}_A . Then, given \mathbf{c} , the matrix \mathbf{C}'_∞ is found.

The next step to have a metric recovery is to find a homography $\mathbf{x}'' = \mathbf{H}_r \mathbf{x}'$ that will recover the metric properties. The lines $\boldsymbol{\ell}'' = (\mathbf{n}''^T, c'')^T$ and $\boldsymbol{\lambda}'' = (\mathbf{m}''^T, d'')^T$ of the recovered solution will be $\boldsymbol{\ell}'' = \mathbf{H}_r^{-T} \boldsymbol{\ell}'$ and $\boldsymbol{\lambda}'' = \mathbf{H}_r^{-T} \boldsymbol{\lambda}'$, which means that $\boldsymbol{\ell}' = \mathbf{H}_r^T \boldsymbol{\ell}''$ and $\boldsymbol{\lambda}' = \mathbf{H}_r^T \boldsymbol{\lambda}''$. This gives

$$\boldsymbol{\ell}'^T \mathbf{C}'_\infty \boldsymbol{\lambda}' = \boldsymbol{\ell}''^T \mathbf{C}''_\infty \boldsymbol{\lambda}'' = 0 \quad (704)$$

which gives

$$\mathbf{C}''_\infty = \mathbf{H}_r \mathbf{C}'_\infty \mathbf{H}_r^T \quad (705)$$

The lines $\boldsymbol{\ell}''$ and $\boldsymbol{\lambda}''$ will then be orthogonal if

$$\mathbf{C}''_\infty = \mathbf{C}_\infty = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \quad (706)$$

which gives $\mathbf{n}''^T \mathbf{m}'' = 0$.

To find a homography \mathbf{H}_r that will ensure this, it is used that $\mathbf{C}'_\infty = \mathbf{H}_r^{-1} \mathbf{C}''_\infty \mathbf{H}_r^{-T}$. This is compared with the singular value decomposition \mathbf{C}'_∞ . This 3×3 matrix is symmetric and of rank 2, which gives the singular value decomposition in the form

$$\mathbf{C}'_\infty = \mathbf{U} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{U}^T \quad (707)$$

where $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$. Note that $\mathbf{U} = \mathbf{V}$ in the singular value decomposition since \mathbf{C}'_∞ is symmetric. This gives

$$\mathbf{C}'_\infty = \mathbf{U} \mathbf{D} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{D} \mathbf{U}^T \quad (708)$$

where $\mathbf{D} = \text{diag}(\sqrt{\sigma_1}, \sqrt{\sigma_2}, \beta)$, and β is a nonzero constant that can be selected freely.

This means that the recovering homography can be found from

$$\mathbf{H}_r^{-1} = \mathbf{U} \mathbf{D} = [\sqrt{\sigma_1} \mathbf{u}_1, \sqrt{\sigma_2} \mathbf{u}_2, \beta \mathbf{u}_3] \quad (709)$$

Then the matrix \mathbf{C}''_∞ will satisfy the condition (706), which means that the lines ℓ'' and λ'' will intersect at a right angle. This means that the metric properties of the image is recovered with the homography

$$\mathbf{x}'' = \mathbf{H}_r \mathbf{x}' \quad (710)$$

Example

To demonstrate metric recovery of an object that has been transformed with a projective homography an example with a square in the plane is considered. This square is shown in Figure 60a, and the transformed square is shown in Figure 60b where the projective effect is clearly seen as parallel lines are no more parallel, and lines that intersect at right angles in the original square do no longer intersect at right angles. To perform metric recovery it is not sufficient that parallel lines are made parallel, it is also required that lines that intersect at right angles in the original image are made to intersect at a right angles in the recovered image. Metric recovery requires 5 equations, and this means that 5 line intersections at right angles are required. The corners of the square gives 4 line intersections, and one additional line crossing of the diagonals of the square is included to has 5 line intersections. This is shown for the original image in Figure 61a. The projective transformation of the line intersections is shown in Figure 61b.

The recovered image using the homography given by (709) is shown in Figure 62a, and the same recovered square with the line crossings is shown in Figure 62b. It is seen that parallel lines are parallel, and that lines that intersect at a right angle in the original image will intersect with a right angle in the recovered image. Note that the recovered square is translated and rotated compared to the original square, and it is scaled by a factor of about 1/2. This is consistent with the result that this type of metric recovery does not recover the similarity transform.

The unknown projective transformation of the example is

$$\mathbf{H}_p = \begin{bmatrix} \cos(\pi/6) & -0.5 \sin(\pi/6) & 0 \\ 0.5 \sin(\pi/6) & \cos(\pi/6) & 0 \\ 0.25 & 0.866 & 2 \end{bmatrix} \quad (711)$$

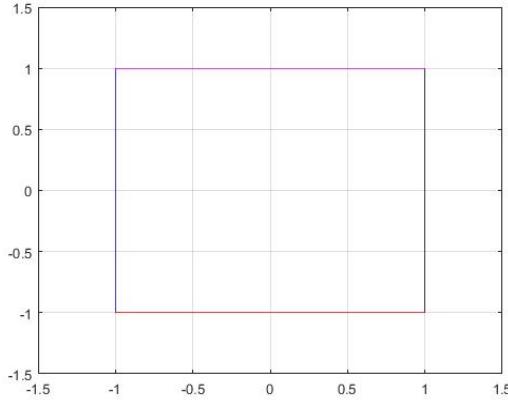
The composite homography from \mathbf{x} to \mathbf{x}'' is found to be

$$\mathbf{H} = \mathbf{H}_r \mathbf{H}_p = \begin{bmatrix} \mathbf{R} & 2\mathbf{t} \\ \mathbf{0}^T & 2 \end{bmatrix} = 2 \begin{bmatrix} 0.5\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (712)$$

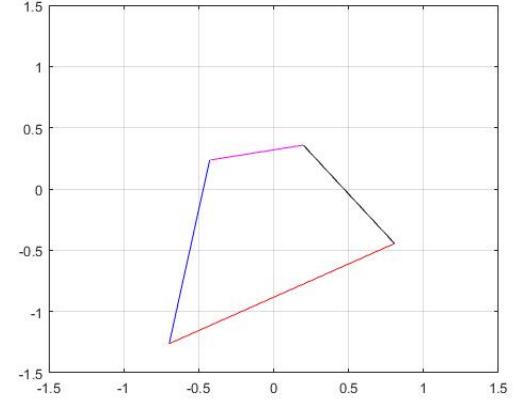
where $\mathbf{t} = [-0.5547, 0]^T$ is the translation of the homography, and

$$\mathbf{R} = \begin{bmatrix} -0.2774 & -0.9608 \\ -0.9608 & 0.2774 \end{bmatrix} \quad (713)$$

satisfies $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and $\det(\mathbf{R}) = -1$, which means that it is a reflection matrix. This means that the composite transformation \mathbf{H} is a similarity transform with scaling factor 0.5. This is in agreement with Figure 62, where it is seen from the coloring of the line crossings that the square has been reflected. The scaling by 1/2 and the translation is also in agreement with Figure 62.

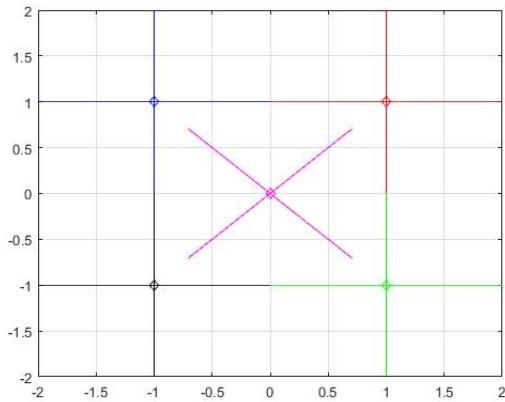


(a) Original square.

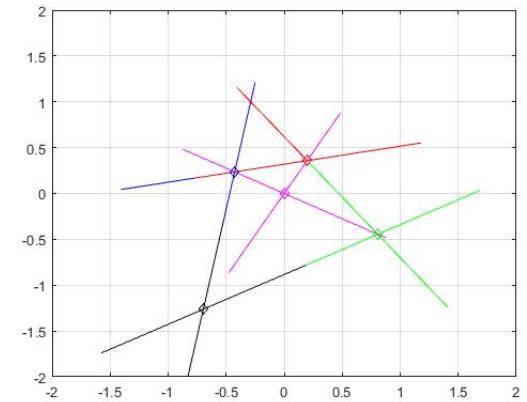


(b) Square after projective transformation.

Figure 60: Square before and after projective transformation.



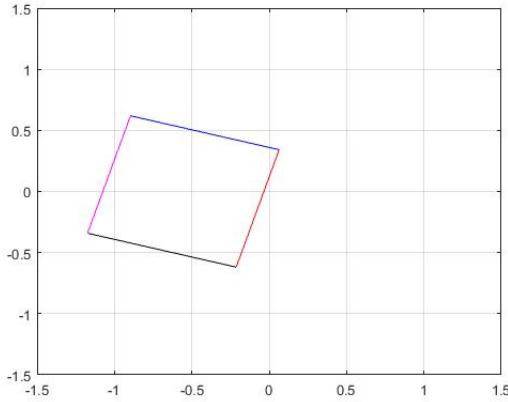
(a) 5 line crossings in scene.



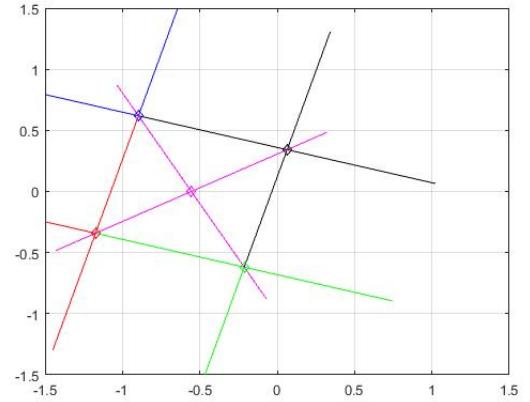
(b) Line crossings after projective transformation of scene.

Figure 61: Line crossings before and after projective transformation

```
% Script for metric recovery from 5 line crossings at right angles.
```



(a) Square after metric recovery of projected square. Note that the original scene differ from the recovered image by a similarity transform.



(b) Line crossings after metric recovery of projected line crossings. The line crossings are at a right angle as in the original scene. Note that the original scene differ from the recovered image by a similarity transform.

Figure 62: Recovered images

```
% Input for illustration: Points to draw a square in the scene.
x1 = [-1;1;1]; x2 = [-1;-1;1]; x3 = [1;-1;1]; x4 = [1;1;1];
% Input: Perspective transformation
K = [cos(pi/6) -0.5*sin(pi/6); 0.5*sin(pi/6) cos(pi/6)]; v = [0;1];, v3 = 2;
Hp = [K [0;0]; v'*K v3]
% Transformed square
xp1 = Hp*x1; xp2 = Hp*x2; xp3 = Hp*x3; xp4 = Hp*x4;

DrawSquare(x1,x2,x3,x4,[-1.5 1.5 -1.5 1.5],1);
DrawSquare(xp1/xp1(3),xp2/xp2(3),xp3/xp3(3),xp4/xp4(3),[-1.5 1.5 -1.5 1.5],2);
% Conic matrix in scene
Ci = [1 0 0;0 1 0;0 0 0];
% Transformed conic matrix
Cip = Hp*Ci*Hp';
%Input: Lines that intersect at a right angle in scene
L1 = [1; 0; 1]; M1 = [L1(2); - L1(1); 1];
L2 = [1; 0; -1]; M2 = [L2(2); - L2(1); 1];
L3 = [1; 0; -1]; M3 = [L3(2); - L3(1); -1];
L4 = [1; 0; 1]; M4 = [L4(2); - L4(1); -1];
L5 = [1; 1; 0]; M5 = [L5(2); - L5(1); 0];
DrawLineCrossings(L1,M1,L2,M2,L3,M3,L4,M4,L5,M5,[-2 2 -2 2],4);
% Transformed lines
L1p = Hp'\L1; M1p = Hp'\M1;
L2p = Hp'\L2; M2p = Hp'\M2;
L3p = Hp'\L3; M3p = Hp'\M3;
L4p = Hp'\L4; M4p = Hp'\M4;
```

```

L5p = Hp'\L5; M5p = Hp'\M5;
DrawLineCrossings(L1p,M1p,L2p,M2p,L3p,M3p,L4p,M4p,L5p,M5p,[-2 2 -2 2],5);

% Coefficient matrix for calculation of conic matrix used in metric
% recovery
A = [Lines2Ai(L1p,M1p); Lines2Ai(L2p,M2p); Lines2Ai(L3p,M3p); ...
       Lines2Ai(L4p,M4p); Lines2Ai(L5p,M5p)];
[U,S,V] = svd(A); c = V(:,6);
C = [c(1) c(2)/2 c(4)/2; c(2)/2 c(3) c(5)/2; c(4)/2 c(5)/2 c(6)];
C = C/C(3,3); C = C*Cip(3,3);
% Calculation of perspective transformation for use in metric recovery
[uc,sc,vc] = svd(C);
invHr = [uc(:,1)*sqrt(sc(1,1)) uc(:,2)*sqrt(sc(2,2)) 0.5*uc(:,3)/uc(3,3)];
Hr = inv(invHr)
% Metric recovery of points in square
xpp1 = Hr*xp1; xpp2 = Hr*xp2; xpp3 = Hr*xp3; xpp4 = Hr*xp4;
DrawSquare(xpp1/xpp1(3),xpp2/xpp2(3),xpp3/xpp3(3),xpp4/xpp4(3),[-1.5 1.5 -1.5 1.5],3);
% Metric recovery of lines
L1pp = invHr'*L1p; M1pp = invHr'*M1p;
L2pp = invHr'*L2p; M2pp = invHr'*M2p;
L3pp = invHr'*L3p; M3pp = invHr'*M3p;
L4pp = invHr'*L4p; M4pp = invHr'*M4p;
L5pp = invHr'*L5p; M5pp = invHr'*M5p;
DrawLineCrossings(L1pp,M1pp,L2pp,M2pp,L3pp,M3pp,L4pp,M4pp,L5pp,M5pp,[-1.5 1.5 -1.5 1.5],6);

% Test if recovered lines intersect at right angle
InnerProduct1 = L1pp(1:2)'*M1pp(1:2);
InnerProduct2 = L2pp(1:2)'*M2pp(1:2);
InnerProduct3 = L3pp(1:2)'*M3pp(1:2);
InnerProduct4 = L4pp(1:2)'*M4pp(1:2);
InnerProduct5 = L5pp(1:2)'*M5pp(1:2);

H = Hr*Hp
R = H(1:2,1:2)
R'*R
det(R)

%%%%%
function Ai = Lines2Ai(L,M)
% This function calculates the coefficient row Ai when
% L'*C*M is converted to Ai*c where C = [a b/2 d/2; b/2 c e/2; d/2 e/2 f]
% and c =[a b c d e f]'

Ai = [L(1)*M(1) 0.5*(L(1)*M(2) + L(2)*M(1)) L(2)*M(2) ...
       0.5*(L(1)*M(3) + L(3)*M(1)) ...
       0.5*(L(2)*M(3) + L(3)*M(2)) L(3)*M(3)];

```

```

end

%%%%%%%
function DrawLineCrossings(L1,M1,L2,M2,L3,M3,L4,M4,L5,M5, axisVector, figureNumber )
% The function draws the line crossings of L1,Mi, i = 1,2,...,5

xLM1 = cross(L1,M1); xLM1 = xLM1/xLM1(3);
aL1 = [L1(2);-L1(1)]; aL1 = aL1/norm(aL1); aM1 = [M1(2);-M1(1)]; aM1 = aM1/norm(aM1);
x11 = xLM1(1:2)-aL1; x12 = xLM1(1:2)+aL1; x13 = xLM1(1:2) -aM1; x14 = xLM1(1:2)+aM1;

xLM2 = cross(L2,M2); xLM2 = xLM2/xLM2(3);
aL2 = [L2(2);-L2(1)]; aL2 = aL2/norm(aL2); aM2 = [M2(2);-M2(1)]; aM2 = aM2/norm(aM2);
x21 = xLM2(1:2)-aL2; x22 = xLM2(1:2)+aL2; x23 = xLM2(1:2) -aM2; x24 = xLM2(1:2)+aM2;

xLM3 = cross(L3,M3); xLM3 = xLM3/xLM3(3);
aL3 = [L3(2);-L3(1)]; aL3 = aL3/norm(aL3); aM3 = [M3(2);-M3(1)]; aM3 = aM3/norm(aM3);
x31 = xLM3(1:2)-aL3; x32 = xLM3(1:2)+aL3; x33 = xLM3(1:2) -aM3; x34 = xLM3(1:2)+aM3;

xLM4 = cross(L4,M4); xLM4 = xLM4/xLM4(3);
aL4 = [L4(2);-L4(1)]; aL4 = aL4/norm(aL4); aM4 = [M4(2);-M4(1)]; aM4 = aM4/norm(aM4);
x41 = xLM4(1:2)-aL4; x42 = xLM4(1:2)+aL4; x43 = xLM4(1:2) -aM4; x44 = xLM4(1:2)+aM4;

xLM5 = cross(L5,M5); xLM5 = xLM5/xLM5(3);
aL5 = [L5(2);-L5(1)]; aL5 = aL5/norm(aL5); aM5 = [M5(2);-M5(1)]; aM5 = aM5/norm(aM5);
x51 = xLM5(1:2)-aL5; x52 = xLM5(1:2)+aL5; x53 = xLM5(1:2) -aM5; x54 = xLM5(1:2)+aM5;

figure(figureNumber);clf;
plot([x11(1) x12(1)], [x11(2) x12(2)], 'b', [x13(1) x14(1)], [x13(2) x14(2)], 'b', ...
[x21(1) x22(1)], [x21(2) x22(2)], 'r', [x23(1) x24(1)], [x23(2) x24(2)], 'r', ...
[x31(1) x32(1)], [x31(2) x32(2)], 'g', [x33(1) x34(1)], [x33(2) x34(2)], 'g', ...
[x41(1) x42(1)], [x41(2) x42(2)], 'k', [x43(1) x44(1)], [x43(2) x44(2)], 'k', ...
[x51(1) x52(1)], [x51(2) x52(2)], 'm', [x53(1) x54(1)], [x53(2) x54(2)], 'm');

grid
axis(axisVector); hold on;
plot(xLM1(1), xLM1(2), 'db', xLM2(1), xLM2(2), 'dr', ...
xLM3(1), xLM3(2), 'dg', xLM4(1), xLM4(2), 'dk', ...
xLM5(1), xLM5(2), 'dm');

end

%%%%%%
function DrawSquare(x1,x2,x3,x4, axisVector, figureNumber )
% The function draws the square with corner points x1, x2, x3 and x4.

figure(figureNumber);clf;
plot([x1(1) x2(1)], [x1(2) x2(2)], 'b', ...

```

```
[x2(1) x3(1)], [x2(2) x3(2)],'r', ...
[x3(1) x4(1)], [x3(2) x4(2)],'k', ...
[x4(1) x1(1)], [x4(2) x1(2)],'m');

grid
axis(axisVector); hold on;
end
```

□

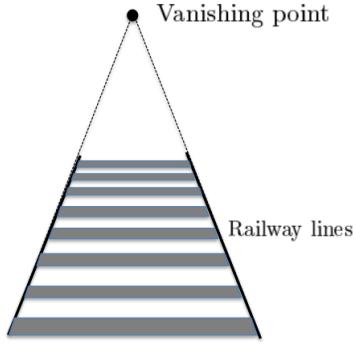


Figure 63: Image of two parallel railway lines, where the perspective of the image causes the two parallel lines to intersect at the horizon in a vanishing point.

16 Calibration

16.1 Introduction

The calculation of the camera parameter matrix from observations is called calibration. The calibration of a camera is important, and is used to find the camera parameter matrix \mathbf{K} . Then, when \mathbf{K} has been found, the camera is said to be calibrated, and the normalized image coordinates $\tilde{\mathbf{s}}$ can be found from the pixel coordinates $\tilde{\mathbf{p}}$ with $\tilde{\mathbf{s}} = \mathbf{K}^{-1}\tilde{\mathbf{p}}$. This means that for a calibrated camera, the camera model simplifies to

$$\lambda \tilde{\mathbf{s}} = [\mathbf{R}_o^c \quad \mathbf{t}_{co}^c] \tilde{\mathbf{r}}_{op}^o \quad (714)$$

where $\tilde{\mathbf{r}}_{op}^o = [x_o, y_o, z_o, 1]^T$ is the homogeneous position of the point in the scene in the object frame, and $\lambda = z$ is the depth coordinate, which is the z coordinate of the Euclidean position $\mathbf{r}_{op}^c = [\mathbf{R}_o^c, \mathbf{t}_{co}^c] \tilde{\mathbf{r}}_{op}^o = [x, y, z]^T$ of the point in the scene in the camera frame.

16.2 Vanishing points in an image

The pinhole model of a camera has a projective effect from the 3D Euclidean scene to the 2D normalized image plane. This means that a point at infinity can be mapped into the image by the projective transformation. A point at infinity that is mapped into an image in this way is called a vanishing point. This effect is a well-known fact from daily life, where, e.g., an image of a pair of railway lines will converge to a point in the image (Figure 63). In the same way parallel lines of a building may converge in a point in the image. Moreover, two orthogonal sets or parallel lines in the horizontal plane will define two points at infinity, and the line through these two points will be the line at infinity which represents the horizon, as shown in Figure 64.

16.3 The image of a vanishing point

Let the camera model be $\mathbf{C} = \mathbf{K}[\mathbf{I} \mid \mathbf{0}]$, which means that the camera frame and the object frame coincides. Then the homogeneous pixel coordinates $\tilde{\mathbf{p}}$ of the image of a homogeneous

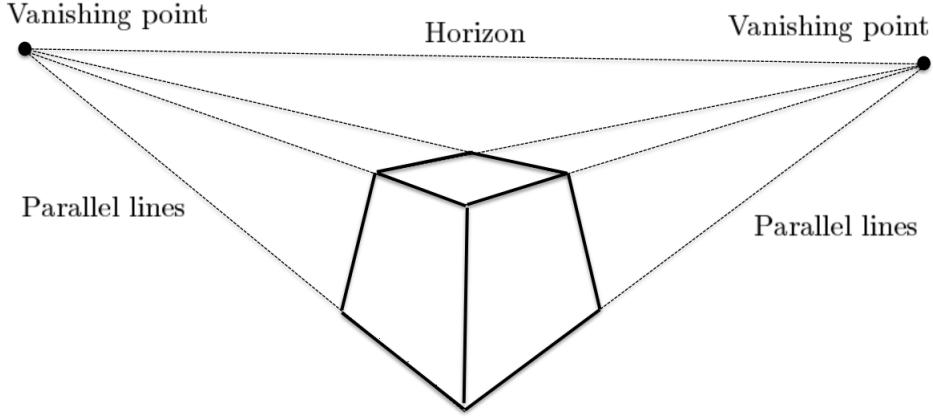


Figure 64: Two vanishing points defined by two sets of parallel lines, where the lines in each of the two sets are horizontal. The horizon is the line at infinity defined by the two vanishing points.

point $\tilde{\mathbf{r}} = [\mathbf{r}^T, 1]^T$ in the coinciding camera and object frame will be

$$\lambda \tilde{\mathbf{p}} = \mathbf{K} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ 1 \end{bmatrix} = \mathbf{K} \mathbf{r} \quad (715)$$

where λ is the depth coordinate. Consider the line defined by the points

$$\tilde{\mathbf{r}} = \begin{bmatrix} \mathbf{r}_0 \\ 1 \end{bmatrix} + \mu \begin{bmatrix} \mathbf{a} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_0 + \mu \mathbf{a} \\ 1 \end{bmatrix} \quad (716)$$

Here \mathbf{r}_0 is a Euclidean point on the line, \mathbf{a} is the direction vector and μ is the parameter that varies along the line. Then, if $\mu \rightarrow \infty$, the point $\tilde{\mathbf{r}}$ on the line will tend to the vanishing point

$$\tilde{\mathbf{a}} = \begin{bmatrix} \mathbf{a} \\ 0 \end{bmatrix} \quad (717)$$

Note that the vanishing point is given by the direction vector \mathbf{a} only, and the vanishing point will be the same for all \mathbf{r}_0 for a given \mathbf{a} .

The homogeneous pixel coordinates of the image of the vanishing point is

$$\tilde{\mathbf{z}} = \mathbf{K} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ 0 \end{bmatrix} = \mathbf{K} \mathbf{a} \quad (718)$$

while the homogeneous coordinates in the normalized image coordinates is

$$\tilde{\mathbf{s}} = \mathbf{a} \quad (719)$$

This is the image of a line through the camera center in the direction of \mathbf{a} . This is illustrated in Figure 65. Again, note that the image of the vanishing point does not depend on the displacement of the line relative to the camera center.

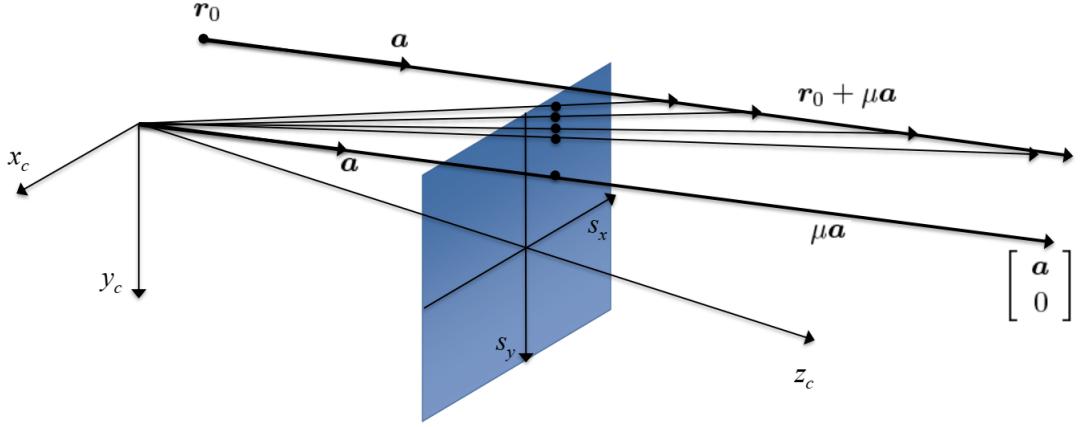


Figure 65: The image of a vanishing point of a line with direction vector \mathbf{a} is the same as the image of line from the optical center with direction vector \mathbf{a} .

16.4 The image of a vanishing line

Consider a plane $\pi = [\mathbf{u}^T, d]^T$ with normal vector \mathbf{u} , and with distance $d/\|\mathbf{u}\|$ from the plane to the origin. Consider the two lines

$$\tilde{\mathbf{r}}_1 = \begin{bmatrix} \mathbf{r}_{10} \\ 1 \end{bmatrix} + \mu_1 \begin{bmatrix} \mathbf{a}_1 \\ 0 \end{bmatrix}, \quad \tilde{\mathbf{r}}_2 = \begin{bmatrix} \mathbf{r}_{20} \\ 1 \end{bmatrix} + \mu_2 \begin{bmatrix} \mathbf{a}_2 \\ 0 \end{bmatrix} \quad (720)$$

in the plane π , where the unit direction vectors \mathbf{a}_1 and \mathbf{a}_2 are not parallel. Then the normal vector of the plane will be

$$\mathbf{u} = \mathbf{a}_1 \times \mathbf{a}_2 \quad (721)$$

The lines will tend to the vanishing points

$$\tilde{\mathbf{a}}_1 = \begin{bmatrix} \mathbf{a}_1 \\ 0 \end{bmatrix}, \quad \tilde{\mathbf{a}}_2 = \begin{bmatrix} \mathbf{a}_2 \\ 0 \end{bmatrix} \quad (722)$$

These two vanishing points define a vanishing line with Plücker coordinates

$$(\mathbf{l}, \mathbf{l}') = (\mathbf{0}, \mathbf{u}) \quad (723)$$

which follows from (446). The vanishing line must be in the plane π since both vanishing points are in the plane. The vanishing line (723) can also be found as the intersection of the plane $\pi = (\mathbf{u}, d)$ and the plane at infinity $\pi_\infty = (\mathbf{0}, 1)$ which is seen from (508) to be $\ell = (\mathbf{0} \times \mathbf{u}, \mathbf{u}) = (\mathbf{0}, \mathbf{u})$.

Then the plane through the origin $(\mathbf{0}, 1)$ of the camera frame and the vanishing line $(\mathbf{0}, \mathbf{u})$ is given by

$$\pi_1 = (\mathbf{u}, 0) \quad (724)$$

which follows from (478). The image of the vanishing line is the intersection of the plane $(\mathbf{u}, 0)$ and the image frame. Note that the vectors \mathbf{a}_1 and \mathbf{a}_2 with starting point in the origin will

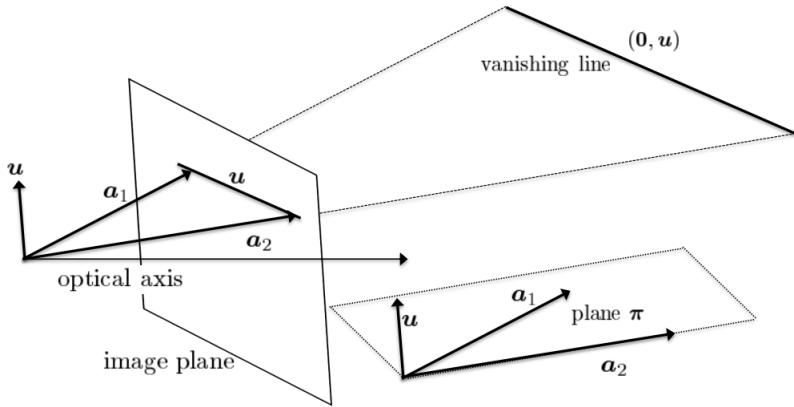


Figure 66: The plane $\pi = (u, d)$ intersects the plane at infinity in the vanishing line $(0, u)$. The image of this vanishing line is the line u in the normalized image plane.

intersect the normalized image plane at $s_1 = \gamma_1 a_1$ and $s_2 = \gamma_2 a_2$. This means that the image of the vanishing line will be the line through the image points s_1 and s_2 , which means that the line in the image plane is $\ell = s_1 \times s_2 = \gamma_1 \gamma_2 a_1 \times a_2 = \gamma_1 \gamma_2 u$. Leaving out the scaling, this gives the line

$$\ell = u \quad (725)$$

in the normalized image plane. This is illustrated in Figure 66.

16.5 Calibration from scene constraints

The usual technique for camera calibration is based on constraints in the scene that are used to find the matrix

$$\mathbf{B} = \mathbf{K}^{-T} \mathbf{K}^{-1} = (\mathbf{K} \mathbf{K}^T)^{-1} \quad (726)$$

which is called the image of the absolute conic. When \mathbf{B} is found, the camera parameter matrix \mathbf{K} can be computed with Cholesky decomposition. This can be done with the Cholesky decomposition $\mathbf{B} = \mathbf{U}^T \mathbf{U}$, and then by computing $\mathbf{K} = \mathbf{U}^{-1}$.

This typically involves lines or directions that are orthogonal in the scene, and the associated vanishing points. Let the camera model be $\mathbf{C} = \mathbf{K}[\mathbf{R} | \mathbf{t}]$. Then a vanishing point in the direction direction \mathbf{a} will be imaged to

$$\lambda \mathbf{z} = \mathbf{K} \mathbf{R} \mathbf{a} \quad (727)$$

where λ is the depth factor. This gives

$$\mathbf{a} = \lambda (\mathbf{K} \mathbf{R})^{-1} \mathbf{z} = \lambda \mathbf{R}^T \mathbf{K}^{-1} \mathbf{z} \quad (728)$$

First we consider the case where there are the two vanishing points \mathbf{a}_1 and \mathbf{a}_2 are orthogonal so that $\mathbf{a}_1^T \mathbf{a}_2 = 0$. The image points are $\lambda_1 \mathbf{z}_1 = \mathbf{K} \mathbf{R} \mathbf{a}_1$ and $\lambda_2 \mathbf{z}_2 = \mathbf{K} \mathbf{R} \mathbf{a}_2$. Then

$$\mathbf{a}_1^T \mathbf{a}_2 = \lambda_1 \lambda_2 \mathbf{z}_1^T \mathbf{K}^{-T} \mathbf{R} \mathbf{R}^T \mathbf{K}^{-1} \mathbf{z}_2 = \lambda_1 \lambda_2 \mathbf{z}_1^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{z}_2 \quad (729)$$

This gives the condition

$$\mathbf{z}_1^T \mathbf{B} \mathbf{z}_2 = 0 \quad (730)$$

This is a scalar equation that gives one constraint on the matrix \mathbf{B} .

Another interesting case is that the scene includes a plane π and a line L that is orthogonal to the plane. An example is when the plane is the floor, and the line is vertical. Suppose that the line has direction vector \mathbf{a} , and the plane has direction vector \mathbf{u} . This means that the direction vector \mathbf{a} of the line will be in the same direction as normal vector \mathbf{u} of the plane, which means that $\mathbf{u} = \gamma \mathbf{a}$ for some nonzero scalar γ . Then the line will have a vanishing point in the image at $\mathbf{z} = \mathbf{K} \mathbf{R} \mathbf{a}$. This follows as the points are mapped with the homography $\mathbf{H} = \mathbf{K} \mathbf{R}$, and it follows that lines are mapped with $\mathbf{H}^{-T} = (\mathbf{K} \mathbf{R})^{-T}$. The image of the vanishing line of the plane will be $\ell = (\mathbf{K} \mathbf{R})^{-T} \mathbf{u}$. This gives

$$\ell = (\mathbf{K} \mathbf{R})^{-T} \gamma \mathbf{a} = \gamma (\mathbf{K} \mathbf{R})^{-T} (\mathbf{K} \mathbf{R})^{-1} \mathbf{z} = \gamma \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{z} \quad (731)$$

which gives

$$\ell = \gamma \mathbf{B} \mathbf{z} \quad (732)$$

This gives the constraint

$$\ell^* \mathbf{B} \mathbf{z} = 0 \quad (733)$$

which gives two constraints on \mathbf{B} .

Another method which was presented in [82] is to use a plane with pose

$$\mathbf{T}_o^c = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (734)$$

where the z axis with direction \mathbf{r}_3 is normal to the plane. Then, if there are at 4 points with known position, the vectors $\mathbf{h}_1 = \mathbf{K} \mathbf{r}_1$, $\mathbf{h}_2 = \mathbf{K} \mathbf{r}_2$ and $\mathbf{h}_3 = \mathbf{K} \mathbf{t}$ can be found with PnP techniques. Then from $\mathbf{r}_1^T \mathbf{r}_2 = 0$, and $\mathbf{r}_1^T \mathbf{r}_1 = \mathbf{r}_2^T \mathbf{r}_2 = 1$, it is found that $\mathbf{h}_1^T \mathbf{B} \mathbf{h}_2 = 0$ and $\mathbf{h}_1^T \mathbf{B} \mathbf{h}_1 = \mathbf{h}_2^T \mathbf{B} \mathbf{h}_2$. The two scalar equations give 2 constraints on the matrix \mathbf{B} .

16.6 Expressions for the image of the absolute conic

Let the intrinsic parameters be given written in the form

$$\mathbf{K} = \begin{bmatrix} k_1 & s & u_0 \\ 0 & k_2 & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (735)$$

The image of the absolute conic $\mathbf{B} = (\mathbf{K} \mathbf{K}^T)^{-1}$ is then given by

$$\mathbf{B} = \frac{1}{k_1^2 k_2^2} \begin{bmatrix} k_2^2 & -s k_2 & -u_0 k_2^2 + v_0 s k_2 \\ -s k_2 & k_1^2 + s^2 & k_2 s u_0 - k_1^2 v_0 - s^2 v_0 \\ -u_0 k_2^2 + v_0 s k_2 & k_2 s u_0 - k_1^2 v_0 - s^2 v_0 & k_1^2 k_2^2 + k_1^2 v_0^2 + (k_2 u_0 - s v_0)^2 \end{bmatrix} \quad (736)$$

which can also be written

$$\mathbf{B} = \begin{bmatrix} \frac{1}{k_1^2} & -\frac{s}{k_1^2 k_2} & \frac{v_0 s - u_0 k_2}{k_1^2 k_2} \\ -\frac{s}{k_1^2 k_2} & \frac{1}{k_2^2} + \frac{s^2}{k_1^2 k_2^2} & \frac{s(k_2 u_0 - s v_0)}{k_1^2 k_2^2} - \frac{v_0}{k_2^2} \\ \frac{v_0 s - u_0 k_2}{k_1^2 k_2} & \frac{s(k_2 u_0 - s v_0)}{k_1^2 k_2^2} - \frac{v_0}{k_2^2} & 1 + \frac{v_0^2}{k_2^2} + \frac{(k_2 u_0 - s v_0)^2}{k_1^2 k_2^2} \end{bmatrix} \quad (737)$$

In the case of zero skew, $s = 0$ which gives

$$\mathbf{B} = \frac{1}{k_1^2 k_2^2} \begin{bmatrix} k_2^2 & 0 & -u_0 k_2^2 \\ 0 & k_1^2 & -k_1^2 v_0 \\ -u_0 k_2^2 & -k_1^2 v_0 & k_1^2 k_2^2 + k_1^2 v_0^2 + (k_2 u_0)^2 \end{bmatrix} \quad (738)$$

This gives the constraint $B_{12} = B_{21} = 0$. If the principal point (u_0, v_0) is known, the origin can be translated to the principal point, and the image of the absolute conic becomes

$$\mathbf{B} = \begin{bmatrix} \frac{1}{k_1^2} & 0 & 0 \\ 0 & \frac{1}{k_2^2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (739)$$

If, in addition, the pixels are assumed to be quadratic, then $k_1 = k_2$, which gives the additional constraint $B_{11} = B_{22}$.

The dual absolute image conic $\mathbf{B}^* = \mathbf{B}^{-1} = \mathbf{K} \mathbf{K}^T$ is

$$\mathbf{B}^* = \begin{bmatrix} k_1^2 + s^2 + u_0^2 & k_2 s + u_0 v_0 & u_0 \\ k_2 s + u_0 v_0 & k_2^2 + v_0^2 & v_0 \\ u_0 & v_0 & 1 \end{bmatrix} \quad (740)$$

If it is assumed that $s = 0$ and $k_1 = k_2 = k$, then

$$\mathbf{B}^* = \begin{bmatrix} k^2 + u_0^2 & u_0 v_0 & u_0 \\ u_0 v_0 & k^2 + v_0^2 & v_0 \\ u_0 & v_0 & 1 \end{bmatrix} \quad (741)$$

Another set of assumptions is that there is zero skew, which gives $s = 0$, and that the principal point is known, in which case $u_0 = v_0 = 0$. Then the matrix becomes

$$\mathbf{B}^* = \begin{bmatrix} k_1^2 & 0 & 0 \\ 0 & k_2^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (742)$$

The intrinsic parameters can be calculated from the components $\{b_{ij}\}$ of the matrix \mathbf{B} as [82]

$$v_0 = \frac{b_{12}b_{13} - b_{11}b_{23}}{b_{11}b_{22} - b_{12}^2} \quad (743)$$

$$\lambda = b_{33} - \frac{b_{13}^2 + v_0(b_{12}b_{13} - b_{11}b_{23})}{b_{11}} \quad (744)$$

$$k_1 = \sqrt{\frac{\lambda}{b_{11}}} \quad (745)$$

$$k_2 = \sqrt{\frac{\lambda b_{11}}{b_{12}b_{13} - b_{11}b_{23}}} \quad (746)$$

$$s = -\frac{b_{12}k_1^2}{\lambda} \quad (747)$$

$$u_0 = \frac{sv_0}{k_1} - \frac{b_{13}k_1^2}{\lambda} \quad (748)$$

Example

```

%Test of expressions for intrinsic parameters from B = inv(K*K')
K = [1500 0.3 640; 0 1490 512; 0 0 1]
B = inv(K*K');

bb = B(1,2)*B(1,3) - B(1,1)*B(2,3);
cc = B(1,1)*B(2,2) - B(1,2)^2;
v0 = bb/cc;
lambda = B(3,3) - (B(1,3)^2 + v0*bb)/(B(1,1));
k1 = sqrt(lambda/B(1,1));
k2 = sqrt(lambda*B(1,1)/cc);
s = -(B(1,2)*k1*k1*k2)/lambda;
u0 = (s*v0)/k1 - (B(1,3)*k1*k1)/lambda;

Kc = [k1 s u0; 0 k2 v0; 0 0 1] % Kc = K is expected

```

16.7 Intrinsic calibration with checkerboard planes: Zhang's method

This section presents Zhang's method for calibration [82] is presented. In this method the camera parameter matrix \mathbf{K} is computed from P4P for at least 3 checkerboard planes, where the unit orthogonal vectors are found. The image of the absolute conic $\mathbf{B} = \mathbf{K}^{-T}\mathbf{K}^{-1}$ is then found from 2 constraints for each plane related to these orthogonal unit vectors. This technique is presented as Algorithm 8.2 in [30], and is described in textbooks like [41] and [67]. A modified version is used in the MATLAB and OpenCV toolboxes of Bouguet². A comparison of toolboxes for calibration is presented in [58].

In the following presentation it is assumed that 3 checkerboard planes are used. Consider 3 object planes π_1 , π_2 and π_3 as shown in Figure 67. Define an object frame 1 so that the z coordinate is zero on the object plane π_1 . In the same way, the object frame 2 is defined so that the z coordinate is zero on π_2 , and the object frame 3 is defined so that the z coordinate is zero on π_3 .

In each object plane π_j there are 4 points with coordinates in object frame j given by $\tilde{\mathbf{r}}_{ji}^j = [x_{ji}, y_{ji}, 0]^T$. The corresponding homogeneous vector of the position in the xy plane of frame j is

$$\tilde{\mathbf{x}}_{ji} = \begin{bmatrix} x_{ji} \\ y_{ji} \\ 1 \end{bmatrix}$$

The normalized image coordinates of the point $\tilde{\mathbf{r}}_{ji}^j$ are given by

$$\tilde{\mathbf{s}}_{ji} = \mathbf{\Pi}\tilde{\mathbf{r}}_{ci}^c = [\mathbf{R}_j \quad \mathbf{t}_j] \tilde{\mathbf{r}}_{ji}^j = \mathbf{M}_j \tilde{\mathbf{x}}_{ji}$$

where

$$\mathbf{M}_j = [\mathbf{r}_{j1} \quad \mathbf{r}_{j2} \quad \mathbf{t}_j]$$

The pixel coordinates are then

$$\tilde{\mathbf{p}}_{ji} = \mathbf{K}\tilde{\mathbf{s}}_{ji} = \mathbf{K}\mathbf{M}_j \tilde{\mathbf{x}}_{ji}$$

²<http://www.vision.caltech.edu/bouguetj/>

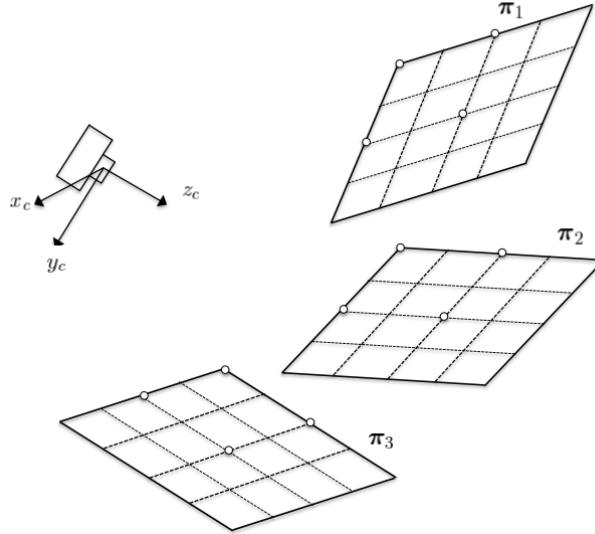


Figure 67: Three planes π_1 , π_2 and π_3 , where each plane has four known points.

This defines a homography $\mathbf{H}_j = \mathbf{K}\mathbf{M}_j$ for the plane π_j so that

$$\tilde{\mathbf{p}}_{ji} = \mathbf{H}_j \tilde{\mathbf{x}}_{ji}$$

The homography $\mathbf{H}_j = [\mathbf{h}_{j1} \ \mathbf{h}_{j2} \ \mathbf{h}_{j3}]$ for the plane π_j can be found from four point mappings $(\tilde{\mathbf{p}}_{ji}, \tilde{\mathbf{x}}_{ji})$ where $\tilde{\mathbf{x}}_{ji}$ is a point on the plane π_j . This is done by setting up the equation $\mathbf{A}_j \mathbf{h}_j = \mathbf{0}$ where $\mathbf{h}_j = [\mathbf{h}_{j1}^T, \mathbf{h}_{j2}^T, \mathbf{h}_{j3}^T]^T$, and then using $\mathbf{h}_j = \gamma \mathbf{v}_{j9}$, where \mathbf{v}_{j9} is the last column in \mathbf{V}_j found from $\mathbf{A}_j = \mathbf{U}_j \mathbf{\Sigma}_j \mathbf{V}_j^T$.

Suppose that the homographies $\mathbf{H}_j = [\mathbf{h}_{j1} \ \mathbf{h}_{j2} \ \mathbf{h}_{j3}]$ have been found for 3 different planes π_j , $j = 1, 2, 3$. Then, for each plane

$$[\mathbf{h}_1 \ \mathbf{h}_2 \ \mathbf{h}_3] = \mathbf{K} [\mathbf{r}_1 \ \mathbf{r}_2 \ \mathbf{t}] = [\mathbf{K}\mathbf{r}_1 \ \mathbf{K}\mathbf{r}_2 \ \mathbf{K}\mathbf{t}]$$

and it follows that

$$\mathbf{r}_1 = \mathbf{K}^{-1} \mathbf{h}_1 \quad \text{and} \quad \mathbf{r}_2 = \mathbf{K}^{-1} \mathbf{h}_2$$

The orthogonality of the rotation matrix implies that

$$\mathbf{r}_1^T \mathbf{r}_2 = 0 \tag{749}$$

$$\mathbf{r}_1^T \mathbf{r}_1 = \mathbf{r}_2^T \mathbf{r}_2 \tag{750}$$

This gives two conditions on \mathbf{B} for each plane, which are written

$$\mathbf{h}_1^T \mathbf{B} \mathbf{h}_2 = 0 \tag{751}$$

$$\mathbf{h}_1^T \mathbf{B} \mathbf{h}_1 - \mathbf{h}_2^T \mathbf{B} \mathbf{h}_2 = 0 \tag{752}$$

It is used that \mathbf{B} is a symmetric matrix. To account for the symmetry of \mathbf{B} the elements of the matrix is written in terms of 6 independent variables b_1, \dots, b_6 as

$$\mathbf{B} = \begin{bmatrix} b_1 & b_2 & b_4 \\ b_2 & b_3 & b_5 \\ b_4 & b_5 & b_6 \end{bmatrix}$$

The notation $\mathbf{h}_1 = \mathbf{u} = [u_1, u_2, u_3]^T$ and $\mathbf{h}_2 = \mathbf{v} = [v_1, v_2, v_3]^T$ is used. The two conditions for each plane are then given as

$$\mathbf{u}^T \mathbf{B} \mathbf{v} = u_1 v_1 b_1 + (u_1 v_2 + u_2 v_1) b_2 + u_2 v_2 b_3 + (u_1 v_3 + u_3 v_1) b_4 + (u_2 v_3 + u_3 v_2) b_5 + u_3 v_3 b_6$$

and

$$\begin{aligned} \mathbf{u}^T \mathbf{B} \mathbf{u} - \mathbf{v}^T \mathbf{B} \mathbf{v} = & (u_1^2 - v_1^2) b_1 + 2(u_1 u_2 - v_1 v_2) b_2 + (u_2^2 - v_2^2) b_3 \\ & + 2(u_1 u_3 - v_1 v_3) b_4 + 2(u_2 u_3 - v_2 v_3) b_5 + (u_3^2 - v_3^2) b_6 \end{aligned}$$

This is reformulated as $\mathbf{a}_1 \mathbf{b} = 0$ and $\mathbf{a}_2 \mathbf{b} = 0$ where the elements of \mathbf{B} have been stacked in the vector

$$\mathbf{b} = [b_1 \ b_2 \ b_3 \ b_4 \ b_5 \ b_6]^T$$

and

$$\begin{aligned} \mathbf{a}_1 &= [u_1 v_1 \ u_1 v_2 + u_2 v_1 \ u_2 v_2 \ u_1 v_3 + u_3 v_1 \ u_2 v_3 + u_3 v_2 \ u_3 v_3] \\ \mathbf{a}_2 &= [u_1^2 - v_1^2 \ 2(u_1 u_2 - v_1 v_2) \ u_2^2 - v_2^2 \ 2(u_1 u_3 - v_1 v_3) \ 2(u_2 u_3 - v_2 v_3) \ u_3^2 - v_3^2] \end{aligned}$$

There will be 2 conditions for each plane, and 3 planes gives 6 conditions. This leads to the expression

$$\mathbf{A} \mathbf{b} = \mathbf{0}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_{11} \\ \mathbf{a}_{12} \\ \mathbf{a}_{21} \\ \mathbf{a}_{22} \\ \mathbf{a}_{31} \\ \mathbf{a}_{32} \end{bmatrix}$$

A solution for \mathbf{b} is then found with a singular value decomposition of \mathbf{A}

$$\mathbf{A} = \sum_{i=1}^6 \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \quad \mathbf{u}_i \in \mathbb{R}^{2n}, \quad \mathbf{v}_i \in \mathbb{R}^6 \quad (753)$$

which gives $\mathbf{b} = k \mathbf{v}_6$. Then the matrix \mathbf{B} is found from the elements of \mathbf{b} .

Finally the camera matrix \mathbf{K} is found from \mathbf{B} using Cholesky decomposition. Here some care must be exercised as there are different variants of the Cholesky decomposition. The Cholesky decomposition of a symmetric positive definite matrix \mathbf{B} is defined in the reference work [27] as $\mathbf{B} = \mathbf{L} \mathbf{L}^T$ where \mathbf{L} is a lower triangular matrix, that is, a matrix where all the elements above the diagonal are zero. However, in the standard MATLAB function the Cholesky decomposition is defined as $\mathbf{B} = \mathbf{G}^T \mathbf{G}$ where \mathbf{G} is upper triangular, that is, a matrix where all the elements below the diagonal are zero. This is appropriate for this problem because \mathbf{K} and \mathbf{K}^{-1} are upper triangular, for example,

$$\mathbf{K} = \begin{bmatrix} 750 & 0 & 512 \\ 0 & 750 & 640 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{K}^{-1} = \begin{bmatrix} 0.0013 & 0 & -0.6827 \\ 0 & 750 & -0.8533 \\ 0 & 0 & 1 \end{bmatrix}$$

Cholesky decomposition based on $\mathbf{B} = \mathbf{G}^T \mathbf{G}$ where \mathbf{G} is upper triangular then gives \mathbf{K}^{-1} , and \mathbf{K} is finally found by matrix inversion.

Python script

```

import numpy as np

def skewm(r):
    # The skewm symmetric form of a vector
    S = np.array([[0, -r[2], r[1]], [r[2], 0, -r[0]], [-r[1], r[0], 0]])
    return S

def expso3(u):
    R = np.identity(3) + np.sinc(np.linalg.norm(u)/np.pi)*skewm(u) \
        + 0.5*(np.sinc(np.linalg.norm(u)/(2*np.pi)))**2 * skewm(u) @ skewm(u)
    return R

def Rt2T(R,t):
    T = np.identity(4)
    T[0:3,0:3] = R; T[0:3,3] = t
    return T

def rc2H(x,xp):
    # Homography H = [r1,r2,t] from point is a plane
    A1 = np.hstack([x[0,0]*skewm(xp[:,0]), x[1,0]*skewm(xp[:,0]), skewm(xp[:,0])])
    A2 = np.hstack([x[0,1]*skewm(xp[:,1]), x[1,1]*skewm(xp[:,1]), skewm(xp[:,1])])
    A3 = np.hstack([x[0,2]*skewm(xp[:,2]), x[1,2]*skewm(xp[:,2]), skewm(xp[:,2])])
    A4 = np.hstack([x[0,3]*skewm(xp[:,3]), x[1,3]*skewm(xp[:,3]), skewm(xp[:,3])])
    A = np.vstack((A1, A2, A3, A4))
    # Singlar value decomposition
    U,S,V = np.linalg.svd(A);
    # Nullspace solution
    scale = np.sign(V[8,8])*np.linalg.norm(V[8,0:3])
    h1 = V[8,0:3]/scale
    h2 = V[8,3:6]/scale
    h3 = V[8,6:9]/scale
    H = np.zeros((3,3))
    H[:,0] = h1; H[:,1] = h2; H[:,2] = h3
    return H

def Hom2B(H1, H2, H3):
    # Calculate B = inv(K*K') from the homographies H1, H2, H3 of three planes
    # where K is the camera calibration matrix
    # Matrix rows
    A1, A2 = calc_h1h2_rows(H1[:,0],H1[:,1])
    A3, A4 = calc_h1h2_rows(H2[:,0],H2[:,1])
    A5, A6 = calc_h1h2_rows(H3[:,0],H3[:,1])

```

```

# A7 = [0 1 0 0 0 0]'; % B(1,2) = B(2,1) = 0;
# A8 = [1 0 -1 0 0 0]'; % B(1,1) = B(2,2);
A = np.vstack((A1, A2, A3, A4, A5, A6))
U,Sigma,Vt = np.linalg.svd(A)
V = Vt.T
B = np.zeros((3,3))
B[0,0] = V[0,5];
B[0,1] = V[1,5];
B[1,1] = V[2,5];
B[0,2] = V[3,5];
B[1,2] = V[4,5];
B[2,2] = V[5,5];
B[1,0] = B[0,1];
B[2,0] = B[0,2];
B[2,1] = B[1,2];

B = B/B[2,2];
return B

def calc_h1h2_rows(u,v):
    # Calculate constraint rows A1' and A2' for single view homography
    # H of plane from the original constraints h1'*B*h2 = 0
    # h1'*B*h1 - h2'*B*h2 = 0 where B = inv(K*K')
    # h1 = H(:,1), h2 = H(:,2)
    # The matrix
    # B = [b1 b2 b4;
    #       b2 b3 b5;
    #       b4 b5 b6]
    # is stacked in the array b = [b1 b2 ... b6]
    # New constraints:
    # A1*b = 0 and A2*b = 0

    A1 = [u[0]*v[0],
           u[0]*v[1]+u[1]*v[0],
           u[1]*v[1],
           u[0]*v[2]+u[2]*v[0],
           u[1]*v[2]+u[2]*v[1],
           u[2]*v[2]]
    A2 = [u[0]*u[0]-v[0]*v[0],
           2*u[0]*u[1]-2*v[0]*v[1],
           u[1]*u[1]-v[1]*v[1],
           2*u[0]*u[2]-2*v[0]*v[2],
           2*u[1]*u[2]-2*v[1]*v[2],
           u[2]*u[2]-v[2]*v[2]]
    return A1, A2

```

```

def points2pixels(r,Tco,K):
    # Homogeneous parameters in camera frame
    rc = Tco @ r
    c = np.zeros((3,4))
    # Sensor readings in pixel coordinates
    for i in range (0, 4):
        c[:,i] = K @ rc[0:3:,i]/rc[2,i]
    return c

#Calculate intrinsic camera matrix K from a single view of three planes
#using the homographies of the three planes.
#Reference: Siciliano Section 10.5
#Ma, Soatto, Kosecka, Sastry Section 6.5.3.
#Solution from Hartley and Zisserman, Algorithm 8.2 p. 225.

# 4 Corners of a square
r = np.array([[0, 0, 0, 1], [0.1, 0, 0, 1], [0.1, 0.1, 0, 1], [0, 0.1, 0, 1]])
r = r.T
# Camera parameter matrix
K = np.array([[750, 0, 640], [0, 750, 512], [0, 0, 1]])
ex = np.array([1, 0, 0]); ey = np.array([0, 1, 0]); ez = np.array([0, 0, 1])
# Plane 1
Rco = np.identity(3)
ococ = np.array([0, 0, 0.5])
Tco = Rt2T(Rco,ococ)
c1 = points2pixels(r,Tco,K);
# Plane 2
Rco = expso3(0*ez) @ expso3(np.pi/4*ex) @ expso3(np.pi/4*ey)
ococ = np.array([0, 0, 0.5])
Tco = Rt2T(Rco,ococ)
c2 = points2pixels(r,Tco,K);
#Plane 3
Rco = expso3(np.pi/3*ez) @ expso3(-np.pi/3*ex) @ expso3(-np.pi/3*ey)
ococ = np.array([0, 0, 0.5])
Tco = Rt2T(Rco,ococ)
c3 = points2pixels(r,Tco,K);

# Given r and c1,c2,c3, calculate the homographies for the 3 planes
H1 = rc2H(r,c1)
H2 = rc2H(r,c2)
H3 = rc2H(r,c3)
# Calculate B = inv(K*K')
B = Hom2B(H1, H2, H3);
# Solve B = Kinv'*Kinv where K is upper diagonal
K = np.linalg.inv(np.linalg.cholesky(B));

```

```

# Normalize
K = K.transpose()/K[2,2]

np.set_printoptions(formatter={'float': '{: 0.2f}'.format})
print ('Estiamted kamera parameter matrix:')
print (K)

MATLAB script

% Calculate intrinsic camera matrix K from a single view of three planes
% using the homographies of the three planes.
% Reference: Siciliano Section 10.5
% Ma, Soatto, Kosecka, Sastry Section 6.5.3.
% Solution from Hartley and Zisserman, Algorithm 8.2 p. 225.

% 4 Corners of a square
r = [[0 0 0 1]', [0.1 0 0 1]', [0.1 0.1 0 1]', [0 0.1 0 1']];
% Camera parameter matrix
K = [750      0      640 ; 0      750      512 ; 0          0          1];
%Plane 1
Rco = eye(3); ococ = [0 0 0.5]'; Tco = [Rco ococ; 0 0 0 1];
% Generate synthetic pixel readings c1, c2 and c3 for three planes
c1 = points2pixels(r,Tco,K);
%Plane 2
Rco = AngleAxis2R(0, [0 0 1]')*AngleAxis2R(pi/4, [1 0 0]')...
    *AngleAxis2R(pi/4, [0 1 0]');
ococ = [0 0 0.5]'; Tco = [Rco ococ; 0 0 0 1];
c2 = points2pixels(r,Tco,K);
%Plane 3
Rco = AngleAxis2R(pi/3, [0 0 1]')*AngleAxis2R(-pi/3, [1 0 0]')...
    *AngleAxis2R(-pi/3, [0 1 0]');
ococ = [0 0 0.5]'; Tco = [Rco ococ; 0 0 0 1];
c3 = points2pixels(r,Tco,K);

% Given r and c1,c2,c3, calculate the homographies for the 3 planes
H1 = rc2H(r,c1);
H2 = rc2H(r,c2);
H3 = rc2H(r,c3);
% Calculate B = inv(K*K')
B = Hom2B(H1, H2, H3);
% Solve B = Kinv'*Kinv where K is upper diagonal
K = inv(chol(B));
% Normalize
K = K/K(3,3)

%%%%%%%%%%%%%
function H = rc2H(x,xp)

```

```

% Planar homography using sensor readings and object data
A1 = [x(1,1)*Skew(xp(:,1)) x(2,1)*Skew(xp(:,1)) Skew(xp(:,1))];
A2 = [x(1,2)*Skew(xp(:,2)) x(2,2)*Skew(xp(:,2)) Skew(xp(:,2))];
A3 = [x(1,3)*Skew(xp(:,3)) x(2,3)*Skew(xp(:,3)) Skew(xp(:,3))];
A4 = [x(1,4)*Skew(xp(:,4)) x(2,4)*Skew(xp(:,4)) Skew(xp(:,4))];
A = [A1; A2; A3; A4];

%Singlar value decomposition
[U,S,V] = svd(A);

%Nullspace solution
zeta1 = sign(V(9,9)* 1/sqrt(V(1:3,9)'*V(1:3,9)));
zeta2 = sign(V(9,9)* 1/sqrt(V(4:6,9)'*V(4:6,9)));
h1 = zeta1 * V(1:3,9);
h2 = zeta2 * V(4:6,9);
h3 = zeta2 * V(7:9,9);
H = [h1 h2 h3];

%%%%%%%%%%%%%
function B = Hom2B(H1, H2, H3)
% Calculate B = inv(K*K') from the homographies H1, H2, H3 of three planes
% where K is the camera calibration matrix

% Matrix rows
[A1, A2] = calc_h1h2_rows(H1(:,1),H1(:,2));
[A3, A4] = calc_h1h2_rows(H2(:,1),H2(:,2));
[A5, A6] = calc_h1h2_rows(H3(:,1),H3(:,2));
% A7 = [0 1 0 0 0 0]'; % B(1,2) = B(2,1) = 0;
% A8 = [1 0 -1 0 0 0]'; % B(1,1) = B(2,2);

A = [ A1 A2 A3 A4 A5 A6]';

%Singlar value decomposition
[U,Sigma,V] = svd(A);

% Extracting B from last column of V
B = zeros(3,3);
B(1,1) = V(1,6);
B(1,2) = V(2,6);
B(2,2) = V(3,6);
B(1,3) = V(4,6);
B(2,3) = V(5,6);
B(3,3) = V(6,6);
B(2,1) = B(1,2);
B(3,1) = B(1,3);

```

```

B(3,2) = B(2,3);
B = B/B(3,3);

%%%%%%%
function [A1, A2] = calc_h1h2_rows(u,v)
% Calculate constraint rows A1' and A2' for single view homography
% H of plane from the original constraints
% h1'*B*h2 = 0
% h1'*B*h1 - h2'*B*h2 = 0
% where B = inv(K*K')
% h1 = H(:,1), h2 = H(:,2)
% The matrix
% B = [b1 b2 b4;
%       b2 b3 b5;
%       b4 b5 b6]
% is stacked in vector
% b = [b1 b2 ... b6]'
% New constraints:
% A1*b = 0 and A2*b = 0

A1 = [u(1)*v(1); u(1)*v(2)+u(2)*v(1) ; ...
       u(2)*v(2); u(1)*v(3)+u(3)*v(1) ; ...
       u(2)*v(3)+u(3)*v(2); u(3)*v(3)];
A2 = [u(1)*u(1)-v(1)*v(1); ...
       2*u(1)*u(2)-2*v(1)*v(2);...
       u(2)*u(2)-v(2)*v(2); ...
       2*u(1)*u(3)-2*v(1)*v(3);...
       2*u(2)*u(3)-2*v(2)*v(3); ...
       u(3)*u(3)-v(3)*v(3)];

%%%%%%
function c = points2pixels(r, Tco,K)
%Homogeneous parameters in camera frame
rc = Tco*r;
%Camera matrix
P = K*[1 0 0 0; 0 1 0 0; 0 0 1 0];
%Sensor readings in pixel coordinates
for i = (1:4)
    c(:,i) = P*rc(:,i)/rc(3,i);
end

%%%%%%
function R = AngleAxis2R(theta, r)
S = [0 -r(3) r(2); ...
      r(3) 0 -r(1);...
      -r(2) r(1) 0];

```

```

R = cos(theta) * eye(3) + sin(theta)*S + (1-cos(theta))*r*r';

%%%%%
function S = Skew(k)
S = [0 -k(3) k(2); ...
      k(3) 0 -k(1);...
      -k(2) k(1) 0];

```

16.8 Intrinsic calibration with checkerboard planes: Bouguet's method

Bouguet's method for intrinsic calibration [12] is implemented in a Matlab calibration toolbox. The method gives a direct solution for the camera parameter matrix \mathbf{K} which is used as a starting point for an iterative optimization of the solution for \mathbf{K} . The main idea of Bouguet's method is to use the vanishing points of the checkerboard planes to determine the image of the absolute conic $\mathbf{B} = (\mathbf{K}\mathbf{K}^T)^{-1}$. The basic observation which the method is based on is that the homography that is determined for each checkerboard plane is $\mathbf{H} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}]$, where the two first columns are the two first columns for the rotation matrix $\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$ from the camera frame to the checkerboard frame. This means that \mathbf{r}_1 and \mathbf{r}_2 are orthogonal, which gives the condition

$$\mathbf{r}_1^T \mathbf{r}_2 = 0 \quad (744)$$

In addition, since \mathbf{r}_1 and \mathbf{r}_2 are of unit length, the diagonals of the square defined by \mathbf{r}_1 and \mathbf{r}_2 will be orthogonal, which gives the condition

$$(\mathbf{r}_1 + \mathbf{r}_2)^T (\mathbf{r}_1 - \mathbf{r}_2) = 0 \quad (745)$$

It is seen that the first condition (744) is the same as the condition (749) used in Zhang's method, while the second condition (745) is equivalent to the second condition (750) in Zhang's method. The difference in the formulation is Bouguet's focus on the fact that \mathbf{r}_1 and \mathbf{r}_2 point at vanishing points in orthogonal directions, and that also $\mathbf{r}_1 + \mathbf{r}_2$ and $\mathbf{r}_1 - \mathbf{r}_2$ point at vanishing points in orthogonal directions.

Example

The toolbox of Bouguet has many option for which intrinsic parameters to calibrate. One option which will be described in the following is that the position of the optical center u_0 and v_0 is known, and that the skew $s = 0$. Then the only parameters to be found are k_1 and k_2 , and if the principal point (u_0, v_0) is compensated for, the image of the absolute conic is

$$\mathbf{B} = \begin{bmatrix} 1/k_1^2 & 0 & 0 \\ 0 & 1/k_2^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (746)$$

Let $\mathbf{z}_1 = \mathbf{r}_1$, $\mathbf{z}_2 = \mathbf{r}_2$, $\mathbf{z}_3 = \mathbf{r}_1 + \mathbf{r}_2$ and $\mathbf{z}_4 = \mathbf{r}_1 - \mathbf{r}_2$, and let $\mathbf{z}_i = [a_i, b_i, c_i]^T$ for $i = 1 \dots 4$. Then the conditions (744) and (745) give

$$\mathbf{z}_1^T \mathbf{B} \mathbf{z}_2 = \frac{a_1 a_2}{k_1^2} + \frac{b_1 b_2}{k_2^2} + c_1 c_2 = 0 \quad (747)$$

$$\mathbf{z}_3^T \mathbf{B} \mathbf{z}_4 = \frac{a_3 a_4}{k_1^2} + \frac{b_3 b_4}{k_2^2} + c_3 c_4 = 0 \quad (748)$$

and the the parameter are found to be

$$k_1 = \sqrt{\frac{a_1 a_2 b_3 b_4 - a_3 a_4 b_1 b_2}{b_1 b_2 c_3 c_4 - b_3 b_4 c_1 c_2}} \quad (759)$$

$$k_2 = \sqrt{\frac{a_1 a_2 b_3 b_4 - a_3 a_4 b_1 b_2}{a_3 a_4 c_3 c_4 - a_1 a_2 c_3 c_4}} \quad (760)$$

□

16.9 Calibration from three vanishing points

Images of man-made structures like buildings will have parallel lines, and it may be possible to identify three sets of parallel lines that are orthogonal. As an example, there may be one set of parallel vertical lines, and one set of parallel lines along the front of the building, and one set of parallel lines along one of the the sides of the building. Each of the three sets of parallel line will define a vanishing point as illustrated in Figure 68. This means that three vanishing points $(\mathbf{v}_1, 0)$, $(\mathbf{v}_2, 0)$ and $(\mathbf{v}_3, 0)$ can be found, where \mathbf{v}_1 , \mathbf{v}_2 and \mathbf{v}_3 are orthogonal unit vectors. Then the 3 conditions

$$\mathbf{v}_1^T \mathbf{v}_2 = \mathbf{v}_2^T \mathbf{v}_3 = \mathbf{v}_3^T \mathbf{v}_1 = 0 \quad (761)$$

will give the 3 conditions

$$\mathbf{z}_1^T \mathbf{B} \mathbf{z}_2 = \mathbf{z}_2^T \mathbf{B} \mathbf{z}_3 = \mathbf{z}_3^T \mathbf{B} \mathbf{z}_1 = 0 \quad (762)$$

where $\mathbf{z}_i = \mathbf{K} \mathbf{R} \mathbf{v}_i$ is the image of \mathbf{v}_i . This gives 3 condition for each image. At least two images will be required to find a general \mathbf{K} with 5 parameters. If it is assumed that the skew is zero, and that $k_1 = k_2$, there are only 3 parameters in the camera parameter matrix, and it is possible to find the parameters from one images.

16.10 Calibration by decomposition of the camera matrix

If the camera matrix

$$\mathbf{C} = \mathbf{K} [\mathbf{R} \ \mathbf{t}] = [\mathbf{K} \mathbf{R} \ \mathbf{K} \mathbf{t}] \quad (763)$$

can be found from a calibration procedure based on, e.g., point observations

$$z \tilde{\mathbf{p}} = \mathbf{C} \tilde{\mathbf{r}}_{o,p}^o \quad (764)$$

Then it is possible to decompose the camera matrix \mathbf{C} so that the matrices \mathbf{K} and \mathbf{R}_o^c and the vector \mathbf{t}_{co}^c are recovered.

The matrix $\mathbf{C} = \{C_{ij}\}$ is found from $\tilde{\mathbf{p}} = [u, v, 1]^T$ and $\tilde{\mathbf{r}}_{o,p}^o = [x, y, z, 1]^T$ by considering

$$zu = C_{11}x + C_{12}y + C_{13}z + C_{14} \quad (765)$$

$$zv = C_{21}x + C_{22}y + C_{23}z + C_{24} \quad (766)$$

$$z = C_{31}x + C_{32}y + C_{33}z + C_{34} \quad (767)$$

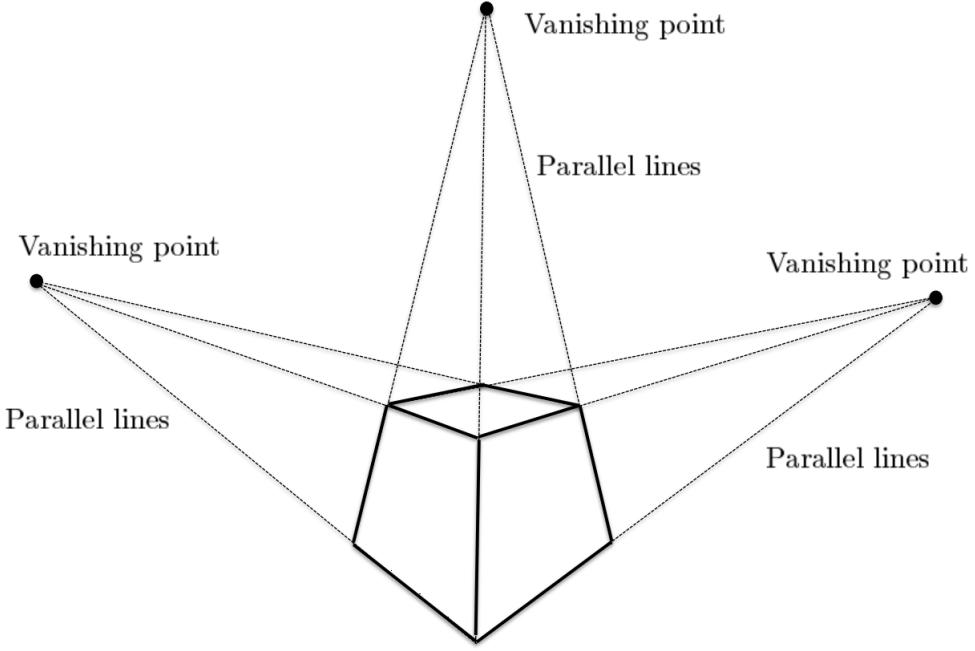


Figure 68: Three vanishing points that are found from defined by three sets of parallel lines.

Here z is given by the last equation, and insertion of this expression in the two first equations gives

$$C_{11}x + C_{12}y + C_{13}z + C_{14} - C_{31}xu - C_{32}yu - C_{33}zu - C_{34}u = 0 \quad (768)$$

$$C_{21}x + C_{22}y + C_{23}z + C_{24} - C_{31}xv - C_{32}yv - C_{33}zv - C_{34}v = 0 \quad (769)$$

Then \mathbf{C} can be found from

$$\mathbf{A}\mathbf{c} = \mathbf{0} \quad (770)$$

where $\mathbf{c} = [C_{11}, C_{12}, \dots, C_{34}]^T$ and

$$\mathbf{A} = \begin{bmatrix} x & y & z & 1 & 0 & 0 & 0 & -xu & -yu & -zu & -u \\ 0 & 0 & 0 & 0 & x & y & z & 1 & -xv & -yv & -zv & -v \end{bmatrix} \quad (771)$$

The matrix \mathbf{C} has 12 unknowns, where 11 are independent, and 6 point mappings are required to solve for \mathbf{C} . Note that the points cannot be coplanar. Therefore, a 3D calibration rig is typically used where the points are on 3 orthogonal planes.

Suppose that the camera matrix has been found to be

$$\mathbf{C} = [\mathbf{M} \quad \mathbf{n}] \quad (772)$$

where \mathbf{M} are the 3 first columns of \mathbf{C} , and \mathbf{n} is the last column. Then the problem is to find \mathbf{K} and \mathbf{R} from

$$\mathbf{M} = \mathbf{K}\mathbf{R} \quad (773)$$

with \mathbf{M} known, and then to find \mathbf{t} from $\mathbf{n} = \mathbf{Kt}$ where \mathbf{n} is known, and \mathbf{K} has been found in the first step.

The decomposition of $\mathbf{M} = \mathbf{KR}$ can be calculated due to the properties of the matrices \mathbf{K} and \mathbf{R} . The camera parameter matrix has the structure

$$\mathbf{K} = \begin{bmatrix} k_u & k_s & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (774)$$

where a skew factor k_s is included. The skew factor will often be assumed to be zero, which is the case if the coordinate axes of the pixel coordinates are orthogonal. The skew factor k_s is proportional to $\cot \theta$, where θ is the angle between the axes of u and v .

All the elements below the diagonal are zero, and the matrix is said to be upper-diagonal. Moreover, all the elements on the diagonal are positive. The inverse camera parameter matrix is

$$\mathbf{K}^{-1} = \begin{bmatrix} 1/k_u & -k_s/(k_u k_v) & -u_0/k_u + v_0 k_s/(k_u k_v) \\ 0 & 1/k_v & -v_0/k_v \\ 0 & 0 & 1 \end{bmatrix} \quad (775)$$

which is upper triangular and has positive elements on the diagonal. The rotation matrix \mathbf{R} is an orthogonal matrix with determinant equal to unity. The inverse $\mathbf{R}^{-1} = \mathbf{R}^T$ is also an orthogonal matrix with determinant equal to unity.

The decomposition can be found using QR decomposition. This is a decomposition of a matrix \mathbf{A} given by

$$\mathbf{A} = \mathbf{QU} \quad (776)$$

where \mathbf{Q} is an orthogonal matrix, and \mathbf{U} is an upper triangular matrix. Then, if \mathbf{A} is known and the diagonal elements of \mathbf{U} are positive, a unique solution \mathbf{Q} and \mathbf{U} is found by QR decomposition, which is implemented as a library function in Matlab and Python. The decomposition of the camera matrix $\mathbf{M} = \mathbf{KR}$ is not in the form of a QR problem, however, it can be formulated as a QR problem by inverting the matrix equation, which gives

$$\mathbf{M}^{-1} = \mathbf{R}^T \mathbf{K}^{-1} \quad (777)$$

This is a QR problem, as the matrix \mathbf{R}^T is orthogonal, and the matrix \mathbf{K}^{-1} is upper diagonal with positive elements on the diagonal. Therefore, \mathbf{R}^T and \mathbf{K}^{-1} are found from QR decomposition of \mathbf{M}^{-1} , and \mathbf{R} and \mathbf{K} are then recovered.

An alternative solution is based on reordering the elements through flipping and transposing the matrices. A Matlab function for this is

```
function [K R] = rqFlip(M)
    [R,K] = qr(flipud(M)');
    K = flipud(K');
    K = fliplr(K);
    R = flipud(R');
```

Library functions for QR will in general not ensure that the diagonal elements of the upper diagonal matrix are positive. Therefore, it will be necessary to perform sign changes in a systematic way to get the right result. This can be done with a sign matrix

$$\mathbf{S} = \text{diag}(\text{sgn}(K_{11}), \text{sgn}(K_{22}), \text{sgn}(K_{33})) \quad (778)$$

The sign change is then done with

$$\mathbf{K}_s = \mathbf{KS} \quad (779)$$

$$\mathbf{R}_s = \mathbf{SR} \quad (780)$$

Then $\mathbf{K}_s \mathbf{R}_s = \mathbf{KSSR} = \mathbf{KR} = \mathbf{M}$, where the matrix \mathbf{K}_s has positive elements on the diagonal, and \mathbf{R}_s is a rotation matrix.

MATLAB script

```
% Input parameters
Ka = [1500 5 640; 0 1490 512; 0 0 1];
Ra = Rotx(pi/4)* Roty(-pi/3) * Rotz(pi/8);
M = Ka*Ra;

[K,R] = rqInv(M); % QR with inverse matrices
%[K,R] = rqFlip(P); % QR with flipped matrices

% Change signs to positive on diagonal of K
S = diag(sign(diag(K)));
K = K * S
R = S * R % K*R = K*S*S*R

% Test
deltaK = K - Ka % Zero matrix expected
deltaR = Ra'*R % Identity matrix expected

%%%%%
function [K R] = rqInv(M)
    [invR, invK] = qr(inv(M));
    K = inv(invK); R = inv(invR);
```

Python script

The calculation is simpler in Python since there is a RQ decomposition available in scipy.

```
import numpy as np
from scipy import linalg

def skewm(r):
    # The skewm symmetric form of a vector
    return np.array([[0, -r[2], r[1]], [r[2], 0, -r[0]], [-r[1], r[0], 0]])

def expso3(u):
```

```

    return np.identity(3) + np.sinc(np.linalg.norm(u)/np.pi)*skewm(u) \
    + 0.5*(np.sinc(np.linalg.norm(u)/(2*np.pi)))**2 * skewm(u) @ skewm(u)

# Input parameters
Ka = np.array([[1500, 5, 640], [0, 1490, 512], [0, 0, 1]]);
ex = np.array([1, 0, 0]); ey = np.array([0, 1, 0]); ez = np.array([0, 0, 1])
Ra = expso3(np.pi/4*ex) @ expso3(-np.pi/3*ey) @ expso3(np.pi/8*ez)

M = Ka @ Ra
K, R = linalg.rq(M)

# Change signs to positive on diagonal of K
S = np.zeros((3,3))
d = np.sign(np.diag(K))
S[0,0] = d[0]
S[1,1] = d[1]
S[2,2] = d[2]
K = K @ S
R = S @ R

print(Ka, '\n', Ra, '\n')
print(K, '\n', R)

```

17 Stereo vision with calibrated cameras

17.1 Introduction

Stereo vision is important in applications where the use of two cameras makes it possible to get depth information from 2D image data. Intuitively, based on how human vision takes advantage of stereo information, it is to be expected that stereo vision will be important also in computer vision. The stereo vision problem requires additional geometric insight and specialized geometric methods that will be introduced and explained in the following. Major references are found in [30, 41]. The presentation will first focus on stereo vision with calibrated cameras, which is based on a description with normalized image coordinates. An important tool in this regard is the essential matrix, which is derived from geometric arguments, and which is important in connection with point correspondences between two images in a stereo arrangement.

17.2 The geometry of a stereo arrangement

Consider two cameras imaging the same scene in a stereo arrangement. The description of the stereo arrangement is based the definition where the camera frame of camera 1 is frame 1, and the camera frame of camera 2 is frame 2. The position of a point P in the frame of camera 1 is \vec{r}_1 , which is the vector from the origin of frame 1 to the point P . The corresponding coordinate vector in the coordinates of frame 1 is $\mathbf{r}_1^1 = [x_1, y_1, z_1]^T$. The position in of the point P in frame 2 is \vec{r}_2 , and the corresponding coordinate vector in the coordinates of frame 2 is $\mathbf{r}_2^2 = [x_2, y_2, z_2]^T$. The position of frame 1 relative to camera frame 2 is given by the vector

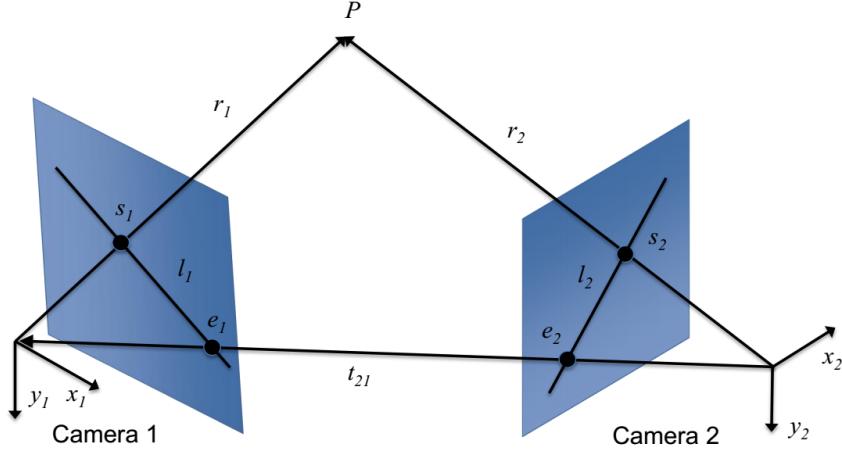


Figure 69: Two cameras in a stereo arrangement. The point P is imaged in normalized image coordinates as s_1 in image 1 and s_2 in image 2. The figure shows the epipole e_1 in image 1 which is the image of the origin of frame 2, and the epipole e_2 in image 1 which is the image of the origin of frame 2. Also the epipolar lines ℓ_1 in image 1 and ℓ_2 in image 2 are shown.

\vec{t}_{21} , which has the coordinate vector \mathbf{t}_{21}^2 in the coordinates of frame 2. It follows that

$$\vec{r}_2 = \vec{r}_1 + \vec{t}_{21} \quad (781)$$

The stereo arrangement is shown in Figure 69.

The transformation from frame 2 to frame 1 is given by

$$\mathbf{T}_1^2 = \begin{bmatrix} \mathbf{R}_1^2 & \mathbf{t}_{21}^2 \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (782)$$

This means that the position of point P in the two cameras are related by the homogeneous position vectors $\tilde{\mathbf{r}}_1^1 = [(\mathbf{r}_1^1)^T, 1]^T$ and $\tilde{\mathbf{r}}_2^2 = [(\mathbf{r}_2^2)^T, 1]^T$ as

$$\tilde{\mathbf{r}}_2^2 = \mathbf{T}_1^2 \tilde{\mathbf{r}}_1^1 \quad (783)$$

The corresponding Euclidean formulation of the transformation is

$$\mathbf{r}_2^2 = \mathbf{R}_1^2 \mathbf{r}_1^1 + \mathbf{t}_{21}^2 \quad (784)$$

The imaging model is

$$\lambda_1 \mathbf{s}_1 = \mathbf{r}_1^1, \quad \lambda_2 \mathbf{s}_2 = \mathbf{r}_2^2 \quad (785)$$

$$\mathbf{p}_1 = \mathbf{K}_1 \mathbf{s}_1, \quad \mathbf{p}_2 = \mathbf{K}_2 \mathbf{s}_2 \quad (786)$$

where $\mathbf{s}_1 = [s_{x1}, s_{y1}, 1]^T$ and $\mathbf{s}_2 = [s_{x2}, s_{y2}, 1]^T$ are the homogeneous vectors of normalized image coordinates for the two cameras, $\mathbf{p}_1 = [u_1, v_1, 1]^T$ and $\mathbf{p}_2 = [u_2, v_2, 1]^T$ are the homogeneous pixel coordinates. The depth coordinates are expressed as $\lambda_1 = z_1$ and $\lambda_2 = z_2$.

17.3 Epipolar constraints and the essential matrix

The epipolar constraint of the stereo imaging results from the observation that the vectors \vec{r}_1 , \vec{r}_2 and \vec{t}_{21} are in the same plane (Figure 69). This means that the triple scalar product of the three vectors is zero, that is,

$$\vec{r}_2 \cdot (\vec{t}_{21} \times \vec{r}_1) = 0 \quad (787)$$

The coordinate form of this in the coordinates of frame 2 is

$$(\mathbf{r}_2^2)^T (\mathbf{t}_{21}^2)^\times \mathbf{R}_1^2 \mathbf{r}_1^1 = 0 \quad (788)$$

This is the epipolar constraint of the stereo imaging system. The epipolar constraint is usually expressed in terms of the essential matrix, which is defined by

$$\mathbf{E} = (\mathbf{t}_{21}^2)^\times \mathbf{R}_1^2 \quad (789)$$

Then from $\lambda_1 \mathbf{s}_1 = \mathbf{r}_1^1$ and $\lambda_2 \mathbf{s}_2 = \mathbf{r}_2^2$ it follows that the epipolar constraint can be written in the standard form with the homogeneous normalized image coordinates as

$$\mathbf{s}_2^T \mathbf{E} \mathbf{s}_1 = 0 \quad (790)$$

The essential matrix is \mathbf{E} the product of a skew symmetric matrix and a rotation matrix. The skew symmetric matrix is of rank 2, while the rotation matrix is full rank. This implies that the essential matrix is of rank 2.

The epipolar constraint is formulated in terms of homogeneous vectors \mathbf{s}_1 and \mathbf{s}_2 , and the constraint can be scaled freely without changing its geometric implications. This means that the essential matrices \mathbf{E} and $\alpha \mathbf{E}$ will be equivalent for all nonzero scalars α . In particular, it is noted that \mathbf{E} and $-\mathbf{E}$ are equivalent essential matrices.

17.4 The essential matrix and epipolar lines

The epipolar line in image 2 is defined by

$$\ell_2 = \mathbf{E} \mathbf{s}_1 \quad (791)$$

It is seen that the line is defined in terms of the essential matrix and the point \mathbf{s}_1 in image 1. This means that the essential matrix gives a mapping from a point \mathbf{s}_1 in image 1 to the epipolar line ℓ_2 in image 2. The significance of the epipolar line follows from the calculation

$$\mathbf{s}_2^T \ell_2 = \mathbf{s}_2^T \mathbf{E} \mathbf{s}_1 = 0 \quad (792)$$

This means that the point \mathbf{s}_2 in image 2 is on the epipolar line ℓ_2 defined by the point \mathbf{s}_1 in image 1.

In the same way the epipolar line

$$\ell_1 = \mathbf{E}^T \mathbf{s}_2 \quad (793)$$

in image 1 is defined as a mapping from the point \mathbf{s}_2 in image 2 to the epipolar line ℓ_1 in image 1. This line has the property that

$$\ell_1^T \mathbf{s}_1 = \mathbf{s}_2^T \mathbf{E} \mathbf{s}_1 = 0 \quad (794)$$

This means that the point s_1 is on the epipolar line ℓ_1 defined by the point s_2 in image 2.

A useful consequence of this is that if s_1 and s_2 represent the same point P in 3D, then the point s_1 must be on the epipolar line ℓ_1 , and the point s_2 must be on the epipolar line ℓ_2 .

Of special interest is the case where the point to be imaged is the origin of one of the cameras. If the point to be imaged with camera 2 is the origin of camera 1, then $\mathbf{r}_2^2 = \mathbf{t}_{21}^2$. The corresponding normalized image point is called the epipole \mathbf{e}_2 , which is defined by

$$\lambda_2 \mathbf{e}_2 = \mathbf{t}_{21}^2 \quad (795)$$

where λ_2 is the depth coordinate. Note that the epipole is a normalized image coordinate vector, which means that the third coordinate of \mathbf{e}_2 is unity. This means that λ_2 is equal to the third coordinate of \mathbf{t}_{21}^2 . In the same way the image in camera 1 of the origin of camera 2 is $\mathbf{r}_1^1 = -\mathbf{R}_2^1 \mathbf{t}_{21}^2$ which corresponds to the epipole \mathbf{e}_1 defined by

$$\lambda_1 \mathbf{e}_1 = -(\mathbf{R}_1^2)^T \mathbf{t}_{21}^2 \quad (796)$$

where normalization of \mathbf{e}_1 is ensured by setting λ_1 equal to the third coordinate of the vector $(\mathbf{R}_1^2)^T \mathbf{t}_{21}^2$.

It is seen that the two epipoles satisfy the conditions

$$\mathbf{E} \mathbf{e}_1 = -(\mathbf{t}_{21}^2)^\times \mathbf{R}_1^2 \mathbf{R}_2^1 \mathbf{t}_{21}^2 = -(\mathbf{t}_{21}^2)^\times \mathbf{t}_{21}^2 = \mathbf{0} \quad (797)$$

$$\mathbf{E}^T \mathbf{e}_2 = -\mathbf{R}_2^1 (\mathbf{t}_{21}^2)^\times \mathbf{t}_{21}^2 = \mathbf{0} \quad (798)$$

This result in combination with the definition of the epipolar lines show that

$$\ell_1^T \mathbf{e}_1 = \mathbf{0}, \quad \ell_2^T \mathbf{e}_2 = \mathbf{0} \quad (799)$$

which means that the epipole in image 1 is on the epipolar line in image 1, and the epipole in image 2 is on the epipolar line in image 2. Note that this is valid in image 1 for any epipolar line $\ell_1 = \mathbf{E}^T \mathbf{s}_2$, and in image 2 for any epipolar line $\ell_2 = \mathbf{E} \mathbf{s}_1$. This shows that the epipolar point \mathbf{e}_1 in image 1 is on all epipolar lines ℓ_1 , which means that all epipolar lines in image 1 must be through the epipole \mathbf{e}_1 . In the same way, all epipolar lines in image 2 must be through the epipole \mathbf{e}_2 .

From this it follows that if all epipolar lines in a picture intersect at one point, then this point is the image of the other camera.

To sum up: Suppose that s_1 is the image of a point in camera 1, and that s_2 is the image of the same point in camera 2. Then the epipolar line $\ell_1 = \mathbf{E}^T \mathbf{s}_2$ in image 1 will be a line through the image point s_1 and the epipole \mathbf{e}_1 . In the same way the epipolar line $\ell_2 = \mathbf{E} \mathbf{s}_1$ in image 2 will be a line through the image point s_2 and the epipole \mathbf{e}_2 .

Example

An example of a stereo arrangement is described in the following with reference to Figure 71. A frame 0 is defined with the z axis horizontal, and the y axis downwards. Frame 1 is the camera frame of camera 1, and frame 2 is the camera frame of camera 2. Frames 1 and 2 are defined by the displacements

$$\mathbf{T}_1^0 = \begin{bmatrix} \mathbf{R}_1^0 & \mathbf{t}_{01}^0 \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{T}_2^0 = \begin{bmatrix} \mathbf{R}_2^0 & \mathbf{t}_{02}^0 \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (800)$$

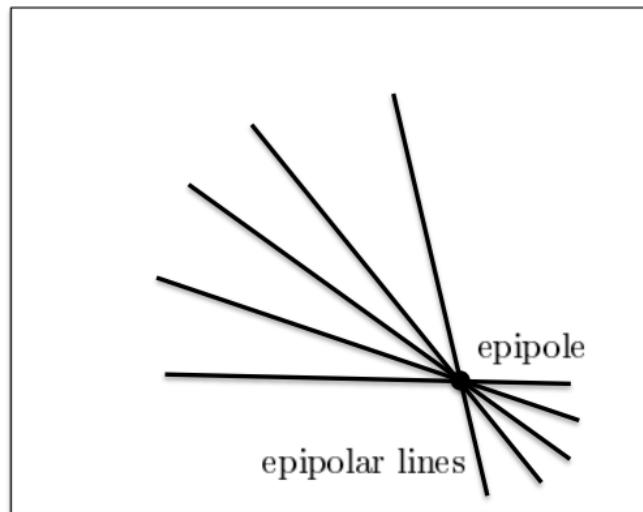


Figure 70: All epipolar lines in an image will pass through the epipole of the image.

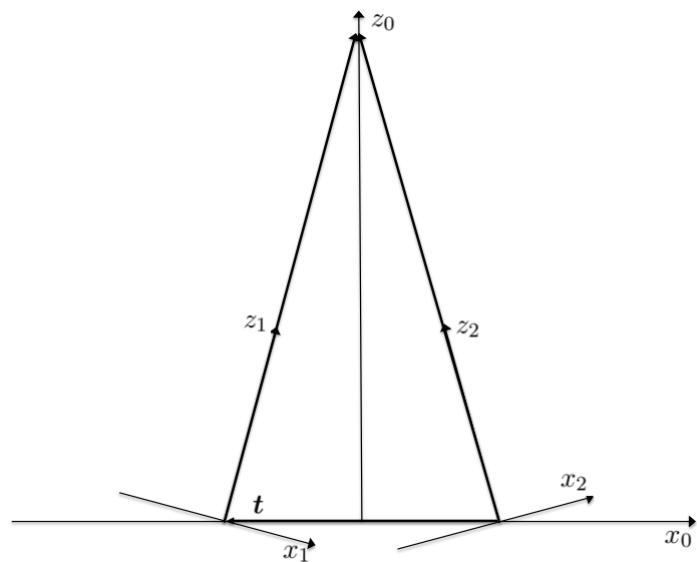


Figure 71: Stereo arrangement.

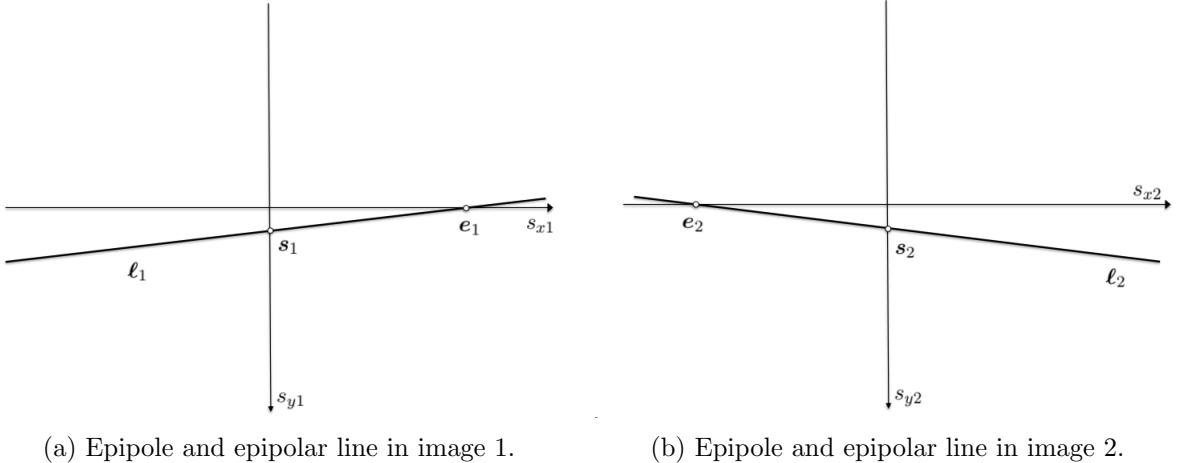


Figure 72: Epipoles and epipolar lines

where $\mathbf{R}_1^0 = \mathbf{R}_y(\pi/12)$, $\mathbf{t}_{01}^0 = -0.5\mathbf{e}_x$, $\mathbf{R}_2^0 = \mathbf{R}_y(-\pi/12)$ and $\mathbf{t}_{02}^0 = 0.5\mathbf{e}_x$ where $\mathbf{R}_y(\alpha)$ is the rotation matrix of a rotation α about the y axis and $\mathbf{e}_x = [1, 0, 0]^T$. Then the displacement between the cameras is given by

$$\mathbf{R} = \mathbf{R}_1^2 = (\mathbf{R}_2^0)^T \mathbf{R}_1^0 = \mathbf{R}_y(\pi/6) = \begin{bmatrix} 0.866 & 0 & 0.5 \\ 0 & 1 & 0 \\ -0.5 & 0 & 0.866 \end{bmatrix} \quad (801)$$

and

$$\mathbf{t} = \mathbf{t}_{21}^2 = (\mathbf{R}_2^0)^T (\mathbf{t}_{01}^0 - \mathbf{t}_{02}^0) = \mathbf{R}_y(\pi/12) \mathbf{e}_x = \begin{bmatrix} -0.9659 \\ 0 \\ 0.2588 \end{bmatrix} \quad (802)$$

It follows that the essential matrix is

$$\mathbf{E} = \mathbf{t}^\times \mathbf{R} = \begin{bmatrix} 0 & -0.2588 & 0 \\ -0.2588 & 0 & 0.9659 \\ 0 & -0.9659 & 0 \end{bmatrix} \quad (803)$$

which is a matrix of rank 2. The epipoles are found to be

$$\mathbf{e}_1 = -\frac{\mathbf{R}^T \mathbf{t}}{\lambda_1} = \begin{bmatrix} 3.7321 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{e}_2 = \frac{\mathbf{t}}{\lambda_2} = \begin{bmatrix} -3.7321 \\ 0 \\ 1 \end{bmatrix} \quad (804)$$

Suppose that the point P that is imaged with both cameras has position

$$\mathbf{r}_0^0 = \begin{bmatrix} 0 \\ 0.2 \\ \frac{1}{2 \tan(\pi/12)} \end{bmatrix} \quad (805)$$

Then the 3D position in the camera frames are

$$\mathbf{r}_1^1 = (\mathbf{R}_1^0)^T (\mathbf{r}_0^0 - \mathbf{t}_{01}^0) = \begin{bmatrix} 0 \\ 0.2000 \\ 1.93190 \end{bmatrix}, \quad \mathbf{r}_2^2 = (\mathbf{R}_2^0)^T (\mathbf{r}_0^0 - \mathbf{t}_{02}^0) = \begin{bmatrix} 0 \\ 0.2000 \\ 1.93190 \end{bmatrix} \quad (806)$$

The resulting normalized image coordinates are

$$s_1 = \frac{\mathbf{r}_1^1}{1.93190} = \begin{bmatrix} 0 \\ 0.1035 \\ 1.0000 \end{bmatrix}, \quad s_2 = \frac{\mathbf{r}_2^2}{1.93190} = \begin{bmatrix} 0 \\ 0.1035 \\ 1.0000 \end{bmatrix} \quad (807)$$

The epipolar lines corresponding to s_1 and s_2 are

$$\ell_1 = \mathbf{E}^T s_2 = \begin{bmatrix} -0.0268 \\ -0.9659 \\ 0.1000 \end{bmatrix}, \quad \ell_2 = \mathbf{E}s_1 = \begin{bmatrix} -0.0268 \\ 0.9659 \\ -0.1000 \end{bmatrix} \quad (808)$$

The epipolar lines and the epipoles are shown in Figure 72. It is verified that s_1 and e_1 are on ℓ_1 , and the s_2 and e_2 are on ℓ_2 by calculating $s_1^T e_1$ and $s_2^T e_2$ and scaling the results, or by calculating the inner products $s_1^T \ell_1$, $e_1^T \ell_1$, $s_2^T \ell_2$ and $e_2^T \ell_2$, which are all equal to zero.

```
% Script to generate a stereo arrangement, and to calculate the
% essential matrix, epipoles and epipolar lines
skewm =@(u) [0 -u(3) u(2); u(3) 0 -u(1);-u(2) u(1) 0];
vex3 = @(u) [u(3,2); u(1,3); u(2,1)];
expso3 =@(u) eye(3)+sinc(norm(u)/pi)*skewm(u) ...
    +0.5*(sinc(norm(u)/(2*pi)))^2*(skewm(u))^2;
ex = [1;0;0]; ey = [0;1;0]; ez = [0;0;1];

R01 = expso3((15/180)*pi*ey); R02 = expso3((-15/180)*pi*ey);
t01 = -0.5*ex; t02 = -t01;
T01 = [R01 t01; 0 0 0 1]; T02 = [R02 t02; 0 0 0 1];
T10 = inv(T01); T20 = inv(T02);
T21 = T20*T01; R21 = T21(1:3,1:3); t21 = T21(1:3,4);
% Displacement from camera 2 to camera 2
R = R21; t = t21; tc = skewm(t);
% Esential matrix
E = tc*R;
% Epipoles
e1 = -R'*t; e2 = t;
e1 = e1/e1(3); e2 = e2/e2(3); % Normalized homogeneous vectors

% 3D point to be imaged
r0 = [0;0.2;0.5/tan((15/180)*pi)];
r1 = R01'*(r0 - t01); r2 = R02'*(r0 - t02)
s1 = r1/r1(3); s2 = r2/r2(3); % Normalized homogeneous vectors
% Resulting epipolar lines
L1 = E'*s2; L2 = E*s1;

%Test if epipolar line goes through epipole and image point (zeros expected)
L1'*s1
L1'*e1
L2'*s2
L2'*e2
```

□

17.5 The essential matrix for standard camera matrices

Let the two cameras be described by the camera models

$$\lambda_1 \mathbf{s}_1 = \mathbf{P}_1 \tilde{\mathbf{r}}, \quad \lambda_2 \mathbf{s}_2 = \mathbf{P}_2 \tilde{\mathbf{r}} \quad (809)$$

Here \mathbf{s}_i is the homogeneous vector of normalized image coordinates, $\tilde{\mathbf{r}} = [\mathbf{r}^T, 1]^T$ is the homogeneous position vector of the world point P to be imaged.

Let the camera matrices be given by the standard camera matrices

$$\mathbf{P}_1 = [\mathbf{I} \mid \mathbf{0}], \quad \mathbf{P}_2 = [\mathbf{R} \mid \mathbf{t}] \quad (810)$$

This gives

$$\lambda_1 \mathbf{s}_1 = \mathbf{r}, \quad \lambda_2 \mathbf{s}_2 = \mathbf{R}\mathbf{r} + \mathbf{t} \quad (811)$$

The definition of these camera matrices implies that the world point P is given in the frame of camera 1, and that the position vector is $\mathbf{r} = \mathbf{r}_1^1$. The rotation matrix is $\mathbf{R} = \mathbf{R}_1^2$, and the translation is $\mathbf{t} = \mathbf{t}_{21}^2$, which is the translation from frame 2 to frame 1 in the coordinates of frame 2. Here frame 1 is the camera frame of camera 1 and frame 2 is the camera frame of camera 2. The homogeneous transformation matrix

$$\mathbf{T}_1^2 = \begin{bmatrix} \mathbf{R}_1^2 & \mathbf{t}_{21}^2 \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3) \quad (812)$$

is then the displacement from frame 2 to frame 1. The displacement given by \mathbf{T}_1^2 is also written (\mathbf{R}, \mathbf{t}) . The scaling parameters are given by $\lambda_1 = z_1$ and $\lambda_2 = z_2$, where z_1 is the z coordinate of \mathbf{r} , and z_2 is the z coordinate of $\mathbf{R}\mathbf{r} + \mathbf{t}$.

The resulting essential matrix is written

$$\mathbf{E} = \mathbf{t}^\times \mathbf{R} \quad (813)$$

The epipole \mathbf{e}_1 in image 1 and the epipole \mathbf{e}_2 in image 2, which are both given in normalized image coordinates, satisfy

$$\lambda_1 \mathbf{e}_1 = -\mathbf{R}^T \mathbf{t}, \quad \lambda_2 \mathbf{e}_2 = \mathbf{t} \quad (814)$$

The essential matrix is used in the epipolar constraint $\mathbf{s}_2^T \mathbf{E} \mathbf{s}_1 = 0$, and the in the mapping from a point in one image to a epipolar line in the other image according to $\ell_2 = \mathbf{E} \mathbf{s}_1$ and $\ell_1 = \mathbf{E}^T \mathbf{s}_2$. The vectors that are involved are homogeneous, and this means that the essential matrix can be scaled, so that the essential matrices \mathbf{E} and $\alpha \mathbf{E}$ will be equivalent for all $\alpha \neq 0$. This means that the insertion of $\mathbf{t} = \lambda_2 \mathbf{e}_2$ (813) and a rescaling gives an equivalent expression for the essential matrix

$$\mathbf{E} = \mathbf{e}_2^\times \mathbf{R} \quad (815)$$

where the epipole \mathbf{e}_2 in image 2 is used in place of \mathbf{t} .

The essential matrix \mathbf{E} is an element of the set \mathcal{E} called the essential space, which is defined by

$$\mathcal{E} = \{ \mathbf{E} | \mathbf{E} = \mathbf{t}^\times \mathbf{R}, \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3 \} \quad (816)$$

17.6 Recovery of rotation and translation from the essential matrix

Consider the problem of determining the displacement (\mathbf{R}, \mathbf{t}) which satisfies $\mathbf{E} = \mathbf{t}^\times \mathbf{R}$ for a given essential matrix $\mathbf{E} \in \mathcal{E}$. Then, it is shown in [30] that if the SVD of the essential matrix is $\mathbf{E} = \mathbf{U}\Sigma\mathbf{V}^T$ there are two possible solutions given by

$$(\mathbf{t}_1^\times, \mathbf{R}_1) = (\mathbf{U}\mathbf{Z}\mathbf{U}^T, \mathbf{U}\mathbf{W}^T\mathbf{V}^T) \quad (817)$$

$$(\mathbf{t}_2^\times, \mathbf{R}_2) = (-\mathbf{U}\mathbf{Z}\mathbf{U}^T, \mathbf{U}\mathbf{W}\mathbf{V}^T) \quad (818)$$

where

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \in SO(3), \quad \mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (819)$$

It is seen that $\mathbf{W} = \mathbf{R}_z(\pi/2)$ is the rotation by angle $\pi/2$ about the z axis, while $\mathbf{Z} = -\mathbf{z}^\times$, where $\mathbf{z} = [0, 0, 1]^T$. It is noted that in [41] the negative of this matrix is termed \mathbf{Z} .

The matrices \mathbf{U} and \mathbf{V} from the SVD are orthogonal but not necessarily rotation matrices. This means that the solutions \mathbf{R}_1 and \mathbf{R}_2 may be reflection matrices. This is fixed somewhat pragmatically by changing the sign of the matrix.

The geometric interpretation of the two solutions can be investigated by observing that since \mathbf{U} is an orthogonal matrix satisfying $\mathbf{U}\mathbf{U}^T = \mathbf{I}$, it follows that

$$\mathbf{t}_1^\times = -\mathbf{U}\mathbf{z}^\times\mathbf{U}^T = -(\mathbf{U}\mathbf{z})^\times \quad (820)$$

$$\mathbf{t}_2^\times = \mathbf{U}\mathbf{z}^\times\mathbf{U}^T = (\mathbf{U}\mathbf{z})^\times \quad (821)$$

and it follows that

$$\mathbf{t}_1 = -\mathbf{U}\mathbf{z}, \quad \mathbf{t}_2 = \mathbf{U}\mathbf{z} \quad (822)$$

where $\mathbf{U} \in O(3)$ is a rotation or reflection matrix. It is seen that $\mathbf{t}_1 = -\mathbf{t}_2$, which means that the two solutions for the translation are in opposite directions. The essential matrix is homogeneous, and may be scaled by a nonzero scalar. This means that both \mathbf{E} and $-\mathbf{E}$ represent the same essential matrix, and each of them gives the two solutions in (817) and (818). This means that there are 4 possible solution corresponding to an essential matrix, and these solutions are given by

$$[\mathbf{R}_1, \mathbf{t}], \quad [\mathbf{R}_1, -\mathbf{t}], \quad [\mathbf{R}_2, \mathbf{t}] \quad \text{or} \quad [\mathbf{R}_2, -\mathbf{t}] \quad (823)$$

where $\mathbf{t} = \mathbf{t}_1$.

Example

```
% Function for computing the 4 solutions of the displacement (R,t)
% for a given essential matrix E
```

```
function [R1,R2,t1,t2] = recoverFromEssential(E)
%RECOVERFROMESSENTIAL Recovers rotation matrix R and translation t so
% that E = tc*R where tc is the skew symmetric form tc.
% Two solutions are found for R and two for t. The resulting four
% solutions are (R1,t1), (R1,t2), (R2,t1) and (R2,t2)
```

```

W = [0 -1 0; 1 0 0; 0 0 1];
Z = [0 1 0; -1 0 0; 0 0 0];

[U,S,V] = svd(E);
sigma = (S(1,1) + S(2,2))*0.5;
E = U*diag(sigma,sigma,0)*V';
[U,S,V] = svd(E);

% Two solutions for recovered rotation matrix
R1 = U*W*V'; R2 = U*W'*V';
% Enforce det(R) = 1 to get rotation matrix
if det(R1) < 0
    R1 = -R1;
end
if det(R2) < 0
    R2 = -R2;
end
tc = U*Z*U';
t1 = [tc(3,2); tc(1,3); tc(2,1)]; t2 = -t1;
end

```

□

17.7 A closer look at the recovery of displacement from the essential matrix

In this section a more detailed analysis on the recovery problem is presented based on [41]. The following result will be shown: nonzero matrix $\mathbf{E} \in \mathbb{R}^{3 \times 3}$ is an essential matrix if and only if \mathbf{E} has a singular value decomposition which satisfies

$$\mathbf{E} = \mathbf{U}\Sigma\mathbf{V}^T, \quad \Sigma = \text{diag}(\sigma, \sigma, 0), \quad \mathbf{U}, \mathbf{V} \in SO(3) \quad (824)$$

which means that \mathbf{E} has two equal singular values $\sigma > 0$ and one singular value equal to zero.

First it is shown that an essential matrix has a SVD given by (824). Suppose that an essential matrix $\mathbf{E} \in \mathcal{E}$ is given. Then there exists at least one displacement $(\mathbf{R}, \mathbf{t}) \in SE(3)$ so that $\mathbf{E} = \mathbf{t}^\times \mathbf{R}$. Let the singular value decomposition of the essential matrix be $\mathbf{E} = \mathbf{U}\Sigma\mathbf{V}^T$. Then from $\mathbf{V}^T \mathbf{V} = \mathbf{I}$ it follows that

$$\mathbf{E}\mathbf{E}^T = \mathbf{U}\Sigma\mathbf{V}^T \mathbf{V}\Sigma\mathbf{U}^T = \mathbf{U}\Sigma^2\mathbf{U}^T \quad (825)$$

Define the rotation matrix $\mathbf{R}_0 \in SO(3)$ which rotates the vector \mathbf{t} so that it is aligned with the z axis. This means that $\mathbf{R}_0 \mathbf{t} = \tau \mathbf{z}$, where $\mathbf{z} = [0, 0, 1]^T$ and $\tau = \|\mathbf{t}\|$. Then

$$\mathbf{t}^\times = \tau(\mathbf{R}_0^T \mathbf{z})^\times = \tau \mathbf{R}_0^T \mathbf{z}^\times \mathbf{R}_0 \quad (826)$$

This leads to

$$\mathbf{E}\mathbf{E}^T = \mathbf{t}^\times \mathbf{R} \mathbf{R}^T \mathbf{t}^{\times T} = \mathbf{t}^\times \mathbf{t}^{\times T} = \mathbf{R}_0^T (\tau^2 \mathbf{z}^\times \mathbf{z}^{\times T}) \mathbf{R}_0 = \mathbf{R}_0^T \begin{bmatrix} \tau^2 & 0 & 0 \\ 0 & \tau^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{R}_0 \quad (827)$$

where it is used that

$$\mathbf{z}^\times \mathbf{z}^{\times T} = -\mathbf{z}^\times \mathbf{z}^\times = \mathbf{z}^T \mathbf{z} \mathbf{I} - \mathbf{z} \mathbf{z}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (828)$$

From (825) and (827) it follows that

$$\boldsymbol{\Sigma} = \begin{bmatrix} \tau & 0 & 0 \\ 0 & \tau & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (829)$$

It is tempting to conclude that \mathbf{U} is equal to \mathbf{R}_0^T , but it will be seen in the following that this is not the case.

Next let

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \in SO(3) \quad (830)$$

be the rotation matrix of a rotation by an angle $\pi/2$ about the z axis, and define

$$\mathbf{Z} = \mathbf{z}^\times = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (831)$$

It is straightforward to verify by direct computation that

$$\mathbf{W} \boldsymbol{\Sigma} = \tau \mathbf{Z} \quad (832)$$

This gives

$$\mathbf{E} = \mathbf{t}^\times \mathbf{R} = \mathbf{R}_0^T (\tau \mathbf{z}^\times) \mathbf{R}_0 \mathbf{R} = \mathbf{R}_0^T \mathbf{W} \boldsymbol{\Sigma} \mathbf{R}_0 \mathbf{R} \quad (833)$$

This means that \mathbf{E} has a singular value decomposition given by

$$\boldsymbol{\Sigma} = \text{diag}(\tau, \tau, 0), \quad \mathbf{U} = \mathbf{R}_0^T \mathbf{W}, \quad \mathbf{V} = (\mathbf{R}_0 \mathbf{R})^T \quad (834)$$

where $\mathbf{U}, \mathbf{V} \in SO(3)$. It is noted that this gives

$$\mathbf{E} \mathbf{E}^T = \mathbf{R}_0^T \mathbf{W} \boldsymbol{\Sigma} \mathbf{R}_0 \mathbf{R} \mathbf{R}^T \mathbf{R}_0^T \boldsymbol{\Sigma} \mathbf{W}^T \mathbf{R}_0 = \mathbf{R}_0^T \boldsymbol{\Sigma}^2 \mathbf{R}_0^T \quad (835)$$

where it is used that $\mathbf{W} \boldsymbol{\Sigma}^2 \mathbf{W}^T = \boldsymbol{\Sigma}^2$, which is verified by direct computation. This is consistent with (827).

Next, it is shown that if the SVD is given by (824), then \mathbf{E} is an essential matrix. Suppose that the SVD of \mathbf{E} is given by (824). Define the two displacements $(\mathbf{R}_1, \mathbf{t}_1) \in SE(3)$ and $(\mathbf{R}_2, \mathbf{t}_2) \in SE(3)$ by

$$\mathbf{t}_1^\times = \tau \mathbf{U} \mathbf{Z} \mathbf{U}^T, \quad \mathbf{R}_1 = \mathbf{U} \mathbf{W}^T \mathbf{V}^T \quad (836)$$

$$\mathbf{t}_2^\times = \tau \mathbf{U} \mathbf{Z}^T \mathbf{U}^T, \quad \mathbf{R}_2 = \mathbf{U} \mathbf{W} \mathbf{V}^T \quad (837)$$

Then, since $\mathbf{U} \mathbf{U}^T = \mathbf{I}$ and $\tau \mathbf{Z} \mathbf{W}^T = \tau \mathbf{Z}^T \mathbf{W} = \boldsymbol{\Sigma}$ it follows that

$$\mathbf{t}_1^\times \mathbf{R}_1 = \tau \mathbf{U} \mathbf{Z} \mathbf{W}^T \mathbf{V}^T = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T = \mathbf{E} \quad (838)$$

$$\mathbf{t}_2^\times \mathbf{R}_2 = \tau \mathbf{U} \mathbf{Z}^T \mathbf{W} \mathbf{V}^T = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T = \mathbf{E} \quad (839)$$

This means that \mathbf{E} is the essential matrix for the two displacements $(\mathbf{R}_1, \mathbf{t}_1)$ and $(\mathbf{R}_2, \mathbf{t}_2)$.

17.8 The twisted pair ambiguity

The first solution (2005) of the translation in can be written

$$\mathbf{t}_1^\times = \tau \mathbf{U} \mathbf{Z} \mathbf{U}^T = \mathbf{U}(\tau \mathbf{z}^\times) \mathbf{U}^T = (\mathbf{U} \tau \mathbf{z})^\times \quad (840)$$

while the second solution (2006) can be written

$$\mathbf{t}_2^\times = \tau \mathbf{U} \mathbf{Z}^T \mathbf{U}^T = \mathbf{U}(-\tau \mathbf{z}^\times) \mathbf{U}^T = -(\mathbf{U} \tau \mathbf{z})^\times \quad (841)$$

This shows that the two solutions for the translation have opposite sign, that is,

$$\mathbf{t}_2 = -\mathbf{t}_1 \quad (842)$$

The orthogonal matrices of the SVD (824) are given by $\mathbf{U} = \mathbf{R}_0^T \mathbf{W}$ and $\mathbf{V} = (\mathbf{R}_0 \mathbf{R})^T$. The first solution (2005) for the rotation is then

$$\mathbf{R}_1 = \mathbf{U} \mathbf{W}^T \mathbf{V}^T = \mathbf{R}_0^T \mathbf{W} \mathbf{W}^T \mathbf{R}_0 \mathbf{R} = \mathbf{R} \quad (843)$$

The second solution (2006) is

$$\mathbf{R}_2 = \mathbf{U} \mathbf{W} \mathbf{V}^T = \mathbf{R}_0^T \mathbf{W} \mathbf{W} \mathbf{R}_0 \mathbf{R} = \mathbf{R}_0^T \mathbf{R}_z(\pi) \mathbf{R}_0 \mathbf{R} \quad (844)$$

When the two solutions are compared, it is seen that \mathbf{R}_2 has an additional rotation $\mathbf{R}_0^T \mathbf{R}_z(\pi) \mathbf{R}_0$. This is a rotation by an angle π about the axis of the translation \mathbf{t} . This follows since the rotation starts with the rotation \mathbf{R}_0^T , which aligns the z axis with the \mathbf{t} vector, followed by a rotation π about the current z axis, and then a rotation \mathbf{R}_0 , and finally the rotation \mathbf{R} . This means that the two solutions differ by a rotation by an angle π about the axis \mathbf{t} vector. The two solutions $(\mathbf{R}_1, \mathbf{t}_1)$ and $(\mathbf{R}_2, -\mathbf{t}_1)$ are therefore referred to as the twisted pair ambiguity.

17.9 Comments on the SVD of the essential matrix

First, consider the case where $\mathbf{R} = \mathbf{I}$, which means that the two cameras have parallel optical axes. Then

$$\mathbf{E} = \mathbf{t}^\times = \tau(\mathbf{R}_0^T \mathbf{z})^\times = \tau \mathbf{R}_0^T \mathbf{z}^\times \mathbf{R}_0 = \mathbf{R}_0^T \mathbf{W} \Sigma \mathbf{R}_0 \quad (845)$$

This shows that a possible SVD is given by $\mathbf{U} = \mathbf{R}_0^T \mathbf{W} \in SO(3)$ and $\mathbf{V} = \mathbf{R}_0^T \in SO(3)$.

Next, consider the case of a general $\mathbf{R} \in SO(3)$ it follows that

$$\mathbf{E} = \mathbf{t}^\times \mathbf{R} = \mathbf{R}_0^T \mathbf{W} \Sigma \mathbf{R}_0 \mathbf{R} \quad (846)$$

where a possible SVD is given by $\mathbf{U} = \mathbf{R}_0^T \mathbf{W} \in SO(3)$ and $\mathbf{V} = (\mathbf{R}_0 \mathbf{R})^T \in SO(3)$.

It is noted that the SVD of \mathbf{E} is not unique, and that

$$\mathbf{E} = \mathbf{R}_0^T \mathbf{W} \Sigma \mathbf{R}_0 \mathbf{R} = \mathbf{R}_0^T \mathbf{W} \Sigma \mathbf{W}^T \mathbf{W} \mathbf{R}_0 \mathbf{R} = \mathbf{R}_0^T \Sigma \mathbf{W} \mathbf{R}_0 \mathbf{R} \quad (847)$$

implies that the alternative SVD with $\mathbf{U}_1 = \mathbf{R}_0^T$ and $\mathbf{V}_1 = (\mathbf{W} \mathbf{R}_0 \mathbf{R})^T$ will be valid since $\mathbf{E} = \mathbf{U}_1 \Sigma \mathbf{V}_1^T$.

The matrix \mathbf{R}_0 can be computed by noting that $\mathbf{R}_0 \mathbf{t} = \tau \mathbf{z}$. This means that the rows of the rotation matrix

$$\mathbf{R} = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{bmatrix} \quad (848)$$

must satisfy $\mathbf{r}_1^T \mathbf{t} = \mathbf{r}_2^T \mathbf{t} = 0$, and $\mathbf{r}_3^T \mathbf{t} = \tau$. The last row is therefore the unit vector

$$\mathbf{r}_3 = \frac{\mathbf{t}}{\tau} \quad (849)$$

The remaining two rows must be orthogonal to \mathbf{t} . A method for computing \mathbf{r}_1 is to find a vector \mathbf{a} which is different from \mathbf{t} , and then find \mathbf{r}_1 as the unit vector along $\mathbf{a} \times \mathbf{t}$. Finally, \mathbf{r}_2 is found as the unit vector along $\mathbf{t} \times \mathbf{r}_1$. A pragmatic method for computing a vector which is different from \mathbf{t} is to find the largest component t_{\max} in \mathbf{t} , and set this component to zero in \mathbf{a} . In addition, the two remaining components in \mathbf{a} are set equal to t_{\max} .

Example

```

skewm =@(u) [0 -u(3) u(2); u(3) 0 -u(1);-u(2) u(1) 0];
expso3 =@(u) eye(3)+sinc(norm(u)/pi)*skewm(u) ...
+0.5*(sinc(norm(u)/(2*pi)))^2*(skewm(u))^2;

t = [5;4;3]; tc = skewm(t); tau = norm(t);
R = expso3([1;2;0]);
W = expso3([0;0;pi/2]);
% Essential matrix
E = tc*R;
[U1,S,V1] = svd(E);
%Check if U1 and V1 are rotation matrices
S
det(U1);
det(V1);

% Compute the rotation matrix which rotates t to [0;0;tau]
% Compute a vector a which is different from t
% by setting the largest element in t to zero and setting the
% remaining two components to tmax.
[tmax,max_ind] = max(t);
a = t; a(max_ind) = 0;
a(mod((max_ind),3)+1) = tmax; a(mod((max_ind+1),3)+1) = tmax;
% Compute a vector w1 which is orthogonal to t
w1 = cross(a,t);
% Compute a vector w2 which is orthogobal to t and w2
w2 = cross(t,w1);
r1 = w1/norm(w1),; r2 = w2/norm(w2); r3 = t/tau;
% Assemble the rotation matrix
R0 = [r1'; r2'; r3'];

```

```

% Compute SVD with U,V in SO(3)
U = R0'*W;
V = R'*R0';
Esvd = U*S*V';

% Scale essential matrices by setting element (3,3) to unity
E = E/E(3,3)
Esvd = Esvd/Esvd(3,3)

```

□

17.10 Selection of the recovered displacement from the essential matrix

It has been shown that there are two solutions for the displacement between the cameras in a stereo setup. In addition, both \mathbf{E} and $-\mathbf{E}$ are possible solutions for the essential matrix. This means that there are four possible solutions for the displacement (\mathbf{R}, \mathbf{t}) . As explained in [30], given an essential matrix $\mathbf{E} = \mathbf{U}\text{diag}(1, 1, 0)\mathbf{V}^T$ and the first camera given by $\mathbf{P} = [\mathbf{I}, \mathbf{0}]$ there are 4 possible solutions for the second camera \mathbf{P}_2 given by

$$[\mathbf{R}_1, \mathbf{t}], \quad [\mathbf{R}_1, -\mathbf{t}], \quad [\mathbf{R}_2, \mathbf{t}] \quad \text{or} \quad [\mathbf{R}_2, -\mathbf{t}] \quad (850)$$

The right solution is selected so that the points have positive depth and therefore are visible from both cameras. This is referred to as the cheirality condition.

17.11 Calculation of the essential matrix with the eight-point algorithm

The essential matrix can be calculated from $n \geq 8$ image correspondences with the eight-point algorithm, which is presented in detail in [41, 30]. This solution is similar to the DLT method for the calculation of homographies, and relies on a least squares solution of a set of linear equations where the singular value decomposition is used.

Suppose that \mathbf{s}_1^i and \mathbf{s}_2^i are corresponding homogeneous points in calibrated image coordinates so that the epipolar constraint is

$$\mathbf{s}_2^T \mathbf{E} \mathbf{s}_1 = 0 \quad (851)$$

where $\mathbf{E} = \{e_{ij}\}$ is the essential matrix. The essential matrix has 9 elements. Due to the homogeneous characteristics of the matrix there are 8 independent elements. This means that 8 equations are required to determine the elements of \mathbf{E} up to a scale.

Let the coordinates of the calibrated image vectors be given by $\mathbf{s}_1 = [x_1, y_1, z_1]^T$ and $\mathbf{s}_2 = [x_2, y_2, z_2]^T$, and defined the column vector

$$\mathbf{e} = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \end{bmatrix}, \quad \mathbf{e}_i = \begin{bmatrix} e_{1i} \\ e_{2i} \\ e_{3i} \end{bmatrix} \quad (852)$$

where $\mathbf{e}_i, i = 1, 2, 3$ are the column vectors of \mathbf{E} . Then the epipolar constraint can be written

$$[x_2 x_1, x_2 y_1, x_2 z_1, y_2 x_1, y_2 y_1, y_2 z_1, z_2 x_1, z_2 y_1, z_2 z_1] \mathbf{e} = 0 \quad (853)$$

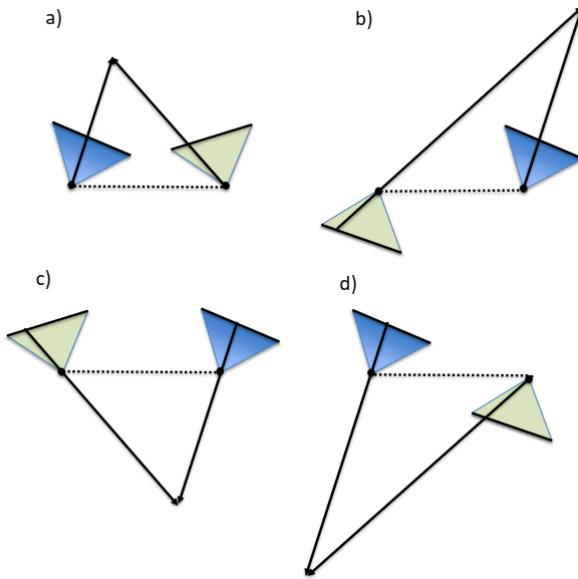


Figure 73: The four solutions for the displacement between the cameras for a given essential matrix. Camera 1, which is in blue, has the same position and orientation in all the four solutions. Camera 2, which is in green has four different displacements. The Euclidean point has been determined by triangulation given the corresponding image coordinate vectors s_1 and s_2 . The only valid solution is a) where the point is visible for both cameras. The solutions a) and b) are a twisted pair, where the second solution has a translation in the opposite direction, and is rotated by an angle π about the translation axis. Also, the solutions c) and d) form a twisted pair.

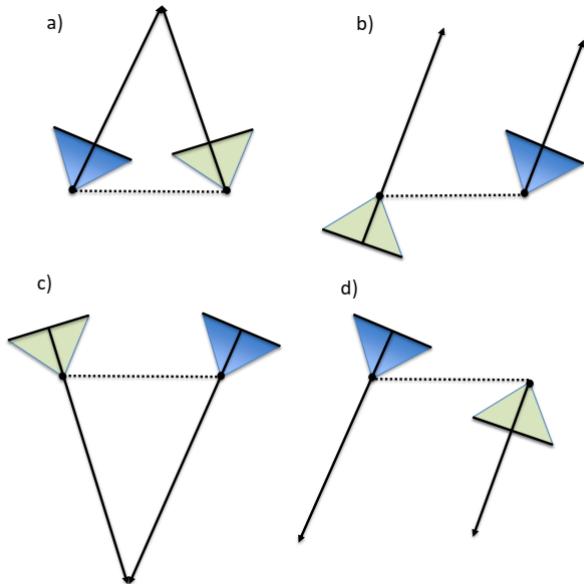


Figure 74: A special case for the four solutions of the displacement of camera 2 given the essential matrix. Solutions b) and d) result in a triangulation problem with parallel lines, where the resulting Euclidean point is a point at infinity. Again, only solution a) is valid.

A solution for \mathbf{E} can then be found from $n \geq 8$ point correspondences $(\mathbf{s}_1^i, \mathbf{s}_2^i)$, which each gives the equation $\mathbf{A}_i \mathbf{e} = 0$, where

$$\mathbf{A}_i = [x_2^i x_1^i, x_2^i y_1^i, \dots, z_2^i z_1^i] \quad (854)$$

The solution is then found from

$$\mathbf{A} \mathbf{e} = \mathbf{0}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_n \end{bmatrix} \quad (855)$$

as $\mathbf{e} = \mathbf{v}_8$, where \mathbf{v}_8 is the the last column in \mathbf{V} found from the singular value decomposition $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$.

Before the calculation is is done, the data sets $\{\mathbf{s}_1^i\}$ and $\{\mathbf{s}_2^i\}$ must be normalized to

$$\mathbf{y}_1^i = \mathbf{N}_1 \mathbf{s}_1^i, \quad \mathbf{y}_2^i = \mathbf{N}_2 \mathbf{s}_2^i \quad (856)$$

so that the inhomogeneous parts of \mathbf{y}_1^i and \mathbf{y}_2^i have the centroid at the origin, and and average distance from the origin of $\sqrt{2}$. It is noted that the scaling matrices \mathbf{N}_1 and \mathbf{N}_2 are similarity transformations which are calculated as in (194). These normalized data set are then used to calculate a normalized \mathbf{E}_n that satisfies

$$\mathbf{y}_2^{iT} \mathbf{E}_n \mathbf{y}_1^i = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{s}_2^{iT} \mathbf{N}_2^T \mathbf{E}_n \mathbf{N}_1 \mathbf{s}_1^i = \mathbf{0} \quad (857)$$

The essential matrix is then recovered as

$$\mathbf{E} = \mathbf{N}_2^T \mathbf{E}_n \mathbf{N}_1 \quad (858)$$

One more correction is required. The essential matrix is of rank 2, and has two nonzero singular values that are equal, and one singular value that is zero. Therefore, to ensure that the computed essential matrix satisfy this condition, the matrix is corrected using the singular value decomposition. For good results this should be done for the normalized essential matrix \mathbf{E}_n . The singular value decomposition is

$$\mathbf{E}_n = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \sigma_3 \mathbf{u}_3 \mathbf{v}_3^T \quad (859)$$

Then the solution which satisfies the condition on the singular values that is closest to \mathbf{E}_n in the Frobenius norm is found by substituting $(\sigma_1 + \sigma_2)/2$ for the two first singular values, and zero for the last. This is called a projection of the matrix. A simpler solution is used, where the freedom of scaling is used, and the solution is

$$\mathbf{E}_{nr} = \mathbf{u}_1 \mathbf{v}_1^T + \mathbf{u}_2 \mathbf{v}_2^T \quad (860)$$

Then the projected essential matrix is computed as

$$\mathbf{E} = \mathbf{N}_2^T \mathbf{E}_{nr} \mathbf{N}_1 \quad (861)$$

Finally, the displacement (\mathbf{R}, \mathbf{t}) can be calculated from \mathbf{E} . There are two solutions $+\mathbf{E}$ and $-\mathbf{E}$ for the essential matrix, and therefore 4 solutions (\mathbf{R}, \mathbf{t}) .

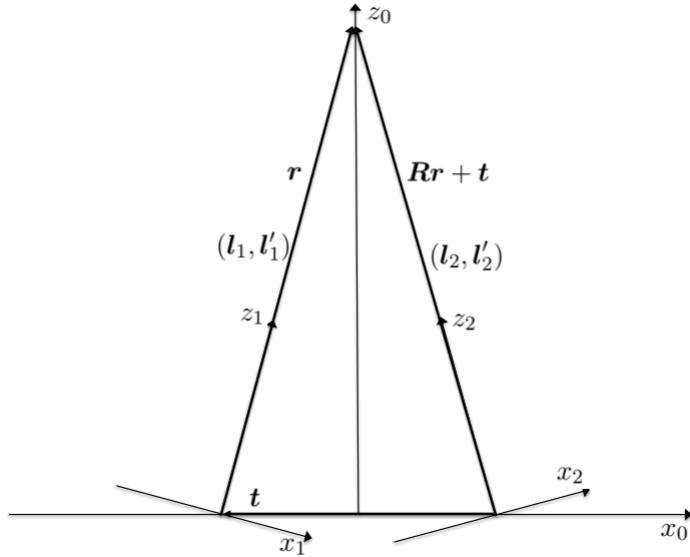


Figure 75: Stereo arrangement. The 3D point is found by triangulation. In this case the depth is positive for both cameras, which shows that this is the valid solution.

17.12 Triangulation to determine the valid displacement of the recovery problem

Triangulation can be used to determine the valid displacement given the 4 solutions from the recovery problem of a given essential matrix. The triangulation determines the position of the 3D point which corresponds to the normalized image coordinates of the two cameras for a given recovery solution. Then for a recovery solution to be valid, it is required that the depth of the 3D point is positive for both cameras, which is necessary for the 3D point to be visible for both cameras. Triangulation can be done with several different techniques. It should be noted that the recovered solutions can be expected to lead to a triangulation problem with parallel or close to parallel lines, which means that the triangulation procedure should be able to handle lines that intersect at a point at infinity.

Example

The stereo arrangement shown in Figure 75 is described in the following. A frame 0 is defined with the z axis horizontal, and the y axis downwards. Frame 1 is the camera frame of camera 1, and frame 2 is the camera frame of camera 2. Frames 1 and 2 are defined by the displacements

$$\mathbf{T}_1^0 = \begin{bmatrix} \mathbf{R}_1^0 & \mathbf{t}_{01}^0 \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{T}_2^0 = \begin{bmatrix} \mathbf{R}_2^0 & \mathbf{t}_{02}^0 \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (862)$$

where $\mathbf{R}_1^0 = \mathbf{R}_y(\pi/12)$, $\mathbf{t}_{01}^0 = -0.5\mathbf{e}_x$, $\mathbf{R}_2^0 = \mathbf{R}_y(-\pi/12)$ and $\mathbf{t}_{02}^0 = 0.5\mathbf{e}_x$ where $\mathbf{R}_y(\alpha)$ is the rotation matrix of a rotation α about the y axis and $\mathbf{e}_x = [1, 0, 0]^T$. Then the displacement between the cameras is given by

$$\mathbf{R} = \mathbf{R}_2^0 = (\mathbf{R}_2^0)^T \mathbf{R}_1^0 = \mathbf{R}_y(\pi/6) = \begin{bmatrix} 0.866 & 0 & 0.5 \\ 0 & 1 & 0 \\ -0.5 & 0 & 0.866 \end{bmatrix} \quad (863)$$

and

$$\mathbf{t} = \mathbf{t}_{21}^2 = (\mathbf{R}_2^0)^T (\mathbf{t}_{01}^0 - \mathbf{t}_{02}^0) = \mathbf{R}_y(\pi/12) \mathbf{e}_x = \begin{bmatrix} -0.9659 \\ 0 \\ 0.2588 \end{bmatrix} \quad (864)$$

It follows that the essential matrix is

$$\mathbf{E} = \mathbf{t}^\times \mathbf{R} = \begin{bmatrix} 0 & -0.2588 & 0 \\ -0.2588 & 0 & 0.9659 \\ 0 & -0.9659 & 0 \end{bmatrix} \quad (865)$$

The rotation matrix \mathbf{R}_0 which rotates the \mathbf{t} vector to the z axis of frame 2 is found by inspection of the figure of the arrangement to be

$$\mathbf{R}_0 = \mathbf{R}_y((5/12)\pi) = \begin{bmatrix} 0.2588 & 0 & 0.9659 \\ 0 & 1.0000 & 0 \\ -0.9659 & 0 & 0.2588 \end{bmatrix} \quad (866)$$

which is verified by

$$\mathbf{R}_0 \mathbf{t} = \begin{bmatrix} 0.2588 & 0 & 0.9659 \\ 0 & 1.0000 & 0 \\ -0.9659 & 0 & 0.2588 \end{bmatrix} \begin{bmatrix} -0.9659 \\ 0 \\ 0.2588 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (867)$$

A reconstruction from a given essential matrix will give the first solution $\mathbf{R}_1 = \mathbf{R}$ and the second solution

$$\mathbf{R}_2 = \mathbf{R}_0^T \mathbf{R}_z(\pi) \mathbf{R}_0 = \begin{bmatrix} 0.866 & 0 & -0.5 \\ 0 & -1 & 0 \\ -0.5 & 0 & -0.866 \end{bmatrix} \quad (868)$$

It is straightforward to verify that

$$\mathbf{t}^\times \mathbf{R}_1 = \mathbf{E}, \quad -\mathbf{t}^\times \mathbf{R}_1 = -\mathbf{E}, \quad \mathbf{t}^\times \mathbf{R}_2 = -\mathbf{E}, \quad -\mathbf{t}^\times \mathbf{R}_2 = \mathbf{E} \quad (869)$$

The twisted pair solution corresponding to Figure 75 is shown in Figure 76.

Example

```
% Script to generate a stereo arrangement, and to calculate the
% 4 possible displacements recovered from an essential matrix
skewm = @(u) [0 -u(3) u(2); u(3) 0 -u(1); -u(2) u(1) 0];
vex3 = @(u) [u(3,2); u(1,3); u(2,1)];
expso3 = @(u) eye(3)+sinc(norm(u)/pi)*skewm(u) ...
    +0.5*(sinc(norm(u)/(2*pi)))^2*(skewm(u))^2;
ex = [1;0;0]; ey = [0;1;0]; ez = [0;0;1];

R01 = expso3((15/180)*pi*ey); R02 = expso3((-15/180)*pi*ey);
t01 = -0.5*ex; t02 = -t01;
T01 = [R01 t01; 0 0 0 1]; T02 = [R02 t02; 0 0 0 1];
T10 = inv(T01); T20 = inv(T02);
T21 = T20*T01; R21 = T21(1:3,1:3); t21 = T21(1:3,4);
```

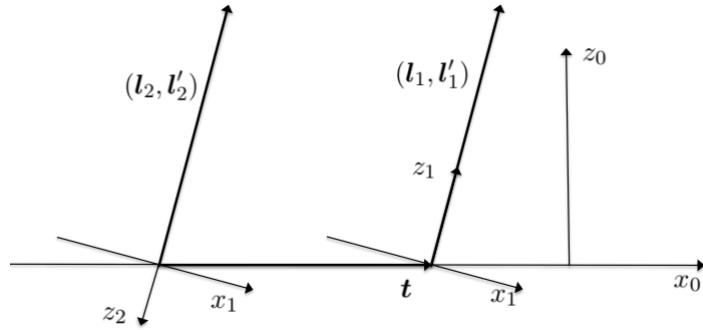


Figure 76: Twisted pair solution for stereo arrangement. Frame 2 has been rotated by an angle π about the \mathbf{t} vector, which is along the x_0 axis, and the translation has changed sign. The situation where the normalized image coordinates are $\mathbf{s}_1 = \mathbf{s}_2 = [0, 0, 1]^T$ is indicated, which gives parallel lines in the triangulation problem. In this case the depth is positive for camera 1 and negative for camera 2, which shows that this is not a valid solution.

```

R = R21; t = t21; tc = skewm(t);
E = tc*R;

% Rotation matrix that rotates t to z
R0 = expso3((75/180)*pi*ey)
R0*t
% Calculation of the second rotation
R2 = R0'*expso3(pi*ez)*R0*R
% Resulting essential matrices
E1 = tc*R
E2 = -tc*R
E3 = tc*R2
E4 = - tc*R2

```

□

17.13 Triangulation using the midpoint

First triangulation using the midpoint of the two lines defined by the image coordinates.

Suppose that a point correspondence $(\mathbf{s}_1, \mathbf{s}_2)$ has been established between the normalized image coordinates \mathbf{s}_1 in image 1 and \mathbf{s}_2 in image 2, which means that the same 3D point is mapped to \mathbf{s}_1 and \mathbf{s}_2 . Then from the camera matrices $\mathbf{P}_1 = [\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}_2 = [\mathbf{R} \mid \mathbf{t}]$ it follows that

$$\lambda_1 \mathbf{s}_1 = \mathbf{r} \quad (870)$$

$$\lambda_2 \mathbf{s}_2 = \mathbf{R}\mathbf{r} + \mathbf{t} \quad (871)$$

This means that the Plücker coordinates of the line from camera 1 to the point is

$$(\mathbf{l}_1, \mathbf{l}'_1) = (\mathbf{s}_1, \mathbf{0}) \quad (872)$$

where the line is referenced to the origin of camera frame 1. The line from camera 2 to the point is given by

$$(\mathbf{l}_2, \mathbf{l}'_2) = (\mathbf{R}^T \mathbf{s}_2, -\mathbf{R}^T (\mathbf{t}^\times \mathbf{s}_2)) \quad (873)$$

Ideally, the two lines should intersect at the 3D point \mathbf{r} , however, the two lines will in general not intersect, and a point must be found as an approximation to \mathbf{r} . A possible solution is to use the midpoint of the common normal, which can be found as described in Section 10.6.

17.14 Linear triangulation by minimizing the algebraic error

Triangulation can also be done with a linear technique. The equations for the calibrated image coordinates is then

$$\begin{aligned} \lambda_1 \mathbf{s}_1 &= \mathbf{P}_1 \tilde{\mathbf{r}} \\ \lambda_2 \mathbf{s}_2 &= \mathbf{P}_2 \tilde{\mathbf{r}} \end{aligned}$$

where $\mathbf{s}_1 = [x_1, y_1, 1]^T$, and $\mathbf{s}_2 = [x_2, y_2, 1]^T$. Let the camera matrices be written in terms of their row vectors as

$$\mathbf{P}_i = \begin{bmatrix} \mathbf{p}_i^{1T} \\ \mathbf{p}_i^{2T} \\ \mathbf{p}_i^{3T} \end{bmatrix} \quad (874)$$

The scale factors can be eliminated by premultiplication with \mathbf{s}_i^\times , which gives $\mathbf{s}_i^\times \mathbf{P}_i \tilde{\mathbf{r}} = \mathbf{0}$. This gives for each camera the set of equations

$$\begin{bmatrix} y_i \mathbf{p}_i^{3T} - \mathbf{p}_i^{2T} \\ -x_i \mathbf{p}_i^{3T} + \mathbf{p}_i^{1T} \\ x_i \mathbf{p}_i^{2T} + y_i \mathbf{p}_i^{1T} \end{bmatrix} \tilde{\mathbf{r}} = \mathbf{0} \quad (875)$$

This gives 3 equations for each camera, where only 3 equations are linearly independent, and one row can be left out. With both cameras this leads to $\mathbf{A} \tilde{\mathbf{r}} = \mathbf{0}$, where

$$\mathbf{A} \tilde{\mathbf{r}} = \begin{bmatrix} x_1 \mathbf{p}_1^{3T} - \mathbf{p}_1^{1T} \\ y_1 \mathbf{p}_1^{3T} - \mathbf{p}_1^{2T} \\ x_2 \mathbf{p}_2^{3T} - \mathbf{p}_2^{1T} \\ y_2 \mathbf{p}_2^{3T} - \mathbf{p}_2^{2T} \end{bmatrix} \tilde{\mathbf{r}} = \mathbf{0} \quad (876)$$

is a matrix of dimension 4×4 . The least-squares solution is

$$\tilde{\mathbf{r}} = c \mathbf{v}_4 \quad (877)$$

where \mathbf{v}_4 is the last column of the matrix \mathbf{V} of the singular value decomposition $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$. The scale of the homogeneous vector $\tilde{\mathbf{r}}$ can then be selected so that the fourth element is unity, which gives $\tilde{\mathbf{r}} = [\mathbf{r}^T, 1]^T$. This means that the Euclidean position \mathbf{r} of the point has been determined without any ambiguity in scale.

17.15 Triangulation using the reprojection error

A third method triangulation is to define a points $\hat{\mathbf{s}}_1$ and $\hat{\mathbf{s}}_2$ which minimize the reprojection error

$$d(\mathbf{s}_1, \hat{\mathbf{s}}_1)^2 + d(\mathbf{s}_2, \hat{\mathbf{s}}_2)^2 \quad (878)$$

subject to $\hat{\mathbf{s}}_2^T \mathbf{E} \hat{\mathbf{s}}_1 = 0$. This means that the points $\hat{\mathbf{s}}_1$ and $\hat{\mathbf{s}}_2$ satisfy the epipolar constraint, which means that the lines defined by $\hat{\mathbf{s}}_1$ and $\hat{\mathbf{s}}_2$ intersect at a 3D point. A method for solving this is presented in [30].

17.16 3D structure recovery

When the standard camera model $\mathbf{P}_1 = [\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}_2 = [\mathbf{R} \mid \mathbf{t}]$ is used, the normalized image coordinates are given by (811) as $\lambda_1 \mathbf{s}_1 = \mathbf{r}$ and $\lambda_2 \mathbf{s}_2 = \mathbf{R}\mathbf{r} + \mathbf{t}$. The two equations can be combined as $\lambda_2 \mathbf{s}_2 = \lambda_1 \mathbf{R}\mathbf{s}_1 + \mathbf{t}$ where λ_1 and λ_2 are depth parameters, which are assumed to be unknown.

Suppose that the displacement $(\mathbf{R}, \gamma\mathbf{t})$ has been determined from the eight-point algorithm where $\mathbf{E} = \mathbf{t}^\times \mathbf{R}$ has been found and the displacement $(\mathbf{R}, \gamma\mathbf{t})$ has been found by selecting one of the solutions of the twisted pair ambiguity. Note that the translation will only be determined up to a scale $\gamma\mathbf{t}$ where $\gamma \neq 0$ is an unknown scalar. Then the normalized image coordinates are given by

$$\lambda_1 \mathbf{s}_1 = \mathbf{r} \quad (879)$$

$$\lambda_2 \mathbf{s}_2 = \mathbf{R}\mathbf{r} + \gamma\mathbf{t} \quad (880)$$

It is assumed that n point correspondences $(\mathbf{s}_1^i, \mathbf{s}_2^i)$, $i = 1, \dots, n$ are given, while \mathbf{R} and \mathbf{t} has been found from the eight-point algorithm. Then the equations

$$\lambda_2^i \mathbf{s}_2^i = \lambda_1^i \mathbf{R}\mathbf{s}_1^i + \gamma\mathbf{t} \quad (881)$$

for the point correspondences are linear in the unknown parameters λ_1^i , λ_2^i and γ . The depth parameters λ_1^i and λ_2^i will be the depths for the same point \mathbf{r}^i , and if one of them is found, the other can be calculated from the displacement (\mathbf{R}, \mathbf{t}) . Therefore the depth parameter λ_2^i is eliminated by premultiplication with $\mathbf{s}_2^i \times$, which gives

$$\lambda_1^i (\mathbf{s}_2^i)^\times \mathbf{R}\mathbf{s}_1^i + \gamma (\mathbf{s}_2^i)^\times \mathbf{t} = \mathbf{0} \quad (882)$$

This equation for point correspondence i can be written

$$\begin{bmatrix} (\mathbf{s}_2^i)^\times \mathbf{R}\mathbf{s}_1^i & (\mathbf{s}_2^i)^\times \mathbf{t} \end{bmatrix} \begin{bmatrix} \lambda_1^i \\ \gamma \end{bmatrix} = \mathbf{0} \quad (883)$$

Then all the depth parameters λ_1^i and the translation scaling γ can be found by solving

$$\mathbf{M}\boldsymbol{\lambda} = \mathbf{0} \quad (884)$$

where the unknown parameters are given by the vector

$$\boldsymbol{\lambda} = [\lambda_1^1, \lambda_1^2, \dots, \lambda_1^n, \gamma]^T \quad (885)$$

and

$$\mathbf{M} = \begin{bmatrix} (\mathbf{s}_2^1)^\times \mathbf{R} \mathbf{s}_1^1 & 0 & \dots & 0 & (\mathbf{s}_2^1)^\times \mathbf{t} \\ 0 & (\mathbf{s}_2^2)^\times \mathbf{R} \mathbf{s}_2^2 & \dots & 0 & (\mathbf{s}_2^2)^\times \mathbf{t} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & (\mathbf{s}_2^n)^\times \mathbf{R} \mathbf{s}_1^n & (\mathbf{s}_2^n)^\times \mathbf{t} \end{bmatrix} \quad (886)$$

where \mathbf{M} has dimension $2n \times (n + 1)$.

The solution is found from the singular value decomposition $\mathbf{M} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$ as the last column of the \mathbf{V} matrix, giving $\boldsymbol{\lambda} = c \mathbf{v}_{n+1}$. Note that $\boldsymbol{\lambda}$ is found up to a single scale, which is the same for all points. This uncertainty in scale is due to the fact that the translation \mathbf{t} between the cameras is only determined up to a scale.

17.17 Triangulation with n known camera matrices

In this section a method for the determination of a homogeneous world point \mathbf{X} is presented based on triangulation with $n \geq 2$ cameras with known camera matrices. Suppose that the camera matrices of n cameras are known and given by $\mathbf{P}_1 = [\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}_i = [\mathbf{R}_i \mid \mathbf{t}_i]$ for $i = 2, \dots, n$, where \mathbf{R}_i and \mathbf{t}_i are known. In this case it is assumed that \mathbf{t}_i has been determined without any ambiguity in scale. The equations for the calibrated image coordinates is then

$$\begin{aligned} \lambda_1 \mathbf{s}_1 &= \mathbf{P}_1 \mathbf{X} \\ \lambda_2 \mathbf{s}_2 &= \mathbf{P}_2 \mathbf{X} \\ &\vdots \\ \lambda_n \mathbf{s}_n &= \mathbf{P}_n \mathbf{X} \end{aligned}$$

The scale factors can be eliminated by premultiplication with \mathbf{s}_i^\times , which gives $\mathbf{s}_i^\times \mathbf{P}_i \mathbf{X} = \mathbf{0}$, and

$$\mathbf{A} \mathbf{X} = \mathbf{0} \quad (887)$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{s}_1^\times \mathbf{P}_1 \\ \mathbf{s}_2^\times \mathbf{P}_2 \\ \vdots \\ \mathbf{s}_n^\times \mathbf{P}_n \end{bmatrix} \quad (888)$$

is a matrix of dimension $3n \times 4$. Note that there are three rows in \mathbf{A} for each condition $\mathbf{s}_i^\times \mathbf{P}_i \mathbf{X} = \mathbf{0}$. As in the DLT computation of a homography the same result would be obtained with the two first rows of each condition, which would give an matrix \mathbf{A} of dimension $2n \times 4$.

The solution is

$$\mathbf{X} = c \mathbf{v}_4 \quad (889)$$

where \mathbf{v}_4 is the last column of the matrix \mathbf{V} of the singular value decomposition $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$. The scale of the homogeneous vector \mathbf{X} can then be selected so that the fourth element is unity, which gives $\mathbf{X} = [\mathbf{r}^T, 1]^T$. This means that the Euclidean position \mathbf{r} of the point has been determined without any ambiguity in scale.

17.18 Determination of a line from stereo vision

Suppose that the scene includes a line \mathbf{L} which is imaged in two cameras in a stereo arrangement with camera models

$$\lambda_1 \mathbf{s}_1 = \mathbf{P}_1 \mathbf{X}, \quad \lambda_2 \mathbf{s}_2 = \mathbf{P}_2 \mathbf{X} \quad (890)$$

where the cameras are described with standard camera matrices

$$\mathbf{P}_1 = [\mathbf{I} \mid \mathbf{0}], \quad \mathbf{P}_2 = [\mathbf{R} \mid \mathbf{t}] \quad (891)$$

and the homogeneous position in the scene is $\mathbf{X} = [\mathbf{r}^T, 1]^T$. This gives

$$\lambda_1 \mathbf{s}_1 = \mathbf{r} \quad (892)$$

$$\lambda_2 \mathbf{s}_2 = \mathbf{R}\mathbf{r} + \mathbf{t} \quad (893)$$

A point \mathbf{X} on the line \mathbf{L} will then be mapped to a point $\mathbf{s}_1 = \mathbf{P}_1 \mathbf{X}$ on a line ℓ_1 in image 1. In the same way the points \mathbf{X} will be mapped to a point $\mathbf{s}_2 = \mathbf{P}_2 \mathbf{X}$ on a line ℓ_2 in image 2.

The lines ℓ_1 and ℓ_2 will satisfy the conditions $\ell_1^T \mathbf{s}_1 = 0$ and $\ell_2^T \mathbf{s}_2 = 0$. This gives $\pi_1^T \mathbf{X} = 0$ and $\pi_2^T \mathbf{X} = 0$, where

$$\pi_1 = \mathbf{P}_1^T \ell_1 = \begin{bmatrix} \ell_1 \\ 0 \end{bmatrix} \quad (894)$$

$$\pi_2 = \mathbf{P}_2^T \ell_2 = \begin{bmatrix} \mathbf{R}^T \ell_2 \\ \mathbf{t}^T \ell_2 \end{bmatrix} \quad (895)$$

The geometric interpretation is that π_1 is the plane through the origin of camera 1 and the line ℓ_1 , and π_2 is the plane through the origin of camera 1 and the line ℓ_2 . The intersection of the two planes is the line \mathbf{L} .

The two lines ℓ_1 and ℓ_2 are found from the singular value decomposition of the equations

$$\mathbf{A}_1 \ell_1 = 0, \quad \mathbf{A}_2 \ell_2 = 0 \quad (896)$$

where

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{s}_{11}^T \\ \vdots \\ \mathbf{s}_{1,n_1}^T \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} \mathbf{s}_{21}^T \\ \vdots \\ \mathbf{s}_{2,n_2}^T \end{bmatrix} \quad (897)$$

In the notation used in connection with Plücker coordinates, the planes are written

$$\pi_1 = (\mathbf{u}, u_4) = (\ell_1, 0) \quad (898)$$

$$\pi_2 = (\mathbf{v}, v_4) = (\mathbf{R}^T \ell_2, \mathbf{t}^T \ell_2) \quad (899)$$

As explained in the next section, the line \mathbf{L} is then given in Plücker coordinates as $\mathbf{L} = (\mathbf{u} \times \mathbf{v}, u_4 \mathbf{v} - v_4 \mathbf{u})$, which gives

$$\mathbf{L} = (\ell_1^\times (\mathbf{R}^T \ell_2), -(\mathbf{t}^T \ell_2) \ell_1) \quad (900)$$

17.19 Geometric interpretation of the determination of a line from stereo vision

The line \mathbf{L} is given in Plücker coordinates as $\mathbf{L} = (\mathbf{l}, \mathbf{l}')$ where \mathbf{l} is the unit direction vector of the line, and $\mathbf{l}' = \mathbf{q} \times \mathbf{l}$ is the moment of the line, where \mathbf{q} is the point on the line that is closest to the origin. It follows that \mathbf{l} and \mathbf{l}' are orthogonal vectors. A plane is written in the form $\mathbf{u} = (\mathbf{u}, u_4)$ where \mathbf{u} is the normal vector of the plane, and $-u_4/|\mathbf{u}|$ is the distance from the origin to the plane in the direction of \mathbf{u} .

Two planes $\mathbf{u} = (\mathbf{u}, u_4)$ and (\mathbf{v}, v_4) which intersect at the line \mathbf{L} will define the line in dual Plücker coordinates as

$$\mathbf{L}^* = (u_4\mathbf{v} - v_4\mathbf{u}, \mathbf{u} \times \mathbf{v}) \quad (901)$$

The line is then found from the dual line by exchanging the two vectors of the Plücker coordinates, which gives

$$\mathbf{L} = (\mathbf{u} \times \mathbf{v}, u_4\mathbf{v} - v_4\mathbf{u}) \quad (902)$$

Suppose that the line $\mathbf{L} = (\mathbf{l}, \mathbf{l}')$ is in the plane π_1 which passes through the origin of camera 1. Let $\mathbf{P}_1 = [\mathbf{I} \mid \mathbf{0}]$ be the camera model of camera 1. This means that the scene is described in the coordinates of c_1 , which is camera frame 1. Moreover, let \mathbf{L} be referenced to c_1 . This means that $\mathbf{l}' = \mathbf{q} \times \mathbf{l}$, where \mathbf{q} is a vector from the origin of c_1 to the line \mathbf{L} . It follows that \mathbf{l}' is normal to the plane π_1 as \mathbf{q} is in the plane. The plane is therefore given by $\pi_1 = (\mathbf{l}, 0)$, where the zero scalar follows as the plane is through the origin.

The line ℓ_1 is in the image plane of camera 1. The line is given by

$$\ell_1 = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \mathbf{n} \\ c \end{bmatrix} \quad (903)$$

where the unit vector $\mathbf{n} = [a, b]^T$ is the normal vector to the line in the image plane, and $-c$ is the distance from the optical axis to the line in the direction of \mathbf{n} . Note that the optical axis is the z_1 axis of camera frame 1. In the standard case where $\mathbf{P}_1 = [\mathbf{I} \mid \mathbf{0}]$, the plane π_1 through the origin of frame 1 and the line ℓ_1 is given in Plücker notation as $\pi_1 = (\ell_1, 0)$. It is seen that this means that the origin of frame 1 is in the plane, which is obviously correct, and that the vector ℓ_1 is normal to the plane. It is interesting to observe that the vector ℓ_1 is the sum of the vector \mathbf{n} , which is normal to the line in the image plane, and a vector of length c along the z_1 axis, which is orthogonal to the image plane.

In the image plane the line satisfies $\ell_1^T \mathbf{s}_1 = 0$ for all image points \mathbf{s}_1 corresponding to a point. In this case the line is represented by the homogeneous vector ℓ_1 . The point $\mathbf{X} = [\mathbf{r}^T, 1]^T$ is a homogeneous vector on the line, and \mathbf{r} is the Euclidean position vector of the point. Insertion of $\lambda_1 \mathbf{s} = \mathbf{X}$ gives $\ell_1^T \mathbf{r} = 0$. This means that the vector ℓ_1 in 3D is orthogonal to the vector \mathbf{r} from the origin of frame 1 to the point \mathbf{X} on the line. The point \mathbf{X} is an arbitrary point on the line \mathbf{L} . Therefore ℓ_1 is orthogonal to all vectors \mathbf{r} from the origin of frame 1 to the line, and it follows that ℓ_1 is normal to the plane π_1 through the origin of frame 1 and the line \mathbf{L} . Moreover, the line in the image plane 1 defined by ℓ_1 as a homogeneous vector will be on the plane.

17.20 Planar homography in stereo vision

In the presentation so far the epipolar constraint has been used to determine the essential matrix with the eight-point algorithm. Then the displacement between the cameras can be determined and scene reconstruction is possible. In the eight-point algorithm the points must be in what is called general position, which means that there are certain condition on the position of the points for the algorithm to work. In particular, the eight points cannot be on the same plane. This calls for further results, since there are many applications where the points of the scene will be on a common plane. This situation will be discussed in this section, and a four-point algorithm will be presented.

Consider two cameras in a stereo arrangement with camera model $P_1 = [\mathbf{I} \mid \mathbf{0}]$ and $P_2 = [\mathbf{R} \mid \mathbf{t}]$, where camera frame 1 is referred to as frame 1. Suppose that the points of the scene is on a plane $\boldsymbol{\pi} = [\mathbf{n}^T, -d]^T$ where \mathbf{n} is the unit normal vector pointing away from camera 1. The vector \mathbf{n} is given in the coordinates of camera frame 1. Consider a homogeneous point $\mathbf{X}_1 = [\mathbf{r}_1^T, 1]^T$ which represents a Euclidean point $\mathbf{r}_1 \in \mathbb{R}^3$ in the plane $\boldsymbol{\pi}$, where \mathbf{r}_1 is given in the coordinates of frame 1. Then $\boldsymbol{\pi}^T \mathbf{X}_1 = 0$ which gives $\mathbf{n}^T \mathbf{r}_1 = d$. The distance from the origin of frame 1 to the plane is therefore $d > 0$.

The distance from the origin of camera 1 to the plane $\boldsymbol{\pi}$ is

$$d = \mathbf{n}^T \mathbf{r}_1 \quad (904)$$

It follows that

$$\frac{\mathbf{n}^T \mathbf{r}_1}{d} = 1 \quad (905)$$

which will be used in the following.

The point \mathbf{r}_1 will be given by

$$\mathbf{r}_2 = \mathbf{R}\mathbf{r}_1 + \mathbf{t} = \mathbf{R}\mathbf{r}_1 + \mathbf{t} \frac{\mathbf{n}^T \mathbf{r}_1}{d} = \left(\mathbf{R} + \frac{1}{d} \mathbf{t} \mathbf{n}^T \right) \mathbf{r}_1 \quad (906)$$

in the frame of camera 2, where (905) has been inserted. This can be written as the homography

$$\mathbf{r}_2 = \mathbf{H}\mathbf{r}_1 \quad (907)$$

where the homography matrix is given by

$$\mathbf{H} = \mathbf{R} + \frac{1}{d} \mathbf{t} \mathbf{n}^T \quad (908)$$

Note that \mathbf{t} is given in frame 2, while \mathbf{n} is given in frame 1.

The image models for the cameras give

$$\lambda_1 \mathbf{s}_1 = \mathbf{r}_1^1, \quad \lambda_2 \mathbf{s}_2 = \mathbf{r}_2^2 \quad (909)$$

where λ_1 and λ_2 are the depth parameters of the point. This gives $\lambda_1 \mathbf{s}_2 = \lambda_2 \mathbf{H} \mathbf{s}_1$, and because of the freedom of scaling, the homography can be written

$$\mathbf{s}_2 = \mathbf{H} \mathbf{s}_1 \quad (910)$$

The homography \mathbf{H} is said to be induced by the plane π , and \mathbf{H} is referred to as a planar homography.

If \mathbf{s}_1 is the image point in camera 1 of a point \mathbf{X}_1 in the plane π , then $\mathbf{s}_2 = \mathbf{H}\mathbf{s}_1$ is the image point in camera 2 of the same point.

```
% Arrangement of cameras and plane. The cameras are mounted so that t
% is parallel to the plane, with distance d from the camera origins to the
% plane.
theta = pi/8;
R = Roty(2*theta); % R^2_1
n = Roty(-theta)*[0; 0; 1]; d = 2; Plane = [n;-d];
t = Roty(theta)*[-2*d*tan(theta); 0; 0]; % t_21^2
E = Skew(t)*R;
e2 = t/t(3);

H = R + t*n'/d % Homography

r11 = [0;0;d/cos(theta)]; % Point on plane in camera frame 1
delta1 = [r11;1]'*Plane % Test that r11 is on the plane
r21 = H*r11 % Corresponding point in camera frame 2
r21 = R*r11 + t % Test if point satisfies displacement transformation

p = 0.1*[1;1;1];
r12 = r11 + (eye(3) - n*n')*p; % Point on plane in camera frame 1
[r12;1]'*Plane % Test that r12 is on the plane

r22 = H*r12 % Corresponding point in camera frame 2
r22 = R*r12 + t % Test if point satisfies displacement transformation
```

□

17.21 Computation of the planar homography

The planar homography

$$\mathbf{s}_2 = \mathbf{H}\mathbf{s}_1 \quad (911)$$

where $\mathbf{H} = \mathbf{R} + \mathbf{t}\mathbf{n}^T/d$ can be computed from point correspondences in the usual way for a homography from $n \geq 4$ point correspondences $(\mathbf{s}_1^i, \mathbf{s}_2^i)$. This is done based on the equation $(\mathbf{s}_2^i)^T \mathbf{H} \mathbf{s}_1^i = 0$, which gives three equations, where two of the equations are linearly independent. The two equations can be selected as

$$\begin{bmatrix} \mathbf{0}^T & -s_{2,3}^i s_1^T & s_{2,2}^i s_1^T \\ s_{2,3}^i s_1^T & \mathbf{0}^T & -s_{2,1}^i s_1^T \end{bmatrix} \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix} = \mathbf{0} \quad (912)$$

as in (290), with $\mathbf{s}_2^i = [s_{2,1}^i, s_{2,2}^i, s_{2,3}^i]^T$, and where $(\mathbf{h}^i)^T$ is row i in \mathbf{H} . The point correspondences $(\mathbf{s}_1^i, \mathbf{s}_2^i)$ are the corresponding calibrated image points for image 1 and 2, which both

are the image of the point \mathbf{X}_i in the plane π . It is required that there are at least four points \mathbf{x}_i on π , and that no three of them are on the same line. Due to the homogeneous properties of the problem, the solution obtained for the homography has an arbitrary scaling, and can be written

$$\mathbf{H}_\gamma = \gamma \mathbf{H} = \gamma(\mathbf{R} + \mathbf{t}\mathbf{n}^T/d) \quad (913)$$

where γ is an unknown nonzero scaling factor and $\mathbf{H} = \mathbf{R} + \mathbf{t}\mathbf{n}^T/d$.

It is noted that

$$\mathbf{H}_\gamma^T \mathbf{H}_\gamma = \gamma^2 (\mathbf{I} + \mathbf{c}\mathbf{n}^T + \mathbf{n}\mathbf{c}^T + \|\mathbf{c}\|^2 \mathbf{n}\mathbf{n}^T) \quad (914)$$

where $\mathbf{c} = \mathbf{R}^T \mathbf{t}/d$. Then the vector $\mathbf{m} = \mathbf{c}^\times \mathbf{n}$ is orthogonal to both \mathbf{c} and \mathbf{n} , and it follows that

$$\mathbf{H}_\lambda^T \mathbf{H}_\lambda \mathbf{m} = \gamma^2 \mathbf{m} \quad (915)$$

This means that γ^2 is an eigenvalue of $\mathbf{H}_\gamma^T \mathbf{H}_\gamma$. It will be shown in the following that the second largest eigenvalue of $\mathbf{H}^T \mathbf{H}$ is $\lambda_2 = 1$, which means that γ^2 is the second largest eigenvalue of $\mathbf{H}_\gamma^T \mathbf{H}_\gamma$. This means that the second largest singular value of \mathbf{H}_γ is $\sigma_2(\mathbf{H}_\lambda) = |\gamma|$. This means that \mathbf{H} can be determined up to a sign by $\mathbf{H} = \mathbf{H}_\gamma / \sigma_2(\mathbf{H}_\gamma)$.

Let the singular value decomposition of \mathbf{H} be

$$\mathbf{H} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \sigma_3 \mathbf{u}_3 \mathbf{v}_3^T \quad (916)$$

It is noted that the second largest singular value is $\sigma_2 = 1$.

17.22 Recovery of the displacement from the planar homography

Consider the case where the homography \mathbf{H} is known, and the task is to recover \mathbf{R} , \mathbf{t}/d and \mathbf{n} so that $\mathbf{H} = \mathbf{R} + (\mathbf{t}/d)\mathbf{n}^T$ where \mathbf{n} is a unit vector. It is possible to recover two solutions \mathbf{R} , \mathbf{t}/d and \mathbf{n} , where \mathbf{t} can be recovered up to a scale [79, 41].

The recovery is based on the observation that $\mathbf{H}\mathbf{w} = \mathbf{R}\mathbf{w}$ for any vector \mathbf{w} which is orthogonal to the normal vector \mathbf{n} of the plane, so that $\mathbf{n}^T \mathbf{w} = 0$. Since the rotation matrix preserves the norm of a vector in the sense that $\|\mathbf{R}\mathbf{w}\| = \|\mathbf{w}\|$, this implies that $\|\mathbf{H}\mathbf{w}\| = \|\mathbf{w}\|$ for any vector \mathbf{w} that is orthogonal to \mathbf{n} . The first step of the method is to find two orthogonal unit vectors \mathbf{w}_1 and \mathbf{w}_2 that are orthogonal to \mathbf{n} , and then to compute $\mathbf{n} = \mathbf{w}_1^\times \mathbf{w}_2$, so that $\mathbf{w}_1, \mathbf{w}_2, \mathbf{n}$ form a set of orthogonal unit vectors. Note that \mathbf{n} is unknown, and only the matrix \mathbf{H} is given.

Define the orthogonal matrix

$$\mathbf{W} = [\mathbf{w}_1 \ \mathbf{w}_2 \ \mathbf{w}_3] \quad (917)$$

where \mathbf{w}_1 and \mathbf{w}_2 are orthogonal to \mathbf{n} , and $\mathbf{n} = \mathbf{w}_3 = \mathbf{w}_1^\times \mathbf{w}_2$. It follows that

$$\mathbf{H}\mathbf{w}_1 = \mathbf{R}\mathbf{w}_1 \quad (918)$$

$$\mathbf{H}\mathbf{w}_2 = \mathbf{R}\mathbf{w}_2 \quad (919)$$

$$\mathbf{H}\mathbf{w}_3 = \mathbf{R}\mathbf{n} + \mathbf{t}/d \quad (920)$$

Then

$$\mathbf{Z} = \mathbf{W}^T \mathbf{H}^T \mathbf{H} \mathbf{W} = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ a & b & c \end{bmatrix} \quad (921)$$

where

$$a = (\mathbf{H}\mathbf{w}_1)^T \mathbf{H}\mathbf{w}_3 = (\mathbf{R}\mathbf{w}_1)^T \mathbf{t}/d \quad (922)$$

$$b = (\mathbf{H}\mathbf{w}_2)^T \mathbf{H}\mathbf{w}_3 = (\mathbf{R}\mathbf{w}_2)^T \mathbf{t}/d \quad (923)$$

$$c = (\mathbf{H}\mathbf{w}_3)^T \mathbf{H}\mathbf{w}_3 = 1 + 2(\mathbf{R}\mathbf{n})^T \mathbf{t}/d + \|\mathbf{t}\|^2/d^2 \quad (924)$$

The eigenvalues of \mathbf{Z} are found from the characteristic polynomial

$$\det(\lambda \mathbf{I} - \mathbf{Z}) = \det \begin{bmatrix} \lambda - 1 & 0 & -a \\ 0 & \lambda - 1 & -b \\ -a & -b & \lambda - c \end{bmatrix} = (\lambda - 1)(\lambda^2 - (1 + c)\lambda - (a^2 + b^2 - c)) \quad (925)$$

The eigenvalues are found to be

$$\lambda_1 = \frac{1 + c + \sqrt{(1 - c)^2 + 4(a^2 + b^2)}}{2} \quad (926)$$

$$\lambda_2 = 1 \quad (927)$$

$$\lambda_3 = \frac{1 + c - \sqrt{(1 - c)^2 + 4(a^2 + b^2)}}{2} \quad (928)$$

It follows that

$$\lambda_1 \geq \lambda_2 = 1 \geq \lambda_3 \quad (929)$$

This is seen by first assuming that $a^2 + b^2 = 0$, which leads to $\lambda_1 = \max(1, c)$ and $\lambda_3 = \min(1, c)$. Then if $a^2 + b^2 > 0$ it follows that $\lambda_1 > \min(1, c)$ and $\lambda_3 < \max(1, c)$.

It is noted that since \mathbf{W} is orthogonal, it follows that the eigenvalues of $\mathbf{W}^T \mathbf{H}^T \mathbf{H} \mathbf{W}$ are equal to the eigenvalues of $\mathbf{H}^T \mathbf{H}$. This follows since $\mathbf{W}^T \mathbf{H}^T \mathbf{H} \mathbf{W} \mathbf{m} = \lambda \mathbf{m}$ is equivalent to $\mathbf{H}^T \mathbf{H}(\mathbf{W} \mathbf{m}) = \lambda(\mathbf{W} \mathbf{m})$. Moreover, it is noted that the eigenvalues of the symmetric matrix $\mathbf{H}^T \mathbf{H}$ are equal to the singular values of $\mathbf{H}^T \mathbf{H}$.

The singular value decomposition of \mathbf{H} is given by

$$\mathbf{H} = \mathbf{U} \Sigma \mathbf{V}^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \sigma_3 \mathbf{u}_3 \mathbf{v}_3^T \quad (930)$$

where $\sigma_2 = 1$.

The singular value decomposition of $\mathbf{H}^T \mathbf{H}$ is given by

$$\mathbf{H}^T \mathbf{H} = \mathbf{V} \Sigma^2 \mathbf{V}^T = \sigma_1^2 \mathbf{v}_1 \mathbf{v}_1^T + \sigma_2^2 \mathbf{v}_2 \mathbf{v}_2^T + \sigma_3^2 \mathbf{v}_3 \mathbf{v}_3^T \quad (931)$$

which gives

$$\mathbf{V}^T \mathbf{H}^T \mathbf{H} \mathbf{V} = \Sigma^2 = \text{diag}(\sigma_1^2, 1, \sigma_3^2) \quad (932)$$

To proceed, we consider a vector \mathbf{w} which satisfies $\mathbf{w}^T \mathbf{n} = 0$. Then define the vector $\mathbf{x} = [x_1, x_2, x_3]^T$ by $\mathbf{x} = \mathbf{V}^T \mathbf{w}$, which gives

$$\mathbf{w} = \mathbf{V} \mathbf{x} = \mathbf{v}_1 x_1 + \mathbf{v}_2 x_2 + \mathbf{v}_3 x_3 \quad (933)$$

which means that \mathbf{w} is given as a linear combination of the output singular vectors \mathbf{v}_i . Then

$$\|\mathbf{H}\mathbf{w}\|^2 = \|\mathbf{H}\mathbf{V}\mathbf{x}\|^2 = \mathbf{x}^T \mathbf{V}^T \mathbf{H}^T \mathbf{H} \mathbf{V} \mathbf{x} = \mathbf{x}^T \Sigma^2 \mathbf{x} = \lambda_1 x_1^2 + x_2^2 + \lambda_3 x_3^2 \quad (934)$$

while

$$\|\mathbf{R}\mathbf{w}\|^2 = \|\mathbf{RV}\mathbf{x}\|^2 = \mathbf{x}^T \mathbf{V}^T \mathbf{R}^T \mathbf{RV}\mathbf{x} = \mathbf{x}^T \mathbf{x} = x_1^2 + x_2^2 + x_3^2 \quad (935)$$

Since $\mathbf{H}\mathbf{w} = \mathbf{R}\mathbf{w}$ it follows that

$$\sigma_1^2 x_1^2 + x_2^2 + \sigma_3^2 x_3^2 = x_1^2 + x_2^2 + x_3^2 \quad (936)$$

which is equivalent to $\sqrt{\sigma_1^2 - 1}x_1 = \pm\sqrt{1 - \sigma_3^2}x_3$. Two orthogonal solutions are possible for \mathbf{x} , which is

$$\mathbf{x}_1 = [\alpha, 0, \pm\beta]^T, \quad \mathbf{x}_2 = [0, 1, 0]^T \quad (937)$$

where

$$\alpha = \sqrt{\frac{1 - \sigma_3^2}{\sigma_1^2 - \sigma_3^2}}, \quad \beta = \sqrt{\frac{\sigma_1^2 - 1}{\sigma_1^2 - \sigma_3^2}} \quad (938)$$

This leads to the two orthogonal matrices

$$\mathbf{A}_1 = [\mathbf{v}_2 \quad \mathbf{a}_1 \quad \mathbf{v}_2^\times \mathbf{a}_1], \quad \mathbf{A}_2 = [\mathbf{v}_2 \quad \mathbf{a}_2 \quad \mathbf{v}_2^\times \mathbf{a}_2] \quad (939)$$

where

$$\mathbf{a}_1 = \frac{\sqrt{1 - \sigma_3^2}\mathbf{v}_1 + \sqrt{\sigma_1^2 - 1}\mathbf{v}_3}{\sqrt{\sigma_1^2 - \sigma_3^2}}, \quad \mathbf{a}_2 = \frac{\sqrt{1 - \sigma_3^2}\mathbf{v}_1 - \sqrt{\sigma_1^2 - 1}\mathbf{v}_3}{\sqrt{\sigma_1^2 - \sigma_3^2}} \quad (940)$$

are unit vectors that are orthogonal to \mathbf{n} . Both vectors are linear combinations of \mathbf{v}_1 and \mathbf{v}_3 , and it follows that they are orthogonal to \mathbf{v}_2 . Since \mathbf{v}_2 is orthogonal to \mathbf{n} , it follows that \mathbf{n} must be along $\mathbf{a}_1^\times \mathbf{v}_2$ for the solution with \mathbf{A}_1 , and along $\mathbf{a}_2^\times \mathbf{v}_2$ for the solution with \mathbf{A}_2 .

Next it is noted that for the solution with \mathbf{A}_1 ,

$$\mathbf{R}\mathbf{v}_2 = \mathbf{H}\mathbf{v}_2, \quad \mathbf{R}\mathbf{a}_1 = \mathbf{H}\mathbf{a}_1, \quad \mathbf{R}(\mathbf{a}_1^\times \mathbf{v}_2) = (\mathbf{R}\mathbf{a}_1)^\times (\mathbf{R}\mathbf{v}_2) = (\mathbf{H}\mathbf{a}_1)^\times (\mathbf{H}\mathbf{v}_2) \quad (941)$$

and for the solution with \mathbf{A}_2 ,

$$\mathbf{R}\mathbf{v}_2 = \mathbf{H}\mathbf{v}_2, \quad \mathbf{R}\mathbf{a}_2 = \mathbf{H}\mathbf{a}_2, \quad \mathbf{R}(\mathbf{a}_2^\times \mathbf{v}_2) = (\mathbf{R}\mathbf{a}_2)^\times (\mathbf{R}\mathbf{v}_2) = (\mathbf{H}\mathbf{a}_2)^\times (\mathbf{H}\mathbf{v}_2) \quad (942)$$

Define the matrices

$$\mathbf{B}_1 = [\mathbf{H}\mathbf{v}_2 \quad \mathbf{H}\mathbf{a}_1 \quad (\mathbf{H}\mathbf{v}_2)^\times \mathbf{H}\mathbf{a}_1], \quad \mathbf{B}_2 = [\mathbf{H}\mathbf{v}_2 \quad \mathbf{H}\mathbf{a}_2 \quad (\mathbf{H}\mathbf{v}_2)^\times \mathbf{H}\mathbf{a}_2] \quad (943)$$

Then

$$\mathbf{R}\mathbf{A}_1 = \mathbf{B}_1, \quad \mathbf{R}\mathbf{A}_2 = \mathbf{B}_2 \quad (944)$$

and it follows that two solutions for the rotation \mathbf{R} are given by

$$\mathbf{R}_1 = \mathbf{B}_1 \mathbf{A}_1^T, \quad \mathbf{R}_2 = \mathbf{B}_2 \mathbf{A}_2^T \quad (945)$$

Finally, it is seen that

$$\mathbf{n}_1 = \mathbf{v}_2^\times \mathbf{a}_1, \quad \mathbf{n}_1 = \mathbf{v}_2 \times \mathbf{a}_2 \quad (946)$$

and $\mathbf{H}\mathbf{n}_i = \mathbf{R}_i \mathbf{n}_i + (1/d)\mathbf{t} \mathbf{n}_i^T \mathbf{n}_i = \mathbf{R}_i \mathbf{n}_i + (1/d)\mathbf{t}$. Then \mathbf{t}/d can be found as

$$\mathbf{t}_i/d = (\mathbf{H} - \mathbf{R}_i)\mathbf{n}_i \quad (947)$$

This method gives the four solutions

$$\begin{aligned}
 \mathbf{R}_1 &= \mathbf{B}_1 \mathbf{A}_1^T, & \mathbf{n}_1 &= \mathbf{v}_2^\times \mathbf{a}_1, & \mathbf{t}_1/d &= (\mathbf{H} - \mathbf{R}_1)\mathbf{n}_1 \\
 \mathbf{R}_2 &= \mathbf{B}_2 \mathbf{A}_2^T, & \mathbf{n}_2 &= \mathbf{v}_2^\times \mathbf{a}_2, & \mathbf{t}_2/d &= (\mathbf{H} - \mathbf{R}_2)\mathbf{n}_2 \\
 \mathbf{R}_3 &= \mathbf{R}_1, & \mathbf{n}_3 &= -\mathbf{n}_1, & \mathbf{t}_3/d &= -\mathbf{t}_1/d \\
 \mathbf{R}_4 &= \mathbf{R}_2, & \mathbf{n}_4 &= -\mathbf{n}_2, & \mathbf{t}_4/d &= -\mathbf{t}_2/d
 \end{aligned} \tag{948}$$

The correct solution must satisfy $\mathbf{n}^T \mathbf{z} > 0$, where $\mathbf{z} = [0, 0, 1]^T$, as the normal \mathbf{n} is assumed to point away from camera 1. This condition will eliminate two of the solutions. The last false solution must be eliminated in some other way.

Example

```

% Script for recovery of R, t/d and n from the planar homography
% H = R + t*n'/d
% Functions
skew =@(u) [0 -u(3) u(2); u(3) 0 -u(1);-u(2) u(1) 0];
expso3 =@(u) eye(3)+sinc(norm(u)/pi)*skew(u)+0.5*(sinc(norm(u)/(2*pi)))^2*(skew(u))^2;

% Input parameters
R = expso3(0.2*[0;1;0])
t = [1;0;0]; n = [0;0;1]; d = 2;
H = R + t*n'/d

[U,S,V] = svd(H);
if det(V) <0
    V = -V; U = -U;
end

s1 = S(1,1); s2 = S(2,2); s3 = S(3,3); v1 = V(:,1); v2 = V(:,2); v3 = V(:,3);
a1 = (sqrt(1-s3^2)*v1 + sqrt(s1^2-1)*v3)/(sqrt(s1^2-s3^2));
a2 = (sqrt(1-s3^2)*v1 - sqrt(s1^2-1)*v3)/(sqrt(s1^2-s3^2));
n1 = cross(v2,a1); n2 = cross(v2,a2);
A1 = [v2 a1 n1];
A2 = [v2 a2 n2];
B1 = [H*v2 H*a1 cross(H*v2,H*a1)]
B2 = [H*v2 H*a2 cross(H*v2,H*a2)]

R1 = B1*A1'; R2 = B2*A2';
t1 = (H-R1)*n1; t2 = (H-R2)*n2;
Ri = [R R1 R2]
ti = [t t1 t2]
ni = [n n1 n2]

```

□

17.23 The planar homography and the essential matrix

Consider stereo vision with the standard camera model $P_1 = [\mathbf{I} \mid \mathbf{0}]$ and $P_2 = [\mathbf{R} \mid \mathbf{t}]$ and a plane $\pi = [\mathbf{n}^T, -d]^T$ where \mathbf{n} is a unit vector. The essential matrix \mathbf{E} and the planar homography \mathbf{H} are given by

$$\mathbf{E} = \mathbf{t}^* \mathbf{R}, \quad \mathbf{H} = \mathbf{R} + \mathbf{t} \mathbf{n}^T \quad (949)$$

Two calibrated image points \mathbf{s}_1 and \mathbf{s}_2 points related by the homography $\mathbf{s}_2 = \mathbf{H} \mathbf{s}_1$ will also satisfy the epipolar constraint $\mathbf{s}_2^T \mathbf{E} \mathbf{s}_1 = 0$.

If \mathbf{s}_1 is the image of a point that is not necessarily in the plane π , then \mathbf{s}_2 will be on the epipolar line $\ell_2 = \mathbf{E} \mathbf{s}_1$.

The calibrated image points are transformed according to $\mathbf{s}_2 = \mathbf{H} \mathbf{s}_1$. It follows that the epipolar lines of the two images transform according to the transformation rule for lines, which is

$$\ell_2 = \mathbf{H}^{-T} \ell_1 \Leftrightarrow \ell_1 = \mathbf{H}^T \ell_2 \quad (950)$$

The essential matrix is given by the planar homography as

$$\mathbf{E} = \mathbf{t}^* \mathbf{H} \quad (951)$$

This follows from $\mathbf{t}^* \mathbf{H} = \mathbf{t}^* (\mathbf{R} + \frac{1}{d} \mathbf{t} \mathbf{n}^T) = \mathbf{t}^* \mathbf{R} = \mathbf{E}$ where it is used that $\mathbf{t}^* \mathbf{t} = \mathbf{0}$.

Another result is

$$\mathbf{E}^T \mathbf{H} + \mathbf{H}^T \mathbf{E} = \mathbf{0} \quad (952)$$

which follows from

$$\mathbf{0} = \mathbf{s}_2^T \mathbf{E} \mathbf{s}_1 = \mathbf{s}_1^T \mathbf{H}^T \mathbf{E} \mathbf{s}_1 \quad (953)$$

which shows that $\mathbf{H}^T \mathbf{E}$ is skew symmetric.

18 Stereo vision with uncalibrated cameras*

18.1 The fundamental matrix

Consider the case where the image point is described in pixel coordinates. Then

$$\mathbf{x}_1 = \mathbf{K}_1 \mathbf{s}_1, \quad \mathbf{x}_2 = \mathbf{K}_2 \mathbf{s}_2 \quad (954)$$

$$\lambda_1 \mathbf{s}_1 = \mathbf{P}_1 \mathbf{X}, \quad \lambda_2 \mathbf{s}_2 = \mathbf{P}_2 \mathbf{X} \quad (955)$$

where \mathbf{K}_1 is the camera parameter matrix of camera 1, and \mathbf{K}_2 is the camera parameter matrix of camera 2, and

$$\mathbf{P}_1 = [\mathbf{I} \mid \mathbf{0}], \quad \mathbf{P}_2 = [\mathbf{R} \mid \mathbf{t}] \quad (956)$$

The resulting essential matrix is $\mathbf{E} = \mathbf{t}^* \mathbf{R}$. The epipolar constraint in terms of the pixel coordinates is given by

$$\mathbf{x}_2^T \mathbf{F} \mathbf{x}_1 = 0 \quad (957)$$

where

$$\mathbf{F} = \mathbf{K}_2^{-T} \mathbf{E} \mathbf{K}_1^{-1} \quad (958)$$

is the fundamental matrix. This follows from

$$0 = \mathbf{s}_2^T \mathbf{E} \mathbf{s}_1 = \mathbf{x}_2^T \mathbf{K}_2^{-T} \mathbf{E} \mathbf{K}_1^{-1} \mathbf{x}_1 \quad (959)$$

The essential matrix \mathbf{E} is of rank 2, while the matrices \mathbf{K}_1 and \mathbf{K}_2 are of rank 3. Therefore the fundamental matrix \mathbf{F} is of rank 2

In this case the epipolar lines are given in pixel coordinates as

$$\ell_1 = \mathbf{F}^T \mathbf{x}_2, \quad \ell_2 = \mathbf{F} \mathbf{x}_1 \quad (960)$$

while the epipoles \mathbf{z}_1 and \mathbf{z}_2 in pixel coordinates are given by

$$\lambda_1 \mathbf{z}_1 = -\mathbf{K}_1 \mathbf{R}^T \mathbf{t}, \quad \lambda_2 \mathbf{z}_2 = \mathbf{K}_2 \mathbf{t} \quad (961)$$

Suppose that \mathbf{x}_1 in camera 1 and the point \mathbf{x}_2 in camera 2 are the pixel coordinates of the same point. Then the epipolar line $\ell_1 = \mathbf{F}^T \mathbf{x}_2$ in image 1 will be a line through the image point \mathbf{x}_1 and the epipole \mathbf{z}_1 . In the same way the epipolar line $\ell_2 = \mathbf{F} \mathbf{x}_1$ in image 2 will be a line through the image point \mathbf{x}_2 and the epipole \mathbf{z}_2 .

Example

```
% Draw epipolar lines and epipole using essential matrix
% Define geometry
theta = pi/2;
R = Roty(theta); % R^2_1
t = Roty(theta/1.5)*[-0.5; 0.1; 0.1]; % t_21^2
E = Skew(t)*R; % Essential matrix
e2 = t/t(3); % Epipole in image 2

% Point 1 in Image 1. Epipolar line and epipole in Image 2.
st11 = [0;0;1]; r11 = st11*0.2; % Point in image 1
r21 = R*r11 + t; st21 = r21/r21(3); % The same point in image 2
L21 = E*st11; % Epipolar line in image 2 corresponding to point 1 in image 1.
figure(1);clf; DrawEpipolar(L21, e2, st21, [-1 1 -1 1], 2);

% Point 2 in Image 1. Epipolar line and epipole in Image 2.
st12 = [0;0.5;1]; r12 = st12*0.2;
r22 = R*r12 + t; st22 = r22/r22(3);
L22 = E*st12;
figure(2);clf; DrawEpipolar(L22, e2, st22, [-1 1 -1 1], 2);

% Draw epipolar lines and epipole using fundamental matrix
K = [1500 0 640; 0 1500 512; 0 0 1]; Kinv = inv(K); % Camera parameter matrix
F = Kinv'*E*Kinv; % Fundamental matrix
ep2 = K*e2; % Epipole in image 2 in pixel coordinates
% Point 1 in Image 1. Epipolar line and epipole in Image 2.
pt11 = K*st11; Lp21 = F*pt11;
pt21 = K*st21;
```

```

figure(4);clf; DrawEpipolar(Lp21, ep2, pt21,[0 1280 0 1048],1500);

% Point 2 in Image 1. Epipolar line and epipole in Image 2.
pt12 = K*st12; Lp22 = F*pt12;
pt22 = K*st22;
figure(5);clf; DrawEpipolar(Lp22, ep2, pt22,[0 1280 0 1048],1500);

%%%%%%%%%%%%%
function DrawEpipolar(L,e,s, axisVector, length)
L = L/norm(L(1:2)); % Scaling of epipolar to get unit normal vector
a = [L(2);-L(1)]; % Direction vector of epipolar line
n = L(1:2); % Normal vector of line
x = [-n*L(3);1]; % Point on line closest to origin
x1 = x(1:2)-length*a; x2 = x(1:2)+length*a; % Two points on the line
plot([x1(1) x2(1)], [x1(2) x2(2)], 'b', e(1), e(2), 'dr', s(1), s(2), 'sk')
grid
axis(axisVector);
end

```

□

18.2 Transformation of skew symmetric matrices

Let $\mathbf{R} \in SO(3)$ be a rotation matrix, which means that $\det(\mathbf{R}) = 1$ and $\mathbf{R}^{-1} = \mathbf{R}^T$. It is well known that the skew symmetric form of $\mathbf{R}\mathbf{u}$ is

$$(\mathbf{R}\mathbf{u})^\times = \mathbf{R}\mathbf{u}^\times \mathbf{R}^T \quad (962)$$

and that

$$(\mathbf{R}\mathbf{u})^\times (\mathbf{R}\mathbf{v}) = \mathbf{R}(\mathbf{u}^\times \mathbf{v}) \quad (963)$$

This can be extended [22, 41] to the skew symmetric form $(\mathbf{K}\mathbf{u})^\times$ where $\mathbf{K} \in \mathbb{R}^{3 \times 3}$ is a matrix of full rank. The result is

$$(\mathbf{K}\mathbf{u})^\times = \det(\mathbf{K})\mathbf{K}^{-T}\mathbf{u}^\times \mathbf{K}^{-1} \quad (964)$$

This is shown by defining $\mathbf{w} = \mathbf{K}\mathbf{u}$, and considering the expression

$$(\mathbf{K}^{-1}\mathbf{w})^\times = \det(\mathbf{K}^{-1})\mathbf{K}^T\mathbf{w}^\times \mathbf{K} \quad (965)$$

As noted in [41] the expressions on both sides are linear maps from \mathbb{R}^3 to $\mathbb{R}^{3 \times 3}$. Therefore, if this is valid when \mathbf{w} is equal to the one of the basis vectors $[1, 0, 0]^T$, $[0, 1, 0]^T$ and $[0, 0, 1]^T$, then it is valid for all \mathbf{w} . It can be shown that the result is valid for the three basis vectors by direct computation at the element level of both sides of (965), and with considerable patience, it is found that $(\mathbf{K}^{-1}\mathbf{w})^\times = \mathbf{K}^T\mathbf{w}^\times \mathbf{K} / \det(\mathbf{K})$. Then the result follows from $\det(\mathbf{K}^{-1}) = \det(\mathbf{K})^{-1}$.

The result for a general full rank matrix \mathbf{K} in (964) is consistent with the result (962) for the rotation matrix, since $\mathbf{R}^{-T} = \mathbf{R}$, and $\mathbf{R}^{-1} = \mathbf{R}^T$.

It is useful to introduce the cofactor matrix of \mathbf{K} , which is written \mathbf{K}^C . Then it is well-known from linear algebra that

$$\mathbf{K}^{-1} = \frac{1}{\det(\mathbf{K})} (\mathbf{K}^C)^T \quad (966)$$

which means that the cofactor matrix of \mathbf{K} can be written

$$\mathbf{K}^C = \det(\mathbf{K}) \mathbf{K}^{-T} \quad (967)$$

Since the determinant is a scalar, it follows that

$$(\mathbf{K}^C)^C = \gamma \mathbf{K} \quad (968)$$

where γ is a nonzero scalar. This means that in a homogeneous setting, $(\mathbf{K}^C)^C$ is equal to \mathbf{K} .

An useful result that follows from (964) is

$$(\mathbf{K}\mathbf{u})^\times (\mathbf{K}\mathbf{v}) = \det(\mathbf{K}) \mathbf{K}^{-T} \mathbf{u}^\times \mathbf{K}^{-1} \mathbf{K}\mathbf{v} = \mathbf{K}^C (\mathbf{u}^\times \mathbf{v}) \quad (969)$$

which is consistent with the result (962) for the rotation matrix.

Example 1

Consider the camera matrix

$$\mathbf{K} = \begin{bmatrix} 1500 & 0 & 640 \\ 0 & 1500 & 512 \\ 0 & 0 & 1 \end{bmatrix} \quad (970)$$

and the vectors $\mathbf{w}_1 = [1, 0, 0]^T$, $\mathbf{w}_2 = [0, 1, 0]^T$ and $\mathbf{w}_3 = [0, 0, 1]^T$. Then it can be verified in MATLAB that

$$(\mathbf{K}\mathbf{w}_1)^\times = \det(\mathbf{K}) \mathbf{K}^{-T} \mathbf{w}_1^\times \mathbf{K}^{-1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1500 \\ 0 & 1500 & 0 \end{bmatrix} \quad (971)$$

$$(\mathbf{K}\mathbf{w}_2)^\times = \det(\mathbf{K}) \mathbf{K}^{-T} \mathbf{w}_2^\times \mathbf{K}^{-1} = \begin{bmatrix} 0 & 0 & 1500 \\ 0 & 0 & 0 \\ -1500 & 0 & 0 \end{bmatrix} \quad (972)$$

$$(\mathbf{K}\mathbf{w}_3)^\times = \det(\mathbf{K}) \mathbf{K}^{-T} \mathbf{w}_3^\times \mathbf{K}^{-1} = \begin{bmatrix} 0 & -1 & 512 \\ 1 & 0 & -640 \\ -512 & 640 & 0 \end{bmatrix} \quad (973)$$

□

Example 2

Let $\mathbf{s}_1, \dots, \mathbf{s}_4$ be four homogeneous points in the normalized image plane, and let $\mathbf{p}_1, \dots, \mathbf{p}_4$ be the corresponding pixel points, where $\mathbf{p}_i = \mathbf{K}\mathbf{s}_i$. Suppose that two lines are given in the

normalized image plane as $\ell_1 = \mathbf{s}_1 \times \mathbf{s}_2$ and $\ell_2 = \mathbf{s}_3 \times \mathbf{s}_4$. Moreover suppose that the lines intersect at $\mathbf{z} = \ell_1 \times \ell_2$. Then the lines will be transformed to the lines

$$\lambda_1 = \mathbf{p}_1^\times \mathbf{p}_2 = (\mathbf{K} \mathbf{s}_1)^\times (\mathbf{K} \mathbf{s}_2) = \mathbf{K}^C (\mathbf{s}_1 \times \mathbf{s}_2) = \mathbf{K}^C \ell_1 \quad (974)$$

$$\lambda_2 = \mathbf{p}_3^\times \mathbf{p}_4 = (\mathbf{K} \mathbf{s}_3)^\times (\mathbf{K} \mathbf{s}_4) = \mathbf{K}^C (\mathbf{s}_3 \times \mathbf{s}_4) = \mathbf{K}^C \ell_2 \quad (975)$$

The intersection point of the lines λ_1 and λ_2 in the pixel plane is

$$\mathbf{w} = \lambda_1 \times \lambda_2 = (\mathbf{K}^C \ell_1)^\times (\mathbf{K}^C \ell_2) = (\mathbf{K}^C)^C \mathbf{z} = \gamma \mathbf{K} \mathbf{z} \quad (976)$$

As the scaling factor γ can be left out, this is consistent with the mapping $\mathbf{w} = \mathbf{K} \mathbf{z}$. \square

18.3 The image of a line in Plücker coordinates with an uncalibrated camera

Consider the camera model $\mathbf{P} = [\mathbf{M} | \mathbf{m}]$ and a 3D line $\mathbf{L} = (x_4 \mathbf{y} - y_4 \mathbf{x}, \mathbf{x} \times \mathbf{y})$ defined by the two homogeneous points $\mathbf{X} = (\mathbf{x}, x_4)$ and $\mathbf{Y} = (\mathbf{y}, y_4)$ in 3D. The 3D points are mapped to the pixel points

$$\mathbf{p} = \mathbf{P} \mathbf{X} = \mathbf{M} \mathbf{x} + x_4 \mathbf{m} \quad (977)$$

$$\mathbf{q} = \mathbf{P} \mathbf{Y} = \mathbf{M} \mathbf{y} + y_4 \mathbf{m} \quad (978)$$

The image of the 3D line \mathbf{L} in pixel coordinates is then found from the pixel points $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{q}}$ as

$$\ell = \mathbf{p} \times \mathbf{q} \quad (979)$$

$$= (\mathbf{M} \mathbf{x} + x_4 \mathbf{m})^\times (\mathbf{M} \mathbf{y} + y_4 \mathbf{m}) \quad (980)$$

$$= (\mathbf{M} \mathbf{x})^\times (\mathbf{M} \mathbf{y}) + x_4 \mathbf{m}^\times (\mathbf{M} \mathbf{y}) - y_4 \mathbf{m} \times (\mathbf{M} \mathbf{x}) \quad (981)$$

$$= \det(\mathbf{M}) \mathbf{M}^{-T} \mathbf{x}^\times \mathbf{y} + \mathbf{m}^\times \mathbf{M} (y_4 \mathbf{x} - x_4 \mathbf{y}) \quad (982)$$

$$= \begin{bmatrix} \mathbf{m} \times \mathbf{M} & \det(\mathbf{M}) \mathbf{M}^{-T} \end{bmatrix} \begin{bmatrix} x_4 \mathbf{y} - y_4 \mathbf{x} \\ \mathbf{x}^\times \mathbf{y} \end{bmatrix} \quad (983)$$

This result was derived in [7].

18.4 Transformation of a line in Plücker coordinates by a homography

Consider a line given by Plücker coordinates in vector form as

$$\mathbf{L} = \begin{bmatrix} \mathbf{a} \\ \mathbf{m} \end{bmatrix} = \begin{bmatrix} x_4 \mathbf{y} - y_4 \mathbf{x} \\ \mathbf{x} \times \mathbf{y} \end{bmatrix} \quad (984)$$

The line is defined by the two homogenous points $\tilde{\mathbf{x}} = [\mathbf{x}^T, x_4]^T$ and $\tilde{\mathbf{y}} = [\mathbf{y}^T, y_4]^T$. The homography \mathbf{H} transforms the homogeneous points to the homogeneous points $\tilde{\mathbf{x}}' = [(\mathbf{x}')^T, x'_4]^T$ and $\tilde{\mathbf{y}}' = [(\mathbf{y}')^T, y'_4]^T$ given by $\tilde{\mathbf{x}}' = \mathbf{H} \tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}' = \mathbf{H} \tilde{\mathbf{y}}$. The resulting line transformation

$$\mathbf{L}' = \mathbf{P}_H \mathbf{L} \quad (985)$$

from \mathbf{L} to

$$\mathbf{L}' = \begin{bmatrix} \mathbf{a}' \\ \mathbf{m}' \end{bmatrix} = \begin{bmatrix} x'_4 \mathbf{y}' - y'_4 \mathbf{x}' \\ \mathbf{x}'^\times \mathbf{y}' \end{bmatrix} \quad (986)$$

is to be found. The derivation is taken from [6].

Suppose that the homography is given by

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v_4 \end{bmatrix} \quad (987)$$

Then

$$\begin{bmatrix} \mathbf{x}' \\ x'_4 \end{bmatrix} = \mathbf{H} \begin{bmatrix} \mathbf{x} \\ x_4 \end{bmatrix} = \begin{bmatrix} \mathbf{A}\mathbf{x} + x_4\mathbf{t} \\ \mathbf{v}^T\mathbf{x} + v_4x_4 \end{bmatrix} \quad (988)$$

$$\begin{bmatrix} \mathbf{y}' \\ y'_4 \end{bmatrix} = \mathbf{H} \begin{bmatrix} \mathbf{y} \\ y_4 \end{bmatrix} = \begin{bmatrix} \mathbf{A}\mathbf{y} + y_4\mathbf{t} \\ \mathbf{v}^T\mathbf{y} + v_4y_4 \end{bmatrix} \quad (989)$$

Then the transformed direction vector is

$$\mathbf{a}' = x'_4\mathbf{y}' - y'_4\mathbf{x}' \quad (990)$$

$$= (\mathbf{v}^T\mathbf{x} + v_4x_4)(\mathbf{A}\mathbf{y} + y_4\mathbf{t}) - (\mathbf{v}^T\mathbf{y} + v_4y_4)(\mathbf{A}\mathbf{x} + x_4\mathbf{t}) \quad (991)$$

$$= \mathbf{v}^T(\mathbf{x}\mathbf{A}\mathbf{y} - \mathbf{y}\mathbf{A}\mathbf{x}) + \mathbf{v}^T(y_4\mathbf{x} - x_4\mathbf{y})\mathbf{t} + v_4\mathbf{A}(x_4\mathbf{y} - y_4\mathbf{x}) \quad (992)$$

$$= \mathbf{A}(\mathbf{y}\mathbf{x}^T - \mathbf{x}\mathbf{y}^T)\mathbf{v} - \mathbf{t}\mathbf{v}^T\mathbf{a} + \mathbf{A}v_4\mathbf{a} \quad (993)$$

$$= -\mathbf{A}\mathbf{v}^\times\mathbf{m} + (\mathbf{A}v_4 - \mathbf{t}\mathbf{v}^T)\mathbf{a} \quad (994)$$

where it is used that $\mathbf{u}^T\mathbf{w}\mathbf{z} = \mathbf{z}\mathbf{u}^T\mathbf{w} = \mathbf{z}\mathbf{w}^T\mathbf{u}$, and that $\mathbf{a}^\times = (\mathbf{x}^\times\mathbf{y})^\times = \mathbf{y}\mathbf{x}^T - \mathbf{x}\mathbf{y}^T$. The transformed moment is found to be

$$\mathbf{m}' = \mathbf{x}' \times \mathbf{y}' \quad (995)$$

$$= (\mathbf{A}\mathbf{x} + x_4\mathbf{t}) \times (\mathbf{A}\mathbf{y} + y_4\mathbf{t}) \quad (996)$$

$$= (\mathbf{A}\mathbf{x}) \times (\mathbf{A}\mathbf{y}) + \mathbf{t}^\times\mathbf{A}(x_4\mathbf{y} - y_4\mathbf{x}) \quad (997)$$

$$= \det(\mathbf{A})\mathbf{A}^{-T}\mathbf{m} + \mathbf{t}^\times\mathbf{A}\mathbf{a} \quad (998)$$

It is seen that the line transformation $\mathbf{L}' = \mathbf{P}_H\mathbf{L}$ is given by

$$\mathbf{P}_H = \begin{bmatrix} \mathbf{A}v_4 - \mathbf{t}\mathbf{v}^T & -\mathbf{A}\mathbf{v}^\times \\ \mathbf{t}^\times\mathbf{A} & \det(\mathbf{A})\mathbf{A}^{-T} \end{bmatrix} \quad (999)$$

Example

Suppose that

$$\mathbf{P}_H = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{21} & \mathbf{P}_{22} \end{bmatrix} \quad (1000)$$

is known. Then it is used that $\det(\mathbf{B}^{-1}) = 1/\det(\mathbf{B})$ and $\det(\alpha\mathbf{B}) = \alpha^n \det(\mathbf{B})$ when $\mathbf{B} \in \mathbb{R}^{3 \times 3}$. This gives $\det \mathbf{P}_{22} = \det(\mathbf{A})^2$. The elements of \mathbf{H} can then be found from [6]

$$\mathbf{A} = \pm \sqrt{|\det(\mathbf{P}_{22})|} \mathbf{P}_{22}^{-T} \quad (1001)$$

$$\mathbf{t}^\times = \mathbf{P}_{21}\mathbf{A}^{-1} \quad (1002)$$

$$\mathbf{v}^\times = -\mathbf{A}^{-1}\mathbf{P}_{12} \quad (1003)$$

$$v_4\mathbf{I} = (\mathbf{P}_{11} + \mathbf{t}\mathbf{v}^T)\mathbf{A}^{-1} \quad (1004)$$

Example

If the homography describes rigid motion, which means that the homography is given by

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3) \quad (1005)$$

then

$$\mathbf{P}_H = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{t}^{\times} \mathbf{R} & \mathbf{R} \end{bmatrix} \quad (1006)$$

which is the usual screw transformation for Plücker lines. \square

18.5 Fundamental matrix for standard cameras

Suppose that the standard camera matrices $\mathbf{P}_1 = \mathbf{K}_1[\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}_2 = \mathbf{K}_2[\mathbf{R} \mid \mathbf{t}]$ are used, and that the world point is given by $\mathbf{X} = [\mathbf{r}^T, 1]^T$. Then the homogeneous pixel coordinate vectors \mathbf{x}_1 and \mathbf{x}_2 are given by

$$\lambda_1 \mathbf{x}_1 = \mathbf{P}_1 \mathbf{X} = \mathbf{K}_1 \mathbf{r} \quad (1007)$$

$$\lambda_2 \mathbf{x}_2 = \mathbf{P}_2 \mathbf{X} = \mathbf{K}_2(\mathbf{R} \mathbf{r} + \mathbf{t}) \quad (1008)$$

The fundamental matrix is given by

$$\mathbf{F} = \mathbf{K}_2^{-T} \mathbf{E} \mathbf{K}_1^{-1} = \mathbf{K}_2^{-T} \mathbf{t}^{\times} \mathbf{R} \mathbf{K}_1^{-1} = \mathbf{K}_2^{-T} \mathbf{t}^{\times} \mathbf{K}_2^{-1} \mathbf{K}_2 \mathbf{R} \mathbf{K}_1^{-1} = \gamma(\mathbf{K}_2 \mathbf{t})^{\times} \mathbf{K}_2 \mathbf{R} \mathbf{K}_1^{-1} \quad (1009)$$

where $\gamma = 1 / \det(\mathbf{K}_2)$. Let $\mathbf{z}_2 = \mathbf{K}_2 \mathbf{e}_2$ be the epipole in image 2 in pixels coordinates. Then, as the fundamental matrix can be scaled freely, and $\mathbf{e}_2 = \lambda_2 \mathbf{t}$, it follows that

$$\mathbf{F} = (\mathbf{K}_2 \mathbf{e}_2)^{\times} \mathbf{K}_2 \mathbf{R} \mathbf{K}_1^{-1} = \mathbf{z}_2^{\times} \mathbf{K}_2 \mathbf{R} \mathbf{K}_1^{-1} \quad (1010)$$

18.6 Fundamental matrix for standard cameras

Suppose that the standard camera matrices $\mathbf{P}_1 = [\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}_2 = [\mathbf{M} \mid \mathbf{m}]$ are used, and that the world point is given by $\mathbf{X} = [\mathbf{r}^T, 1]^T$. Then the homogeneous pixel coordinate vectors \mathbf{x}_1 and \mathbf{x}_2 are given by

$$\lambda_1 \mathbf{x}_1 = \mathbf{P}_1 \mathbf{X} = \mathbf{r} \quad (1011)$$

$$\lambda_2 \mathbf{x}_2 = \mathbf{P}_2 \mathbf{X} = \mathbf{M} \mathbf{r} + \mathbf{m} \quad (1012)$$

Then the fundamental matrix is

$$\mathbf{F} = \mathbf{m}^{\times} \mathbf{M} \quad (1013)$$

This is verified from

$$\mathbf{x}_2^T \mathbf{F} \mathbf{x}_1 = \underbrace{(\mathbf{M} \mathbf{r})^T (\mathbf{m}^{\times} \mathbf{M} \mathbf{r})}_{=0} + \underbrace{\mathbf{m}^T \mathbf{m}^{\times} \mathbf{M} \mathbf{r}}_{=0} = 0 \quad (1014)$$

The fundamental matrix is unique up to a scale. Note that this agrees with the expression for the essential matrix which is $\mathbf{E} = \mathbf{t}^{\times} \mathbf{R}$.

It is noted that

$$\mathbf{F}^T \mathbf{m} = \mathbf{0} \quad (1015)$$

18.7 Calculation of the fundamental matrix from point correspondences

Suppose that \mathbf{x}_1 and \mathbf{x}_2 are corresponding homogeneous points in pixel coordinates so that the epipolar constraint is written

$$\mathbf{x}_2^{i\text{T}} \mathbf{F} \mathbf{x}_1^i = 0 \quad (1016)$$

where $\mathbf{F} = \{f_{ij}\}$ is the fundamental matrix. The fundamental matrix has 9 elements. Due to the homogeneous characteristics of the matrix there are 8 independent elements. This means that 8 equations are required to determine the elements of F within a scale.

Let the coordinates of the vectors be given by $\mathbf{x}_1 = [x_1, y_1, 1]^T$ and $\mathbf{x}_2 = [x_2, y_2, 1]^T$. Then the epipolar constraint is

$$x_2 x_1 f_{11} + x_2 y_1 f_{12} + x_2 f_{13} + y_2 x_1 f_{21} + y_2 y_1 f_{22} + y_2 f_{23} + x_1 f_{31} + y_1 f_{32} + f_{33} = 0 \quad (1017)$$

which can be written

$$[x_2 x_1, x_2 y_1, x_2, y_2 x_1, y_2 y_1, y_2, x_1, y_1, 1] \mathbf{f} = \mathbf{0} \quad (1018)$$

where $\mathbf{f} = [\mathbf{f}_1^T, \mathbf{f}_2^T, \mathbf{f}_3^T]^T$ and \mathbf{f}_i is column i in \mathbf{F} . A solution for \mathbf{F} can then be found from 8 point mappings, which each gives the equation

$$\mathbf{A}_i \mathbf{f} = \mathbf{0}, \quad \mathbf{A}_i = [x_2^i x_1^i, x_2^i y_1^i, x_2^i, y_2^i x_1^i, y_2^i y_1^i, y_2^i, x_1^i, y_1^i, 1] \quad (1019)$$

The solution is then found from

$$\mathbf{A} \mathbf{f} = \mathbf{0}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_8 \end{bmatrix} \quad (1020)$$

as $\mathbf{f} = \mathbf{v}_8$, where \mathbf{v}_8 is the the last column in \mathbf{V} found from the singular value decomposition $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$.

The data sets $\{\mathbf{x}_1^i\}$ and $\{\mathbf{x}_2^i\}$ must be normalized to

$$\mathbf{y}_1^i = \mathbf{N}_1 \mathbf{x}_1^i, \quad \mathbf{y}_2^i = \mathbf{N}_2 \mathbf{x}_2^i \quad (1021)$$

so that the inhomogeneous parts of \mathbf{y}_1^i and \mathbf{y}_2^i have the centroid at the origin, and and average distance from the origin of $\sqrt{2}$. These normalized data set are then used to calculate a normalized \mathbf{F}_n that satisfies

$$\mathbf{y}_2^{i\text{T}} \mathbf{F}_n \mathbf{y}_1^i = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{x}_2^{i\text{T}} \mathbf{N}_2^T \mathbf{F}_n \mathbf{N}_1 \mathbf{x}_1^i = \mathbf{0} \quad (1022)$$

The fundamental matrix is then recovered as

$$\mathbf{F} = \mathbf{N}_2^T \mathbf{F}_n \mathbf{N}_1 \quad (1023)$$

One more correction is required. The true fundamental matrix is of rank 2. Therefore, it must be ensured that the computed solution for the fundamental matrix has rank 2. This should be done for the normalized fundamental matrix \mathbf{F}_n , where the rank condition is ensured using the singular value decomposition $\mathbf{F}_n = \sum_{i=1}^3 \sigma_i \mathbf{u}_i \mathbf{v}_i^T$. Then the solution with rank 2 that is closest to \mathbf{F}_n in the Frobenius norm is $\mathbf{F}_{nr} = \sum_{i=1}^2 \sigma_i \mathbf{u}_i \mathbf{v}_i^T$, which means that the smallest singular value is set to zero. Then the fundamental matrix is computed as $\mathbf{F} = \mathbf{N}_2^T \mathbf{F}_{nr} \mathbf{N}_1$.

18.8 Sampson error of the uncalibrated epipolar constraint

A further improvement of the accuracy of the computed fundamental matrix is possible can be achieved by minimizing the reprojection error associated with the fundamental matrix. The data sets $\mathbf{x}_1^i = [x_1^i, y_1^i, 1]^T$ and $\mathbf{x}_2^i = [x_2^i, y_2^i, 1]^T$ are given from observations, and let \mathbf{F} be the computed fundamental matrix. Let $\hat{\mathbf{x}}_1^i = [\hat{x}_1^i, \hat{y}_1^i, 1]^T$ and $\hat{\mathbf{x}}_2^i = [\hat{x}_2^i, \hat{y}_2^i, 1]^T$ be the corresponding data sets that satisfy the epipolar constraint

$$\hat{\mathbf{x}}_2^{iT} \mathbf{F} \hat{\mathbf{x}}_1^i = 0 \quad (1024)$$

Then the reprojection error is

$$d_r(\mathbf{F})^2 = \sum_{i=1}^n (\mathbf{x}_1^i - \hat{\mathbf{x}}_1^i)^T (\mathbf{x}_1^i - \hat{\mathbf{x}}_1^i) + (\mathbf{x}_2^i - \hat{\mathbf{x}}_2^i)^T (\mathbf{x}_2^i - \hat{\mathbf{x}}_2^i) \quad (1025)$$

$$= \sum_{i=1}^n (x_1^i - \hat{x}_1^i)^2 + (y_1^i - \hat{y}_1^i)^2 + (x_2^i - \hat{x}_2^i)^2 + (y_2^i - \hat{y}_2^i)^2 \quad (1026)$$

Define $\mathbf{q}^i = [x_1^i, y_1^i, x_2^i, y_2^i]^T$, $\hat{\mathbf{q}}^i = [\hat{x}_1^i, \hat{y}_1^i, \hat{x}_2^i, \hat{y}_2^i]^T$ and $\delta \mathbf{q}^i = \hat{\mathbf{q}}^i - \mathbf{q}^i$, which gives

$$d_r(\mathbf{F})^2 = \delta \mathbf{q}^T \delta \mathbf{q} \quad (1027)$$

Moreover, let $g(\mathbf{q}) = \mathbf{x}_2^T \mathbf{F} \mathbf{x}_1$, and let the fundamental matrix be decomposed in terms of its column vectors \mathbf{f}_i and row vectors \mathbf{f}^i as

$$\mathbf{F} = [\mathbf{f}_1 \quad \mathbf{f}_2 \quad \mathbf{f}_3] = \begin{bmatrix} \mathbf{f}^{1T} \\ \mathbf{f}^{2T} \\ \mathbf{f}^{3T} \end{bmatrix} \quad (1028)$$

The Sampson error is found from the linear approximation $0 = g(\hat{\mathbf{q}}) = g(\mathbf{q}) + \nabla^T \mathbf{g}(\mathbf{q}) \delta \mathbf{q}$, which gives

$$\delta \mathbf{q}^2 = \frac{g(\mathbf{q})^2}{|\nabla^T \mathbf{g}(\mathbf{q})|^2} \quad (1029)$$

The gradient is

$$\nabla^T \mathbf{g}(\mathbf{q}) = \left[\frac{\partial g}{\partial x_1} \quad \frac{\partial g}{\partial y_1} \quad \frac{\partial g}{\partial x_2} \quad \frac{\partial g}{\partial y_2} \right] = [\mathbf{f}_1^T \mathbf{x}_2^i \quad \mathbf{f}_2^T \mathbf{x}_2^i \quad \mathbf{f}^{1T} \mathbf{x}_1^i \quad \mathbf{f}^{2T} \mathbf{x}_1^i] \quad (1030)$$

The reprojection is then found from the Sampson error for the fundamental matrix is found to be

$$d_r^2(\mathbf{F}) = \sum_{i=1}^n \frac{(\mathbf{x}_2^{iT} \mathbf{F} \mathbf{x}_1^i)^2}{(\mathbf{f}_1^T \mathbf{x}_2^i)^2 + (\mathbf{f}_2^T \mathbf{x}_2^i)^2 + (\mathbf{f}^{1T} \mathbf{x}_1^i)^2 + (\mathbf{f}^{2T} \mathbf{x}_1^i)^2} \quad (1031)$$

which is minimized with respect to the elements of \mathbf{F} .

18.9 The ambiguity in the reconstruction from two uncalibrated cameras

Reconstruction is considered based on the standard camera matrices

$$\mathbf{P}_1 = \mathbf{K}_1 [\mathbf{I} \quad \mathbf{0}] \quad (1032)$$

$$\mathbf{P}_2 = \mathbf{K}_2 [\mathbf{R} \quad \mathbf{t}] \quad (1033)$$

are used. Note that the matrices \mathbf{K}_1 and \mathbf{K}_2 are unknown as it is assumed that the cameras are not calibrated.

Then the homogeneous uncalibrated image coordinate vectors \mathbf{x}_1 and \mathbf{x}_2 , which are the pixel coordinate vectors, are given by

$$\lambda_1 \mathbf{x}_1 = \mathbf{P}_1 \mathbf{X} \quad (1034)$$

$$\lambda_2 \mathbf{x}_2 = \mathbf{P}_2 \mathbf{X} \quad (1035)$$

If \mathbf{x}_1 and \mathbf{x}_2 are given, there is a certain ambiguity in the determination of the transformation \mathbf{T} between the two cameras. To describe this ambiguity, the homography $\mathbf{H} \in \mathbb{R}^{4 \times 4}$ is introduced. The homography is by definition invertible. Then, it follows that

$$\lambda_1 \mathbf{x}_1 = \mathbf{P}_1 \mathbf{X} = (\mathbf{P}_1 \mathbf{H}^{-1})(\mathbf{H} \mathbf{X}) = \mathbf{P}_{1p} \mathbf{X}_p \quad (1036)$$

$$\lambda_2 \mathbf{x}_2 = \mathbf{P}_2 \mathbf{X} = (\mathbf{P}_2 \mathbf{H}^{-1})(\mathbf{H} \mathbf{X}) = \mathbf{P}_{2p} \mathbf{X}_p \quad (1037)$$

Here,

$$\mathbf{P}_{1p} = \mathbf{P}_1 \mathbf{H}^{-1} \quad (1038)$$

$$\mathbf{P}_{2p} = \mathbf{P}_2 \mathbf{H}^{-1} \quad (1039)$$

are called the projective camera models, which correspond to the projective 3D point

$$\mathbf{X}_p = \mathbf{H} \mathbf{X} \quad (1040)$$

The ambiguity of the reconstruction from uncalibrated views is that there is no way to determine the projective homography \mathbf{H} from the uncalibrated image data \mathbf{x}_1 and \mathbf{x}_2 , therefore a reconstruction will result in a projected scene description \mathbf{X}_p .

18.10 Decomposition of projective homography

The homography \mathbf{H} which appears in the camera models $\mathbf{P}_{1p} = \mathbf{P}_1 \mathbf{H}^{-1}$ can be decomposed in an affine transformation \mathbf{H}_a , a projective transformation \mathbf{H}_p and a rigid displacement \mathbf{H}_e so that

$$\mathbf{H} = \mathbf{H}_p \mathbf{H}_a \mathbf{H}_e \Leftrightarrow \mathbf{H}^{-1} = \mathbf{H}_e^{-1} \mathbf{H}_a^{-1} \mathbf{H}_p^{-1} \quad (1041)$$

Then, as intermediate representations between the real scene \mathbf{X} and the projective scene \mathbf{X}_p it is convenient to define a Euclidean scene $\mathbf{X}_e = \mathbf{H}_e \mathbf{X}$ and an affine scene $\mathbf{X}_a = \mathbf{H}_a \mathbf{X}_e$ so that

$$\mathbf{X}_p = \mathbf{H}_p \mathbf{X}_a = \mathbf{H}_p \mathbf{H}_a \mathbf{X}_e = \mathbf{H}_p \mathbf{H}_a \mathbf{H}_e \mathbf{X} \quad (1042)$$

In the same way intermediate cameras models are defined for camera 1. The Euclidean camera model is $\mathbf{P}_{1e} = \mathbf{P}_1 \mathbf{H}_e^{-1}$, and the the affine camera is $\mathbf{P}_a = \mathbf{P}_e \mathbf{H}_a^{-1}$ so that

$$\mathbf{P}_{1p} = \mathbf{P}_{1a} \mathbf{H}_p^{-1} = \mathbf{P}_{1e} \mathbf{H}_a^{-1} \mathbf{H}_p^{-1} = \mathbf{P}_1 \mathbf{H}_e^{-1} \mathbf{H}_a^{-1} \mathbf{H}_p^{-1} \quad (1043)$$

It is then straightforward to verify that

$$\mathbf{P}_{1p} \mathbf{X}_p = \mathbf{P}_{1a} \mathbf{X}_a = \mathbf{P}_{1e} \mathbf{X}_e = \mathbf{P}_1 \mathbf{X} \quad (1044)$$

The rigid displacement is written

$$\mathbf{H}_e = \begin{bmatrix} \mathbf{R}_e & \mathbf{t}_e \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3) \quad (1045)$$

It is noted that

$$\mathbf{X}_e = \begin{bmatrix} \mathbf{R}_e & \mathbf{t}_e \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{X} \quad (1046)$$

which is a rotation \mathbf{R}_e and a translation \mathbf{t}_e .

Expressions for the transformations \mathbf{H}_a and \mathbf{H}_p will now be developed for camera 1. The assumptions are that the Euclidean camera model is the standard model $\mathbf{P}_{1e} = \mathbf{K}_1[\mathbf{I} \mid \mathbf{0}]$, while the projective camera model is $\mathbf{P}_{1p} = [\mathbf{I} \mid \mathbf{0}]$. This means that the transformations \mathbf{H}_a and \mathbf{H}_p must satisfy

$$[\mathbf{I} \mid \mathbf{0}] = \mathbf{K}_1 [\mathbf{I} \mid \mathbf{0}] \mathbf{H}_a^{-1} \mathbf{H}_p^{-1} \quad (1047)$$

This is will be satisfied if the transformations are selected so that

$$\mathbf{H}_a^{-1} = \begin{bmatrix} \mathbf{K}_1^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{H}_p^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_4 \end{bmatrix} \quad (1048)$$

This is verified from the calculation

$$\mathbf{P}_{1e} \mathbf{H}_a^{-1} \mathbf{H}_p^{-1} = \mathbf{K}_1 [\mathbf{I} \mid \mathbf{0}] \begin{bmatrix} \mathbf{K}_1^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_4 \end{bmatrix} = [\mathbf{I} \mid \mathbf{0}] \quad (1049)$$

18.11 Camera models in the uncalibrated reconstruction

As described in the previous section, camera 1 will have a Euclidean model $\mathbf{P}_{1e} = \mathbf{K}_1[\mathbf{I}, \mathbf{0}]$ and a projective model $\mathbf{P}_{1p} = [\mathbf{I}, \mathbf{0}]$. The affine model is then

$$\mathbf{P}_{1a} = \mathbf{P}_{1e} \mathbf{H}_a^{-1} = \mathbf{K}_1 [\mathbf{I} \mid \mathbf{0}] \begin{bmatrix} \mathbf{K}_1^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} = [\mathbf{I} \mid \mathbf{0}] \quad (1050)$$

The Euclidean model for camera 2 is $\mathbf{P}_{2e} = \mathbf{K}_2[\mathbf{R}, \mathbf{t}] = [\mathbf{K}_2\mathbf{R}, \mathbf{K}_2\mathbf{t}]$. Then the affine model is

$$\mathbf{P}_{2a} = \mathbf{P}_{2e} \mathbf{H}_a^{-1} = [\mathbf{K}_2\mathbf{R} \quad \mathbf{K}_2\mathbf{t}] \begin{bmatrix} \mathbf{K}_1^{-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} = [\mathbf{K}_2\mathbf{R}\mathbf{K}_1^{-1} \quad \mathbf{K}_2\mathbf{t}] \quad (1051)$$

The projective model is found to be

$$\mathbf{P}_{2p} = [\mathbf{K}_2\mathbf{R}\mathbf{K}_1^{-1} \quad \mathbf{K}_2\mathbf{t}] \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_4 \end{bmatrix} = [\mathbf{K}_2\mathbf{R}\mathbf{K}_1^{-1} + \mathbf{K}_2\mathbf{t}\mathbf{v}^T \quad v_4\mathbf{K}_2\mathbf{t}] \quad (1052)$$

To summarize, the camera models for camera 1 are

$$\mathbf{P}_{1e} = [\mathbf{K}_1 \mid \mathbf{0}] \quad (1053)$$

$$\mathbf{P}_{1a} = [\mathbf{I} \mid \mathbf{0}] \quad (1054)$$

$$\mathbf{P}_{1p} = [\mathbf{I} \mid \mathbf{0}] \quad (1055)$$

The camera models for camera 2 are

$$\mathbf{P}_{2e} = [\mathbf{K}_2 \mathbf{R} \quad \mathbf{K}_2 \mathbf{t}] \quad (1056)$$

$$\mathbf{P}_{2a} = [\mathbf{K}_2 \mathbf{R} \mathbf{K}_1^{-1} \quad \mathbf{K}_2 \mathbf{t}] \quad (1057)$$

$$\mathbf{P}_{2p} = [\mathbf{K}_2 \mathbf{R} \mathbf{K}_1^{-1} + \mathbf{K}_2 \mathbf{t} \mathbf{v}^T \quad v_4 \mathbf{K}_2 \mathbf{t}] \quad (1058)$$

The scene is given in the Euclidean, affine and projective representations as

$$\mathbf{X}_e = \begin{bmatrix} \mathbf{R}_e & \mathbf{t}_e \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{X} \quad (1059)$$

$$\mathbf{X}_a = \begin{bmatrix} \mathbf{K}_1 & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{X}_e \quad (1060)$$

$$\mathbf{X}_p = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{v}^T/v_4 & 1/v_4 \end{bmatrix} \mathbf{X}_a \quad (1061)$$

18.12 The fundamental matrix and the projective ambiguity

Consider two camera models $\mathbf{P}_{1pa} = [\mathbf{I} \mid \mathbf{0}]$, $\mathbf{P}_{2pa} = [\mathbf{M}_a \mid \mathbf{m}_a]$ and $\mathbf{P}_{1pb} = [\mathbf{I} \mid \mathbf{0}]$, $\mathbf{P}_{2pb} = [\mathbf{M}_b \mid \mathbf{m}_b]$ that are due to the projective ambiguity. This means that the two camera models correspond to the same set of two cameras with fundamental matrix \mathbf{F} . This means that the fundamental matrix can be written in the two alternative forms $\mathbf{F} = \mathbf{m}_a^\times \mathbf{M}_a$ and $\mathbf{F} = \mathbf{m}_b^\times \mathbf{M}_b$. This means that up to a scale,

$$\mathbf{m}_a^\times \mathbf{M}_a = \mathbf{m}_b^\times \mathbf{M}_b \quad (1062)$$

Moreover, since \mathbf{F} is of rank two, and

$$\mathbf{F}^T \mathbf{m}_a = \mathbf{0}, \quad \mathbf{F}^T \mathbf{m}_b = \mathbf{0} \quad (1063)$$

it follows that \mathbf{m}_a and \mathbf{m}_b are linearly dependent, so that $\mathbf{m}_a = \gamma \mathbf{m}_b$ for some scalar γ . This implies that the models $\mathbf{P}_{2pa} = [\mathbf{M}_a \mid \mathbf{m}_a]$ and $\mathbf{P}_{2pb} = [\mathbf{M}_b \mid \mathbf{m}_b]$ correspond to the same fundamental matrix \mathbf{F} whenever

$$\mathbf{M}_a = \mathbf{M}_b + \mathbf{m}_b \mathbf{v}^T \quad (1064)$$

for some $\mathbf{v} \in \mathbb{R}^3$, which is seen from

$$\mathbf{F} = \mathbf{m}_a^\times \mathbf{M}_a = \gamma \mathbf{m}_b^\times (\mathbf{M}_b + \mathbf{m}_b \mathbf{v}^T) = \gamma \mathbf{m}_b^\times \mathbf{M}_b + \gamma \mathbf{m}_b^\times \mathbf{m}_b \mathbf{v}^T = \gamma \mathbf{m}_b^\times \mathbf{M}_b \quad (1065)$$

It follows that the two camera model are related by the projective transformation \mathbf{H}_{ps} as

$$[\mathbf{M}_a \quad \mathbf{m}_a] = [\mathbf{M}_b \quad \mathbf{m}_b] \mathbf{H}_{ps} \quad (1066)$$

where

$$\mathbf{H}_{ps} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_4 \end{bmatrix} \quad (1067)$$

18.13 A projective camera model from the fundamental matrix

A projective reconstruction from uncalibrated cameras is based on first finding the fundamental matrix \mathbf{F} from point correspondences. Then a camera model is established from the fundamental matrix. Due to the ambiguity of the uncalibrated reconstruction, the resulting camera models, and hence the reconstruction will be projective.

The projective reproduction is therefore the procedure to find a camera model $\mathbf{P}_{1p} = [\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}_{2p} = [\mathbf{M} \mid \mathbf{m}]$ from a given fundamental matrix \mathbf{F} . The fundamental matrix is related to the camera model by $\mathbf{F} = \mathbf{m}^\times \mathbf{M}$.

Then a projective camera model can be found from the fundamental matrix as

$$\mathbf{P}_{1p} = [\mathbf{I} \quad \mathbf{0}] \quad (1068)$$

$$\mathbf{P}_{2p} = [-\mathbf{m}^\times \mathbf{F} \quad \mathbf{m}] \quad (1069)$$

where $\mathbf{M} = -\mathbf{m}^\times \mathbf{F}$ and \mathbf{m} is determined from the equation

$$\mathbf{F}^T \mathbf{m} = \mathbf{0} \quad (1070)$$

This solution for the camera model is called the canonical decomposition of a fundamental matrix into two camera models [41].

The vector \mathbf{m} is found with the singular value decomposition $\mathbf{F}^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + 0 \cdot \mathbf{u}_3 \mathbf{v}_3^T$, which gives $\mathbf{m} = \mathbf{v}_3$. It is verified that

$$\mathbf{m}^\times \mathbf{M} = -\mathbf{m}^\times \mathbf{m}^\times \mathbf{F} = \underbrace{\mathbf{m}^T \mathbf{m}}_{=1} \mathbf{I} \mathbf{F} - \mathbf{m} \underbrace{\mathbf{m}^T \mathbf{F}}_{=0} = \mathbf{F} \quad (1071)$$

The matrix $-\mathbf{m}^\times \mathbf{F}$ which is used in this canonical decomposition is singular, as opposed to the Euclidean and affine version $\mathbf{K} \mathbf{R} \mathbf{K}^{-1}$.

18.14 Projective reconstruction from uncalibrated views

The projective reconstruction is done based on the camera model

$$\lambda_1 \mathbf{x}_1 = \mathbf{P}_{1p} \mathbf{X}_p \quad (1072)$$

$$\lambda_2 \mathbf{x}_2 = \mathbf{P}_{2p} \mathbf{X}_p \quad (1073)$$

where $\mathbf{P}_{1p} = [\mathbf{I}, \mathbf{0}]$ and $\mathbf{P}_{2p} = [-\mathbf{m}^\times \mathbf{F}, \mathbf{m}]$, where the fundamental matrix \mathbf{F} has been found from using the eight-point algorithm, and \mathbf{m} is found from $\mathbf{F}^T \mathbf{m} = \mathbf{0}$.

Both equations of the camera model are homogeneous, and the depth factors λ_1 and λ_2 are eliminated by premultiplication with \mathbf{x}_i^\times . This gives

$$\mathbf{x}_1^\times \mathbf{P}_{1p} \mathbf{X}_p = \mathbf{0} \quad (1074)$$

$$\mathbf{x}_2^\times \mathbf{P}_{2p} \mathbf{X}_p = \mathbf{0} \quad (1075)$$

where \mathbf{x}_1 and \mathbf{x}_2 are the known uncalibrated image coordinates, $\mathbf{P}_{1p} = [\mathbf{I}, \mathbf{0}]$ and \mathbf{P}_{2p} have been found from the fundamental matrix. As usual, this is rewritten in the form

$$\mathbf{A} \mathbf{X}_p = \mathbf{0} \quad (1076)$$

which is solved for \mathbf{X}_p using the singular value decomposition. This results in a projective scene \mathbf{X}_p in the sense that $\mathbf{X}_p = \mathbf{H}_p \mathbf{X}$ where \mathbf{H}_p is a projective homography and \mathbf{X} is the physical scene in Euclidean space.

18.15 Affine reconstruction from projective reconstruction

Suppose that a projective reconstruction has been done, and the result is the camera model $\mathbf{P}_{1p} = [\mathbf{I}, \mathbf{0}]$ and $\mathbf{P}_{2p} = [\mathbf{M}, \mathbf{m}]$ and the scene \mathbf{X}_p . The scene is a projective transformation of the real world \mathbf{X} , and can be written as already explained by $\mathbf{X}_p = \mathbf{H}_{ps}\mathbf{X}_a = \mathbf{H}_{ps}\mathbf{H}_{as}\mathbf{X}$, where \mathbf{X}_a is an affine representation of the real world. The affine scene \mathbf{X}_a is characterized by the property that parallel lines are parallel, which means that all vanishing points have been transformed to infinity, and will not appear in the image. The goal is to find the real Euclidean scene, and one method is to do this by first finding the affine scene. This was introduced as a stratified reconstruction method in [21].

The next step is therefore to find the projective transformation

$$\mathbf{H}_{ps}^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_4 \end{bmatrix} \quad (1077)$$

so that an affine reconstruction $\mathbf{X}_a = \mathbf{H}_{ps}^{-1}\mathbf{X}_p$ can be found. A solution to this problem is to find the plane π_∞ at infinity in the projective scene, and select the last row in \mathbf{H}_{ps}^{-1} so that $\pi_\infty = [\mathbf{v}^T, v_4]^T$. Then any vanishing point $\mathbf{Z}_p = [\mathbf{z}^T, 1]^T$ in the projective scene will be on the plane at infinity π_∞ , which means that

$$\pi_\infty^T \mathbf{Z}_p = [\mathbf{v}^T \ v_4] \begin{bmatrix} \mathbf{z} \\ 1 \end{bmatrix} = 0 \quad (1078)$$

This implies that in the affine scene, the vanishing point is mapped to the point

$$\mathbf{Z}_a = \mathbf{H}_{ps}^{-1}\mathbf{Z}_p = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_4 \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{z} \\ 0 \end{bmatrix} \quad (1079)$$

which is at infinity in the direction \mathbf{z} . This shows that the scene given by \mathbf{X}_a is affine.

The plane at infinity can be found in the projective scene if three vanishing points can be identified.

18.16 Affine reconstruction from uncalibrated views

An affine reconstruction is possible directly from the uncalibrated image data of two cameras if the displacement between the two cameras is a pure translation [41]. In this case the camera models are $\mathbf{P}_1 = \mathbf{K}[\mathbf{I}, \mathbf{0}]$ and $\mathbf{P}_2 = \mathbf{K}[\mathbf{I}, \mathbf{t}]$ where it is assumed that the intrinsic parameters are the same so that $\mathbf{K}_1 = \mathbf{K}_2 = \mathbf{K}$. Then

$$\lambda_1 \mathbf{x}_1 = \mathbf{K} [\mathbf{I} \ \mathbf{0}] \mathbf{X} \quad (1080)$$

$$\lambda_2 \mathbf{x}_2 = \mathbf{K} [\mathbf{I} \ \mathbf{t}] \mathbf{X} \quad (1081)$$

and it follows that

$$\lambda_1 \mathbf{x}_1 = \lambda_2 \mathbf{x}_2 + \mathbf{m} \quad (1082)$$

where $\mathbf{m} = \mathbf{K}\mathbf{t}$. It is noted that in this case, the fundamental matrix is $\mathbf{F} = \mathbf{m}^\times$, which means that \mathbf{m} is found from the fundamental matrix using the epipolar constraint $\mathbf{x}_2^T \mathbf{F} \mathbf{x}_1 = 0$. An

equation for the scale factor λ_1 can then be found by eliminating the λ_2 term by premultiplication with \mathbf{x}_2^\times . In the same way an equation for λ_2 can be found by premultiplication with \mathbf{x}_1^\times . This gives

$$\lambda_1 \mathbf{x}_2^\times \mathbf{x}_1 = \mathbf{x}_2^\times \mathbf{m} \quad (1083)$$

$$\lambda_2 \mathbf{x}_1^\times \mathbf{x}_2 = -\mathbf{x}_1^\times \mathbf{m} \quad (1084)$$

Premultiplication of the first equation with $(\mathbf{x}_2^\times \mathbf{x}_1)^\text{T}$ and the second equation with $(\mathbf{x}_1^\times \mathbf{x}_2)^\text{T}$ then gives the expressions

$$\lambda_1 = \frac{(\mathbf{x}_2^\times \mathbf{x}_1)^\text{T} (\mathbf{x}_2^\times \mathbf{m})}{(\mathbf{x}_2^\times \mathbf{x}_1)^2}, \quad \lambda_2 = -\frac{(\mathbf{x}_1^\times \mathbf{x}_2)^\text{T} (\mathbf{x}_1^\times \mathbf{m})}{(\mathbf{x}_1^\times \mathbf{x}_2)^2} \quad (1085)$$

18.17 Euclidean reconstruction

Suppose that the affine camera models are

$$\mathbf{P}_{1a} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \quad (1086)$$

$$\mathbf{P}_{2a} = \begin{bmatrix} \mathbf{M} & \mathbf{m} \end{bmatrix} \quad (1087)$$

where $\mathbf{M} = \mathbf{K} \mathbf{R} \mathbf{K}^{-1}$ and $\mathbf{m} = \mathbf{K} \mathbf{t}$ has been determined. The next step is to find \mathbf{K} from \mathbf{M} , so that the affine transformation

$$\mathbf{H}_a^{-1} = \begin{bmatrix} \mathbf{K}^{-1} & \mathbf{0} \\ \mathbf{0}^\text{T} & 1 \end{bmatrix} \quad (1088)$$

can be found. This is done by first finding the matrix $\boldsymbol{\omega}^* = \mathbf{K} \mathbf{K}^\text{T}$. An expression for $\boldsymbol{\omega}^*$ is found by noting that

$$\mathbf{M} \boldsymbol{\omega}^* \mathbf{M}^\text{T} = (\mathbf{K} \mathbf{R} \mathbf{K}^{-1})(\mathbf{K} \mathbf{K}^\text{T})(\mathbf{K}^{-\text{T}} \mathbf{R}^\text{T} \mathbf{K}^\text{T}) = \mathbf{K} \mathbf{K}^\text{T} = \boldsymbol{\omega}^* \quad (1089)$$

This gives

$$\boldsymbol{\omega}^* = \mathbf{M} \boldsymbol{\omega}^* \mathbf{M}^\text{T} \quad (1090)$$

which can be used to solve for $\boldsymbol{\omega}^*$. Then \mathbf{K} can be found by Cholesky decomposition.

18.18 Calibration from three sets of orthogonal parallel lines

An image of parallel lines may include the vanishing point, which is the image of the point at infinity where the parallel lines intersect. If there are three sets of parallel lines, where the lines of the three sets are orthogonal, then there will be three vanishing points. These vanishing points can be used for calibration. Such parallel lines can for example be found as the outline of a building, or it can be the edges of a cube specially designed for calibration.

Let the camera model be $\mathbf{P} = \mathbf{K}[\mathbf{R}, \mathbf{t}]$. Suppose that three sets L_1 , L_2 and L_3 of parallel lines are found, and that the unit direction vectors of the lines of the set L_i is \mathbf{a}_i , so that \mathbf{a}_1 , \mathbf{a}_2 and \mathbf{a}_3 are the orthogonal unit vectors of the world coordinate frame w . This means that the parallel lines are aligned with the axes of the world coordinate frame.

The lines of set L_1 , which all have direction vector \mathbf{a}_1 will intersect at the point $\mathbf{X}_1 = [\mathbf{a}_1^\text{T}, 0]^\text{T}$, which is the point at infinity in the direction of \mathbf{a}_1 . In the same way, the linear of the set L_2

will intersect at the point $\mathbf{X}_2 = [\mathbf{a}_2^T, 0]^T$, and the lines of the set L_3 will intersect at the point $\mathbf{X}_2 = [\mathbf{a}_3^T, 0]^T$.

The points \mathbf{X}_1 , \mathbf{X}_2 and \mathbf{X}_3 at infinity will be imaged as the vanishing points \mathbf{z}_1 , \mathbf{z}_2 and \mathbf{z}_3 where the vanishing points are given by

$$\mathbf{z}_i = \mathbf{P}\mathbf{X}_i = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{X}_i \quad (1091)$$

which means that the vanishing point satisfies

$$\mathbf{z}_i = \mathbf{K}\mathbf{R}\mathbf{a}_i, \quad \mathbf{a}_i = \mathbf{R}^T \mathbf{K}^{-1} \mathbf{z}_i \quad (1092)$$

Then

$$\mathbf{a}_i^T \mathbf{a}_j = \mathbf{z}_i^T \mathbf{K}^{-T} \mathbf{R} \mathbf{R}^T \mathbf{K}^{-1} \mathbf{z}_j = \mathbf{z}_i^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{z}_j = \mathbf{z}_i^T \boldsymbol{\omega} \mathbf{z}_j \quad (1093)$$

The orthogonality of the three vectors \mathbf{a}_1 , \mathbf{a}_2 and \mathbf{a}_3 therefore gives the conditions

$$\mathbf{z}_1^T \boldsymbol{\omega} \mathbf{z}_2 = 0, \quad \mathbf{z}_2^T \boldsymbol{\omega} \mathbf{z}_3 = 0, \quad \mathbf{z}_3^T \boldsymbol{\omega} \mathbf{z}_1 = 0 \quad (1094)$$

where $\boldsymbol{\omega} = (\mathbf{K}\mathbf{K}^T)^{-1}$. This gives 3 conditions on $\boldsymbol{\omega}$, which is a symmetric matrix, and has 5 unknowns.

19 Conics*

19.1 Introduction

Conics are second-order curves in the Euclidean plane. Conics include circles, ellipses, parabolas and hyperbolas. Points and lines in the plane are given by homogeneous vectors, while a conic is given by a homogeneous matrix. This section presents conics and how conics are used in connection with homographies. The use of conics in vision is treated in great detail in [30], while basic theory on conics in projective geometry is found in, e.g., [61].

19.2 Description of conics

A conic, which is also called a conic section, is the curve that results from the intersection of a plane with a cone in a 3-dimensional Euclidean space. This curve as seen in the Euclidean plane defined by the intersection will be a circle, ellipse, hyperbola or parabola depending on the geometry of the intersection. This is a well-known concept from classic geometry.

Let $\mathbf{x} = [x_1, x_2, x_3]^T$ be a homogeneous point representing the Euclidean point $\mathbf{p} = [x, y]^T = [x_1/x_3, x_2/x_3]^T$ in the xy plane, where x_3 is the homogeneous coordinate. The point \mathbf{x} will be on the conic defined by the homogenous matrix \mathbf{C} if

$$\mathbf{x}^T \mathbf{C} \mathbf{x} = 0 \quad (1095)$$

It is obvious that this condition will not be changed when the scaling of \mathbf{x} or \mathbf{C} is changed. The homogeneous matrix \mathbf{C} is given in the standard form [30]

$$\mathbf{C} = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix} \quad (1096)$$

The curve of the conic is described in homogeneous coordinates as

$$ax_1^2 + bx_1x_2 + cx_2^2 + dx_1x_3 + ex_2x_3 + fx_3^2 = 0 \quad (1097)$$

The corresponding curve in Euclidean geometry is found by dividing this expression by the homogeneous coordinate x_3 , which gives

$$ax^2 + bxy + cy^2 + dx + ey + f = 0 \quad (1098)$$

A conic can be classified as three different types of curves depending on the determinant of the upper left 2×2 submatrix of \mathbf{C} , which is $-(b^2 - 4ac)$. The classification is

- If $b^2 - 4ac < 0$, the conic will be an ellipse.
- If $b^2 - 4ac = 0$, the conic will be a hyperbola.
- If $b^2 - 4ac > 0$, the conic will be a parabola.

It is noted that if $a = c$ and $b = 0$, the conic will be a circle, which is a special case of an ellipse.

Note that the quadratic form is formulated in terms of homogeneous coordinates. Therefore also the coefficient matrix is homogeneous, which means that it can be scaled by a scalar and still represent the same conic. This means that the conic is described with 5 independent parameters as one of the 6 parameters can be selected depending on the scaling.

In computations it can be advantageous to write the expression for the conic in the linear form

$$\begin{bmatrix} x_1^2 & x_1x_2 & x_2^2 & x_1x_3 & x_2x_3 & x_3^2 \end{bmatrix} \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}^T = 0 \quad (1099)$$

where $\mathbf{c} = [a, b, c, d, e, f]^T$ is the vector of the coefficients.

Then, for a general conic \mathbf{C} , the coefficients of the conic can be determined from 5 points with the equation

$$\mathbf{A}\mathbf{c} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \mathbf{a}_3^T \\ \mathbf{a}_4^T \\ \mathbf{a}_5^T \end{bmatrix} \mathbf{c} = \mathbf{0} \quad (1100)$$

where

$$\mathbf{a}_i^T = \begin{bmatrix} x_{1i}^2 & x_{1i}x_{2i} & x_{2i}^2 & x_{1i}x_{3i} & x_{2i}x_{3i} & x_{3i}^2 \end{bmatrix} \quad (1101)$$

is given by the coordinates of $\mathbf{x}_i = [x_{1i}, x_{2i}, x_{3i}]^T$, which is point i . The equation (1100) can be solved for \mathbf{c} in the usual way with the singular value decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, and $\mathbf{c} = \lambda\mathbf{v}_6$, where \mathbf{v}_6 is the last column in \mathbf{V} , and λ is a nonzero scalar that can be selected.

Example 1

A circle is given by

$$a(x^2 + y^2) + dx + ey + f = 0 \quad (1102)$$

This can be written

$$a \left(x^2 + y^2 + \frac{d}{a}x + \frac{e}{a}y + \frac{f}{a} \right) = a \left(\left(x + \frac{d}{2a} \right)^2 + \left(y + \frac{e}{2a} \right)^2 - \frac{d^2 + e^2 - 4af}{4a^2} \right) = 0 \quad (1103)$$

In terms of the center (x_0, y_0) and the radius r this is written

$$a \left((x - x_0)^2 + (y - y_0)^2 - r^2 \right) = 0 \quad (1104)$$

which gives

$$x_0 = -\frac{d}{2a}, \quad y_0 = -\frac{e}{2a}, \quad r^2 = \frac{d^2 + e^2 - 4af}{4a^2} \quad (1105)$$

It is noted that if $a = 1$, then

$$(x - x_0)^2 + (y - y_0)^2 - r^2 = x^2 + y^2 - 2x_0x - 2y_0y + x_0^2 + y_0^2 - r^2 = 0 \quad (1106)$$

which can be written $\mathbf{x}^T \mathbf{C} \mathbf{x} = 0$, where

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ -x_0 & -y_0 & x_0^2 + y_0^2 - r^2 \end{bmatrix} \quad (1107)$$

This gives $A = C = 1$, $B = 0$, $D = -2x_0$, $E = -2y_0$ and $F = x_0^2 + y_0^2 - r^2$. Then $B^2 - 4AC = -4 < 0$. \square

Example 2

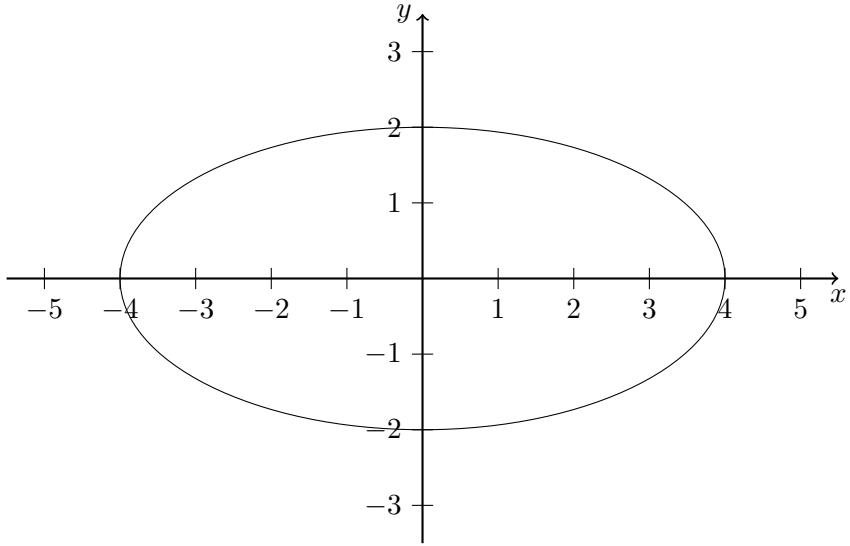


Figure 77: Ellipse with $a = 4$, $b = 2$ and zero offset.

An example of an ellipse is

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0 \quad (1108)$$

This can be written $\mathbf{x}^T \mathbf{C} \mathbf{x} = 0$, where

$$\mathbf{C} = \begin{bmatrix} \frac{1}{a^2} & 0 & 0 \\ 0 & \frac{1}{b^2} & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad (1109)$$

which gives $B^2 - 4AC = -4/(a^2b^2) < 0$. \square

Example 3

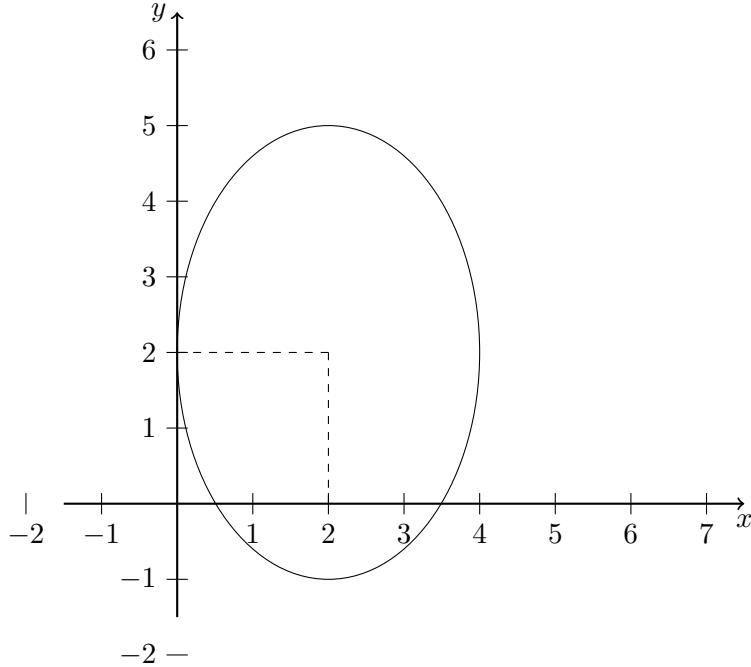


Figure 78: Ellipse with $a = 2$, $b = 3$ and offset $x_0 = 2$ and $y_0 = 2$.

An example of an ellipse with offset is

$$\frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} - 1 = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 2\frac{x_0x}{a^2} - 2\frac{y_0y}{b^2} + \frac{x_0^2}{a^2} + \frac{y_0^2}{b^2} - 1 = 0 \quad (1110)$$

This can be written $\mathbf{x}^T \mathbf{C} \mathbf{x} = 0$ where

$$\mathbf{C} = \begin{bmatrix} \frac{1}{a^2} & 0 & -\frac{x_0}{a^2} \\ 0 & \frac{1}{b^2} & -\frac{y_0}{b^2} \\ -\frac{x_0}{a^2} & -\frac{y_0}{b^2} & \frac{x_0^2}{a^2} + \frac{y_0^2}{b^2} - 1 \end{bmatrix} \quad (1111)$$

\square

Example 4

A parabola is given by

$$y^2 - 4ax = 0 \quad (1112)$$

which gives

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & -2a \\ 0 & 1 & 0 \\ -2a & 0 & 0 \end{bmatrix} \quad (1113)$$

where $A = B = 0$, $C = 1$, $D = 0$, $E = -4a$ and $F = 0$, which gives $B^2 - 4AC = 0$. \square

Example 5

An example of a hyperbola is

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} - 1 = 0 \quad (1114)$$

which gives

$$\mathbf{C} = \begin{bmatrix} \frac{1}{a^2} & 0 & 0 \\ 0 & -\frac{1}{b^2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1115)$$

where $A = 1/a^2$, $B = 0$, $C = -1/b^2$, $D = E = 0$ and $F = -1$, which gives $B^2 - 4AC = 4/(a^2b^2) > 0$. \square

19.3 Determination of an ellipse from 5 points

Example 1:

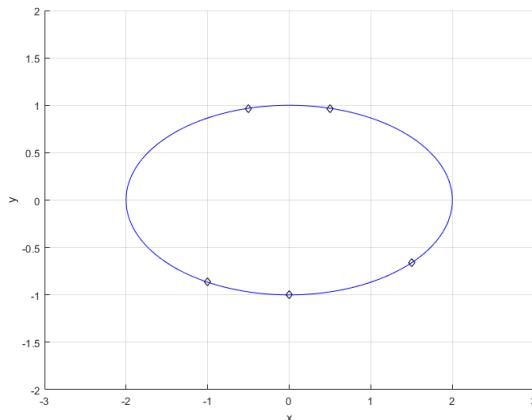


Figure 79: Ellipse found from 5 points

Calculation of conic from 5 points.

```
% Input: 5 points on an ellipse.
% Ellipse: x^2/a^2 + y^2/b^2 - 1 = 0
% |y| = (b/a)*sqrt(a*a - x*x)
a = 2; b = 1; x0 = 1; y0 = 1;
x = [-1 -0.5 0 0.5 1.5]'
```

```

y = (b/a)*sqrt(a*a - x.*x);
for i = 1:size(x)
    y(i) = (-1)^i*y(i)
end
z = ones(size(x));

% Calculation of 5 conditions on c
A = zeros(5,6);
A = [x.^2, x.*y, y.^2, x.*z, y.*z, z.^2]
[U,S,V] = svd(A);
% Result: Coefficient vector c = (a,b,c,d,e,f) of a circle
c = V(:,6)/norm(V(:,6)); % Scaled to unit vector
C = [c(1) c(2)/2 c(4)/2;c(2)/2 c(3) c(5)/2;c(4)/2 c(5)/2 c(6)];
C = C/(-C(3,3))

% Plot points
figure(1);clf;
axis([-3 3 -3 3]); grid; hold on;
plot(x,y,'dk')
% Plot ellipse
a = sqrt(1/C(1,1)); b = sqrt(1/C(2,2));
t = 0:0.01:2*pi;
Xp = a*cos(t); Yp = b*sin(t);
plot(Xp,Yp,'b')

% Test: Check if the points are on the conic
d = zeros(size(x,1),1); X = [x,y,z];
for i = 1:size(x)
    d(i) = X(i,:)*C*X(i,:)';
end
d % Zeros expected

```

□

19.4 Determination of a circle from 3 points

If there are additional conditions on the coefficients of the conic, then fewer than 5 points may be sufficient. This is the case for a circle, where there are two conditions $a = c$ and $b = 0$ on the coefficient. This means that if \mathbf{C} represents a circle, then it is sufficient with three points on the circle to define \mathbf{C} . The conic is then

$$\mathbf{C} = \begin{bmatrix} a & 0 & d/2 \\ 0 & a & e/2 \\ d/2 & e/2 & f \end{bmatrix} \quad (1116)$$

and the expression for the curve is

$$a(x_1^2 + x_2^2) + dx_1x_3 + ex_2x_3 + fx_3^2 = 0 \quad (1117)$$

The linear equation for the coefficients associated with the point \mathbf{x} is then

$$\begin{bmatrix} x_1^2 + x_2^2 & x_1 x_3 & x_2 x_3 & x_3^2 \end{bmatrix} \begin{bmatrix} a & d & e & f \end{bmatrix}^T = 0 \quad (1118)$$

where $\mathbf{c} = [a, d, e, f]^T$ is the vector if the coefficients of the conic.

Example

A circle is defined by the three points

$$\mathbf{x}_1 = [1 \ 0 \ 1]^T, \quad \mathbf{x}_2 = [0 \ 1 \ 1]^T, \quad \mathbf{x}_3 = [1 \ 2 \ 1]^T \quad (1119)$$

Then the circle can be found from the three conditions of the linear equation

$$\mathbf{A}\mathbf{c} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 5 & 1 & 2 & 1 \end{bmatrix} \mathbf{c} = 0 \quad (1120)$$

The singular value decomposition of \mathbf{A} is $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}$ where

$$\mathbf{U} = \begin{bmatrix} -0.2232 & -0.9412 & 0.2535 \\ -0.2511 & -0.1958 & -0.9480 \\ -0.9419 & 0.2752 & 0.1926 \end{bmatrix} \quad (1121)$$

$$\Sigma = \begin{bmatrix} 5.8984 & 0 & 0 & 0 \\ 0 & 1.1701 & 0 & 0 \\ 0 & 0 & 0.9163 & 0 \end{bmatrix} \quad (1122)$$

$$\mathbf{V} = \begin{bmatrix} -0.8788 & 0.2043 & 0.2931 & 0.3162 \\ -0.1975 & -0.5692 & 0.4869 & -0.6325 \\ -0.3619 & 0.3031 & -0.6141 & -0.6325 \\ -0.2401 & -0.7365 & -0.5476 & 0.3162 \end{bmatrix} \quad (1123)$$

As usual, the solution is found by the last column of the \mathbf{V} matrix, where the scaling can be selected. By selecting $a = 1$, the solution is found to be $\mathbf{c} = (1, -2, -2, 1)^T$. The resulting conic curve is

$$x_1^2 + x_2^2 - 2x_1 x_3 - 2x_2 x_3 + x_3^2 = 0 \quad (1124)$$

Insertion of $x = x_1/x_2$ and $y = x_2/x_3$ gives

$$(x - 1)^2 + (y - 1)^2 - 1 = 0 \quad (1125)$$

which is a circle with the center at $(1, 1)$ and radius $r = 1$.

```
% Input: 3 points on the circle
X = [[1 0 1]; [0 1 1]; [1 2 1]]'
% Calculation of 3 conditions on c
A = zeros(3,4)
for i = 1:3
    ai = [X(1,i)^2+X(2,i)^2 X(1,i)*X(3,i) X(2,i)*X(3,i) X(3,i)^2];
    A(i,:) = A(i,:) + ai
end
[U,S,V] = svd(A)
```

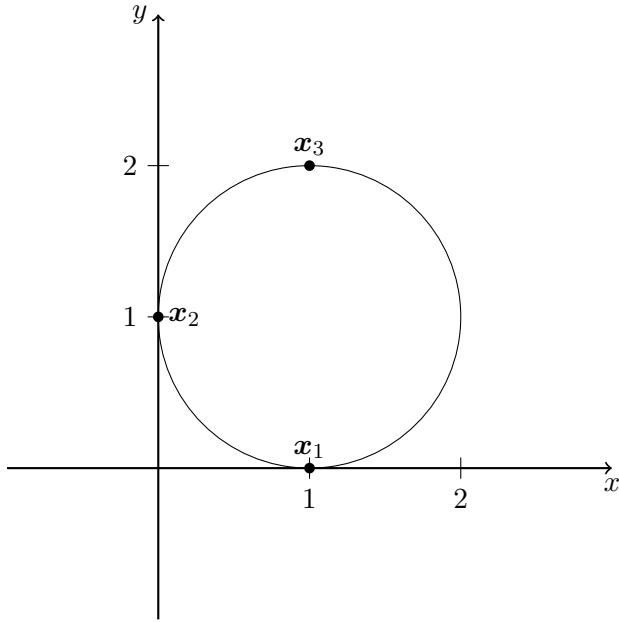


Figure 80: Circle determined from 3 points.

```
% Result: Coefficient vector c = (a,d,e,f) of a circle
c = V(:,4)/V(1,4) % Scaled so that a=1

% Test: Check if the points are on the conic
C = [c(1) 0 c(2)/2;0 c(1) c(3)/2;c(2)/2 c(3)/2 c(4)]
X(:,1)'*C*X(:,1)
X(:,2)'*C*X(:,2)
X(:,3)'*C*X(:,3)
```

□

19.5 Determination of a conic from determinants

The conic $\mathbf{x}^T \mathbf{C} \mathbf{x} = \mathbf{0}$ can also be found from determinants. The coefficients of the conic matrix \mathbf{C} is determined by 5 conditions $\mathbf{a}_i^T \mathbf{c} = 0$, $i = 1, 2, \dots, 5$ where the vectors $\mathbf{a}_i = [x_i^2, x_i y_i, y_i^2, x_i, y_i, 1]^T$ are calculated from 5 points $\mathbf{x}_i = [x_i, y_i, 1]^T$ on the conic, where the coordinates have been scaled so that $x_{3i} = 1$ to simplify the expressions. The conic is then determined from

$$\mathbf{A} \mathbf{c} = \begin{bmatrix} \mathbf{a}_1^T \\ \vdots \\ \mathbf{a}_5^T \end{bmatrix} \mathbf{c} = \mathbf{0} \quad (1126)$$

where the matrix 5×6 matrix \mathbf{A} has rank 5. If the point $\mathbf{x} = [x, y, 1]^T$ is also on the conic, then one more condition $\mathbf{a}^T \mathbf{c} = 0$ on the conic is obtained, where

$$\mathbf{a} = [x^2, x y, y^2, x, y, 1]^T \quad (1127)$$

The resulting 6 conditions can be written

$$\mathbf{A}_d \mathbf{c} = \begin{bmatrix} \mathbf{a}^T \\ \mathbf{A} \end{bmatrix} \mathbf{c} = \mathbf{0} \quad (1128)$$

where

$$\mathbf{A}_d = \begin{bmatrix} x^2 & xy & y^2 & x & y & 1 \\ x_1^2 & x_1 y_1 & y_1^2 & x_1 & y_1 & 1 \\ x_2^2 & x_2 y_2 & y_2^2 & x_2 & y_2 & 1 \\ x_3^2 & x_3 y_3 & y_3^2 & x_3 & y_3 & 1 \\ x_4^2 & x_4 y_4 & y_4^2 & x_4 & y_4 & 1 \\ x_5^2 & x_5 y_5 & y_5^2 & x_5 & y_5 & 1 \end{bmatrix} \quad (1129)$$

The rank of \mathbf{A}_d must be 5, because there are nontrivial solutions to \mathbf{c} . This means that the determinant of the 6×6 matrix \mathbf{A}_d is zero, that is

$$\det(\mathbf{A}_d) = 0 \quad (1130)$$

This determinant is found by developing the determinant from the first row. This gives

$$\det(\mathbf{A}_d) = x^2 \det(\mathbf{A}_{\check{1}}) - xy \det(\mathbf{A}_{\check{2}}) + y^2 \det(\mathbf{A}_{\check{3}}) - x \det(\mathbf{A}_{\check{4}}) + y \det(\mathbf{A}_{\check{5}}) - \det(\mathbf{A}_{\check{6}}) \quad (1131)$$

where $\mathbf{A}_{\check{i}}$ is the 5×5 matrix which is obtained from \mathbf{A} with column i removed. From (1127) it is seen that this can be written

$$\det(\mathbf{A}_d) = \mathbf{a}^T \mathbf{c} \quad (1132)$$

where

$$\mathbf{c} = [\det(\mathbf{A}_{\check{1}}) \ -\det(\mathbf{A}_{\check{2}}) \ \det(\mathbf{A}_{\check{3}}) \ -\det(\mathbf{A}_{\check{4}}) \ \det(\mathbf{A}_{\check{5}}) \ -\det(\mathbf{A}_{\check{6}})]^T \quad (1133)$$

This shows that the vector \mathbf{c} and therefore the matrix \mathbf{C} can be found from the determinants $\det(\mathbf{A}_{\check{i}})$ of submatrices of \mathbf{A}_d . Moreover, the quadratic form $\mathbf{x}^T \mathbf{C} \mathbf{x} = \mathbf{a}^T \mathbf{c}$ of the conic is given by

$$\mathbf{x}^T \mathbf{C} \mathbf{x} = \det(\mathbf{A}_d) \quad (1134)$$

for any point \mathbf{x} , including points x that are not on the conic.

Example

```
% Input: 5 points on a conic
X = [[1 0 1]; [0 1 1]; [-1 0 1]; [0 -1 1]; [0.5 sqrt(3)/2 1]]';
% Calculation of 5 conditions on c
A = zeros(5,6); c = zeros(6,1);
for i = 1:5
    ai = [X(1,i)^2 X(1,i)*X(2,i) X(2,i)^2 X(1,i)*X(3,i) X(2,i)*X(3,i) X(3,i)^2];
    A(i,:) = A(i,:) + ai;
end
c = [det(A(:,2:6)); -det([A(:,1) A(:,3:6)]); det([A(:,1:2) A(:,4:6)]); ...
       -det([A(:,1:3) A(:,5:6)]); det([A(:,1:4) A(:,6)]); -det(A(:,1:5))];
c = c/c(1); % Scale so that a = 1

% Test: Check if the points are on the conic
```

```

C = [c(1) c(2)/2 c(4)/2;c(2)/2 c(3) c(5)/2;c(4)/2 c(5)/2 c(6)]
d = zeros(5,1);
for i = 1:5
    d(i) = X(:,i)'*C*X(:,i);
end
d % Zeros expected

```

□

19.6 Determination of a circle from determinants

In the special case of a circle the coefficients of the conic is found from three points. Then the condition from one more point \mathbf{x} on the circle will be linearly dependent on the original 3 conditions in $\mathbf{A}\mathbf{c} = 0$. Then

$$\mathbf{A}_d \mathbf{c} = \begin{bmatrix} \mathbf{a}^T \\ \mathbf{A} \end{bmatrix} \mathbf{c} = \mathbf{0} \quad (1135)$$

where

$$\mathbf{A}_d = \begin{bmatrix} x^2 + y^2 & x & y & 1 \\ x_1^2 + y_1^2 & x_1 & y_1 & 1 \\ x_2^2 + y_2^2 & x_2 & y_2 & 1 \\ x_3^2 + y_3^2 & x_3 & y_3 & 1 \end{bmatrix} \quad (1136)$$

The matrix \mathbf{A}_d will be of rank 3. This gives

$$0 = \det(\mathbf{A}_d) = (x^2 + y^2) \det(\mathbf{A}_{\check{1}}) - x \det(\mathbf{A}_{\check{2}}) + y \det(\mathbf{A}_{\check{3}}) - \det(\mathbf{A}_{\check{4}}) \quad (1137)$$

where $\mathbf{A}_{\check{i}}$ is the 3×3 matrix which is obtained from \mathbf{A} with column i removed. This means that the conic coefficients are given by

$$\mathbf{c} = [\det(\mathbf{A}_{\check{1}}) \ -\det(\mathbf{A}_{\check{2}}) \ \det(\mathbf{A}_{\check{3}}) \ -\det(\mathbf{A}_{\check{4}})]^T \quad (1138)$$

The quadratic form is given by

$$\mathbf{x}^T \mathbf{C} \mathbf{x} = \det(\mathbf{A}_d) \quad (1139)$$

for all points \mathbf{x} , where $\mathbf{x}^T \mathbf{C} \mathbf{x} = 0$ for points \mathbf{x} on the circle. .

Example

```

% Input: 3 points on the circle
X = [[1 0 1]; [0 1 1]; [1 2 1]]'
% Calculation of 3 conditis on c
A = zeros(3,4); c = zeros(4,1);
for i = 1:3
    ai = [X(1,i)^2+X(2,i)^2 X(1,i)*X(3,i) X(2,i)*X(3,i) X(3,i)^2];
    A(i,:) = A(i,:) + ai
end
% Calculation of c from determinant
c = [det(A(:,2:4)); -det([A(:,1) A(:,3:4)]); ....
      det([A(:,1:2) A(:,4)]); -det(A(:,1:3))];

```

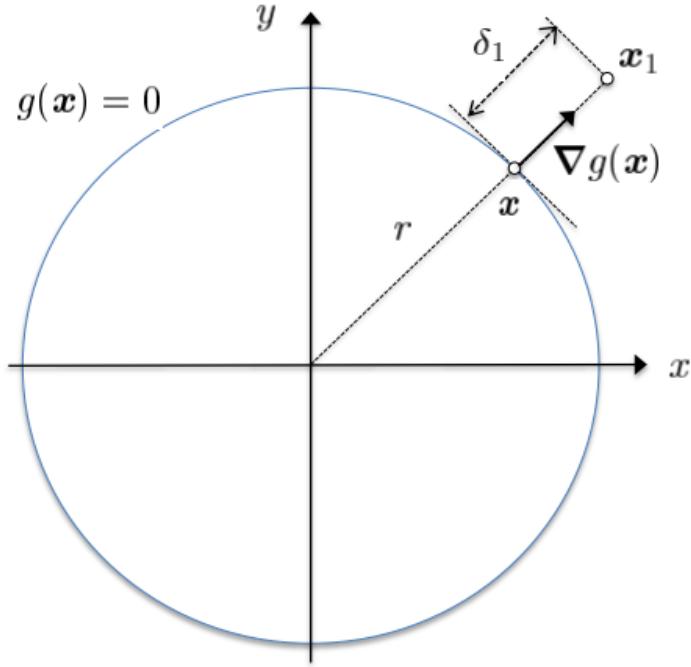


Figure 81: The distance δ_1 from a circle $g(\mathbf{x}) = 0$ to a point \mathbf{x}_1 .

```

c = c/c(1); % Scale so that a = 1

% Test: Check if the points are on the conic
C = [c(1) 0 c(2)/2; 0 c(1) c(3)/2; c(2)/2 c(3)/2 c(4)]
X(:,1)'*C*X(:,1)
X(:,2)'*C*X(:,2)
X(:,3)'*C*X(:,3)

```

19.7 Least-squares determination of a conic from point observations

The determination of conics from points in the previous sections has been based on a minimum number of point, which gives an exact solution of $\mathbf{Ac} = \mathbf{0}$. This solution is found under the assumption that the points are exactly on the conic. In the following the least-squares solution will be discussed for the problem where there are more points that are to be fitted to a conic. An overview is found in [2, 15].

Consider the conic

$$\mathbf{x}^T \mathbf{C} \mathbf{x} = 0 \quad (1140)$$

Define the polynomial

$$g(\mathbf{x}) = \mathbf{x}^T \mathbf{C} \mathbf{x} = ax^2 + bxy + cy^2 + dx + ey + f \quad (1141)$$

which is zero for points on the conic. The points of the conic will be on the curve specified by the conic, and the gradient $\nabla g(\mathbf{x})$ will be a vector which is normal to this curve. Suppose

that the \mathbf{x}_1 is given, and that \mathbf{x} is the closest point on the conic. Then the vector from \mathbf{x} to \mathbf{x}_1 will be along the gradient $\nabla g(\mathbf{x})$ of the curve, and \mathbf{x}_1 will be given by

$$\mathbf{x}_1 = \mathbf{x} + \delta_1 \frac{\nabla g(\mathbf{x})}{|\nabla g(\mathbf{x})|} \quad (1142)$$

where δ_1 is the distance from the conic to the point \mathbf{x}_1 , as shown in Figure 81. Moreover, the value of the polynomial $g(\mathbf{x}_1)$ at \mathbf{x}_1 can be approximated by the first order expression

$$g(\mathbf{x}_1) = g(\mathbf{x}) + |\nabla g(\mathbf{x})|\delta_1 = |\nabla g(\mathbf{x})|\delta_1 \quad (1143)$$

From this it is seen that the distance from \mathbf{x}_1 to the conic can be approximated by

$$\delta_1 = \frac{g(\mathbf{x}_1)}{|\nabla g(\mathbf{x}_1)|} \quad (1144)$$

This distance has been called the Sampson distance [60].

19.8 Least-squares determination of a circle from point observations

Consider the circle $g(\mathbf{x}) = 0$, where

$$g(\mathbf{x}) = a(x^2 + y^2) + dx + ey + f = 0 \quad (1145)$$

Then the gradient is

$$\nabla g = \begin{bmatrix} 2ax + d \\ 2ay + e \\ 0 \end{bmatrix} = 2a \begin{bmatrix} x - x_0 \\ y - y_0 \\ 0 \end{bmatrix} \quad (1146)$$

where $x_0 = -d/(2a)$ and $y_0 = -e/(2a)$ are the coordinates of the center of the circle. It is straightforward to verify that for a circle, the gradient $\nabla g(\mathbf{x})$ at the point \mathbf{x} on the circle has direction from the center x_0 to the point

$$\mathbf{x} = \mathbf{x}_0 + \frac{\nabla g(\mathbf{x})}{2a} \quad (1147)$$

on the circle. The norm of the gradient is

$$|\nabla g|^2 = (2ax + d)^2 + (2ay + e)^2 \quad (1148)$$

$$= 4a^2x^2 + 4adx + d^2 + 4a^2y^2 + 4aey + e^2 \quad (1149)$$

$$= 4a[a(x^2 + y^2) + dx + ey + f] + d^2 + e^2 - 4af \quad (1150)$$

If the point \mathbf{x}_i is assumed to be close to the conic, then the approximation $ax_i^2 + y_i^2 + dx_i + ey_i + f \approx 0$ can be used and the gradient can be approximated by

$$|\nabla g(\mathbf{x})|^2 = d^2 + e^2 - 4af \quad (1151)$$

Then the Sampson distance from the conic to the point x_i can be approximated by

$$\delta_i^2 = \frac{g(\mathbf{x}_i)^2}{|\nabla g(\mathbf{x})|^2} = \frac{(ax_i^2 + bx_iy_i + cy_i^2 + dx_i + ey_i + f)^2}{d^2 + e^2 - 4af} \quad (1152)$$

The expression for the Sampson distance is used in Pratt fit [55] of a circle to a set of points. It is noted that $d^2 + e^2 - 4af = 4a^2r^2$.

If the gradient is evaluated at the point \mathbf{x}_i instead of at the conic, then

$$|\nabla g(\mathbf{x}_i)|^2 = 4a^2(x_i^2 + y_i^2) + 4adx_i + 4aey_i + d^2 + e^2 \quad (1153)$$

The Sampson distance to the point x_i can then be approximated by

$$\delta_i^2 = \frac{g(\mathbf{x}_i)^2}{|\nabla g(\mathbf{x}_i)|^2} = \frac{(a(x_i^2 + y_i^2) + dx_i + ey_i + f)^2}{4a^2(\bar{x}^2 + \bar{y}^2) + 4a\bar{x} + 4a\bar{y} + d^2 + e^2} \quad (1154)$$

where the average values

$$\overline{(x^2 + y^2)} = \frac{1}{n} \sum_{i=1}^n (x_i^2 + y_i^2), \quad \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i, \quad \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i \quad (1155)$$

are used in the denominator. This is used in the Taubin fit [71] of a circle to a set of points.

19.9 The Kåsa fit of a circle to a set of points

A simple method for fitting a set of points to a circle is called the Kåsa fit, which was presented in [35], before the concept of the Sampson error had been introduced. Suppose that a circle is given by the parameters a, d, e, f as

$$g(\mathbf{x}) = a(x^2 + y^2) + dx + ey + f = 0 \quad (1156)$$

It is noted that for $a \neq 0$ this is a circle, and for $a = 0$ this will be a line. In the following it is assumed that $a \neq 0$. The scaling of this equation can be selected freely, and the circle may therefore be written in the form

$$h(\mathbf{x}) = x^2 + y^2 - \theta_1 x - \theta_2 y - \theta_3 = 0 \quad (1157)$$

where $h(\mathbf{x}) = g(\mathbf{x})/a$ and there are three parameters given by $\theta_1 = -d/a$, $\theta_2 = -e/a$ and $\theta_3 = -f/a$.

The equation for the circle can also be written in terms of the parameters x_0, y_0, r as

$$(x - x_0)^2 + (y - y_0)^2 - r^2 = 0 \quad (1158)$$

where x_0 and y_0 are the coordinates of the center of the circle, and r is the radius. The relation between x_0, y_0, r and $\theta_1, \theta_2, \theta_3$ is given by

$$x_0 = -\frac{d}{2a} = \frac{\theta_1}{2}, \quad y_0 = -\frac{e}{2a} = \frac{\theta_2}{2}, \quad r^2 = \frac{d^2 + e^2 - 4af}{4a^2} = \left(\frac{\theta_1}{2}\right)^2 + \left(\frac{\theta_2}{2}\right)^2 + \theta_3 \quad (1159)$$

which is verified by completing the squares.

Consider a point (x_i, y_i) which satisfies

$$(x_i - x_0)^2 + (y_i - y_0)^2 - r_i^2 = 0 \quad (1160)$$

which means that the point is in a circle with center in (x_0, y_0) and the radius r_i . Then

$$h(\mathbf{x}_i) = (x_i - x_0)^2 + (y_i - y_0)^2 - r^2 \quad (1161)$$

It follows from (1158) and (1160) that

$$h(\mathbf{x}_i) = r^2 - r_i^2 \quad (1162)$$

The Kåsa fit [35] is based on the minimization of

$$h(\mathbf{x}_i)^2 = (r^2 - r_i^2)^2 = [(r - r_i)(r + r_i)]^2 = [\delta_i(2r + \delta_i)]^2 \approx (2r\delta_i)^2 \quad (1163)$$

The minimization of $h(\mathbf{x}_i)$ is therefore a minimization of the distance $\delta_i = r_1 - r$ from the point to the circle scaled by two times the radius. This is a weakness with the solution, as it tends to minimize both the radius and the distance from the points to the circle. If the points are only from a small segment of the circle, then the Kåsa fit may result in a too small radius.

The Kåsa fit can be computed as the least-squares solution of the homogeneous equation $g(\mathbf{x}_i) = 0$ for $i = 1, \dots, n$, which gives

$$\underbrace{\begin{bmatrix} x_1^2 + y_1^2 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n^2 + y_n^2 & x_n & y_n & 1 \end{bmatrix}}_A \underbrace{\begin{bmatrix} a \\ d \\ e \\ f \end{bmatrix}}_c = \mathbf{0} \quad (1164)$$

This is the same as the least-squares solution of the conic $\mathbf{x}_i^T \mathbf{Q} \mathbf{x}_i = 0$ for $i = 1, \dots, n$, and the solution is $\mathbf{c} = \gamma \mathbf{v}_4$ where \mathbf{v}_4 is the last column of the matrix \mathbf{V} of the singular value decomposition $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$.

Alternatively, and in line with the original paper [35] of Kåsa, the parameters can be computed as the least squares solution of the inhomogeneous formulation $\theta_1 x_i + \theta_2 y_i + \theta_3 = x_i^2 + y_i^2$ for $i = 1, \dots, n$, which gives

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix}}_B \underbrace{\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}}_\theta = \underbrace{\begin{bmatrix} x_1^2 + y_1^2 \\ \vdots \\ x_n^2 + y_n^2 \end{bmatrix}}_b \quad (1165)$$

which has a least squares solution found from the normal equations

$$(\mathbf{B}^T \mathbf{B}) \boldsymbol{\theta} = \mathbf{B}^T \mathbf{b} \quad (1166)$$

where $\mathbf{B}^T \mathbf{B}$ is a full-rank 3×3 matrix. Then the center and the radius are found from $x_0 = \theta_1/2$, $y_0 = \theta_2/2$ and $r = \sqrt{(\theta_1^2 + \theta_2^2)/4 + \theta_3}$.

The following Matlab function calculates the center of a circle and the radius based on the solution of (1166). The function was taken from the web-page of Professor Chernov, who is the first author of [15].

```

function Par = Kasa(XY)
%-----
% Simple algebraic circle fit (Kasa method)
% Input: XY(n,2) is the array of coordinates of n points
% x(i)=XY(i,1), y(i)=XY(i,2)
% Output: Par = [a b R] is the fitting circle: center (a,b) and radius R
%-----

n = size(XY,1); % number of data points
Z = XY(:,1).*XY(:,1) + XY(:,2).*XY(:,2);
XY1 = [XY ones(n,1)];
P = XY1 \ Z; % Solves normal equation (XY1'*XY1) * P = XY1'*Z
Par = [P(1)/2, P(2)/2, sqrt((P(1)*P(1)+P(2)*P(2))/4+P(3))];

end % Kasa

```

□

19.10 The original Kåsa fit*

The original Kåsa fit [35] is based on the minimization of f_i as defined in (1161) with respect to the parameter set x_0, y_0, r . The loss function to be minimized is then

$$L_K = \sum_{i=1}^n h(x_i)^2 = \sum_{i=1}^n \left[(x_i - x_0)^2 + (y_i - y_0)^2 - r^2 \right]^2 \quad (1167)$$

Then the minimum will satisfy

$$\frac{\partial L_K}{\partial r} = -4r \sum_{i=1}^n \left[(x_i - x_0)^2 + (y_i - y_0)^2 - r^2 \right] = 0 \quad (1168)$$

$$\frac{\partial L_K}{\partial x_0} = -4 \sum_{i=1}^n \left[(x_i - x_0)^2 + (y_i - y_0)^2 - r^2 \right] (x_i - x_0) = 0 \quad (1169)$$

$$\frac{\partial L_K}{\partial y_0} = -4 \sum_{i=1}^n \left[(x_i - x_0)^2 + (y_i - y_0)^2 - r^2 \right] (y_i - y_0) = 0 \quad (1170)$$

Following [35] this is simplified to

$$\sum_{i=1}^n \left[(x_i - x_0)^2 + (y_i - y_0)^2 - r^2 \right] = 0 \quad (1171)$$

$$\sum_{i=1}^n \left[(x_i - x_0)^2 x_i + (y_i - y_0)^2 x_i - r^2 x_i \right] = 0 \quad (1172)$$

$$\sum_{i=1}^n \left[(x_i - x_0)^2 y_i + (y_i - y_0)^2 y_i - r^2 y_i \right] = 0 \quad (1173)$$

which is further simplified by writing out the squares and introducing $s = r^2 - x_0^2 - y_0^2$, which gives

$$2x_0 \sum_{i=1}^n x_i + 2y_0 \sum_{i=1}^n y_i + sn = \sum_{i=1}^n (x_i^2 + y_i^2) \quad (1174)$$

$$2x_0 \sum_{i=1}^n x_i^2 + 2y_0 \sum_{i=1}^n x_i y_i + s \sum_{i=1}^n x_i = \sum_{i=1}^n (x_i^3 + x_i y_i^2) \quad (1175)$$

$$2x_0 \sum_{i=1}^n x_i y_i + 2y_0 \sum_{i=1}^n y_i^2 + s \sum_{i=1}^n y_i = \sum_{i=1}^n (x_i^2 y_i + y_i^3) \quad (1176)$$

which gives three linear equations for the unknowns x_0, y_0, s . The equations can be written in matrix form as

$$\mathbf{A}\mathbf{c} = \mathbf{b} \quad (1177)$$

where $\mathbf{c} = [x_0, y_0, s]^T$ and

$$\mathbf{A} = \begin{bmatrix} 2 \sum x_i & 2 \sum y_i & n \\ 2 \sum x_i^2 & 2 \sum x_i y_i & \sum x_i \\ 2 \sum x_i y_i & 2 \sum y_i^2 & \sum y_i \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \sum (x_i^2 + y_i^2) \\ \sum (x_i^3 + x_i y_i^2) \\ \sum (x_i^2 y_i + y_i^3) \end{bmatrix} \quad (1178)$$

This equation is solved for \mathbf{c} , and the radius is found from $r = \sqrt{s + x_0^2 + y_0^2}$.

19.11 The Rayleigh quotient

Consider the minimization of the function

$$f(\mathbf{x}) = \frac{\mathbf{x}^T \mathbf{P} \mathbf{x}}{\mathbf{x}^T \mathbf{Q} \mathbf{x}} \quad (1179)$$

where $\mathbf{x} \in \mathbb{R}^n$ and \mathbf{P} and \mathbf{Q} are positive definite symmetric $n \times n$ matrices, which means that $\mathbf{x}^T \mathbf{P} \mathbf{x} > 0$ and $\mathbf{x}^T \mathbf{Q} \mathbf{x} > 0$ for all $\mathbf{x} \neq 0$.

The condition for optimality is

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = 2 \frac{\mathbf{P} \mathbf{x} (\mathbf{x}^T \mathbf{Q} \mathbf{x}) - \mathbf{Q} \mathbf{x} (\mathbf{x}^T \mathbf{P} \mathbf{x})}{(\mathbf{x}^T \mathbf{Q} \mathbf{x})^2} = \mathbf{0} \quad (1180)$$

This gives

$$\mathbf{P} \mathbf{x} (\mathbf{x}^T \mathbf{Q} \mathbf{x}) = (\mathbf{x}^T \mathbf{P} \mathbf{x}) \mathbf{Q} \mathbf{x} \quad (1181)$$

and it follows that the minimum is achieved when

$$\mathbf{P} \mathbf{x} = f(\mathbf{x}) \mathbf{Q} \mathbf{x} \quad (1182)$$

The solution for this problem is found from the generalized eigenvalue problem

$$\mathbf{P} \mathbf{x} = \lambda \mathbf{Q} \mathbf{x} \quad (1183)$$

The generalized eigenvalues λ_i and the corresponding generalized eigenvectors \mathbf{v}_i satisfies $(\mathbf{P} - \lambda_i \mathbf{Q}) \mathbf{v}_i = 0$ for $i = 1, \dots, n$. The optimal solution is then $\mathbf{x}_* = k \mathbf{v}_{i*}$ where $k \neq 0$ is a scaling

factor that can be selected freely, and \mathbf{v}_{i^*} is the eigenvector corresponding to the smallest eigenvalue λ_{i^*} . Moreover, the optimal solution is $f(\mathbf{x}_*) = \lambda_{i^*}$.

Comment

A formulation that is often used is that the minimization of $f(\mathbf{x})$ is equivalent to the minimization of

$$g(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x} \quad (1184)$$

subject to the constraint

$$\mathbf{x}^T \mathbf{Q} \mathbf{x} = 1 \quad (1185)$$

It is typically claimed that this can be verified by noting that

$$f(\mathbf{x}) = \frac{\mathbf{x}^T \mathbf{P} \mathbf{x}}{\mathbf{x}^T \mathbf{Q} \mathbf{x}} = \frac{\mathbf{x}'^T \mathbf{P} \mathbf{x}'}{\mathbf{x}'^T \mathbf{Q} \mathbf{x}'} \quad (1186)$$

where $\mathbf{x}' = \gamma \mathbf{x}$ and $\gamma \neq 0$, which means that \mathbf{x} can be scaled so that $\mathbf{x}^T \mathbf{Q} \mathbf{x} = 1$. The new constrained problem can be solved using Lagrange multipliers and the objective function

$$L(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x} + \lambda(1 - \mathbf{x}^T \mathbf{Q} \mathbf{x}) \quad (1187)$$

The condition for optimality is then

$$\mathbf{P} \mathbf{x} = \lambda \mathbf{Q} \mathbf{x} \quad (1188)$$

which is the same condition as in (1183). \square

19.12 The generalized eigenvalue problem

The generalized eigenvalue problem is written

$$\mathbf{A} \mathbf{x} = \lambda \mathbf{B} \mathbf{x} \quad (1189)$$

where $\mathbf{A} = \mathbf{A}^T$ and $\mathbf{B} = \mathbf{B}^T$ are symmetric $n \times n$ matrices, and \mathbf{B} positive definite. Then the matrix \mathbf{B} has n positive eigenvalues $\eta_i > 0$, $i = 1, \dots, n$ and n orthonormal eigenvectors \mathbf{m}_i so that $\mathbf{B} \mathbf{m}_i = \eta_i \mathbf{m}_i$. The matrix $\mathbf{M} = [\mathbf{m}_1, \dots, \mathbf{m}_n]$ is orthogonal, and satisfies $\mathbf{M}^T \mathbf{M} = \mathbf{I}$. It follows that $\mathbf{B} \mathbf{M} = \mathbf{M} \mathbf{D}$, where $\mathbf{D} = \text{diag}(\eta_1, \dots, \eta_n)$, and \mathbf{B} can be transformed to a diagonal form by

$$\mathbf{M}^T \mathbf{B} \mathbf{M} = \mathbf{D} = \text{diag}(\eta_1, \dots, \eta_n) \quad (1190)$$

The generalized eigenvalue problem can then be transformed into the form

$$\mathbf{M}^T \mathbf{A} \mathbf{M} \mathbf{y} = \lambda \mathbf{D} \mathbf{y} \quad (1191)$$

where $\mathbf{x} = \mathbf{M} \mathbf{y}$. Define $\mathbf{N} = \mathbf{D}^{-1/2} \mathbf{M}$ where $\mathbf{D}^{-1/2} = \text{diag}(\eta_1^{-1/2}, \dots, \eta_n^{-1/2})$. Then $\mathbf{N}^T \mathbf{B} \mathbf{N} = \mathbf{I}$, and the generalized eigenvalue problem becomes

$$\mathbf{N}^T \mathbf{A} \mathbf{N} \mathbf{z} = \lambda \mathbf{z} \quad (1192)$$

where $\mathbf{x} = \mathbf{M} \mathbf{y} = \mathbf{N} \mathbf{z}$. It is seen that the eigenvalues of the generalized eigenvalue problem will be the same as the eigenvalues of $\mathbf{N}^T \mathbf{A} \mathbf{N}$.

From Sylvester's law of inertia it follows that the eigenvalues λ_i of $\mathbf{N}^T \mathbf{A} \mathbf{N}$ will be related to the eigenvalues γ_i of \mathbf{A} so that there will be the same number of positive eigenvalues, the same number of eigenvalues equal to zero, and the same number of negative eigenvalues.

If $\mathbf{A} = \mathbf{A}^T$ is positive definite, while $\mathbf{B} = \mathbf{B}^T$ is not necessarily positive definite, the problem can reformulated as

$$\mathbf{B}\mathbf{x} = \frac{1}{\lambda} \mathbf{A}\mathbf{x} \quad (1193)$$

Then it follows that the eigenvalues $1/\lambda_i$ will be related to the eigenvalues η_i of \mathbf{B} so that there will be the same number of positive eigenvalues $1/\lambda_i$ as there are positive eigenvalues η_i , and the same number of negative eigenvalues $1/\lambda_i$ as there are negative eigenvalues η_i . If $\eta_n = 0$, then there will be no corresponding eigenvalue as $1/\lambda_n$ is undefined. This means that in the generalized eigenvalue problem, the eigenvalue λ_i will be positive if η_i is positive, and negative when η_i is negative. If $\eta_i = 0$, then there is no corresponding eigenvalue λ_i .

19.13 Eigenvalues and the Rayleigh quotient

Consider the minimization of the function

$$f(\mathbf{x}) = \frac{\mathbf{x}^T \mathbf{P} \mathbf{x}}{\mathbf{x}^T \mathbf{Q} \mathbf{x}} \quad (1194)$$

where $\mathbf{x} \in \mathbb{R}^n$. Suppose that $\mathbf{P} = \mathbf{P}^T > 0$ is positive definite and $\mathbf{Q} = \mathbf{Q}^T \geq 0$ is positive semidefinite.

A necessary condition for the minimum of $f(\mathbf{x})$ is that the optimal solution \mathbf{x}_* is an eigenvector \mathbf{v}_{i*} of the generalized eigenvalue problem $\mathbf{P}\mathbf{x} = \lambda \mathbf{Q}\mathbf{x}$. The minimum value of the function is then $f(\mathbf{x}_*) = \lambda_{i*}$.

If \mathbf{Q} has one negative eigenvalue and $n - 1$ positive eigenvalues, then the generalized eigenvalue problem will one negative eigenvalue and $n - 1$ positive eigenvalues. The function $f(\mathbf{x})$ cannot be negative. This means that the minimum value of $f(\mathbf{x})$ is achieved with the smallest non-negative eigenvalue of the generalized eigenvalue problem.

If \mathbf{Q} has one eigenvalue equal to zero and $n - 1$ positive eigenvalues, then the generalized eigenvalue problem $\mathbf{Q}\mathbf{x} = (1/\lambda)\mathbf{P}\mathbf{x}$ will one eigenvalue equal to zero, which means that the corresponding λ_i is undefined, and $n - 1$ positive eigenvalues $1/\lambda_i$. Then the minimum value of $f(\mathbf{x})$ is achieved with the smallest eigenvalue λ_i .

19.14 The Pratt fit

The Pratt fit [55] of a circle is based on the minimization of $g/|\nabla g|$ using the approximation (1151) for the gradient. The objective function to be minimized is then

$$L_P(\mathbf{c}) = \sum_{i=1}^n \frac{g(\mathbf{x}_i)^2}{|\nabla g(\mathbf{x})^2|} = \sum_{i=1}^n \frac{[a(x_i^2 + y_i^2) + dx_i + ey_i + f]^2}{d^2 + e^2 - 4af} \quad (1195)$$

which is seen from (1152). The objective function can be written

$$L_P(\mathbf{c}) = \frac{\mathbf{c}^T \mathbf{Z}^T \mathbf{Z} \mathbf{c}}{\mathbf{c}^T \mathbf{N} \mathbf{c}} = \frac{\mathbf{c}^T \mathbf{M} \mathbf{c}}{\mathbf{c}^T \mathbf{N} \mathbf{c}} \quad (1196)$$

where $\mathbf{c} = [a, d, e, f]^T$, $\mathbf{M} = \mathbf{Z}^T \mathbf{Z}$ and

$$\mathbf{Z} = \begin{bmatrix} (x_1^2 + y_1^2) & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ (x_n^2 + y_n^2) & x_n & y_n & 1 \end{bmatrix} \quad (1197)$$

while

$$\mathbf{N} = \begin{bmatrix} 0 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & 0 & 0 & 0 \end{bmatrix} \quad (1198)$$

Here the matrix \mathbf{M} is positive definite, while \mathbf{N} is positive semidefinite with eigenvalues $\lambda_{N1} = 2$, $\lambda_{N2} = \lambda_{N3} = 1$ and $\lambda_{N4} = -2$.

This minimum is found from the generalized eigenvalue problem

$$\mathbf{M}\mathbf{c} = \lambda \mathbf{N}\mathbf{c} \quad (1199)$$

Due to the eigenvalues of \mathbf{N} , this generalized eigenvalue problem has four eigenvalues $\lambda_1, \dots, \lambda_4$ where $\lambda_1 \geq \lambda_2 \geq \lambda_3 > 0$ and $\lambda_4 < 0$.

The minimum of the objective function is $L_P(\mathbf{c}_*) = \lambda_3$, which is found for the optimal solution $\mathbf{c}_* = \mathbf{v}_3$, where \mathbf{v}_3 is the eigenvector corresponding to the smallest positive eigenvalue λ_3 .

19.15 The Taubin fit of a circle

The Taubin fit [71] of a circle is based on the minimization of $g/|\nabla g|$ using the approximation (1153) for the gradient. The objective function to be minimized is then

$$L_T = \frac{1}{n} \sum_{i=1}^n \frac{g(\mathbf{x}_i)^2}{|\nabla g(\mathbf{x}_i)|^2} = \frac{1}{n} \sum_{i=1}^n \frac{[a(x_i^2 + y_i^2) + dx_i + ey_i + f]^2}{4a^2(x^2 + y^2) + 4ad\bar{x} + 4ae\bar{y} + d^2 + e^2} \quad (1200)$$

where the average values

$$\overline{(x^2 + y^2)} = \frac{1}{n} \sum_{i=1}^n (x_i^2 + y_i^2), \quad \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i, \quad \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i \quad (1201)$$

are used in the denominator. The objective function can be written

$$L_P(\mathbf{c}) = \frac{\mathbf{c}^T \mathbf{Z}^T \mathbf{Z} \mathbf{c}}{\mathbf{c}^T \mathbf{N} \mathbf{c}} = \frac{\mathbf{c}^T \mathbf{M} \mathbf{c}}{\mathbf{c}^T \mathbf{N} \mathbf{c}} \quad (1202)$$

where where \mathbf{M} is as in Pratt's fit, and

$$\mathbf{N} = \begin{bmatrix} 4\overline{x^2 + y^2} & 2\bar{x} & 2\bar{y} & 0 \\ 2\bar{x} & 1 & 0 & 0 \\ 2\bar{y} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \frac{1}{n} \begin{bmatrix} 4m_{14} & 2m_{24} & 2m_{34} & 0 \\ 2m_{24} & n & 0 & 0 \\ 2m_{34} & 0 & n & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (1203)$$

where m_{ij} is element ij of \mathbf{M} .

This minimum is found from the generalized eigenvalue problem

$$\mathbf{M}\mathbf{c} = \lambda \mathbf{N}\mathbf{c} \quad (1204)$$

where \mathbf{M} is positive definite, and \mathbf{N} has 3 positive eigenvalues $\lambda_{N1} \geq \lambda_{N2} \geq \lambda_{N3}$, and one eigenvalue $\lambda_{N4} = 0$. The generalized eigenvalue problem will therefore have four eigenvalues $\lambda_1, \dots, \lambda_4$ where $\lambda_1 \geq \lambda_2 \geq \lambda_3 > 0$ while λ_4 is undefined. Then the optimal solution is $\mathbf{c}_* = \mathbf{v}_3$.

Example

```
% Find circle from 5 points with Pratt fit and Taubin fit
% Input: 5 points on the circle
XY = [[1 0 1]; [0 1 1]; [1 2 1]; [1.9 1 1]; [2.1 1 1]];
X = XY(:,1); Y = XY(:,2);
n = size(X,1);

Z = [X.^2+Y.^2, X, Y, ones(size(X))];
M = Z'*Z;
% Pratt fit
N = [0 0 0 -2; 0 1 0 0; 0 0 1 0; -2 0 0 0];
% Taubin fit
% N = [4*M(1,4) 2*M(2,4) 2*M(3,4) 0; 2*M(2,4) n 0 0; 2*M(3,4) 0 n 0; 0 0 0 0]/n;

% Generalized eigenvalue problem
[V,D] = eig(M,N);
[Dsort, ID] = sort(diag(D));
% Select smallest positive eigenvalue
C = V(:,ID(2)); % Pratt fit:
% C = V(:,ID(1)); % Taubin fit

C = C/C(1) % Scaled coefficients

% Center and radius
x0 = -C(2)/(2*C(1))
y0 = -C(3)/(2*C(1))
r = sqrt((C(2)^2 + C(3)^2 - 4*C(1)*C(4))/(4*C(1)^2))
```

19.16 The Taubin fit for an ellipse

Consider the conic $g(\mathbf{x}) = 0$, where

$$g(\mathbf{x}) = ax^2 + bxy + cy^2 + dx + ey + f = \mathbf{a}^T \mathbf{c} = 0 \quad (1205)$$

where $\mathbf{a} = [x^2, xy, y^2, x, y, 1]^T$ and $\mathbf{c} = [a, b, c, d, e, f]^T$. The gradient is

$$\nabla g = (\nabla \mathbf{a}(\mathbf{x})) \mathbf{c} = \begin{bmatrix} 2ax + by + d \\ 2cy + bx + e \\ 0 \end{bmatrix} \quad (1206)$$

where

$$\nabla \mathbf{a}(\mathbf{x}) = \begin{bmatrix} 2x & y & 0 & 1 & 0 & 0 \\ 0 & x & 2y & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (1207)$$

The square norm of the gradient is

$$|\nabla g|^2 = \mathbf{c}^T \mathbf{Q} \mathbf{c} \quad (1208)$$

where

$$\mathbf{Q} = [\nabla \mathbf{a}(\mathbf{x})]^T \nabla \mathbf{a}(\mathbf{x}) = \begin{bmatrix} 4x^2 & 2xy & 0 & 2x & 0 & 0 \\ 2xy & x^2 + y^2 & 2xy & y & x & 0 \\ 0 & 2xy & 4y^2 & 0 & 2y & 0 \\ 2x & y & 0 & 1 & 0 & 0 \\ 0 & x & 2y & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (1209)$$

The Taubin fit is then found by minimizing

$$L_{T_f} = \frac{1}{n} \sum_{i=1}^n \frac{g(\mathbf{x}_i)^2}{|\nabla g(\mathbf{x}_i)|^2} = \frac{1}{n} \sum_{i=1}^n \frac{\mathbf{c}^T \mathbf{a}_i \mathbf{a}_i^T \mathbf{c}}{\mathbf{c}^T \mathbf{Q} \mathbf{c}} \quad (1210)$$

where

$$\mathbf{P} = \frac{1}{n} \sum_{i=1}^n \mathbf{a}_i \mathbf{a}_i^T = \frac{1}{n} \sum_{i=1}^n \begin{bmatrix} x_i^4 & x_i^3 y_i & x_i^2 y_i^2 & x_i^3 & x_i^2 y_i & x_i^2 \\ x_i^3 y_i & x_i^2 y_i^2 & x_i y_i^3 & x_i^2 y_i & x_i y_i^2 & x_i y_i \\ x_i^2 y_i^2 & x_i y_i^3 & y_i^4 & x_i y_i^2 & y_i^3 & y_i^2 \\ x_i^3 & x_i^2 y_i & x_i y_i^2 & x_i^2 & x_i y_i & x_i \\ x_i^2 y_i & x_i y_i^2 & y_i^3 & x_i y_i & y_i^2 & y_i \\ x_i^2 & x_i y_i & y_i^2 & x_i & y_i & 1 \end{bmatrix} \quad (1211)$$

, and

$$\bar{\mathbf{Q}} = \frac{1}{n} \sum_{i=1}^n \begin{bmatrix} 4x_i^2 & 2x_i y_i & 0 & 2x_i & 0 & 0 \\ 2x_i y_i & x_i^2 + y_i^2 & 2x_i y_i & y_i & x_i & 0 \\ 0 & 2x_i y_i & 4y_i^2 & 0 & 2y_i & 0 \\ 2x_i & y_i & 0 & 1 & 0 & 0 \\ 0 & x_i & 2y_i & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (1212)$$

The matrix $\bar{\mathbf{Q}}$ can be expressed in terms of the elements p_{ij} of \mathbf{P} as

$$\bar{\mathbf{Q}} = \begin{bmatrix} 4p_{16} & 2p_{26} & 0 & 2p_{46} & 0 & 0 \\ 2p_{26} & p_{16} + p_{36} & 2p_{26} & p_{56} & p_{46} & 0 \\ 0 & 2p_{26} & 4p_{36} & 0 & 2p_{56} & 0 \\ 2p_{46} & p_{56} & 0 & 1 & 0 & 0 \\ 0 & p_{46} & 2p_{56} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (1213)$$

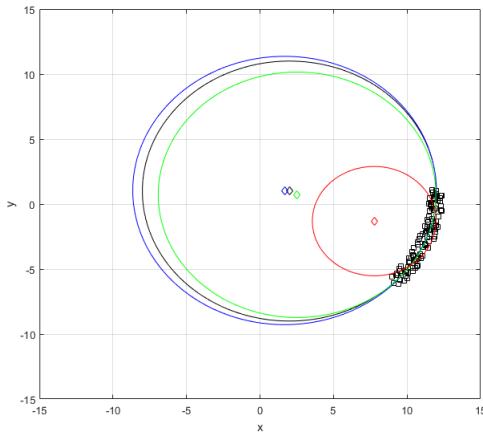


Figure 82: Circles found from 100 noisy points over a sector of $\pi/4$ of the circle shown in black. The Taubin fit (blue) and the Pratt fit (green) gives good solutions. The Kåsa fit (red) underestimates the radius significantly.

19.17 Example of fitting a circle to a set of points

Two calculations were tested where the Taubin fit, the Pratt fit and the Kåsa fit was used to calculate a circle from noisy point data. The results are shown in Figure 82 when the points were from a sector of $\pi/4$ of the circle. The Taubin and the Pratt fit worked well, while the Kåsa fit performed badly and give a very small radius. In the second test 6 points were distributed along a larger sector of the circle. Then all three methods worked well, and there was very little difference between the results, which are shown in Figure 83.

```
% Script for fitting a circle to a set of points

% Generate roughly spherical data
Radius_act=10;
Center_act=[2,1];
NoiseScale=1;

Angles=rand(100,1)*pi/4;
Rot_z=@(ang) [cos(ang), -sin(ang); sin(ang), cos(ang)];
X=zeros(size(Angles,1),2); %initialize
for i=1:size(X,1)
    X(i,:)=([Radius_act,0]+NoiseScale*(rand(1,2)-.5))*Rot_z(Angles(i,1))+Center_act;
end

ParT = CircleFitTaubin(X);
ParP = CircleFitPratt(X);
ParK = DirectFitKaasa(X);
%ParK = DirectFitConic(X);
```

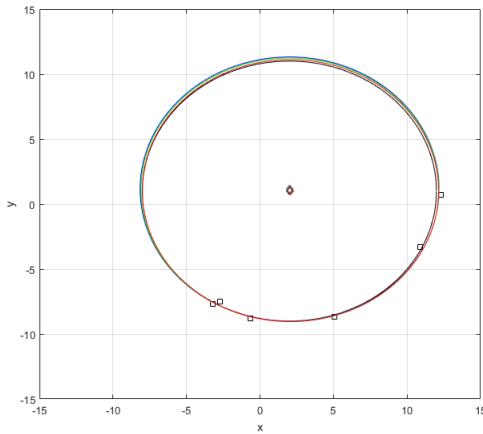


Figure 83: Circles found from 6 point over a sector of π of the circle. The Taubin fit (blue), the Pratt fit (green) and the Kåsa fit (red) gives good solutions that are close to equal. The true circle is shown in black.

```

figure(1);clf;
plot(X(:,1),X(:,2), 'ks')
axis([-15 15 -15 15]); hold on
grid
t = 0:0.01:2*pi; nt = size(t,2);
Circle_act = [Center_act(1)+Radius_act*cos(t); Center_act(2)+Radius_act*sin(t)];
CircleT = [ParT(1)+ParT(3)*cos(t); ParT(2)+ParT(3)*sin(t)];
CircleP = [ParP(1)+ParP(3)*cos(t); ParP(2)+ParP(3)*sin(t)];
CircleK = [ParK(1)+ParK(3)*cos(t); ParK(2)+ParK(3)*sin(t)];

plot(Circle_act(1,:),Circle_act(2,:),'k',Center_act(1),Center_act(2),'kd')
plot(CircleT(1,1:nt),CircleT(2,1:nt),'b',ParT(1),ParT(2),'bd')
plot(CircleP(1,1:nt),CircleP(2,1:nt),'g',ParP(1),ParP(2),'gd')
plot(CircleK(1,1:nt),CircleK(2,1:nt),'r',ParK(1),ParK(2),'rd')

%%%%%%%%%%%%%%%
function Par = CircleFitTaubin(XY)
%-----
% Circle fit by Taubin
% G. Taubin, 'Estimation of planar curves, surfaces and nonplanar
% space curves defined by implicit equations, with applications
% to edge and range image segmentation,' IEEE Transactions on
% Pattern Analysis and Machine Intelligence Vol. 13, pp. 1115{1138, 1991
%
% Input: XY(n,2) is the array of coordinates of n points
% x(i)=XY(i,1), y(i)=XY(i,2)
% Output: Par = [x0,y0,r]

```

```

%-----
n = size(XY,1);      % number of data points
X = XY(:,1); Y = XY(:,2);
% the centroid of the data set
xcentroid = mean(X)
ycentroid = mean(Y)
% Move points to the origin
X = X - xcentroid;
Y = Y - ycentroid;

Z = [X.^2+Y.^2, X, Y, ones(size(X))];
M = Z'*Z/n;
N = [4*M(1,4) 2*M(2,4) 2*M(3,4) 0; 2*M(2,4) n 0 0; 2*M(3,4) 0 n 0;0 0 0 0];
% Generalized eigenvalue problem
[V,D] = eig(M,N);
[Dsort, ID] = sort(diag(D));
% Select second eigenvalue, which is the smallest positive eigenvalue
CF = V(:,ID(1));
A = CF(1); B = CF(2); C = CF(3); D = CF(4);
Par(1) = -B/(2*A)+xcentroid; Par(2) = -C/(2*A)+ycentroid;
Par(3) = sqrt((B^2 +C^2 -4*A*D)/(4*A^2)); r = Par(3)
end      %

%%%%%%%%%%%%%%%
function Par = CircleFitPratt(XY)

%-----
%     Circle fit by Pratt
%     V. Pratt, "Direct least-squares fitting of algebraic surfaces",
%     Computer Graphics, Vol. 21, pages 145-152 (1987)
%
%     Input: XY(n,2) is the array of coordinates of n points
%            x(i)=XY(i,1), y(i)=XY(i,2)
%     Output: Coeff = [A,B,C,D]
%
%     Note: this fit does not use built-in matrix functions (except "mean"),
%           so it can be easily programmed in any programming language
%-----

n = size(XY,1);      % number of data points
X = XY(:,1); Y = XY(:,2);
% the centroid of the data set
xcentroid = mean(X)
ycentroid = mean(Y)
% Move points to the origin

```

```

X = X - xcentroid;
Y = Y - ycentroid;

Z = [X.^2+Y.^2, X, Y, ones(size(X))];
M = Z'*Z/n;
N = [0 0 0 -2;0 1 0 0; 0 0 1 0;-2 0 0 0];
% Generalized eigenvalue problem
[V,D] = eig(M,N);
[Dsort, ID] = sort(diag(D));
% Select second eigenvalue, which is the smallest positive eigenvalue
CF = V(:,ID(2));
A = CF(1); B = CF(2); C = CF(3); D = CF(4);
Par(1) = -B/(2*A)+xcentroid; Par(2) = -C/(2*A)+ycentroid;
Par(3) = sqrt((B^2 +C^2 -4*A*D)/(4*A^2)); r = Par(3)

end      %     CircleFitPratt

%%%%%%%%%%%%%%%
function Par = DirectFitKaasa(XY)
%-----
%     Circle fit by Kåsa
%     I. Kasa. A curve fitting procedure and its error analysis.
%     IEEE Trans. Inst. Meas., 25(1):8{14, 1976.
%     Input: XY(n,2) is the array of coordinates of n points
%            x(i)=XY(i,1), y(i)=XY(i,2)
%     Output: Par = [x0,y0,r]
%-----

n = size(XY,1);      % number of data points
X = XY(:,1); Y = XY(:,2);
xcentroid = mean(X);
ycentroid = mean(Y); % the centroid of the data set
X = X - xcentroid;
Y = Y - ycentroid;

A = zeros(3); b = [0;0;0];
for i = 1:n
    A(1,1) = A(1,1) + 2*X(i); A(1,2) = A(1,2) + 2*Y(i);
    A(2,1) = A(2,1) + 2*X(i)^2; A(2,2) = A(2,2) + 2*X(i)*Y(i);
    A(3,2) = A(3,2) + 2*Y(i)^2;
    b(1) = b(1) + X(i)^2+Y(i)^2; b(2) = b(2) + X(i)^3+X(i)*Y(i)^2;
    b(3) = b(3) + X(i)^2*Y(i)+Y(i)^3;
end
A(1,3) = n; A(2,3) = 0.5*A(1,1); A(3,1) = A(2,2); A(3,3) = 0.5*A(1,2);

```

```

Par = A\b
Par(3) = sqrt(Par(1)^2+Par(2)^2+Par(3))
Par(1) = Par(1)+xcentroid; Par(2) = Par(2) + ycentroid;
end    %

%%%%%%%%%%%%%%%
function Par = DirectFitConic(XY)
%-----
% Circle fit using conic and SVD
% Input: XY(n,2) is the array of coordinates of n points
% x(i)=XY(i,1), y(i)=XY(i,2)
% Output: Par = [x0,y0,r]
%-----

n = size(XY,1);      % number of data points
X = XY(:,1); Y = XY(:,2);
% the centroid of the data set
xcentroid = mean(X)
ycentroid = mean(Y)
% Move points to the origin
X = X - xcentroid;
Y = Y - ycentroid;

Z = [X.^2+Y.^2, X, Y, ones(size(X))];
[U,S,V] = svd(Z);
% Select second eigenvalue, which is the smallest positive eigenvalue
CF = V(:,4);
A = CF(1); B = CF(2); C = CF(3); D = CF(4);
Par(1) = -B/(2*A)+xcentroid; Par(2) = -C/(2*A)+ycentroid;
Par(3) = sqrt((B^2 +C^2 -4*A*D)/(4*A^2)); r = Par(3);
end    %

```

19.18 Lines and conics

The line $\ell = \mathbf{C}\mathbf{x}$ will be a tangent to the conic $\mathbf{x}^T \mathbf{C} \mathbf{x} = 0$ in the point \mathbf{x} . This is seen from noting that \mathbf{x} is on ℓ , which is seen from $\mathbf{x}^T \ell = \mathbf{x}^T \mathbf{C} \mathbf{x} = 0$. To see that ℓ is a tangent to the conic, suppose that the line intersects the conic at another point \mathbf{y} on the conic. Then $\mathbf{y}^T \mathbf{C} \mathbf{y} = 0$ since \mathbf{y} is on the conic, and $\mathbf{y}^T \ell = \mathbf{y}^T \mathbf{C} \mathbf{x} = 0$ since \mathbf{y} is on ℓ . This implies that all the points $\mathbf{x} + \alpha \mathbf{y}$ will be on the conic for all α , which follows from

$$(\mathbf{x} + \alpha \mathbf{y})^T \mathbf{C} (\mathbf{x} + \alpha \mathbf{y}) = \mathbf{x}^T \mathbf{C} \mathbf{x} + 2\alpha \mathbf{y}^T \mathbf{C} \mathbf{x} + \alpha^2 \mathbf{y}^T \mathbf{C} \mathbf{y} = 0$$

This means that all point on the line ℓ will be on the conic unless there is only one point of the conic that is on ℓ .

The dual of the conic $\mathbf{x}^T \mathbf{C} \mathbf{x} = 0$ is defined as

$$\boldsymbol{\ell}^T \mathbf{C}^* \boldsymbol{\ell} = 0 \quad (1214)$$

where $\boldsymbol{\ell}$ is a line tangent to the conic. In the case that \mathbf{C} is nonsingular the point on the conic where the line is tangent is given by

$$\mathbf{x} = \mathbf{C}^{-1} \boldsymbol{\ell} \quad (1215)$$

This gives

$$\mathbf{x}^T \mathbf{C} \mathbf{x} = \boldsymbol{\ell}^T \mathbf{C}^{-1} \mathbf{C} \mathbf{C}^{-1} \boldsymbol{\ell} = \boldsymbol{\ell}^T \mathbf{C}^{-1} \boldsymbol{\ell} = 0 \quad (1216)$$

where it is used that \mathbf{C} is symmetric. It follows that $\mathbf{C}^* = \mathbf{C}^{-1}$ for nonsingular \mathbf{C} . A dual conic has 5 independent parameters, and can be determined by 5 lines that are tangent to the dual conic.

19.19 Homographies and conics

Consider the homography $\mathbf{x}' = \mathbf{H} \mathbf{x}$. Then the expression for a conic $\mathbf{x}^T \mathbf{C} \mathbf{x} = 0$ can be developed as

$$\mathbf{x}^T \mathbf{C} \mathbf{x} = \mathbf{x}'^T \mathbf{H}^{-T} \mathbf{C} \mathbf{H}^{-1} \mathbf{x}' = \mathbf{x}'^T \mathbf{C}' \mathbf{x}' = 0 \quad (1217)$$

which shows that the coefficient matrix of the conic transforms according to

$$\mathbf{C}' = \mathbf{H}^{-T} \mathbf{C} \mathbf{H}^{-1} \quad (1218)$$

For a dual conic $\boldsymbol{\ell}^T \mathbf{C}^* \boldsymbol{\ell} = 0$ the transformation is found using $\boldsymbol{\ell}' = \mathbf{H}^{-T} \boldsymbol{\ell}$, which gives

$$\boldsymbol{\ell}^T \mathbf{C}^* \boldsymbol{\ell} = \boldsymbol{\ell}'^T \mathbf{H} \mathbf{C}^* \mathbf{H}^T \boldsymbol{\ell}' = \boldsymbol{\ell}'^T \mathbf{C}'^* \boldsymbol{\ell}' \quad (1219)$$

which leads to

$$\mathbf{C}'^* = \mathbf{H} \mathbf{C}^* \mathbf{H}^T \quad (1220)$$

19.20 Degenerate conics

Consider two lines $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$. Then the matrix

$$\mathbf{C} = \boldsymbol{\ell}_1 \boldsymbol{\ell}_1^T + \boldsymbol{\ell}_2 \boldsymbol{\ell}_2^T \quad (1221)$$

which is singular of rank 2, defines a degenerate conic by

$$\mathbf{x}^T \mathbf{C} \mathbf{x} = 0 \quad (1222)$$

Then a point \mathbf{x} will be on the conic if it is a point on $\boldsymbol{\ell}_1$ or a point on $\boldsymbol{\ell}_2$, which follows from

$$\mathbf{x}^T \mathbf{C} \mathbf{x} = (\mathbf{x}^T \boldsymbol{\ell}_1)(\boldsymbol{\ell}_1^T \mathbf{x}) + (\mathbf{x}^T \boldsymbol{\ell}_2)(\boldsymbol{\ell}_2^T \mathbf{x})$$

which is zero if \mathbf{x} is on either $\boldsymbol{\ell}_1$ or $\boldsymbol{\ell}_2$. It is noted that $\mathbf{C} \mathbf{x} = \mathbf{0}$ for $\mathbf{x} = \boldsymbol{\ell}_1 \times \boldsymbol{\ell}_2$.

A degenerate dual conic

$$\boldsymbol{\ell}^T \mathbf{C}^* \boldsymbol{\ell} = 0 \quad (1223)$$

is defined by the singular rank 2 matrix

$$\mathbf{C}^* = \mathbf{x}_1 \mathbf{x}_2^T + \mathbf{x}_2 \mathbf{x}_1^T \quad (1224)$$

The lines of the dual conic are the lines that pass through either \mathbf{x}_1 or \mathbf{x}_2 .

19.21 Circular points*

The circular points

$$\mathbf{z}_1 = \begin{bmatrix} 1 \\ i \\ 0 \end{bmatrix}, \quad \mathbf{z}_2 = \begin{bmatrix} 1 \\ -i \\ 0 \end{bmatrix} \quad (1225)$$

are complex points with i as the imaginary unit, and because the last component is zero, they are said to be complex points at infinity. The points are called circular points because both points are solutions to the conic for a circle, which is

$$A(x_1^2 + x_2^2) + Dx_1x_3 + Ex_2x_3 + Fx_3^2 = 0 \quad (1226)$$

For the circular points the coordinates are $x_1 = 1$, $x_2 = \pm i$ and $x_3 = 0$, and the equation becomes $A[1^2 + (\pm i)^2] = 0$. which means that both circular points are on the circle.

An important property of the circular points is that they are invariant under similarity transformations. To verify this, consider

$$\mathbf{z}'_1 = \mathbf{H}_s \mathbf{z}_1 = \begin{bmatrix} s \cos \theta & -s \sin \theta & t_1 \\ s \sin \theta & s \cos \theta & t_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ i \\ 0 \end{bmatrix} = s \begin{bmatrix} \cos \theta - i \sin \theta \\ \sin \theta + i \cos \theta \\ 0 \end{bmatrix} \quad (1227)$$

$$\mathbf{z}'_2 = \mathbf{H}_s \mathbf{z}_2 = \begin{bmatrix} s \cos \theta & -s \sin \theta & t_1 \\ s \sin \theta & s \cos \theta & t_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -i \\ 0 \end{bmatrix} = s \begin{bmatrix} \cos \theta + i \sin \theta \\ \sin \theta - i \cos \theta \\ 0 \end{bmatrix} \quad (1228)$$

This is further developed by noting that $\exp(-i\theta) = \cos \theta - i \sin \theta$, $i \exp(-i\theta) = i \cos \theta + \sin \theta$, $\exp(i\theta) = \cos \theta + i \sin \theta$ and $-i \exp(i\theta) = -i \cos \theta + \sin \theta$. Then the expressions for \mathbf{z}'_1 and \mathbf{z}'_2 can be factored as

$$\mathbf{z}'_1 = s \exp(-i\theta) \begin{bmatrix} 1 \\ i \\ 0 \end{bmatrix} = s \exp(-i\theta) \mathbf{z}_1 \quad (1229)$$

$$\mathbf{z}'_2 = s \exp(i\theta) \begin{bmatrix} 1 \\ -i \\ 0 \end{bmatrix} = s \exp(i\theta) \mathbf{z}_2 \quad (1230)$$

It is seen that for both circular points, the transformation by a similarity transform gives the same circular point except for the scaling by a complex number.

19.22 Degenerate dual conic of the circular points*

A degenerate dual conic can be defined from the circular points with the coefficient matrix

$$\mathbf{C}_\infty^* = \mathbf{z}_1 \mathbf{z}_2^T + \mathbf{z}_2 \mathbf{z}_1^T \quad (1231)$$

Direct calculation of the outer products gives

$$\mathbf{C}_\infty^* = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (1232)$$

It is noted that the line at infinity ℓ_∞ is a line of this dual conic.

Let $\ell_1 = (\mathbf{n}_1^T, c_1)^T$ and $\ell_2 = (\mathbf{n}_2^T, c_2)^T$ be two lines normal vectors \mathbf{n}_1 and \mathbf{n}_2 . The lines are in the scene, which is the Euclidean space. Then

$$\ell_1^T \mathbf{C}_\infty^* \ell_2 = \begin{bmatrix} \mathbf{n}_1^T & c_1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{n}_2 \\ c_2 \end{bmatrix} = \mathbf{n}_1^T \mathbf{n}_2 \quad (1233)$$

This is the scalar product of the normal vectors of the two lines, which is the cosine of the angle θ between the lines with a scaling depending on the magnitude of the normal lines. In the same way it is found that $\ell_i^T \mathbf{C}_\infty^* \ell_i = \mathbf{n}_i^T \mathbf{n}_i$ for $i = 1, 2$. This means that

$$\cos \theta = \frac{\mathbf{n}_1^T \mathbf{n}_2}{\sqrt{(\mathbf{n}_1^T \mathbf{n}_1)(\mathbf{n}_2^T \mathbf{n}_2)}} = \frac{\ell_1^T \mathbf{C}_\infty^* \ell_2}{\sqrt{(\ell_1^T \mathbf{C}_\infty^* \ell_1)(\ell_2^T \mathbf{C}_\infty^* \ell_2)}} \quad (1234)$$

19.23 Transformation of the dual conic of the circular points*

Consider a general projective transformation $\mathbf{x}' = \mathbf{H}_p \mathbf{x}$. The transformation of $\ell_1^T \mathbf{C}_\infty^* \ell_2$ gives

$$\ell_1'^T \mathbf{C}_\infty^* \ell_2' = (\ell_1^T \mathbf{H}_p^{-1})(\mathbf{H}_p \mathbf{C}_\infty^* \mathbf{H}_p^T)(\mathbf{H}_p^{-T} \ell_2) = \ell_1^T \mathbf{C}_\infty^* \ell_2 \quad (1235)$$

This is an important result, as it shows that $\ell_1^T \mathbf{C}_\infty^* \ell_2$ is invariant even under projective transformations.

The case where the two lines ℓ_1 and ℓ_2 are orthogonal is of special interest. In this case $\ell_1^T \mathbf{C}_\infty^* \ell_2 = 0$ and it follows that $\ell_1'^T \mathbf{C}_\infty^* \ell_2' = 0$.

To proceed, it is necessary to study the transformation

$$\mathbf{C}_\infty^* = \mathbf{H}_p \mathbf{C}_\infty^* \mathbf{H}_p^T \quad (1236)$$

closer using the decomposition $\mathbf{H}_p = \mathbf{H}_{ps} \mathbf{H}_{as} \mathbf{H}_s$ of the projective homography

$$\mathbf{H}_p = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{v}^T \mathbf{A} & v_3 \end{bmatrix} \quad (1237)$$

where

$$\mathbf{H}_{ps} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v_3 \end{bmatrix}, \quad \mathbf{H}_{as} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{H}_s = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1238)$$

First, the similarity transformation is investigated. Then the coefficient matrix of the dual conic transforms according to

$$\mathbf{H}_s \mathbf{C}_\infty^* \mathbf{H}_s^T = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} s\mathbf{R}^T & \mathbf{0} \\ \mathbf{t}^T & 1 \end{bmatrix} \quad (1239)$$

This gives

$$\mathbf{H}_s \mathbf{C}_\infty^* \mathbf{H}_s^T = s^2 \mathbf{C}_\infty^* \quad (1240)$$

which shows that the coefficient matrix of the transformed dual conic is the same as the original coefficient matrix times a scaling factor s^2 . This means that $\mathbf{H}_s \mathbf{C}_\infty^* \mathbf{H}_s^T$ and \mathbf{C}_∞^* represents the same conic under a similarity transformation.

It is noted that a projective transformation gives

$$C_{\infty}^* = \mathbf{H}_p C_{\infty}^* \mathbf{H}_p^T \quad (1241)$$

$$= \mathbf{H}_{ps} \mathbf{H}_{as} \mathbf{H}_s C_{\infty}^* \mathbf{H}_s^T \mathbf{H}_{as}^T \mathbf{H}_{ps}^T \quad (1242)$$

$$= s^2 \mathbf{H}_{ps} \mathbf{H}_{as} C_{\infty}^* \mathbf{H}_{as}^T \mathbf{H}_{ps}^T \quad (1243)$$

$$= s^2 \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{v}^T \mathbf{A} & v_3 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{A}^T & \mathbf{A}^T \mathbf{v} \\ \mathbf{0}^T & v_3 \end{bmatrix} \quad (1244)$$

$$= s^2 \begin{bmatrix} \mathbf{A} \mathbf{A}^T & \mathbf{A} \mathbf{A}^T \mathbf{v} \\ \mathbf{v}^T \mathbf{A} \mathbf{A}^T & \mathbf{v}^T \mathbf{A} \mathbf{A}^T \mathbf{v} \end{bmatrix} \quad (1245)$$

20 Quaternions

20.1 Introduction

Unit quaternions are important in applications involving rotations, like robotics, vision, drone control, aerospace and computer graphics. A unit quaternion has 4 parameters, and can be used as a representation of a 3×3 rotation matrix. The main advantage of unit quaternions is that they lead to efficient computations in the numerical integration of the kinematic differential equations, and they lead to equations that are suitable for control system design and analysis. Quaternions were invented by Hamilton in 1843. Quaternions were used in physics, e.g. in the form of Pauli spin matrices, but widespread use of unit quaternions started with the aerospace applications when strapdown inertial navigation systems replaced gimballed inertial systems in the late 1970s [73]. In a strapdown system the angular velocity was measured in the body frame of, e.g. an aeroplane, and the kinematic differential equations had to be numerically integrated to determine the rotation matrix of the plane, and this was efficiently done with unit quaternions. In 1982 a multiplicative extended Kalman filter was developed based on unit quaternions for attitude estimation for spacecraft. Unit quaternions were introduced in nonlinear control systems in the early 1990s, as in [78] and [19], and are now widely used. Also, unit quaternions were introduced in computer graphics [64] to calculate rotational motion in the simulation of motion.

20.2 Quaternion sum, difference and product

It is convenient to describe a quaternion as the sum of a scalar and a 3-dimensional vector. The scalar is then the real part and the vector is the imaginary part in the formulation of Hamilton. A quaternion $\mathbf{q} \in \mathbb{H}$ is then given as

$$\mathbf{q} = \alpha + \beta \quad (1246)$$

where $\alpha \in \mathbb{R}$ is a scalar and $\beta \in \mathbb{R}^3$ is 3-dimensional vector. Let two quaternions be given by $\mathbf{q}_1 = \alpha_1 + \beta_1 \in \mathbb{H}$ and $\mathbf{q}_2 = \alpha_2 + \beta_2 \in \mathbb{H}$. Then addition and subtraction is component-wise and is given by

$$\mathbf{q}_1 \pm \mathbf{q}_2 = \alpha_1 \pm \alpha_2 + \beta_1 \pm \beta_2 \in \mathbb{H} \quad (1247)$$

Multiplication with a scalar γ is defined by

$$\gamma \mathbf{q} = \gamma \alpha + \gamma \beta \in \mathbb{H} \quad (1248)$$

The quaternion product is written

$$\mathbf{q}_1 \circ \mathbf{q}_2 = (\alpha_1 \alpha_2 - \boldsymbol{\beta}_1 \cdot \boldsymbol{\beta}_2) + (\alpha_1 \boldsymbol{\beta}_2 + \alpha_2 \boldsymbol{\beta}_1 + \boldsymbol{\beta}_1 \times \boldsymbol{\beta}_2) \in \mathbb{H} \quad (1249)$$

The quaternion product is associative, which means that

$$\mathbf{q}_1 \circ (\mathbf{q}_2 \circ \mathbf{q}_3) = (\mathbf{q}_1 \circ \mathbf{q}_2) \circ \mathbf{q}_3 = \mathbf{q}_1 \circ \mathbf{q}_2 \circ \mathbf{q}_3 \in \mathbb{H} \quad (1250)$$

for $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3 \in \mathbb{H}$. The quaternion product is not commutative, and its commutator is given by

$$\mathbf{q}_1 \circ \mathbf{q}_2 - \mathbf{q}_2 \circ \mathbf{q}_1 = 2\boldsymbol{\beta}_1 \times \boldsymbol{\beta}_2 \quad (1251)$$

20.3 Quaternion norm, conjugate and inverse

The quaternion inner product is defined by

$$\mathbf{q}_1 \cdot \mathbf{q}_2 = \alpha_1 \alpha_2 + \boldsymbol{\beta}_1 \cdot \boldsymbol{\beta}_2 \quad (1252)$$

It follows that

$$\mathbf{q} \cdot \mathbf{q} = \alpha^2 + \boldsymbol{\beta} \cdot \boldsymbol{\beta} \quad (1253)$$

The norm $\|\mathbf{q}\|$ of a quaternion \mathbf{q} is defined by

$$\|\mathbf{q}\|^2 = \mathbf{q} \cdot \mathbf{q} = \alpha^2 + \boldsymbol{\beta} \cdot \boldsymbol{\beta} \quad (1254)$$

The conjugate quaternion is given by

$$\mathbf{q}^* = \alpha - \boldsymbol{\beta} \quad (1255)$$

The quaternion product of a quaternion and its conjugate is

$$\mathbf{q} \circ \mathbf{q}^* = \mathbf{q}^* \circ \mathbf{q} = \alpha^2 + \boldsymbol{\beta} \cdot \boldsymbol{\beta} \quad (1256)$$

It is seen that this implies that

$$\|\mathbf{q}\|^2 = \mathbf{q} \cdot \mathbf{q} = \mathbf{q} \circ \mathbf{q}^* = \mathbf{q}^* \circ \mathbf{q} \quad (1257)$$

The identity quaternion $\mathbf{q}_{\text{id}} = 1$ is defined in terms of the quaternion product so that

$$\mathbf{q} \circ \mathbf{q}_{\text{id}} = \mathbf{q}_{\text{id}} \circ \mathbf{q} = \mathbf{q} \quad (1258)$$

The inverse quaternion \mathbf{q}^{-1} is defined by

$$\mathbf{q} \circ \mathbf{q}^{-1} = \mathbf{q}_{\text{id}} = 1 \quad (1259)$$

Pre-multiplication with \mathbf{q}^* gives $\mathbf{q}^* \circ \mathbf{q} \circ \mathbf{q}^{-1} = \mathbf{q}^*$, and it follows from $\mathbf{q}^* \circ \mathbf{q} = \|\mathbf{q}\|^2$ and associativity that $\|\mathbf{q}\|^2 \mathbf{q}^{-1} = \mathbf{q}^*$, which gives

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\|\mathbf{q}\|^2} \quad (1260)$$

It follows from

$$1 = \mathbf{q}_1 \circ \mathbf{q}_1^{-1} = \mathbf{q}_1 \circ (\mathbf{q}_2 \circ \mathbf{q}_2^{-1}) \circ \mathbf{q}_1^{-1} = (\mathbf{q}_1 \circ \mathbf{q}_2) \circ (\mathbf{q}_2^{-1} \circ \mathbf{q}_1^{-1})$$

that the inverse of a quaternion product is the product of the inverse quaternions in reversed order, which is written

$$(\mathbf{q}_1 \circ \mathbf{q}_2)^{-1} = \mathbf{q}_2^{-1} \circ \mathbf{q}_1^{-1} \quad (1261)$$

Moreover, the conjugate of the quaternion product is

$$(\mathbf{q}_1 \circ \mathbf{q}_2)^* = (\alpha_1 \alpha_2 - \boldsymbol{\beta}_1 \cdot \boldsymbol{\beta}_2) - (\alpha_1 \boldsymbol{\beta}_2 + \alpha_2 \boldsymbol{\beta}_1 + \boldsymbol{\beta}_1 \times \boldsymbol{\beta}_2) = \mathbf{q}_2^* \circ \mathbf{q}_1^* \quad (1262)$$

Then it follows that

$$(\mathbf{q}_1 \circ \mathbf{q}_2) \circ (\mathbf{q}_1 \circ \mathbf{q}_2)^* = (\mathbf{q}_1 \circ \mathbf{q}_2) \circ (\mathbf{q}_2^* \circ \mathbf{q}_1^*) = (\mathbf{q}_1 \circ \mathbf{q}_1^*) \|\mathbf{q}_2\|^2 = \|\mathbf{q}_1\|^2 \|\mathbf{q}_2\|^2 \quad (1263)$$

which leads to

$$\|\mathbf{q}_1 \circ \mathbf{q}_2\|^2 = \|\mathbf{q}_1\|^2 \|\mathbf{q}_2\|^2 \quad (1264)$$

It is noted that this was called the law of moduli by Hamilton. This is a property of complex numbers, and Hamilton attempted to find a formulation with triplets that satisfies this law. This did not succeed, and it was later proved that this was impossible for the dimension 3. Hamilton later increased the dimension to 4, which lead to his discovery of the quaternions [76].

An interesting result is

$$\begin{aligned} \mathbf{q}_1 \circ \mathbf{q}_2^* &= (\alpha_1 \alpha_2 + \boldsymbol{\beta}_1 \cdot \boldsymbol{\beta}_2) - (\alpha_1 \boldsymbol{\beta}_2 + \alpha_2 \boldsymbol{\beta}_1 + \boldsymbol{\beta}_1 \times \boldsymbol{\beta}_2) \\ &= \mathbf{q}_1 \cdot \mathbf{q}_2 - (\alpha_1 \boldsymbol{\beta}_2 + \alpha_2 \boldsymbol{\beta}_1 + \boldsymbol{\beta}_1 \times \boldsymbol{\beta}_2) \end{aligned} \quad (1265)$$

which shows that the scalar part of the quaternion product $\mathbf{q}_1 \circ \mathbf{q}_2^*$ is the quaternion inner product $\mathbf{q}_1 \cdot \mathbf{q}_2$.

20.4 Vectors as quaternions

A vector \mathbf{v} can be regarded as a quaternion with zero scalar part. It is then possible to formulate the quaternion product of a quaternion $\mathbf{q} = \alpha + \boldsymbol{\beta}$ and a vector \mathbf{v} as

$$\mathbf{q} \circ \mathbf{v} = -\boldsymbol{\beta} \cdot \mathbf{v} + \alpha \mathbf{v} + \boldsymbol{\beta} \times \mathbf{v} \quad (1266)$$

and

$$\mathbf{v} \circ \mathbf{q} = -\boldsymbol{\beta} \cdot \mathbf{v} + \alpha \mathbf{v} - \boldsymbol{\beta} \times \mathbf{v} \quad (1267)$$

It follows that the quaternion product of two vectors \mathbf{v}_1 and \mathbf{v}_2 is

$$\mathbf{v}_1 \circ \mathbf{v}_2 = -\mathbf{v}_1 \cdot \mathbf{v}_2 + \mathbf{v}_1 \times \mathbf{v}_2 \quad (1268)$$

The conjugate of a vector is $\mathbf{v}^* = -\mathbf{v}$. The quaternion magnitude of a vector \mathbf{v} is consistent with the vector magnitude, which is seen from

$$\|\mathbf{v}\|^2 = \mathbf{v} \circ \mathbf{v}^* = -\mathbf{v} \circ \mathbf{v} = \mathbf{v} \cdot \mathbf{v} \quad (1269)$$

20.5 The quaternion as a 4-dimensional column vector

A widely used notation is to represent a quaternions $\mathbf{q} = \alpha + \boldsymbol{\beta} \in \mathbb{H}$ as a 4-dimensional vector

$$[\mathbf{q}] = \begin{bmatrix} \alpha \\ \boldsymbol{\beta} \end{bmatrix} \in \mathbb{R}^4 \quad (1270)$$

The conjugation in this case is

$$[\mathbf{q}^*] = \begin{bmatrix} \alpha \\ -\boldsymbol{\beta} \end{bmatrix} \in \mathbb{R}^4 \quad (1271)$$

The inner product of two quaternions $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{H}$ is then in agreement with the inner product of the corresponding vectors $[\mathbf{q}_1], [\mathbf{q}_2] \in \mathbb{R}^4$, which is written

$$\mathbf{q}_1 \cdot \mathbf{q}_2 = [\mathbf{q}_1]^T [\mathbf{q}_2] \quad (1272)$$

The norm is given by the inner product as

$$\|\mathbf{q}\|^2 = \mathbf{q} \cdot \mathbf{q} = [\mathbf{q}]^T [\mathbf{q}] \quad (1273)$$

while the quaternion product is given in vector form by

$$[\mathbf{q}_1 \circ \mathbf{q}_2] = \begin{bmatrix} \alpha_1 \alpha_2 - \boldsymbol{\beta}_1^T \boldsymbol{\beta}_2 \\ \alpha_1 \boldsymbol{\beta}_2 + \alpha_2 \boldsymbol{\beta}_1 + \boldsymbol{\beta}_1 \times \boldsymbol{\beta}_2 \end{bmatrix} \quad (1274)$$

The norm is given by the first element of the quaternion product in vector form as

$$\|\mathbf{q}\|^2 = [\mathbf{q} \circ \mathbf{q}^*]_1 = \alpha_1 \alpha_2 + \boldsymbol{\beta}_1^T \boldsymbol{\beta}_2 \quad (1275)$$

20.6 Matrix representation of the quaternion product*

The quaternion product can also be written in terms of matrices as

$$[\mathbf{q}_1 \circ \mathbf{q}_2] = \mathbf{Q}_L(\mathbf{q}_1) [\mathbf{q}_2] = \mathbf{Q}_R(\mathbf{q}_2) [\mathbf{q}_1] \quad (1276)$$

where

$$\mathbf{Q}_L(\mathbf{q}) = \begin{bmatrix} \alpha & -\boldsymbol{\beta}^T \\ \boldsymbol{\beta} & \alpha \mathbf{I} + \boldsymbol{\beta} \times \end{bmatrix} \quad (1277)$$

$$\mathbf{Q}_R(\mathbf{q}) = \begin{bmatrix} \alpha & -\boldsymbol{\beta}^T \\ \boldsymbol{\beta} & \alpha \mathbf{I} - \boldsymbol{\beta} \times \end{bmatrix} \quad (1278)$$

It is seen that

$$\mathbf{Q}_L(\mathbf{q}^*) = \mathbf{Q}_L(\mathbf{q})^T, \quad \mathbf{Q}_R(\mathbf{q}^*) = \mathbf{Q}_R(\mathbf{q})^T \quad (1279)$$

It is straightforward to verify that

$$\mathbf{Q}_L(\mathbf{q})^T \mathbf{Q}_L(\mathbf{q}) = \mathbf{Q}_R(\mathbf{q})^T \mathbf{Q}_R(\mathbf{q}) = \mathbf{I} \|\mathbf{q}\|^2 \quad (1280)$$

and that

$$\mathbf{Q}_L(\mathbf{q}) \mathbf{Q}_R(\mathbf{q})^T = \mathbf{Q}_R(\mathbf{q})^T \mathbf{Q}_L(\mathbf{q}) \quad (1281)$$

The matrices can be written in the form

$$\mathbf{Q}_L(\mathbf{q}) = \begin{bmatrix} [\mathbf{q}] & \Psi_L(\mathbf{q}) \end{bmatrix} \quad (1282)$$

$$\mathbf{Q}_R(\mathbf{q}) = \begin{bmatrix} [\mathbf{q}] & \Psi_R(\mathbf{q}) \end{bmatrix} \quad (1283)$$

where

$$\Psi_L(\mathbf{q}) = \begin{bmatrix} -\boldsymbol{\beta}^T \\ \alpha \mathbf{I} + \boldsymbol{\beta}^* \end{bmatrix} \quad (1284)$$

$$\Psi_R(\mathbf{q}) = \begin{bmatrix} -\boldsymbol{\beta}^T \\ \alpha \mathbf{I} - \boldsymbol{\beta}^* \end{bmatrix} \quad (1285)$$

It is noted that

$$\Psi_L(\mathbf{q})^T[\mathbf{q}] = \Psi_R(\mathbf{q})^T[\mathbf{q}] = \mathbf{0} \quad (1286)$$

It is seen that for a vector $\mathbf{v} \in \mathbb{H}$, the vector form is

$$[\mathbf{v}] = \begin{bmatrix} 0 \\ \mathbf{v} \end{bmatrix} \in \mathbb{R}^4 \quad (1287)$$

and

$$[\mathbf{q} \circ \mathbf{v}] = \mathbf{Q}_L(\mathbf{q})[\mathbf{v}] = \Psi_L(\mathbf{q})\mathbf{v}, \quad [\mathbf{v} \circ \mathbf{q}] = \mathbf{Q}_R(\mathbf{q})[\mathbf{v}] = \Psi_R(\mathbf{q})\mathbf{v} \quad (1288)$$

20.7 Hamilton's Representation in complex form*

In this section the original formulation of Hamilton is presented. On October 16, 1843 [76], Hamilton discovered the quaternions as complex numbers with one real part and three imaginary parts. The quaternion is then written in the form

$$q = q_s + q_1i + q_2j + q_3k \quad (1289)$$

where the complex units i , j and k satisfy

$$i^2 = j^2 = k^2 = -1 \quad (1290)$$

$$ij = -ji = k, \quad jk = -kj = i, \quad ki = -ik = j \quad (1291)$$

Multiplication with a scalar α gives

$$\alpha q = \alpha(q_s + q_1i + q_2j + q_3k) = \alpha q_s + \alpha q_1i + \alpha q_2j + \alpha q_3k$$

Let two quaternions be given by $q = q_s + q_1i + q_2j + q_3k$ and $p = p_s + p_1i + p_2j + p_3k$. Addition and subtraction is component-wise and is given by

$$q \pm p = q_s \pm p_s + (q_1 \pm p_1)i + (q_2 \pm p_2)j + (q_3 \pm p_3)k$$

The multiplication of the two quaternions is done with the quaternion product. This is defined in the same way as for the multiplication of complex numbers, which gives

$$qp = (q_s + q_1i + q_2j + q_3k)(p_s + p_1i + p_2j + p_3k) \quad (1292)$$

$$= q_s p_s - q_1 p_1 - q_2 p_2 - q_3 p_3 \quad (1293)$$

$$+ (q_s p_1 + p_s q_1 + q_2 p_3 - q_3 p_2)i \quad (1294)$$

$$+ (q_s p_2 + p_s q_2 + q_3 p_1 - q_1 p_3)j \quad (1295)$$

$$+ (q_s p_3 + p_s q_3 + q_1 p_2 - q_2 p_1)k \quad (1296)$$

The conjugate of a quaternion is obtained by changing the sign of the imaginary parts, which gives the conjugate quaternion as

$$q^* = q_s - iq_1 - jq_2 - kq_3 \quad (1297)$$

Then the quaternion product gives

$$qq^* = q_s^2 + q_1^2 + q_2^2 + q_3^2 \quad (1298)$$

The magnitude $\|q\|$ of a quaternion is defined as

$$\|q\|^2 = q_s^2 + q_1^2 + q_2^2 + q_3^2 \quad (1299)$$

and it follows that

$$\|q\|^2 = qq^* \quad (1300)$$

The inverse quaternion q^{-1} is defined in terms of the quaternion product by $qq^{-1} = 1$, which gives

$$q^{-1} = \frac{q^*}{\|q\|^2} \quad (1301)$$

The inner product of two quaternions can be defined by defining the inner product of the complex units according to

$$i \cdot i = j \cdot j = k \cdot k = 1$$

Then

$$q \cdot q = q_s^2 + q_1^2 + q_2^2 + q_3^2 \quad (1302)$$

and it follows that $q \cdot q = qq^*$.

20.8 Unit quaternions

The unit quaternion is also referred to as the Euler parameters. The unit quaternion $\mathbf{q} = \eta + \boldsymbol{\epsilon}$ is a quaternion with norm 1, and satisfies

$$\|\mathbf{q}\|^2 = \mathbf{q} \cdot \mathbf{q} = \mathbf{q} \circ \mathbf{q}^* = \eta^2 + \boldsymbol{\epsilon} \cdot \boldsymbol{\epsilon} = 1 \quad (1303)$$

which implies that the inverse of the unit quaternion is equal to the conjugate, which is written

$$\mathbf{q}^* = \mathbf{q}^{-1} \quad (1304)$$

A unit quaternion can be described in terms of the Euler parameters $\eta = \cos \frac{\theta}{2}$ and $\boldsymbol{\epsilon} = \mathbf{k} \sin \frac{\theta}{2}$ as

$$\mathbf{q} = \eta + \boldsymbol{\epsilon} = \cos \frac{\theta}{2} + \mathbf{k} \sin \frac{\theta}{2} \quad (1305)$$

where \mathbf{k} is a unit vector. This is in agreement with the unit norm, which is seen from

$$\|\mathbf{q}\|^2 = \cos^2 \frac{\theta}{2} + \sin^2 \frac{\theta}{2} = 1 \quad (1306)$$

Consider a rotation matrix $\mathbf{R} \in SO(3)$ with angle-axis parameters given by the angle θ and the unit vector \mathbf{k} . Then the Rodrigues equation can be written

$$\mathbf{R} = \mathbf{I} + \mathbf{k}^\times \sin \theta + (1 - \cos \theta) \mathbf{k}^\times \mathbf{k}^\times \quad (1307)$$

Insertion of the trigonometric identities

$$\sin \theta = 2 \cos \frac{\theta}{2} \sin \frac{\theta}{2} \quad (1308)$$

$$\cos \theta = 1 - 2 \sin^2 \frac{\theta}{2} \quad (1309)$$

gives the rotation matrix expressed in terms of the Euler parameters as

$$\mathbf{R} = \mathbf{I} + 2\eta \boldsymbol{\epsilon}^\times + 2\boldsymbol{\epsilon}^\times \boldsymbol{\epsilon}^\times \quad (1310)$$

Note that this gives an expression without trigonometric terms.

The trace of the rotation matrix is found using $\boldsymbol{\epsilon}^\times \boldsymbol{\epsilon}^\times = \boldsymbol{\epsilon} \boldsymbol{\epsilon}^T - \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \mathbf{I}$ to be

$$\text{tr} \mathbf{R} = 3 - 4\boldsymbol{\epsilon}^T \boldsymbol{\epsilon} = 3 - 4(1 - \eta^2) = 4\eta^2 - 1 \quad (1311)$$

which gives

$$\eta^2 = \frac{\text{tr} \mathbf{R} + 1}{4} \quad (1312)$$

20.9 Representation of a rotation matrix by a unit quaternion

It is seen from (1310) that the unit quaternions \mathbf{q} and $-\mathbf{q}$ represents the same rotation matrix \mathbf{R} . It is noted that

$$-\eta = -\cos \frac{\theta}{2} = \cos \left(\frac{\theta}{2} + \pi \right) = \cos \frac{\theta + 2\pi}{2} \quad (1313)$$

$$-\boldsymbol{\epsilon} = -\sin \frac{\theta}{2} \mathbf{k} = \sin \left(\frac{\theta}{2} + \pi \right) \mathbf{k} = \sin \frac{\theta + 2\pi}{2} \mathbf{k} \quad (1314)$$

This means that if $\mathbf{q} = \eta + \boldsymbol{\epsilon}$ corresponds to an angle θ of rotation, then $-\mathbf{q} = -\eta - \boldsymbol{\epsilon}$ corresponds to an angle $\theta + 2\pi$ of rotation. This gives the same rotation matrix \mathbf{R} .

20.10 Shepperd's method for calculation of the unit quaternion

In this section it is shown how the unit quaternion of a rotation can be computed from the rotation matrix $\mathbf{R} = \{r_{ij}\}$. Some care must be taken in the formulation of the algorithm to arrive at continuous solutions and division by zero. The solution that is used is Shepperd's method from 1978, which was proposed in [62].

The rotation matrix is given in terms of the Euler parameters by (1310), which in component form gives

$$\mathbf{R} = \mathbf{R}_e(\eta, \boldsymbol{\epsilon}) = \begin{pmatrix} \eta^2 + \epsilon_1^2 - \epsilon_2^2 - \epsilon_3^2 & 2(\epsilon_1 \epsilon_2 - \eta \epsilon_3) & 2(\epsilon_1 \epsilon_3 + \eta \epsilon_2) \\ 2(\epsilon_1 \epsilon_2 + \eta \epsilon_3) & \eta^2 - \epsilon_1^2 + \epsilon_2^2 - \epsilon_3^2 & 2(\epsilon_2 \epsilon_3 - \eta \epsilon_1) \\ 2(\epsilon_1 \epsilon_3 - \eta \epsilon_2) & 2(\epsilon_2 \epsilon_3 + \eta \epsilon_1) & \eta^2 - \epsilon_1^2 - \epsilon_2^2 + \epsilon_3^2 \end{pmatrix} \quad (1315)$$

In addition, the condition

$$\eta^2 + \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 = 1 \quad (1316)$$

is given.

The following notation is introduced to simplify the algorithms:

$$\mathbf{z} = \begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} := 2 \begin{pmatrix} \eta \\ \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{pmatrix} \quad (1317)$$

$$T := r_{11} + r_{22} + r_{33} = \text{tr} \mathbf{R} \quad (1318)$$

and

$$r_{00} := T \quad (1319)$$

This gives the symmetric set of equations

$$z_0^2 = 1 + 2r_{00} - T \quad (1320)$$

$$z_1^2 = 1 + 2r_{11} - T \quad (1321)$$

$$z_2^2 = 1 + 2r_{22} - T \quad (1322)$$

$$z_3^2 = 1 + 2r_{33} - T \quad (1323)$$

that appear from the diagonal elements of \mathbf{R} , while the off-diagonal terms give the equations

$$z_0 z_1 = r_{32} - r_{23} \quad z_2 z_3 = r_{32} + r_{23} \quad (1324)$$

$$z_0 z_2 = r_{13} - r_{31} \quad z_3 z_1 = r_{13} + r_{31} \quad (1325)$$

$$z_0 z_3 = r_{21} - r_{12} \quad z_1 z_2 = r_{21} + r_{12} \quad (1326)$$

The algorithm is as follows:

1. Find the largest element in $\{r_{00}, r_{11}, r_{22}, r_{33}\}$. This element is denoted r_{ii} .

2. Compute

$$|z_i| = \sqrt{1 + 2r_{ii} - T} \quad (1327)$$

3. Determine the sign of z_i from some criterion, like continuity of solution, or $\eta > 0$.

4. Find the remaining z_j from the three equations out of (1324–1326) where the left hand side is $z_j z_i$ for all $j \neq i$. For example, if z_0 was found under step 2 and 3, then the remaining z_j are found from

$$z_1 = (r_{32} - r_{23})/z_0 \quad (1328)$$

$$z_2 = (r_{13} - r_{31})/z_0 \quad (1329)$$

$$z_3 = (r_{21} - r_{12})/z_0 \quad (1330)$$

5. Compute $\eta = z_0/2$ and $\epsilon_k = z_k/2$ for $k = 1, 2, 3$.

Note that this algorithm avoids division by zero as the division is done with the z_i that has the largest absolute value.

Comment

In [67] the solution

$$\eta = \frac{1}{2} \sqrt{r_{11} + r_{22} + r_{33} + 1} \quad (1331)$$

$$\epsilon_1 = \frac{1}{2} \operatorname{sgn}(r_{32} - r_{23}) \sqrt{r_{11} - r_{22} - r_{33} + 1} \quad (1332)$$

$$\epsilon_2 = \frac{1}{2} \operatorname{sgn}(r_{13} - r_{31}) \sqrt{r_{22} - r_{33} - r_{11} + 1} \quad (1333)$$

$$\epsilon_3 = \frac{1}{2} \operatorname{sgn}(r_{21} - r_{12}) \sqrt{r_{33} - r_{11} - r_{22} + 1} \quad (1334)$$

is given, where $\operatorname{sgn}(x) = 1$ for $x > 0$ and $\operatorname{sgn}(x) = -1$ for $x < 0$. This solution has the restriction that $\eta \geq 0$, and there are potential discontinuities due to the sgn function.

MATLAB function

```
function [eta, eps] = ShepperdR2q (R)
% Matlab function to compute a quaternion from rotation matrix using Shepperd's algorithm
% which is stable, does not lose significant precision, and uses only one square root.
% J. Guidance and Control, 1 (1978) 223-224.
z00 = R(1,1) + R(2,2) + R(3,3); % Trace of R
z11 = R(1,1) + R(1,1) - z00;
z22 = R(2,2) + R(2,2) - z00;
z33 = R(3,3) + R(3,3) - z00;

%Find a large zii to avoid division by zero
if z00 >= 0.5
    w = sqrt(1.0 + z00);
    wInv = 1.0/w;
    x = (R(3,2) - R(2,3))*wInv;
    y = (R(1,3) - R(3,1))*wInv;
    z = (R(2,1) - R(1,2))*wInv;
elseif z11 >= 0.5
    x = sqrt(1.0 + z11);
    xInv = 1.0/x;
    w = (R(3,2) - R(2,3))*xInv;
    y = (R(2,1) + R(1,2))*xInv;
    z = (R(3,1) + R(1,3))*xInv;
elseif z22 >= 0.5
    y = sqrt(1.0 + z22);
    yInv = 1.0/y;
    w = (R(1,3) - R(3,1))*yInv;
    x = (R(2,1) + R(1,2))*yInv;
    z = (R(3,2) + R(2,3))*yInv;
else
```

```

z = sqrt(1.0 + z33);
zInv = 1.0/z;
w = (R(2,1) - R(1,2))*zInv;
x = (R(3,1) + R(1,3))*zInv;
y = (R(3,2) + R(2,3))*zInv;
end

eta = 0.5*w;
eps = 0.5*[x; y; z];
end

```

20.11 Rotation of a vector by quaternion conjugation

The rotation of a vector \mathbf{a} by a rotation matrix \mathbf{R} gives the vector \mathbf{Ra} . The rotation can also be done with the corresponding unit quaternion \mathbf{q} in terms of the quaternion product as

$$\mathbf{Ra} = \mathbf{q} \circ \mathbf{a} \circ \mathbf{q}^{-1} \quad (1335)$$

This is verified with the calculation

$$\mathbf{q} \circ \mathbf{a} \circ \mathbf{q}^{-1} = (\eta + \boldsymbol{\epsilon}) \circ \mathbf{a} \circ (\eta - \boldsymbol{\epsilon}) \quad (1336)$$

$$= (-\boldsymbol{\epsilon}^T \mathbf{a} + (\eta \mathbf{a} + \boldsymbol{\epsilon}^\times \mathbf{a})) \circ (\eta - \boldsymbol{\epsilon}) \quad (1337)$$

$$= [-\eta \boldsymbol{\epsilon}^T \mathbf{a} + \eta \mathbf{a}^T \boldsymbol{\epsilon} + (\boldsymbol{\epsilon}^\times \mathbf{a})^T \boldsymbol{\epsilon}] \quad (1338)$$

$$\boldsymbol{\epsilon}^T \mathbf{a} \boldsymbol{\epsilon} + \eta^2 \mathbf{a} + \eta \boldsymbol{\epsilon}^\times \mathbf{a} - \eta \mathbf{a}^\times \boldsymbol{\epsilon} - (\boldsymbol{\epsilon}^\times \mathbf{a})^\times \boldsymbol{\epsilon} \quad (1339)$$

$$= \boldsymbol{\epsilon} \boldsymbol{\epsilon}^T \mathbf{a} + \eta^2 \mathbf{I} \mathbf{a} + \eta \boldsymbol{\epsilon}^\times \mathbf{a} + \eta \boldsymbol{\epsilon}^\times \mathbf{a} - (\mathbf{a} \boldsymbol{\epsilon}^T - \boldsymbol{\epsilon} \mathbf{a}^T) \boldsymbol{\epsilon} \quad (1340)$$

$$= (\boldsymbol{\epsilon} \boldsymbol{\epsilon}^T + \eta^2 \mathbf{I}) \mathbf{a} + 2\eta \boldsymbol{\epsilon}^\times \mathbf{a} - \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \mathbf{I} \mathbf{a} + \boldsymbol{\epsilon} \boldsymbol{\epsilon}^T \mathbf{a} \quad (1341)$$

$$= (2\boldsymbol{\epsilon} \boldsymbol{\epsilon}^T - 2\boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \mathbf{I} + \mathbf{I}) \mathbf{a} + 2\eta \boldsymbol{\epsilon}^\times \mathbf{a} \quad (1342)$$

$$= (\mathbf{I} + 2\eta \boldsymbol{\epsilon}^\times + 2\boldsymbol{\epsilon}^\times \boldsymbol{\epsilon}^\times) \mathbf{a} \quad (1343)$$

Here the scalar part is zero since $\boldsymbol{\epsilon}^T \mathbf{a} = \mathbf{a}^T \boldsymbol{\epsilon}$ and $\boldsymbol{\epsilon}^\times \mathbf{a}$ is orthogonal to $\boldsymbol{\epsilon}$. The vector part is simplified using $\boldsymbol{\epsilon}^\times \boldsymbol{\epsilon}^\times = \boldsymbol{\epsilon} \boldsymbol{\epsilon}^T - \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \mathbf{I}$, $(\boldsymbol{\epsilon}^\times \mathbf{a})^\times = \boldsymbol{\epsilon}^\times \mathbf{a}^\times - \mathbf{a}^\times \boldsymbol{\epsilon}^\times = \mathbf{a} \boldsymbol{\epsilon}^T - \boldsymbol{\epsilon} \mathbf{a}^T$ [20] and $\eta^2 + \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} = 1$. The expression in the bracket is seen from (1310) to be the rotation matrix.

This means that the rotation of a vector from \mathbf{a} to \mathbf{Ra} can be expressed with the unit quaternion \mathbf{q} as the operation $\mathbf{q} \circ \mathbf{a} \circ \mathbf{q}^{-1}$, which is called a conjugation. It is seen that the scalar part of $\mathbf{q} \circ \mathbf{a} \circ \mathbf{q}^{-1}$ is zero, which is in agreement with the scalar part of \mathbf{Ra} , which is zero.

20.12 Vector form of the unit quaternion

The vector form of the unit quaternion is given by

$$[\mathbf{q}] = \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} \quad (1344)$$

The norm is then given by

$$[\mathbf{q}]^T [\mathbf{q}] = \eta^2 + \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} = 1 \quad (1345)$$

The vector form of the rotation equation for quaternions is given by

$$[\mathbf{q} \circ \mathbf{v} \circ \mathbf{q}^*] = \begin{bmatrix} 0 \\ \mathbf{R}\mathbf{v} \end{bmatrix} \quad (1346)$$

20.13 Matrix representation of quaternion product for unit quaternions*

It is noted that

$$\Psi_L(\mathbf{q})^T \Psi_L(\mathbf{q}) = \begin{bmatrix} -\epsilon & \eta \mathbf{I} - \epsilon^\times \end{bmatrix} \begin{bmatrix} -\epsilon^T \\ \eta \mathbf{I} + \epsilon^\times \end{bmatrix} = \mathbf{I} \quad (1347)$$

$$\Psi_R(\mathbf{q})^T \Psi_R(\mathbf{q}) = \begin{bmatrix} -\epsilon & \eta \mathbf{I} + \epsilon^\times \end{bmatrix} \begin{bmatrix} -\epsilon^T \\ \eta \mathbf{I} - \epsilon^\times \end{bmatrix} = \mathbf{I} \quad (1348)$$

where it is used that $\epsilon\epsilon^T + \eta^2 \mathbf{I} - \epsilon^\times \epsilon^\times = \epsilon\epsilon^T - \epsilon^T \epsilon \mathbf{I} + (\eta^2 + \epsilon^T \epsilon) \mathbf{I} - \epsilon^\times \epsilon^\times = \mathbf{I}$.

In matrix form this gives

$$[\mathbf{q} \circ \mathbf{v} \circ \mathbf{q}^*] = \mathbf{Q}_R(\mathbf{q}^*) \mathbf{Q}_L(\mathbf{q}) [\mathbf{v}] \quad (1349)$$

Here

$$\mathbf{Q}_R(\mathbf{q}^*) \mathbf{Q}_L(\mathbf{q}) = \mathbf{Q}_R(\mathbf{q})^T \mathbf{Q}_L(\mathbf{q}) \quad (1350)$$

$$= \begin{bmatrix} [\mathbf{q}]^T \\ \Psi_R(\mathbf{q})^T \end{bmatrix} \begin{bmatrix} [\mathbf{q}] & \Psi_L(\mathbf{q}) \end{bmatrix} \quad (1351)$$

$$= \begin{bmatrix} [\mathbf{q}]^T [\mathbf{q}] & \mathbf{0} \\ \mathbf{0} & \Psi_R(\mathbf{q})^T \Psi_L(\mathbf{q}) \end{bmatrix} \quad (1352)$$

$$= \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \quad (1353)$$

where it is used that

$$\Psi_R(\mathbf{q})^T \Psi_L(\mathbf{q}) = \begin{bmatrix} -\epsilon & \eta \mathbf{I} + \epsilon^\times \end{bmatrix} \begin{bmatrix} -\epsilon^T \\ \eta \mathbf{I} + \epsilon^\times \end{bmatrix} \quad (1354)$$

$$= \epsilon\epsilon^T + \eta^2 \mathbf{I} + 2\eta\epsilon^\times + \epsilon^\times\epsilon^\times \quad (1355)$$

$$= \epsilon\epsilon^T - \epsilon^T \epsilon \mathbf{I} + (\eta^2 + \epsilon\epsilon^T) \mathbf{I} + 2\eta\epsilon^\times + \epsilon^\times\epsilon^\times \quad (1356)$$

$$= \mathbf{R} \quad (1357)$$

Here $\epsilon^\times\epsilon^\times = \epsilon\epsilon^T - \epsilon^T \epsilon \mathbf{I}$ and $\eta^2 + \epsilon^T \epsilon = 1$ is used. It follows that

$$[\mathbf{q} \circ \mathbf{v} \circ \mathbf{q}^*] = \mathbf{Q}_R(\mathbf{q}^*) \mathbf{Q}_L(\mathbf{q}) [\mathbf{v}] = \begin{bmatrix} 0 \\ \mathbf{R}\mathbf{v} \end{bmatrix} \quad (1358)$$

20.14 Composite rotations in terms of quaternions

A composite rotation

$$\mathbf{R} = \mathbf{R}_1 \mathbf{R}_2 \quad (1359)$$

can then be described with the quaternion

$$\mathbf{q} = \mathbf{q}_1 \circ \mathbf{q}_2 \quad (1360)$$

where $\mathbf{q}_1 = \eta_1 + \boldsymbol{\epsilon}_1$ corresponds to \mathbf{R}_1 and $\mathbf{q}_2 = \eta_2 + \boldsymbol{\epsilon}_2$ corresponds to \mathbf{R}_2 . This is seen from

$$\mathbf{q} \circ \mathbf{a} \circ \mathbf{q}^* = (\mathbf{q}_1 \circ \mathbf{q}_2) \circ \mathbf{a} \circ (\mathbf{q}_2^* \circ \mathbf{q}_1^*) \quad (1361)$$

$$= \mathbf{q}_1 \circ (\mathbf{q}_2 \circ \mathbf{a} \circ \mathbf{q}_2^*) \circ \mathbf{q}_1^* \quad (1362)$$

$$= \mathbf{R}_1(\mathbf{R}_2 \mathbf{a}) \quad (1363)$$

$$= \mathbf{R}\mathbf{a} \quad (1364)$$

Here it is used that $\mathbf{q}^* = (\mathbf{q}_1 \circ \mathbf{q}_2)^* = \mathbf{q}_2^* \circ \mathbf{q}_1^*$.

20.15 The deviation between two quaternions

The deviation between two rotation matrices \mathbf{R}_1 and \mathbf{R}_2 can be described by the rotation matrix $\tilde{\mathbf{R}} = \mathbf{R}_1^T \mathbf{R}_2$, which means that $\mathbf{R}_2 = \mathbf{R}_1 \tilde{\mathbf{R}}$. The deviation between the corresponding unit quaternions \mathbf{q}_1 and \mathbf{q}_2 can then be described with $\tilde{\mathbf{q}} = \mathbf{q}_1^* \circ \mathbf{q}_2$, which gives $\mathbf{q}_2 = \mathbf{q}_1 \circ \tilde{\mathbf{q}}$. It follows that $\tilde{\mathbf{q}}$ corresponds to the rotation matrix $\tilde{\mathbf{R}}$. Let (θ_1, \mathbf{k}_1) be the angle axis parameters of the rotation described by \mathbf{R}_1 and \mathbf{q}_1 , and let (θ_2, \mathbf{k}_2) be the angle axis parameters of the rotation described by \mathbf{R}_2 and \mathbf{q}_2 . Moreover, let (ϕ, \mathbf{a}) be the angle axis parameters of the rotation described by the deviation as given by $\tilde{\mathbf{R}}$ and $\tilde{\mathbf{q}}$. Then

$$\tilde{\mathbf{q}} = \mathbf{q}_1^* \circ \mathbf{q}_2 = \cos \frac{\phi}{2} + \sin \frac{\phi}{2} \mathbf{a} \quad (1365)$$

It is noted that

$$\begin{aligned} \mathbf{q}_1^* \circ \mathbf{q}_2 &= \cos \frac{\theta_1}{2} \cos \frac{\theta_2}{2} + \sin \frac{\theta_1}{2} \sin \frac{\theta_2}{2} \mathbf{k}_1 \cdot \mathbf{k}_2 \\ &\quad + \cos \frac{\theta_1}{2} \sin \frac{\theta_2}{2} \mathbf{k}_2 - \cos \frac{\theta_2}{2} \sin \frac{\theta_1}{2} \mathbf{k}_1 - \sin \frac{\theta_1}{2} \sin \frac{\theta_2}{2} \mathbf{k}_1 \times \mathbf{k}_2 \end{aligned} \quad (1366)$$

and

$$\mathbf{q}_1 \cdot \mathbf{q}_2 = \cos \frac{\theta_1}{2} \cos \frac{\theta_2}{2} + \sin \frac{\theta_1}{2} \sin \frac{\theta_2}{2} \mathbf{k}_1 \cdot \mathbf{k}_2 \quad (1367)$$

From these three equations it is seen that the inner product of two unit quaternions is

$$\mathbf{q}_1 \cdot \mathbf{q}_2 = \cos \frac{\phi}{2} \quad (1368)$$

which is the cosine of the half angle between them.

20.16 Kinematic differential equations for quaternions

In this section the kinematic differential equations for the unit quaternion will be developed. Let $\mathbf{R} = \mathbf{R}_b^a$ be the rotation matrix from a to b , and let $\boldsymbol{\omega}^a$ be the angular velocity in the coordinates of the a frame. Time differentiation of the relation

$$\mathbf{R}\mathbf{a} = \mathbf{q} \circ \mathbf{a} \circ \mathbf{q}^* \quad (1369)$$

gives

$$\dot{\mathbf{R}}\mathbf{a} + \mathbf{R}\dot{\mathbf{a}} = \dot{\mathbf{q}} \circ \mathbf{a} \circ \mathbf{q}^* + \mathbf{q} \circ \dot{\mathbf{a}} \circ \mathbf{q}^* + \mathbf{q} \circ \mathbf{a} \circ \dot{\mathbf{q}}^* \quad (1370)$$

Here $\mathbf{R}\dot{\mathbf{a}} = \mathbf{q} \circ \dot{\mathbf{a}} \circ \mathbf{q}^*$, and this gives

$$\dot{\mathbf{R}}\mathbf{a} = \dot{\mathbf{q}} \circ \mathbf{a} \circ \mathbf{q}^* + \mathbf{q} \circ \mathbf{a} \circ \dot{\mathbf{q}}^* \quad (1371)$$

From $\mathbf{q} \circ \mathbf{q}^* = 1$ it follows that $\dot{\mathbf{q}} \circ \mathbf{q}^* + \mathbf{q} \circ \dot{\mathbf{q}}^* = 0$ and therefore $\dot{\mathbf{q}}^* = -\mathbf{q}^* \circ \dot{\mathbf{q}} \circ \mathbf{q}^*$. This gives

$$\dot{\mathbf{R}}\mathbf{a} = \dot{\mathbf{q}} \circ (\mathbf{q}^* \circ \mathbf{q}) \circ \mathbf{a} \circ \mathbf{q}^* - \mathbf{q} \circ \mathbf{a} \circ \mathbf{q}^* \circ \dot{\mathbf{q}} \circ \mathbf{q}^* \quad (1372)$$

$$= (\dot{\mathbf{q}} \circ \mathbf{q}^*) \circ (\mathbf{q} \circ \mathbf{a} \circ \mathbf{q}^*) - (\mathbf{q} \circ \mathbf{a} \circ \mathbf{q}^*) \circ (\dot{\mathbf{q}} \circ \mathbf{q}^*) \quad (1373)$$

$$= (\dot{\mathbf{q}} \circ \mathbf{q}^*) \circ (\mathbf{R}\mathbf{a}) - (\mathbf{R}\mathbf{a}) \circ (\dot{\mathbf{q}} \circ \mathbf{q}^*) \quad (1374)$$

Then the quaternion commutator (1251) gives

$$\dot{\mathbf{R}}\mathbf{a} = 2(\dot{\mathbf{q}} \circ \mathbf{q}^*)^\times (\mathbf{R}\mathbf{a}) \quad (1375)$$

where it is used that the term $\dot{\mathbf{q}} \circ \mathbf{q}^*$ has zero scalar part. This is verified by the calculation

$$\dot{\mathbf{q}} \circ \mathbf{q}^* = (\dot{\eta} + \dot{\epsilon}) \circ (\eta - \epsilon) = (\dot{\eta}\eta + \dot{\epsilon}^T \epsilon) + (\eta\dot{\epsilon} - \dot{\eta}\epsilon + \epsilon \times \dot{\epsilon}) = \eta\dot{\epsilon} - \dot{\eta}\epsilon + \epsilon \times \dot{\epsilon} \quad (1376)$$

where it is used that the scalar part is $\dot{\eta}\eta + \dot{\epsilon}^T \epsilon = (d/dt)(\eta^2 + \epsilon^T \epsilon) = 0$.

Insertion of $\dot{\mathbf{R}} = (\boldsymbol{\omega}^a)^\times \mathbf{R}$ in (1375) gives

$$(\boldsymbol{\omega}^a)^\times \mathbf{R}\mathbf{a} = 2(\dot{\mathbf{q}} \circ \mathbf{q}^*)^\times \mathbf{R}\mathbf{a} \quad (1377)$$

This result is valid for all values of $\mathbf{R}\mathbf{a}$, and it follows that

$$\boldsymbol{\omega}^a = 2\dot{\mathbf{q}} \circ \mathbf{q}^* \quad (1378)$$

If the angular velocity is given in the b frame by $\boldsymbol{\omega}^b$, then it is used that $\boldsymbol{\omega}^a = \mathbf{R}\boldsymbol{\omega}^b = \mathbf{q} \circ \boldsymbol{\omega}^b \circ \mathbf{q}^*$. This gives

$$\boldsymbol{\omega}^b = 2\mathbf{q}^* \circ \dot{\mathbf{q}} \quad (1379)$$

The kinematic differential equations are then found by post-multiplication of (1378) by $\frac{1}{2}\mathbf{q}$ and by pre-multiplication of (1379) by $\frac{1}{2}\mathbf{q}$ to be

$$\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\omega}^a \circ \mathbf{q} \quad (1380)$$

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \circ \boldsymbol{\omega}^b \quad (1381)$$

This can be written out as

$$\dot{\mathbf{q}} = -\frac{1}{2}\boldsymbol{\epsilon}^T \boldsymbol{\omega}^a + \frac{1}{2}(\eta \mathbf{I} - \boldsymbol{\epsilon}^\times) \boldsymbol{\omega}^a \quad (1382)$$

$$\dot{\mathbf{q}} = -\frac{1}{2}\boldsymbol{\epsilon}^T \boldsymbol{\omega}^b + \frac{1}{2}(\eta \mathbf{I} + \boldsymbol{\epsilon}^\times) \boldsymbol{\omega}^b \quad (1383)$$

In terms of the scalar and vector part this is written

$$\dot{\eta} = -\frac{1}{2}\boldsymbol{\epsilon}^T \boldsymbol{\omega}^a \quad (1384)$$

$$\dot{\boldsymbol{\epsilon}} = \frac{1}{2}(\eta \mathbf{I} - \boldsymbol{\epsilon}^\times) \boldsymbol{\omega}^a \quad (1385)$$

when the angular velocity is given in the fixed frame a , and

$$\dot{\eta} = -\frac{1}{2}\boldsymbol{\epsilon}^T \boldsymbol{\omega}^b \quad (1386)$$

$$\dot{\boldsymbol{\epsilon}} = \frac{1}{2}(\eta \mathbf{I} + \boldsymbol{\epsilon}^\times) \boldsymbol{\omega}^b \quad (1387)$$

when the angular velocity is given in the body frame b .

The angular velocity can be expressed in term for the scalar and vector part of the quaternion by evaluating the quaternion products i (1378) and (1379). This gives

$$\boldsymbol{\omega}^a = 2(\eta \dot{\boldsymbol{\epsilon}} - \dot{\eta} \boldsymbol{\epsilon} + \boldsymbol{\epsilon}^\times \dot{\boldsymbol{\epsilon}}) \quad (1388)$$

$$\boldsymbol{\omega}^b = 2(\eta \dot{\boldsymbol{\epsilon}} - \dot{\eta} \boldsymbol{\epsilon} - \boldsymbol{\epsilon}^\times \dot{\boldsymbol{\epsilon}}) \quad (1389)$$

20.17 Kinematic differential equations for composite rotations

Consider the quaternion \mathbf{q}_{ab} corresponding to the rotation matrix \mathbf{R}_b^a from a to b , and the quaternion \mathbf{q}_{bc} corresponding to the rotation matrix \mathbf{R}_c^b from b to c . Then the composite quaternion

$$\mathbf{q}_{ac} = \mathbf{q}_{ab} \circ \mathbf{q}_{bc} \quad (1390)$$

corresponds to the rotation matrix $\mathbf{R}_c^a = \mathbf{R}_b^a \mathbf{R}_c^b$. The kinematic differential equations for the two quaternions \mathbf{q}_{ab} and \mathbf{q}_{bc} are given by

$$\dot{\mathbf{q}}_{ab} = \frac{1}{2}\boldsymbol{\omega}_{ab}^a \circ \mathbf{q}_{ab} = \frac{1}{2}\mathbf{q}_{ab} \circ \boldsymbol{\omega}_{ab}^b \quad (1391)$$

$$\dot{\mathbf{q}}_{bc} = \frac{1}{2}\boldsymbol{\omega}_{bc}^b \circ \mathbf{q}_{bc} = \frac{1}{2}\mathbf{q}_{bc} \circ \boldsymbol{\omega}_{bc}^c \quad (1392)$$

which is consistent with the coordinate transformations

$$\boldsymbol{\omega}_{ab}^a = \mathbf{q}_{ab} \circ \boldsymbol{\omega}_{ab}^b \circ \mathbf{q}_{ab}^{-1} \quad (1393)$$

$$\boldsymbol{\omega}_{bc}^b = \mathbf{q}_{bc} \circ \boldsymbol{\omega}_{bc}^c \circ \mathbf{q}_{bc}^{-1} \quad (1394)$$

The kinematic differential equation for the composite quaternion is then

$$\dot{\mathbf{q}}_{ac} = \dot{\mathbf{q}}_{ab} \circ \mathbf{q}_{bc} + \mathbf{q}_{ab} \circ \dot{\mathbf{q}}_{bc} \quad (1395)$$

$$= \frac{1}{2}\boldsymbol{\omega}_{ab}^a \circ \mathbf{q}_{ab} \circ \mathbf{q}_{bc} + \frac{1}{2}\mathbf{q}_{ab} \circ \boldsymbol{\omega}_{bc}^b \circ \mathbf{q}_{bc} \quad (1396)$$

$$= \frac{1}{2}\boldsymbol{\omega}_{ab}^a \circ \mathbf{q}_{ab} \circ \mathbf{q}_{bc} + \frac{1}{2}\mathbf{q}_{ab} \circ \boldsymbol{\omega}_{bc}^b \circ \mathbf{q}_{ab}^{-1} \circ \mathbf{q}_{ab} \circ \mathbf{q}_{bc} \quad (1397)$$

$$= \frac{1}{2}\boldsymbol{\omega}_{ab}^a \circ \mathbf{q}_{ab} \circ \mathbf{q}_{bc} + \frac{1}{2}\boldsymbol{\omega}_{bc}^a \circ \mathbf{q}_{ab} \circ \mathbf{q}_{bc} \quad (1398)$$

$$= \frac{1}{2}(\boldsymbol{\omega}_{ab}^a + \boldsymbol{\omega}_{bc}^a) \circ \mathbf{q}_{ab} \circ \mathbf{q}_{bc} \quad (1399)$$

$$= \frac{1}{2}\boldsymbol{\omega}_{ac}^a \circ \mathbf{q}_{ac} \quad (1400)$$

Another form of a composite quaternion is used in navigation problems, where the quaternion error is used. Then, if the quaternion from a to b is \mathbf{q}_{ab} , while the estimate is \mathbf{q}_{bc} , then the estimation error \mathbf{q}_{ac} is given by $\mathbf{q}_{ac} \circ \mathbf{q}_{bc} = \mathbf{q}_{ab}$, which gives the expression

$$\mathbf{q}_{ac} = \mathbf{q}_{ab} \circ \mathbf{q}_{cb}^{-1} \quad (1401)$$

for the error quaternion. The kinematic differential equation is found using (??) and $\mathbf{q}_{cb}^{-1} = \mathbf{q}_{bc}$, which gives (1400).

20.18 Normalization of unit quaternions

In numerical computations of a unit quaternion it may happen that the computed quaternion \mathbf{p} is not a unit quaternion, that is, the condition $\mathbf{p} \circ \mathbf{p}^* \neq 1$ does not hold. In that case the computed quaternion can be normalized by

$$\mathbf{q} = \frac{\mathbf{p}}{\|\mathbf{p}\|} \quad (1402)$$

where $\|\mathbf{p}\| = \sqrt{\mathbf{p} \circ \mathbf{p}^*}$, which will ensure the condition $\mathbf{q} \circ \mathbf{q}^* = 1$. This can be used in numerical solution of differential equations involving unit quaternions.

20.19 Logarithm of a unit quaternion

Consider a rotation by an angle θ about the unit vector \mathbf{k} , and let $\phi = \theta/2$ be half of the rotation angle. Let $\phi\mathbf{k}$ be considered as a quaternion. The exponential function of the quaternion $\phi\mathbf{k}$ is defined by

$$\exp(\phi\mathbf{k}) = 1 + \phi\mathbf{k} + \frac{(\phi\mathbf{k}) \circ (\phi\mathbf{k})}{2!} + \frac{(\phi\mathbf{k}) \circ (\phi\mathbf{k}) \circ (\phi\mathbf{k})}{3!} + \dots = \sum_{i=0}^{\infty} \frac{(\phi\mathbf{k})^i}{i!} \quad (1403)$$

where \mathbf{v}^n is the quaternion product of the vector \mathbf{v} of order n . It is noted that $\mathbf{k}^2 = \mathbf{k} \circ \mathbf{k} = -\mathbf{k} \cdot \mathbf{k} = -1$, while $\mathbf{k}^3 = \mathbf{k}^2 \circ \mathbf{k} = -\mathbf{k}$. This gives

$$\exp(\phi\mathbf{k}) = 1 + \phi\mathbf{k} + \frac{(\phi\mathbf{k})^2}{2!} + \frac{(\phi\mathbf{k})^3}{3!} + \frac{(\phi\mathbf{k})^4}{4!} + \frac{(\phi\mathbf{k})^5}{5!} + \dots \quad (1404)$$

$$= \left(1 - \frac{\phi^2}{2!} + \frac{\phi^4}{4!} + \dots\right) + \left(\phi - \frac{\phi^3}{3!} + \frac{\phi^5}{5!} \dots\right) \mathbf{k} \quad (1405)$$

$$= \cos \phi + \mathbf{k} \sin \phi \quad (1406)$$

From this result it is concluded that the unit quaternion can be expressed in terms of the quaternion exponential function as

$$\mathbf{q} = \exp\left(\frac{\theta}{2}\mathbf{k}\right) \quad (1407)$$

In accordance with this the logarithm of the quaternion is the quaternion

$$\log(\mathbf{q}) = \frac{\theta}{2}\mathbf{k} \quad (1408)$$

20.20 Computation of the quaternion from the logarithm

Suppose that the logarithm \mathbf{v} of a quaternion is given, and the quaternion $\mathbf{q} = \exp(\mathbf{v})$ is to be computed numerically. The logarithm can be written in terms of an angle ϕ and a unit vector \mathbf{k} as $\mathbf{v} = \phi\mathbf{k}$. Note that in this case it is assumed that \mathbf{v} is given, while ϕ and \mathbf{k} are not known. It follows that $\exp(\mathbf{v}) = \exp(\phi\mathbf{k}) = \cos\phi + \mathbf{k}\sin\phi$ since \mathbf{k} is a unit vector. Note that \mathbf{k} is undefined when $\|\mathbf{v}\| = |\phi| = 0$. In this case $\mathbf{v} = \mathbf{0}$ and $\exp(\mathbf{0}) = 1$.

Expressed in terms of \mathbf{v} this gives

$$\exp(\mathbf{v}) = \cos\|\mathbf{v}\| + \sin\|\mathbf{v}\|\frac{\mathbf{v}}{\|\mathbf{v}\|} \quad (1409)$$

A potential problem with the expression is that it is undefined for $\|\mathbf{v}\| = 0$. The solution to this problem is to reformulate the expression as

$$\exp(\mathbf{v}) = \cos\|\mathbf{v}\| + \text{sinc}(\|\mathbf{v}\|)\mathbf{v} \quad (1410)$$

where

$$\text{sinc}(x) = \frac{\sin x}{x} \quad (1411)$$

which satisfies $\text{sinc}(0) = 1$. It is easy to see from the Taylor series expansion of the sine function, which is

$$\sin x = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \dots$$

that the following series expansion can be used:

$$\text{sinc}(x) = 1 - \frac{1}{6}x^2 + \frac{1}{120}x^4 - \dots \quad (1412)$$

This means that $\text{sinc}(x)$ is defined, continuous and differentiable for all x , and $\text{sinc}(0) = 1$.

In case a numerically stable function for sinc is not available, the expression

$$\text{sinc}(x) \approx 1 - \frac{x^2}{6} \quad (1413)$$

from the two first terms in the Taylor series expansion can be used for stable and accurate computation of $\text{sinc}(x)$ close to zero.

20.21 Computation of the logarithm

Consider the case where the unit quaternion $\mathbf{q} = \eta + \epsilon$ is given, and the logarithm \mathbf{v} is to be computed so that $\exp(\mathbf{v}) = \mathbf{q}$. Then $\epsilon = \sin\phi\mathbf{k}$ where $\phi = \theta/2$ and \mathbf{k} is a unit vector, and it follows that $\sin\phi = \|\epsilon\|$. Then $\phi = \arcsin(\|\epsilon\|)$, and the logarithm $\mathbf{v} = \phi\mathbf{k}$ is found from

$$\mathbf{v} = \frac{\arcsin\|\epsilon\|}{\|\epsilon\|}\epsilon \quad (1414)$$

The expression involves division by $\|\epsilon\|$, and it must be checked how to handle the situation when $\|\epsilon\| \rightarrow 0$. The Taylor series expansion of the \arcsin function is

$$\arcsin(x) = x + \frac{x^3}{6} + \frac{3x^5}{40} + \dots \quad (1415)$$

and it follows that

$$\frac{\arcsin(x)}{x} = 1 + \frac{x^2}{6} + \frac{3x^4}{40} + \dots \quad (1416)$$

This shows that $\arcsin(x)/x$ is well conditions for all x , and that is can be approximated by $1 + x^2/6$ when x is close to zero.

20.22 Numerical integration of quaternions with Euler's method

As an introductory example, Euler's method for a system with state in a Euclidean space \mathbb{R}^n is presented.

Example

Consider the nonlinear system

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t)] \quad (1417)$$

where $\mathbf{x} \in \mathbb{R}^n$. Euler's method is used to integrate the solution of this type of system. The idea is to use a discrete-time description where the time is considered at $t_k = kh$ where h is the time step. It is noted that $t_{k+1} - t_k = h$. Then given $\mathbf{x}(t_k)$, an approximation of $\mathbf{x}(t_{k+1})$ is calculated by using $\mathbf{f}[\mathbf{x}(t)] \approx \mathbf{f}[\mathbf{x}(t_k)]$ in the whole interval from t_k to t_{k+1} . This gives

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + \int_{t_k}^{t_{k+1}} \mathbf{f}[\mathbf{x}(\tau)] d\tau \approx \mathbf{x}(t_k) + \int_{t_k}^{t_{k+1}} \mathbf{f}[\mathbf{x}(t_k)] d\tau \quad (1418)$$

which gives Euler's method

$$\mathbf{x}(t_{k+1}) = \mathbf{x}(t_k) + h\mathbf{f}[\mathbf{x}(t_k)] \quad (1419)$$

□

Consider the kinematic differential equation

$$\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\omega} \circ \mathbf{q} \quad (1420)$$

where \mathbf{q} is the quaternion from the a frame to the b frame, and $\boldsymbol{\omega} = \boldsymbol{\omega}^a$ is the angular velocity in the a frame. Application of Euler's method gives the integration scheme

$$\mathbf{q}(t_{k+1}) = \mathbf{q}(t_k) + \frac{h}{2}\boldsymbol{\omega}(t_k) \circ \mathbf{q}(t_k) \quad (1421)$$

The result $\mathbf{q}(t_{k+1})$ will in general not be a unit quaternion, even if $\mathbf{q}(t_k)$ is a unit quaternion. Therefore this type of integration scheme must be combined with a renormalization method like dividing the result with its norm, which is done with the assignment

$$\mathbf{q}(t_{k+1}) := \frac{\mathbf{q}(t_{k+1})}{\|\mathbf{q}(t_{k+1})\|} \quad (1422)$$

Other integration methods like Runge-Kutta 4 can be used. Also in this case a renormalization must be used.

20.23 Numerical integration of quaternions with exponential methods

An alternative method for integration of the kinematic differential equation for the quaternion is to use the exponential function for the quaternion increment. It is noted that if the angular velocity $\boldsymbol{\omega}(t)$ is assumed to be constant and given by $\boldsymbol{\omega}(t_k)$ in the time interval from t_k to $t_{k+1} = t_k + h$, then the resulting rotation is described with the quaternion

$$\exp\left(\frac{h}{2}\boldsymbol{\omega}(t_k)\right) \quad (1423)$$

Then the resulting quaternion at t_{k+1} will be

$$\mathbf{q}(t_{k+1}) := \mathbf{q}(t_k) \circ \exp\left(\frac{h}{2}\boldsymbol{\omega}(t_k)\right) \quad (1424)$$

The resulting quaternion will be a unit quaternion whenever $\mathbf{q}(t_k)$ is a unit quaternion. In this case a renormalization will not be required. This method is described in [70], where different Runge-Kutta methods are applied in the Runge-Kutta-Munthe-Kaas framework.

20.24 The power of a quaternion

The power of a quaternion is given by

$$\mathbf{q}^n = \underbrace{\mathbf{q} \circ \mathbf{q} \circ \dots \circ \mathbf{q}}_{n \text{ times}} \quad (1425)$$

when n is an integer. In the general case where α is a positive real number and $\mathbf{q} = \|\mathbf{q}\| \exp\left(\frac{\theta}{2}\mathbf{k}\right)$, then the power of a quaternion is defined in terms of its logarithm $\mathbf{k}\theta/2$ by

$$\mathbf{q}^\alpha = \|\mathbf{q}\|^\alpha \exp\left(\frac{\alpha\theta}{2}\mathbf{k}\right) \quad (1426)$$

20.25 Interpolation of unit quaternions with SLERP

In this section the interpolation between two unit quaternions $\mathbf{q}_1 = \eta_1 + \boldsymbol{\epsilon}_1$ and $\mathbf{q}_2 = \eta_2 + \boldsymbol{\epsilon}_2$ where $\eta_i = \cos(\theta_i/2)$ and $\boldsymbol{\epsilon}_i = \mathbf{k}_i \sin(\theta_i/2)$ for $i = 1, 2$. Let the unit quaternion

$$\mathbf{q} = \eta + \boldsymbol{\epsilon} = \cos \frac{\theta}{2} + \mathbf{k} \sin \frac{\theta}{2} \quad (1427)$$

be defined as the increment from \mathbf{q}_1 to \mathbf{q}_2 , which is given by

$$\mathbf{q} = \mathbf{q}_1^{-1} \circ \mathbf{q}_2 \quad (1428)$$

which gives $\mathbf{q}_1 \circ \mathbf{q} = \mathbf{q}_2$. Interpolation from \mathbf{q}_1 to \mathbf{q}_2 can then be implemented as

$$\mathbf{q}_{\text{int}}(t) = \mathbf{q}_1 \circ \mathbf{q}^t, \quad 0 \leq t \leq 1 \quad (1429)$$

This interpolation scheme was proposed by Shoemake in 1985 [64], and was called SLERP (spherical linear interpolation). The interpolation can also be written

$$\mathbf{q}_{\text{int}}(t) = \mathbf{q}_1 \circ \exp\left(\frac{t\theta}{2}\mathbf{k}\right) \quad (1430)$$

It is seen that the SLERP interpolation is a rotation from an angle of zero to angle of θ about the constant vector \mathbf{k} .

The alternative formulation

$$\mathbf{q}_{\text{int}}(t) = \frac{\sin \frac{(1-t)\theta}{2}}{\sin \frac{\theta}{2}} \mathbf{q}_1 + \frac{\sin \frac{t\theta}{2}}{\sin \frac{\theta}{2}} \mathbf{q}_2 \quad (1431)$$

of SLERP is often used [64], which avoids the quaternion product. This formulation is derived in the next section.

20.26 Derivation of the SLERP formula

The SLERP formula (1431) and can be derived as follows. The quaternion increment can be written

$$\mathbf{q} = (\eta_1 - \boldsymbol{\epsilon}_1) \circ (\eta_2 + \boldsymbol{\epsilon}_2) = (\eta_1 \eta_2 + \boldsymbol{\epsilon}_1^T \boldsymbol{\epsilon}_2) + (\eta_1 \boldsymbol{\epsilon}_2 + \eta_2 \boldsymbol{\epsilon}_1 + \boldsymbol{\epsilon}_1 \times \boldsymbol{\epsilon}_2) \quad (1432)$$

It is seen that

$$\cos \frac{\theta}{2} = \eta_1 \eta_2 + \boldsymbol{\epsilon}_1^T \boldsymbol{\epsilon}_2 \quad (1433)$$

This is equal to the inner product of \mathbf{q}_1 and \mathbf{q}_2 , which is

$$\mathbf{q}_1 \cdot \mathbf{q}_2 = \eta_1 \eta_2 + \boldsymbol{\epsilon}_1^T \boldsymbol{\epsilon}_2 \quad (1434)$$

It follows that

$$\mathbf{q}_1 \cdot \mathbf{q}_2 = \cos \frac{\theta}{2} \quad (1435)$$

In the same way it is found that

$$\mathbf{q}_1 \cdot \mathbf{q}_{\text{int}}(t) = \cos \frac{t\theta}{2} \quad (1436)$$

$$\mathbf{q}_2 \cdot \mathbf{q}_{\text{int}}(t) = \cos \frac{(1-t)\theta}{2} \quad (1437)$$

The trigonometric identity for the sine of a difference of angles gives

$$\sin \left(\frac{(1-t)\theta}{2} \right) = \cos \frac{t\theta}{2} \sin \frac{\theta}{2} - \cos \frac{\theta}{2} \sin \frac{t\theta}{2} \quad (1438)$$

$$\sin \left(\frac{t\theta}{2} \right) = \cos \frac{(1-t)\theta}{2} \sin \frac{\theta}{2} - \cos \frac{\theta}{2} \sin \frac{(1-t)\theta}{2} \quad (1439)$$

Insertion on the inner products (1435), (1436) and (1437), $\mathbf{q}_1 \cdot \mathbf{q}_1 = 1$ and $\mathbf{q}_2 \cdot \mathbf{q}_2 = 1$ gives

$$\mathbf{q}_1 \cdot \mathbf{q}_1 \sin \left(\frac{(1-t)\theta}{2} \right) = \mathbf{q}_1 \cdot \mathbf{q}_{\text{int}}(t) \sin \frac{\theta}{2} - \mathbf{q}_1 \cdot \mathbf{q}_2 \sin \frac{t\theta}{2} \quad (1440)$$

$$\mathbf{q}_2 \cdot \mathbf{q}_2 \sin \left(\frac{t\theta}{2} \right) = \mathbf{q}_2 \cdot \mathbf{q}_{\text{int}}(t) \sin \frac{\theta}{2} - \mathbf{q}_2 \cdot \mathbf{q}_1 \sin \frac{(1-t)\theta}{2} \quad (1441)$$

Here \mathbf{q}_1 can be factored out in the first equation, and \mathbf{q}_2 can be factored out in the second equation. This gives

$$\mathbf{q}_1 \cdot \mathbf{p} = 0, \quad \mathbf{q}_2 \cdot \mathbf{p} = 0 \quad (1442)$$

where

$$\mathbf{p} = \mathbf{q}_{\text{int}}(t) \sin \frac{\theta}{2} - \mathbf{q}_1 \sin \left(\frac{(1-t)\theta}{2} \right) - \mathbf{q}_2 \sin \frac{t\theta}{2} \quad (1443)$$

This means that the components of \mathbf{p} along \mathbf{q}_1 and \mathbf{q}_2 are zero, and as long as \mathbf{q}_1 and \mathbf{q}_2 are not in the same direction, this implies that $\mathbf{p} = \mathbf{0}$, and therefore

$$\mathbf{q}_{\text{int}}(t) \sin \frac{\theta}{2} = \mathbf{q}_1 \sin \left(\frac{(1-t)\theta}{2} \right) + \mathbf{q}_2 \sin \frac{t\theta}{2} \quad (1444)$$

This gives the usual expression (1431) for SLERP interpolation.

20.27 Composite quaternion in the matrix form

The left multiplication of the composite quaternion $\mathbf{q} = \mathbf{q}_1 \circ \mathbf{q}_2$ can be expressed with the matrix $\mathbf{Q}_L(\mathbf{q}) = \mathbf{Q}_L(\mathbf{q}_1)\mathbf{Q}_L(\mathbf{q}_2)$, which is verified with

$$\mathbf{Q}_L(\mathbf{q}) = \mathbf{Q}_L(\mathbf{q}_1)\mathbf{Q}_L(\mathbf{q}_2) \quad (1445)$$

$$= \begin{bmatrix} \eta_1 & -\boldsymbol{\epsilon}_1^T \\ \boldsymbol{\epsilon}_1 & \eta_1 \mathbf{I} + \boldsymbol{\epsilon}_1^\times \end{bmatrix} \begin{bmatrix} \eta_2 & -\boldsymbol{\epsilon}_2^T \\ \boldsymbol{\epsilon}_2 & \eta_2 \mathbf{I} + \boldsymbol{\epsilon}_2^\times \end{bmatrix} \quad (1446)$$

$$= \begin{bmatrix} \eta_1 \eta_2 - \boldsymbol{\epsilon}_1^T \boldsymbol{\epsilon}_2 & -\eta_1 \boldsymbol{\epsilon}_2^T - \eta_1 \boldsymbol{\epsilon}_2^\times - \boldsymbol{\epsilon}_1^T \boldsymbol{\epsilon}_2^\times \\ \eta_1 \boldsymbol{\epsilon}_2 + \eta_1 \boldsymbol{\epsilon}_2 + \boldsymbol{\epsilon}_1^\times \boldsymbol{\epsilon}_2 & -\boldsymbol{\epsilon}_1 \boldsymbol{\epsilon}_2^T + \eta_1 \eta_2 \mathbf{I} + \eta_1 \boldsymbol{\epsilon}_2^\times + \eta_2 \boldsymbol{\epsilon}_1^\times + \boldsymbol{\epsilon}_1^\times \boldsymbol{\epsilon}_2^\times \end{bmatrix} \quad (1447)$$

$$= \begin{bmatrix} \eta_1 \eta_2 - \boldsymbol{\epsilon}_1^T \boldsymbol{\epsilon}_2 & -(\eta_1 \boldsymbol{\epsilon}_2 + \eta_1 \boldsymbol{\epsilon}_2 + \boldsymbol{\epsilon}_1^\times \boldsymbol{\epsilon}_2)^T \\ \eta_1 \boldsymbol{\epsilon}_2 + \eta_1 \boldsymbol{\epsilon}_2 + \boldsymbol{\epsilon}_1^\times \boldsymbol{\epsilon}_2 & (\eta_1 \eta_2 - \boldsymbol{\epsilon}_1^T \boldsymbol{\epsilon}_2) \mathbf{I} + (\eta_1 \boldsymbol{\epsilon}_2 + \eta_2 \boldsymbol{\epsilon}_1 + \boldsymbol{\epsilon}_1^\times \boldsymbol{\epsilon}_2)^\times \end{bmatrix} \quad (1448)$$

where it is used that $(\boldsymbol{\epsilon}_1^\times \boldsymbol{\epsilon}_2)^T = -(\boldsymbol{\epsilon}_2^\times \boldsymbol{\epsilon}_1)^T = \boldsymbol{\epsilon}_1^T \boldsymbol{\epsilon}_2^\times$, $\boldsymbol{\epsilon}_1 \boldsymbol{\epsilon}_2^T = \boldsymbol{\epsilon}_2^\times \boldsymbol{\epsilon}_1^\times + \boldsymbol{\epsilon}_1^T \boldsymbol{\epsilon}_2 \mathbf{I}$, and $(\boldsymbol{\epsilon}_1^\times \boldsymbol{\epsilon}_2)^\times = \boldsymbol{\epsilon}_1^\times \boldsymbol{\epsilon}_2 - \boldsymbol{\epsilon}_2^\times \boldsymbol{\epsilon}_1$.

20.28 The $\sin \theta \mathbf{k}$ vector

The vector

$$\mathbf{e} = \sin \theta \mathbf{k} \quad (1449)$$

can be written

$$\mathbf{e} = 2\eta \boldsymbol{\epsilon} \quad (1450)$$

which follows from the trigonometric identity $\sin \theta = 2 \sin(\theta/2) \cos(\theta/2)$. The vector \mathbf{e} is found from the rotation matrix $\mathbf{R} = r_{ij}$ as

$$\mathbf{e} = \frac{1}{2} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{13} - r_{31} \end{bmatrix} \quad (1451)$$

This vector has been used in kinematics and control of robot manipulators like in [39]. Let the rotation matrix be written $\mathbf{R} = [\mathbf{n}, \mathbf{s}, \mathbf{a}]$ and let the desired rotation matrix be $\mathbf{R}_d = [\mathbf{n}_d, \mathbf{s}_d, \mathbf{a}_d]$. Let the increment in rotation $\tilde{\mathbf{R}}$ be given by $\tilde{\mathbf{R}} = \mathbf{R} \mathbf{R}_d^T$, so that $\mathbf{R} = \tilde{\mathbf{R}} \mathbf{R}_d$. Then if (\mathbf{k}, θ) are the angle and axis parameters of $\tilde{\mathbf{R}}$ it is found that [20]

$$\mathbf{e} = \frac{1}{2} (\mathbf{n}_d^\times \mathbf{n} + \mathbf{s}_d^\times \mathbf{s} + \mathbf{a}_d^\times \mathbf{a}) \quad (1452)$$

20.29 The Rodrigues vector

The Rodrigues vector

$$\boldsymbol{\rho} = \tan \frac{\theta}{2} \mathbf{k} \quad (1453)$$

can be written in terms of the Euler parameters as

$$\boldsymbol{\rho} = \frac{\boldsymbol{\epsilon}}{\eta} \quad (1454)$$

The Rodrigues vector can be computed from the rotation matrix using

$$\boldsymbol{\rho} = \frac{\boldsymbol{\epsilon}}{\eta} = \frac{2\eta\boldsymbol{\epsilon}}{2\eta^2} = \frac{\boldsymbol{\epsilon}}{2(\text{tr}\mathbf{R} + 1)} = \frac{1}{\text{tr}\mathbf{R} + 1} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{13} - r_{31} \end{bmatrix} \quad (1455)$$

where (1312) is used. The rotation matrix as given by Euler parameters i (1310) can be written in terms of the Rodrigues parameters. First an expression for η^2 is found from

$$1 = \eta^2 + \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} = \eta^2 (1 + \boldsymbol{\rho}^T \boldsymbol{\rho}) \quad (1456)$$

which gives

$$\eta^2 = \frac{1}{1 + \boldsymbol{\rho}^T \boldsymbol{\rho}} \quad (1457)$$

The rotation matrix is then found from (1310) to be

$$\mathbf{R} = \mathbf{I} + \frac{2}{1 + \boldsymbol{\rho}^T \boldsymbol{\rho}} (\boldsymbol{\rho}^\times + \boldsymbol{\rho}^\times \boldsymbol{\rho}^\times) \quad (1458)$$

From this result the Cayley transformation can be derived using

$$\mathbf{R}(\mathbf{I} - \boldsymbol{\rho}^\times) = \left[\mathbf{I} + \frac{2}{1 + \boldsymbol{\rho}^T \boldsymbol{\rho}} (\boldsymbol{\rho}^\times + \boldsymbol{\rho}^\times \boldsymbol{\rho}^\times) \right] (\mathbf{I} - \boldsymbol{\rho}^\times) \quad (1459)$$

$$= \mathbf{I} - \boldsymbol{\rho}^\times + \frac{2}{1 + \boldsymbol{\rho}^T \boldsymbol{\rho}} (\boldsymbol{\rho}^\times - \boldsymbol{\rho}^\times \boldsymbol{\rho}^\times \boldsymbol{\rho}^\times) \quad (1460)$$

$$= \mathbf{I} - \boldsymbol{\rho}^\times + \frac{2\boldsymbol{\rho}^\times}{1 + \boldsymbol{\rho}^T \boldsymbol{\rho}} (1 + \boldsymbol{\rho}^T \boldsymbol{\rho}) \quad (1461)$$

$$= \mathbf{I} + \boldsymbol{\rho}^\times \quad (1462)$$

where it is used that

$$\boldsymbol{\rho}^\times \boldsymbol{\rho}^\times \boldsymbol{\rho}^\times = \boldsymbol{\rho}^\times [\boldsymbol{\rho} \boldsymbol{\rho}^T - (\boldsymbol{\rho}^T \boldsymbol{\rho}) \mathbf{I}] = -(\boldsymbol{\rho}^T \boldsymbol{\rho}) \boldsymbol{\rho}^\times \quad (1463)$$

This leads to the Cayley transformation for the rotation matrix, which is written

$$\mathbf{R} = (\mathbf{I} + \boldsymbol{\rho}^\times)(\mathbf{I} - \boldsymbol{\rho}^\times)^{-1} \quad (1464)$$

The Cayley transformation maps a vector \mathbf{u} to a rotation matrix according to

$$\text{Cay}(\mathbf{u}) = (\mathbf{I} + \mathbf{u}^\times)(\mathbf{I} - \mathbf{u}^\times)^{-1} \in SO(3) \quad (1465)$$

It is seen that

$$\mathbf{R} = \text{Cay}(\boldsymbol{\rho}) \quad (1466)$$

It follows that for small θ , the following approximation is valid:

$$\text{Cay}\left(\frac{\theta \mathbf{k}}{2}\right) \approx \mathbf{R} \quad \text{when } \theta \text{ is small} \quad (1467)$$

20.30 The modified Rodrigues vector

The modified Rodrigues parameters

$$\boldsymbol{\mu} = \tan \frac{\theta}{4} \mathbf{k} \quad (1468)$$

can be written

$$\boldsymbol{\mu} = \frac{\boldsymbol{\epsilon}}{1 + \eta} \quad (1469)$$

which is verified by the computation

$$\frac{\sin x}{1 + \cos x} = \frac{2 \sin \frac{x}{2} \cos \frac{x}{2}}{2 \cos^2 \frac{x}{2}} = \frac{\sin \frac{x}{2}}{\cos \frac{x}{2}} \quad (1470)$$

20.31 Kinematic differential equations for the Rodrigues vector

The kinematic differential equation for the Rodrigues vector is found from

$$\dot{\boldsymbol{\rho}} = \frac{d}{dt} \frac{\boldsymbol{\epsilon}}{\eta} = \frac{\eta \dot{\boldsymbol{\epsilon}} - \dot{\eta} \boldsymbol{\epsilon}}{\eta^2} \quad (1471)$$

This gives

$$\dot{\boldsymbol{\rho}} = \frac{1}{2} (I - \boldsymbol{\rho}^\times + \boldsymbol{\rho} \boldsymbol{\rho}^T) \boldsymbol{\omega}^a \quad (1472)$$

$$\dot{\boldsymbol{\rho}} = \frac{1}{2} (I + \boldsymbol{\rho}^\times + \boldsymbol{\rho} \boldsymbol{\rho}^T) \boldsymbol{\omega}^b \quad (1473)$$

It is noted that

$$\frac{d}{dt} \boldsymbol{\rho} \Big|_{\boldsymbol{\rho}=0} = \frac{1}{2} \boldsymbol{\omega}^a = \frac{1}{2} \boldsymbol{\omega}^b \quad (1474)$$

The angular velocity is found from

$$\boldsymbol{\omega}^a = 2\eta^2 \left(\frac{\eta \dot{\boldsymbol{\epsilon}} - \dot{\eta} \boldsymbol{\epsilon}}{\eta^2} + \frac{\boldsymbol{\epsilon}^\times \dot{\boldsymbol{\epsilon}}}{\eta^2} \right) \quad (1475)$$

20.32 Kinematic differential equations for the modified Rodrigues parameters

The Rodrigues parameters are given by

$$\boldsymbol{\rho} = \tan \frac{\theta}{2} \mathbf{k} \quad (1476)$$

The modified Rodrigues parameters are given by

$$\boldsymbol{\mu} = \tan \frac{\theta}{4} \mathbf{k} \quad (1477)$$

It is straightforward to show that

$$\boldsymbol{\mu} = \frac{\sin \frac{\theta}{4}}{\cos \frac{\theta}{4}} \mathbf{k} = \frac{2 \sin \frac{\theta}{4} \cos \frac{\theta}{4}}{2 \cos^2 \frac{\theta}{4}} \mathbf{k} = \frac{\sin \frac{\theta}{2} \mathbf{k}}{1 + \cos \frac{\theta}{2}} = \frac{\boldsymbol{\epsilon}}{1 + \eta} \quad (1478)$$

The kinematic differential equation is found from

$$\dot{\mu} = \frac{\dot{\epsilon}}{1+\eta} - \frac{\dot{\eta}\epsilon}{(1+\eta)^2} \quad (1479)$$

$$= \frac{\eta\omega + \epsilon^\times\omega}{1+\eta} + \frac{(\epsilon^T\omega)\epsilon}{(1+\eta)^2} \quad (1480)$$

$$= \frac{1}{2} \left(\frac{(1+\eta)(\eta\omega + \epsilon^\times) + \epsilon\epsilon^T}{(1+\eta)^2} \right) \omega \quad (1481)$$

$$= \frac{1}{2} \left(\frac{\eta\mathbf{I} + \epsilon^\times + \eta^2\mathbf{I} + \eta\epsilon^\times + \epsilon\epsilon^T}{(1+\eta)^2} \right) \omega \quad (1482)$$

$$= \frac{1}{2} \left(\frac{\eta\mathbf{I} + \eta^2\mathbf{I}}{(1+\eta)^2} + \frac{\epsilon^\times}{1+\eta} + \frac{\epsilon\epsilon^T}{(1+\eta)^2} \right) \omega \quad (1483)$$

$$= \frac{1}{4} ((1 - \mathbf{p}^T\mathbf{p})\mathbf{I} + 2\mathbf{p}^\times + 2\mathbf{p}\mathbf{p}^T) \omega \quad (1484)$$

$$= \frac{1}{4} ((1 + \mathbf{p}^T\mathbf{p})\mathbf{I} + 2\mathbf{p}^\times + 2\mathbf{p}^\times\mathbf{p}^\times) \omega \quad (1485)$$

where the second last step follows from $2(\eta + \eta^2) = 2\eta + 2\eta^2 + 1 - (\eta^2 + \epsilon^T\epsilon) = (1 + \eta)^2 - \epsilon^T\epsilon$.

It is noted that

$$\frac{d}{dt}\mu \Big|_{\mu=0} = \frac{1}{4}\omega^a = \frac{1}{4}\omega^b \quad (1486)$$

20.33 Kinematic differential equations for the $\mathbf{k} \sin \theta$ vector

The vector $\mathbf{e} = \mathbf{k} \sin \theta$ satisfies $\mathbf{e} = 2\eta\epsilon$. The kinematic differential equation is therefore found from $\dot{\mathbf{e}} = 2(\dot{\eta}\epsilon + \eta\dot{\epsilon})$, which gives

$$\dot{\mathbf{e}} = (\eta^2\mathbf{I} - \epsilon\epsilon^T - \eta\epsilon^\times)\omega^a \quad (1487)$$

$$= (\eta^2\mathbf{I} - \epsilon\epsilon^T + \eta\epsilon^\times)\omega^b \quad (1488)$$

It is noted that $\epsilon = \mathbf{0}$ implies $\mathbf{e} = \mathbf{0}$ and $\theta = 0$, which gives

$$\frac{d}{dt}\mathbf{e} \Big|_{\epsilon=0} = \omega^a = \omega^b \quad (1489)$$

21 Exponential description of rotations

21.1 Exponential description of rotation matrices from angle and axis

A rotation matrix can be described by the Rodrigues equation as a rotation by an angle θ about a unit vector \mathbf{k} . The rotation matrix is then

$$\mathbf{R} = \mathbf{I} + \sin \theta \mathbf{k}^\times + (1 - \cos \theta) \mathbf{k}^\times \mathbf{k}^\times \quad (1490)$$

where \mathbf{k}^\times is the skew-symmetric form of \mathbf{k} .

The rotation matrix defined by the rotation θ about \mathbf{k} can also be described by the exponential function. This is done with the matrix exponential function, which for a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is defined by

$$\exp(\mathbf{A}) \triangleq \mathbf{I} + \mathbf{A} + \frac{1}{2!} \mathbf{A}^2 + \frac{1}{3!} \mathbf{A}^3 + \dots = \sum_{i=0}^{\infty} \frac{1}{i!} \mathbf{A}^i \quad (1491)$$

where it is used that $0! = 1$.

Consider the matrix $\mathbf{u}^\times = \theta \mathbf{k}^\times \in \mathbf{R}^{3 \times 3}$. It follows from the definition of the matrix exponential function that $\exp(\theta \mathbf{k}^\times)$ is given by

$$\exp(\theta \mathbf{k}^\times) = \mathbf{I} + \theta (\mathbf{k}^\times) + \frac{\theta^2}{2!} (\mathbf{k}^\times)^2 + \frac{\theta^3}{3!} (\mathbf{k}^\times)^3 + \frac{\theta^4}{4!} (\mathbf{k}^\times)^4 \dots \quad (1492)$$

This series is simplified using

$$(\mathbf{k}^\times)^3 = (\mathbf{k}^\times)^2 \mathbf{k}^\times = (\mathbf{k} \mathbf{k}^T - \mathbf{I}) \mathbf{k}^\times = -\mathbf{k}^\times \quad (1493)$$

where it is used that $\mathbf{k} \mathbf{k}^T \mathbf{k}^\times = \mathbf{k} (\mathbf{k}^T \mathbf{k}^\times) = \mathbf{k} (-\mathbf{k}^\times \mathbf{k})^T = \mathbf{0}$. This gives

$$\begin{aligned} \exp(\theta \mathbf{k}^\times) &= \mathbf{I} + \underbrace{\left(\theta - \frac{\theta^3}{3!} + \dots \right)}_{\sin \theta} \mathbf{k}^\times + \underbrace{\left(\frac{\theta^2}{2!} - \frac{\theta^4}{4!} + \dots \right)}_{1 - \cos \theta} \mathbf{k}^\times \mathbf{k}^\times \\ &= \mathbf{I} + \sin \theta \mathbf{k}^\times + (1 - \cos \theta) \mathbf{k}^\times \mathbf{k}^\times \end{aligned} \quad (1494)$$

$$= \mathbf{I} + \sin \theta \mathbf{k}^\times + (1 - \cos \theta) \mathbf{k}^\times \mathbf{k}^\times \quad (1495)$$

which is seen to be equal to (1490). It is concluded that

$$\mathbf{R} = \exp(\theta \mathbf{k}^\times) \quad (1496)$$

This means that

$$\mathbf{R} = \exp(\mathbf{u}^\times) \quad (1497)$$

where $\mathbf{u}^\times = \theta \mathbf{k}^\times$, and $\mathbf{u} = \theta \mathbf{k}$. The matrix $\hat{\mathbf{u}}$ is accordingly called the logarithm of the rotation matrix, that is,

$$\log(\mathbf{R}) = \mathbf{u}^\times = \theta \mathbf{k}^\times \quad (1498)$$

The transpose of the rotation matrix is

$$\mathbf{R}^T = \mathbf{I} - \sin \theta \mathbf{k}^\times + (1 - \cos \theta) \mathbf{k}^\times \mathbf{k}^\times = \exp(-\theta \mathbf{k}^\times) \quad (1499)$$

where it is used that $(\mathbf{k}^\times)^T = -\mathbf{k}^\times$. This gives

$$\log(\mathbf{R}^T) = -\log(\mathbf{R}) \quad (1500)$$

Note that when the rotation is $\theta = \pi$, the exponential gives

$$\mathbf{R} = \exp(\pi \mathbf{k}^\times) = \mathbf{I} + 2 \mathbf{k}^\times \mathbf{k}^\times = 2 \mathbf{k} \mathbf{k}^T - \mathbf{I} \quad (1501)$$

This is symmetric matrix, and it follows that $\exp(\pi \mathbf{k}^\times) = \exp(-\pi \mathbf{k}^\times)$.

For completeness it is mentioned that the series expansion of the matrix logarithm of a matrix \mathbf{U} is

$$\log(\mathbf{U}) = \log(\mathbf{I} + (\mathbf{U} - \mathbf{I})) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{(\mathbf{U} - \mathbf{I})^k}{k} \quad (1502)$$

which is the Taylor series about the identity matrix.

21.2 Rotation of the logarithm

Let the logarithm of a rotation matrix \mathbf{R} be given by $\log(\mathbf{R}) = \mathbf{u}^\times$. Consider the rotation of the vector form \mathbf{u} of the logarithm by a rotation matrix \mathbf{R}_0 to the vector $\mathbf{R}_0\mathbf{u}$. Then the skew symmetric form of $\mathbf{R}_0\mathbf{u}$ is

$$(\mathbf{R}_0\mathbf{u})^\times = \mathbf{R}_0\mathbf{u}^\times\mathbf{R}_0^T \quad (1503)$$

It is straightforward to verify that this matrix is skew symmetric by comparing the matrix with its transpose. The exponential of this skew symmetric matrix is

$$\exp(\mathbf{R}_0\mathbf{u}^\times\mathbf{R}_0^T) = \mathbf{R}_0 \exp(\mathbf{u}^\times)\mathbf{R}_0^T \quad (1504)$$

This is verified from the series

$$\exp(\mathbf{R}_0\mathbf{u}^\times\mathbf{R}_0^T) = \mathbf{I} + \mathbf{R}_0\mathbf{u}^\times\mathbf{R}_0^T + \frac{1}{2}\mathbf{R}_0\mathbf{u}^\times\mathbf{R}_0^T\mathbf{R}_0\mathbf{u}^\times\mathbf{R}_0^T + \dots \quad (1505)$$

$$= \mathbf{R}_0 \left(\mathbf{I} + \mathbf{u}^\times + \frac{1}{2}\mathbf{u}^\times\mathbf{u}^\times + \dots \right) \mathbf{R}_0^T \quad (1506)$$

$$= \mathbf{R}_0 \exp(\mathbf{u}^\times)\mathbf{R}_0^T \quad (1507)$$

21.3 The norm of the logarithm

The logarithm is the skew symmetric matrix $\theta\mathbf{k}^\times$. The corresponding vector form of the logarithm is $\theta\mathbf{k}$. The norm of the vector form of the logarithm is

$$\|\theta\mathbf{k}\|^2 = (\theta\mathbf{k})^T(\theta\mathbf{k}) = \theta^2\mathbf{k}^T\mathbf{k} = \theta^2 \quad (1508)$$

The Frobenius norm of the skew symmetric form of the unit vector \mathbf{k} is

$$\|\mathbf{k}^\times\|_F^2 = \text{tr}(\mathbf{k}^{\times T}\mathbf{k}^\times) = -\text{tr}(\mathbf{k}^\times\mathbf{k}^\times) = \text{tr}(\mathbf{k}^T\mathbf{k}\mathbf{I} - \mathbf{k}\mathbf{k}^T) = 2 \quad (1509)$$

The Frobenius norm of the logarithm is therefore given by

$$\|\theta\mathbf{k}^\times\|_F^2 = \theta^2\|\mathbf{k}^\times\|_F^2 = 2\theta^2 \quad (1510)$$

21.4 Computation of the exponential the rotation matrix

In optimization and interpolation methods it is necessary to compute the rotation matrix \mathbf{R} as the exponential of a general logarithm \mathbf{u}^\times as

$$\mathbf{R} = \exp(\mathbf{u}^\times) \quad (1511)$$

Note that in this case, \mathbf{u}^\times is given, while θ and \mathbf{k}^\times are unknown. This can be the case in time integration schemes, and in optimization where \mathbf{u} is the increment. In general, it is not recommended to used the series expansion (1491) for computation.

Instead it is used that the angle θ and axis of the rotation \mathbf{k} corresponding corresponding to a logarithm \mathbf{u}^\times will satisfy $\mathbf{u} = \theta\mathbf{k}$, and it follows that

$$\theta = \|\mathbf{u}\|, \quad \mathbf{k} = \frac{\mathbf{u}}{\|\mathbf{u}\|} \quad (1512)$$

Note that this means that \mathbf{k} is undefined for $\mathbf{u} = \mathbf{0}$, which must be handled in some way. Insertion of the expressions from (1512) into the Rodrigues equation (1490) gives

$$\exp(\mathbf{u}^\times) = \mathbf{I} + \sin \|\mathbf{u}\| \frac{\mathbf{u}^\times}{\|\mathbf{u}\|} + (1 - \cos \|\mathbf{u}\|) \frac{\mathbf{u}^\times}{\|\mathbf{u}\|} \frac{\mathbf{u}^\times}{\|\mathbf{u}\|} \quad (1513)$$

Here it is a potential problem that $\|\mathbf{u}\|$ tend to zero, and division by zero in the $\mathbf{u}^\times/\|\mathbf{u}\|$ term must be avoided in numerical solutions. The expression is therefore written as

$$\exp(\mathbf{u}^\times) = \mathbf{I} + \text{sinc}(\|\mathbf{u}\|) \mathbf{u}^\times + \frac{1 - \cos \|\mathbf{u}\|}{\|\mathbf{u}\|^2} \mathbf{u}^\times \mathbf{u}^\times \quad (1514)$$

where

$$\text{sinc}(x) \triangleq \begin{cases} \sin(x)/x & x \neq 0 \\ 1 & x = 0 \end{cases} \quad (1515)$$

This is further developed by noting that $1 - \cos \theta = 2 \sin^2(\theta/2)$, which gives the expression that is recommended for computation:

$$\exp(\mathbf{u}^\times) = \mathbf{I} + \text{sinc}(\|\mathbf{u}\|) \mathbf{u}^\times + \frac{1}{2} \text{sinc}^2\left(\frac{\|\mathbf{u}\|}{2}\right) \mathbf{u}^\times \mathbf{u}^\times \quad (1516)$$

Here the function $\text{sinc}(x)$ is continuous, differentiable and numerically well behaved, which is seen from its Taylor series expansion

$$\text{sinc}(x) = \frac{\sin x}{x} = \frac{x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 + \dots}{x} = 1 - \frac{1}{3!}x^2 + \frac{1}{5!}x^4 + \dots \quad (1517)$$

Note in particular that $\text{sinc}(x)$ is well behaved at $x = 0$.

Note that the library function `sinc` in MATLAB calculates $\sin(\pi x)/(\pi x)$. If a library function for $\text{sinc}(x)$ is not available, it can be calculated as

$$\text{sinc}(x) = \begin{cases} \frac{\sin(x)}{x}, & |x| \geq \delta \\ 1 - \frac{x^2}{6}, & |x| < \delta \end{cases} \quad (1518)$$

where δ is selected so that the error term $(1/5!) \delta^4$ is less than the machine precision, which is the smallest number ϵ such that the difference between 1 and $1 + \epsilon$ is zero in the computation. In MATLAB the usual machine precision is 10^{-16} . Then $\delta = (120 \cdot 10^{-16})^{1/4} = 0.00033$ will give a results within machine precision.

21.5 Computation of the logarithm the rotation matrix for $|\theta| < \pi$

The logarithm is calculated from $\text{tr}(\mathbf{R}) = 1 + 2 \cos \theta$, $\mathbf{R} - \mathbf{R}^T = 2 \sin \theta \mathbf{k}^\times$ and $\log(\mathbf{R}) = \theta \mathbf{k}^\times$, which gives

$$\mathbf{k}^\times = \frac{1}{2 \sin \theta} (\mathbf{R} - \mathbf{R}^T) \quad (1519)$$

The logarithm can then be computed from [51]

$$\theta = \arccos \frac{\text{tr} \mathbf{R} - 1}{2}, \quad 0 \leq \theta \leq \pi \quad (1520)$$

and

$$\log(\mathbf{R}) = \begin{cases} \mathbf{0}, & \theta = 0 \\ \frac{\theta}{2\sin\theta}(\mathbf{R} - \mathbf{R}^T), & 0 < \theta < \pi \end{cases} \quad (1521)$$

To handle the potential problem of calculating $\sin \theta$ when θ approaches zero, the series expansion

$$\frac{x}{\sin x} = x \csc(x) = 1 + \frac{1}{6}x^2 + \frac{7}{360}x^4 + \dots, \quad 0 \leq |x| < \pi \quad (1522)$$

can be used. This gives

$$\log(\mathbf{R}) = \begin{cases} 0.5 \left(1 + \frac{1}{6}\theta^2\right) (\mathbf{R} - \mathbf{R}^T), & \theta < \delta \\ \frac{\theta}{2\sin\theta}(\mathbf{R} - \mathbf{R}^T), & \delta < \theta < \pi - \delta \end{cases} \quad (1523)$$

Example

```
# Python script for the computation of the logarithm for theta < pi
import numpy as np
def skewm(r):
    return np.array([[0, -r[2], r[1]], [r[2], 0, -r[0]], [-r[1], r[0], 0]])
def vex(u):
    return np.array([u[2,1], u[0,2], u[1,0]])
def expS03(u):
    return np.identity(3) + np.sinc(np.linalg.norm(u)/np.pi)*skewm(u) \
        + 0.5*(np.sinc(np.linalg.norm(u)/(2*np.pi)))**2*skewm(u)@skewm(u)
def logS03(R):
    # The vector form of the logarithm in S0(3) for |theta| < pi
    theta = np.arccos(0.5 * (np.trace(R) - 1)) # np.arccos returns in [0,pi]
    if theta < 0.000001:
        f = 1 + (theta**2)/6
    else: # 0.000001 < theta < np.pi - 0.00001
        f = theta / np.sin(theta)
    return f * 0.5 * vex((R-R.T))
# Test
ex = np.array([1, 0, 0]); ey = np.array([0, 1, 0]); ez = np.array([0, 0, 1]);
k = (ex+ey+ez); k = k/np.linalg.norm(k)
R= expS03(k*2*np.pi/3)
#R = np.zeros([3,3]); R[0,0]=-1; R[1,1]=-1; R[2,2]=1
L = logS03(R)
Rc = expS03(L)

np.set_printoptions(formatter={'float': '{: 0.4f}'.format})
print('\n R =\n {} \n L =\n {} \n norm(L)/pi =\n {} \n
Rc =\n {} \n log(R.T@Rc) = \n{}'.
format(R, L, np.linalg.norm(L)/np.pi, Rc, logS03(R.T@Rc)))
```

21.6 Computation of the logarithm the rotation matrix for $|\theta| \leq \pi$

An extension of this solution is to include the case where $|\theta| = \pi$, which implies that $\mathbf{R} - \mathbf{R}^T = \mathbf{0}$. The angle θ can still be computed from (1520), while some other method must be used to

compute \mathbf{k} . A solution for \mathbf{k} at $|\theta| = \pi$ is given in [40], but this involves a check if $|\theta| = \pi$, and it is not suited for numerical computations of \mathbf{k} when $|\theta|$ is close to π since $\sin \theta$ tends to zero and $\mathbf{R} - \mathbf{R}^T$ tends to the zero matrix.

In the following the method of [40] is extended to include computation when $|\theta|$ is close to and equal to π . A solution can be found from the elements of the rotation matrix written in the form

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} c_\theta + k_1^2 v_\theta & k_1 k_2 v_\theta - k_3 s_\theta & k_1 k_3 v_\theta + k_2 s_\theta \\ k_2 k_1 v_\theta + k_3 s_\theta & c_\theta + k_2^2 v_\theta & k_2 k_3 v_\theta - k_1 s_\theta \\ k_3 k_1 v_\theta - k_2 s_\theta & k_3 k_2 v_\theta + k_1 s_\theta & c_\theta + k_3^2 v_\theta \end{bmatrix} \quad (1524)$$

where $s_\theta = \sin \theta$, $c_\theta = \cos \theta$ and $v_\theta = 1 - c_\theta$. It is noted that when $\pi - |\theta|$ is small, then $c_\theta \approx -1$ and $v_\theta \approx 2$. The diagonal gives

$$k_i^2 v_\theta + c_\theta = r_{ii} \quad (1525)$$

while the off-diagonal elements gives

$$2k_1 k_2 v_\theta = r_{12} + r_{21} \quad (1526)$$

$$2k_2 k_3 v_\theta = r_{23} + r_{32} \quad (1527)$$

$$2k_3 k_1 v_\theta = r_{31} + r_{13} \quad (1528)$$

This means that k_1 can be computed from the diagonal elements as

$$k_1 = \sqrt{\frac{r_{11} - c_\theta}{v_\theta}} \quad (1529)$$

Then, if k_1 has been computed, and if it can be ensured that k_1 is not close to zero, then k_2 and k_3 can be found from the off-diagonal elements as

$$k_2 = \frac{r_{12} + r_{21}}{2k_1 v_\theta}, \quad k_3 = \frac{r_{31} + r_{13}}{2k_1 v_\theta} \quad (1530)$$

In general, it is not known if k_1 is nonzero, and if it is sufficiently far from zero. However, at least one of the elements k_1 , k_2 and k_3 must be have a magnitude greater than or equal to $\sqrt{1/3} = 0.5774$ since $\mathbf{k} = [k_1, k_2, k_3]^T$ is a unit vector. Moreover, since v_θ is close to 2 for $|\theta|$ close to π , it can be checked if k_i is sufficiently far from zero by checking $r_{ii} - c_\theta$. The trace is close to minus one, and c_θ is close to minus 1, which gives

$$(r_{11} - c_\theta) + (r_{22} - c_\theta) + (r_{33} - c_\theta) = \text{tr}(\mathbf{R}) - 3 \cos \theta \approx -1 + 3 = 2 \quad (1531)$$

Since $|r_{ii}| \leq 1$ it follows that $0 \leq r_{ii} - c_\theta \leq 2$ for $|\theta| = \pi$, and it follows that at least one of the terms will satisfy $r_{ii} - c_\theta \geq 2/3$. The solution is then to check which of the terms $r_{ii} - c_\theta$ is largest, or, alternatively, find a term $r_{ii} - c_\theta$ larger than, e.g., 0.5, then compute k_i , and finally the remaining elements for the off-diagonal equations.

This gives the following algorithm for $|\theta| > \pi - \delta$:

If $r_{11} - c_\theta \geq 0.5$

$$k_1 = \sqrt{\frac{r_{11} - c_\theta}{v_\theta}}, \quad k_2 = \frac{r_{12} + r_{21}}{2k_1 v_\theta}, \quad k_3 = \frac{r_{31} + r_{13}}{2k_1 v_\theta} \quad (1532)$$

Else if $r_{22} - c_\theta \geq 0.5$ then

$$k_2 = \sqrt{\frac{r_{22} - c_\theta}{v_\theta}}, \quad k_3 = \frac{r_{23} + r_{32}}{2k_2 v_\theta}, \quad k_1 = \frac{r_{12} + r_{21}}{2k_2 v_\theta} \quad (1533)$$

Else (in this case $r_{33} - c_\theta \geq 0.5$)

$$k_3 = \sqrt{\frac{r_{33} - c_\theta}{v_\theta}}, \quad k_1 = \frac{r_{31} + r_{13}}{2k_3 v_\theta}, \quad k_2 = \frac{r_{23} + r_{32}}{2k_3 v_\theta} \quad (1534)$$

Remark

Alternative expressions for the computation of the logarithm are given by [33]

$$\log(\mathbf{R}) = \frac{\arcsin(\|\mathbf{e}\|)}{\|\mathbf{e}\|} \mathbf{e}^\times, \quad \mathbf{e}^\times = \frac{1}{2}(\mathbf{R} - \mathbf{R}^T) \quad (1535)$$

A drawback with this approach is that it is only valid for $|\theta| < \pi/2$. For small $\|\mathbf{e}\|$ the series expansion

$$\frac{\arcsin(x)}{x} = 1 + \frac{x^2}{6} + \frac{3x^4}{40} + \dots \quad (1536)$$

is used, and the solution is calculated using $\frac{\arcsin(x)}{x} = 1 + \frac{x^2}{6}$ when $x \leq \delta$, where $3\delta^4/40$ is less than the machine precision.

Example

```
# Python script for the computation of the logarithm and the exponential
import numpy as np

def skewm(r):
    return np.array([[0, -r[2], r[1]], [r[2], 0, -r[0]], [-r[1], r[0], 0]])
def vex(u):
    return np.array([u[2,1], u[0,2], u[1,0]])
def expSO3(u):
    return np.identity(3) + np.sinc(np.linalg.norm(u)/np.pi)*skewm(u) \
        + 0.5*(np.sinc(np.linalg.norm(u)/(2*np.pi)))**2*skewm(u)@skewm(u)
def logSO3(R):
    # The vector form of the logarithm in SO(3)
    theta = np.arccos(0.5 * (np.trace(R) - 1)) # np.arccos returns in [0,pi]
    if theta < 0.000001:
        f = 1 + (theta**2)/6
        u = f * 0.5 * vex((R-R.T))
    elif theta > np.pi - 0.00001:
        ct = np.cos(theta); vt = 1-ct
        if R[0,0] - ct > 0.5:
            kx = np.sqrt((R[0,0] - ct)/vt)
            ky = (R[0,1] + R[1,0]) / (2*kx*vt)
            kz = (R[2,0] + R[0,2]) / (2*kx*vt)
        elif R[1,1] - ct > 0.5:
```

```

        ky = np.sqrt((R[1,1] - ct)/vt)
        kz = (R[1,2] + R[2,1]) / (2*ky*vt)
        kx = (R[0,1] + R[1,0]) / (2*ky*vt)
    else:
        kz = np.sqrt((R[2,2] - ct)/vt)
        kx = (R[2,0] + R[0,2]) / (2*kz*vt)
        ky = (R[1,2] + R[2,1]) / (2*kz*vt)
        u = theta * np.array([kx, ky, kz])
    else: # 0.000001 < theta < np.pi - 0.00001
        f = theta / np.sin(theta)
        u = f * 0.5 * vex((R-R.T))
    return u

# Test
ex = np.array([1, 0, 0]); ey = np.array([0, 1, 0]); ez = np.array([0, 0, 1]);
k = (ex + ey); k = k/np.linalg.norm(k)
R= expSO3(k*np.pi)
#R = np.zeros([3,3]); R[0,0]=-1; R[1,1]=-1; R[2,2]=1
L = logSO3(R)
Rc = expSO3(L)

np.set_printoptions(formatter={'float': '{: 0.4f}'.format})
print('R =\n {} \n L =\n {} \n norm(L)/pi =\n {} \n Rc =\n {} \n log(R.T@Rc) = \n {}'.
      format(R, L, np.linalg.norm(L)/np.pi, Rc, logSO3(R.T@Rc)))

```

21.7 Position and velocity of a point mass

Consider a point mass m with position $\mathbf{p} \in \mathbb{R}^3$. The velocity of the point mass will then be $\mathbf{v} = (\mathrm{d}/\mathrm{d}t)\mathbf{p} \in \mathbb{R}^3$. If the point mass moves along a trajectory $\mathbf{p}(t)$, then the velocity at time t_1 will be $\mathbf{v}(t_1) = (\mathrm{d}/\mathrm{d}t)\mathbf{p}(t_1)$, which is tangent to the trajectory at the position $\mathbf{p}(t_1)$. Suppose that the mass point moves along a trajectory from $\mathbf{p}(t_1)$ to $\mathbf{p}(t_2)$. Then the distance $d(\mathbf{p}(t_1), \mathbf{p}(t_2))$ along the trajectory from $\mathbf{p}(t_1)$ to $\mathbf{p}(t_2)$ can be calculated from a Riemannian metric

$$\langle \mathbf{v}, \mathbf{v} \rangle = \mathbf{v}^T \mathbf{v} = \|\mathbf{v}\|^2 \quad (1537)$$

so that

$$d(\mathbf{p}(t_1), \mathbf{p}(t_2)) = \int_{t_1}^{t_2} \langle \mathbf{v}(t), \mathbf{v}(t) \rangle^{\frac{1}{2}} \mathrm{d}t = \int_{t_1}^{t_2} \|\mathbf{v}(t)\| \mathrm{d}t \quad (1538)$$

In the following this will be extended to $SO(3)$, where the geometry is more complicated.

21.8 Rotation and angular velocity in $SO(3)$

Rotation is described with a rotation matrix $\mathbf{R} \in SO(3)$. In the description of the time derivatives of the rotation matrix the concept of a tangent plane is useful, as the angular velocity is defined in the tangent plane. Also the logarithm of the rotation matrix is defined in the tangent plane. Moreover, in optimization problems the gradient of the rotation matrix is useful, and the definition of a gradient is done in the tangent plane. In the following the

tangent plane of $SO(3)$ is defined in terms of the angular velocity, and then it is shown how the logarithm and the gradient can be described in the same setting.

Consider a rotation matrix $\mathbf{R} = \mathbf{R}_b^s \in SO(3)$ from the spatial frame s to the body frame b . The time derivative of the rotation matrix is

$$\dot{\mathbf{R}} = \mathbf{R}\boldsymbol{\omega}_b^\times \in T_R SO(3) \quad (1539)$$

where $T_R SO(3)$ is the tangent space of $SO(3)$ at \mathbf{R} . Here $\boldsymbol{\omega}_b^\times = \mathbf{R}^T \dot{\mathbf{R}}$ is the right velocity, which is the skew symmetric form of the angular velocity $\boldsymbol{\omega}_b$ in body coordinates. The time derivative of the rotation matrix can alternatively be written

$$\dot{\mathbf{R}} = \boldsymbol{\omega}_s^\times \mathbf{R} \in T_R SO(3) \quad (1540)$$

where $\boldsymbol{\omega}_s^\times = \dot{\mathbf{R}} \mathbf{R}^T$ is the left velocity, which is the skew symmetric form of the angular velocity $\boldsymbol{\omega}_s$ in spatial coordinates. It follows that $\mathbf{R}\boldsymbol{\omega}_b^\times = \boldsymbol{\omega}_s^\times \mathbf{R}$, which gives

$$\boldsymbol{\omega}_s^\times = \mathbf{R}\boldsymbol{\omega}_b^\times \mathbf{R}^T \quad (1541)$$

The tangent space at the identity $\mathbf{R} = \mathbf{I}$ is defined as $so(3) \stackrel{\triangle}{=} T_I SO(3)$. The time derivative of the rotation matrix at the identity is

$$\dot{\mathbf{R}}|_{\mathbf{R}=\mathbf{I}} = \boldsymbol{\omega}_b^\times = \boldsymbol{\omega}_s^\times \in so(3) \quad (1542)$$

which means that $so(3)$ is the set of skew symmetric matrices. The tangent plane at the identity is of special importance, and is called the Lie algebra $so(3) = T_I SO(3)$ of the Lie group $SO(3)$. The Lie algebra is defined as

$$so(3) = \{\mathbf{u}^\times | \mathbf{u} \in \mathbb{R}^3\} \quad (1543)$$

which is the set of all skew symmetric matrices of dimension 3×3 .

21.9 The kinematic differential equation of the logarithm

In this section the kinematic differential equations for the logarithm is derived. This is useful in applications like numerical integration, Gauss-Newton and gradient search methods, estimation problems using Kalman filters, and in control. The resulting kinematic differential equations involves the left and right Jacobians of $SO(3)$. Early work on this is found in [51], where the topic is the optimization of mechanisms, and in [13] where the topic is feedback control. The derivation of the expressions for the Jacobians is quite involved, and the basis for this is derived for general matrix Lie groups in textbooks on Lie group theory, like [28, 16], while results for $SO(3)$ and to some extent $SE(3)$ is found in [51] and [13].

Consider a rotation described by the rotation matrix

$$\mathbf{R}(t) = \exp(\mathbf{u}(t)^\times) \quad (1544)$$

Then it can be shown that the time derivative of the exponential function is

$$\frac{d}{dt} \exp(\mathbf{u}^\times) = (\Psi_L(\mathbf{u}^\times) \dot{\mathbf{u}})^\times \exp(\mathbf{u}^\times) = \exp(\mathbf{u}^\times) (\Psi_R(\mathbf{u}^\times) \dot{\mathbf{u}})^\times \quad (1545)$$

where $\Psi_L(\mathbf{u}^\times)$ is the left Jacobian, and $\Psi_R(\mathbf{u}^\times)$ is the right Jacobian.

The spatial and body angular velocities are defined by

$$\dot{\mathbf{R}} = \boldsymbol{\omega}_s^\times \mathbf{R} = \mathbf{R} \boldsymbol{\omega}_b^\times \quad (1546)$$

From (1545) and (1546) it follows that the angular velocities are given by the time derivative of the logarithm as

$$\boldsymbol{\omega}_s = \Psi_L(\mathbf{u}^\times) \dot{\mathbf{u}} \quad (1547)$$

$$\boldsymbol{\omega}_b = \Psi_R(\mathbf{u}^\times) \dot{\mathbf{u}} \quad (1548)$$

The kinematic differential equations for the logarithm are then

$$\dot{\mathbf{u}} = \Psi_L(\mathbf{u}^\times)^{-1} \boldsymbol{\omega}_s \quad (1549)$$

$$\dot{\mathbf{u}} = \Psi_R(\mathbf{u}^\times)^{-1} \boldsymbol{\omega}_b \quad (1550)$$

Let $\theta = \|\mathbf{u}\|$. The expressions for the Jacobians are given in closed form

$$\Psi_L(\mathbf{u}^\times) = \mathbf{I} + \left(\frac{1 - \cos \theta}{\theta^2} \right) \mathbf{u}^\times + \left(\frac{\theta - \sin \theta}{\theta^3} \right) \mathbf{u}^{\times 2} \quad (1551)$$

$$\Psi_R(\mathbf{u}^\times) = \mathbf{I} - \left(\frac{1 - \cos \theta}{\theta^2} \right) \mathbf{u}^\times + \left(\frac{\theta - \sin \theta}{\theta^3} \right) \mathbf{u}^{\times 2} \quad (1552)$$

while the closed form solutions of the inverse Jacobians are given by

$$\Psi_L(\mathbf{u}^\times)^{-1} = \mathbf{I} - \frac{1}{2} \mathbf{u}^\times + \frac{1 - \frac{\theta}{2} \cot(\frac{\theta}{2})}{\theta^2} \mathbf{u}^{\times 2} \quad (1553)$$

$$\Psi_R(\mathbf{u}^\times)^{-1} = \mathbf{I} + \frac{1}{2} \mathbf{u}^\times + \frac{1 - \frac{\theta}{2} \cot(\frac{\theta}{2})}{\theta^2} \mathbf{u}^{\times 2} \quad (1554)$$

The coefficients of the Jacobians can be computed around zero with the Taylor series expansions

$$\frac{1 - \cos x}{x^2} = \frac{1 - \left(1 - \frac{x^2}{2!} + \frac{x^4}{4!} \dots\right)}{x^2} = \frac{1}{2!} - \frac{x^2}{4!} \dots \quad (1555)$$

and

$$\frac{x - \sin x}{x^3} = \frac{x - \left(x - \frac{x^3}{3!} + \frac{x^5}{5!} \dots\right)}{x^3} = \frac{1}{3!} - \frac{x^2}{5!} \dots \quad (1556)$$

The coefficient of the last term of the inverse Jacobians can be computed around zero using the Taylor expansion, which is found from

$$\frac{1 - x \cot x}{4x^2} = \frac{1 - x \left(x^{-1} - \frac{1}{3}x + \frac{1}{45}x^3 - \frac{2}{945}x^5 + \dots\right)}{4x^2} \quad (1557)$$

$$= \frac{1}{4} \left(\frac{1}{3} - \frac{1}{45}x^2 + \frac{2}{945}x^4 - \dots \right) \quad (1558)$$

Example

```

# Script for the computation of the Jacobians and their inverses
import numpy as np

def skewm(r):
    return np.array([[0, -r[2], r[1]], [r[2], 0, -r[0]], [-r[1], r[0], 0]])
def vex(u):
    return np.array([u[2,1], u[0,2], u[1,0]])

def expSO3(u):
    return np.identity(3) + np.sinc(np.linalg.norm(u)/np.pi)*skewm(u) \
        + 0.5*(np.sinc(np.linalg.norm(u)/(2*np.pi)))**2*skewm(u)@skewm(u)
def logSO3(R):
    # The vector form of the logarithm in SO(3) (Iserles, 2006)
    eh = 0.5*(R-R.T)  # eh = sin(theta)k^skew
    en = np.linalg.norm(vex(eh)) # en = |sin(theta)|
    if en < 0.000001:
        g = 1 + (en**2)/6
    else:
        g = (np.arcsin(en)/en)
    return vex(g*eh)

# Unit vectors
ex = np.array([1, 0, 0]); ey = np.array([0, 1, 0]); ez = np.array([0, 0, 1]);
np.set_printoptions(formatter={'float': '{: 0.4f}'.format})

def J_L(u):
    theta = np.linalg.norm(u); uh = skewm(u)
    if theta > 0.000001:
        a = (1-np.cos(theta))/(theta**2)
        b = (theta - np.sin(theta))/(theta**3)
    else:
        a = 0.5 - theta**2/24
        b = 1/3 - theta**2/120
    return np.eye(3) + a*uh + b*uh@uh

def J_L_inv(u):
    theta = np.linalg.norm(u); thetahalf = theta/2
    uh = skewm(u)
    if theta > 0.000001:
        a = (1 - thetahalf/np.tan(thetahalf))/(theta**2)
    else:
        a = 1/12 - 1/180*theta**2
    return np.eye(3) - 0.5*uh + a*uh@uh

def J_R(u):
    return J_L(-u)

```

```

def J_R_inv(u):
    return J_L_inv(-u)

# Testing

u1 = ex*np.pi/3; u2 = -ey*np.pi/6; u3 = ez*0.0000001
u4 = ex + ey - ez

JL1 = J_L(u1); JL2 = J_L(u2); JL3 = J_L(u3); JL4 = J_L(u4)
JLinv1 = J_L_inv(u1); JLinv2 = J_L_inv(u2);
JLinv3 = J_L_inv(u3); JLinv4 = J_L_inv(u4);
print('\n JL1 = \n {} \n JL2 = \n {} \n JL3 = \n {} \n JL4 = \n {}'.
      format(JL1, JL2, JL3, JL4))

print('\n JL1@JLinv1 = \n {} \n JL2@JLinv2 = \n {} \n JL3@JLinv3 = \n {} \n JL4@JLinv4 = \n {}'.
      format(JL1@JLinv1, JL2@JLinv2, JL3@JLinv3, JL4@JLinv4))

JR1 = J_R(u1); JRinv1 = J_R_inv(u1)
print('\n JR1@JRinv1 = \n {} \n'.format(JR1@JRinv1))

```

22 Gradient search on $SO(3)$

22.1 Distances in $SO(3)$

Distances in the usual Euclidean space \mathbb{R}^3 are intuitive and easy to understand. The distance $d(\mathbf{p}, \mathbf{q})$ between two points $\mathbf{p} = [x_p, y_p, z_p]^T$ and $\mathbf{q} = [x_q, y_q, z_q]^T$ is simply the length of a straight line from \mathbf{p} to \mathbf{q} . This is written

$$d(\mathbf{p}, \mathbf{q}) = \|\mathbf{q} - \mathbf{p}\| \quad (1559)$$

where

$$\|\mathbf{a}\| = \sqrt{a_1^2 + a_2^2 + a_3^2} \quad (1560)$$

is the Euclidean norm of a vector $\mathbf{a} = [a_1, a_2, a_3]^T$, which means that the square of the distance is

$$d(\mathbf{p}, \mathbf{q})^2 = \|\mathbf{q} - \mathbf{p}\|^2 = (\mathbf{q} - \mathbf{p})^T(\mathbf{q} - \mathbf{p}) \quad (1561)$$

An obvious property of this distance in \mathbb{R}^3 is that it is bi-invariant, which means that the distance between the points is unchanged if both points are given the same offset, that is

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{p} - \mathbf{r}, \mathbf{q} - \mathbf{r}) \quad (1562)$$

which is straightforward to verify by the calculation

$$d(\mathbf{p} - \mathbf{r}, \mathbf{q} - \mathbf{r})^2 = [(\mathbf{q} - \mathbf{r}) - (\mathbf{p} - \mathbf{r})]^T[(\mathbf{q} - \mathbf{r}) - (\mathbf{p} - \mathbf{r})] = (\mathbf{q} - \mathbf{p})^T(\mathbf{q} - \mathbf{p}) = d(\mathbf{p}, \mathbf{q}) \quad (1563)$$

In $SO(3)$ the concept of a distance between two rotation matrices \mathbf{R}_1 and \mathbf{R}_2 is less obvious. Still the concept of a distance is needed in applications. One example is the calculation of a rotation matrix that can represent the average of a set of measured rotation matrices. A second example is in calibration problems there is a need to minimize the offset in rotation based on measured rotation matrices. Finally, in attitude control systems where the rotation of a rigid body is controlled, there is a need to have an expression for the difference between two rotation matrices.

In the following the concept of distances and differences in $SO(3)$ will be made clear, and methods for the calculation averages and solutions to calibration problems will be presented. The presentation will follow the terminology in [29] where angular, chordal and quaternion distances are presented.

22.2 Invariance of distance measures

In $SO(3)$ the concept of invariance of a distance measure involves left invariance and right invariance. If a distance measure is both left and right invariant, then it is said to be bi-invariant [51]. To define this, consider two rotation matrices $\mathbf{R}_1, \mathbf{R}_2 \in SO(3)$. A distance function is said to be right invariant if

$$d(\mathbf{R}_1, \mathbf{R}_2) = d(\mathbf{R}_1 \mathbf{R}_R, \mathbf{R}_2 \mathbf{R}_R) \quad (1564)$$

for all $\mathbf{R}_R \in SO(3)$, while it is left invariant if

$$d(\mathbf{R}_1, \mathbf{R}_2) = d(\mathbf{R}_L \mathbf{R}_1, \mathbf{R}_L \mathbf{R}_2) \quad (1565)$$

for all $\mathbf{R}_L \in SO(3)$.

A distance function d on $SO(3)$ is said to be bi-invariant if it is both left and right invariant, which means that

$$d(\mathbf{R}_1, \mathbf{R}_2) = d(\mathbf{R}_L \mathbf{R}_1, \mathbf{R}_L \mathbf{R}_2) = d(\mathbf{R}_1 \mathbf{R}_R, \mathbf{R}_2 \mathbf{R}_R) \quad (1566)$$

for all $\mathbf{R}_L, \mathbf{R}_R \in SO(3)$.

22.3 Norms of the logarithm

A rotation matrix can be written in terms of a unit vector \mathbf{k} and an angle θ as

$$\mathbf{R} = \exp(\theta \mathbf{k}^\times) = \mathbf{I} + \sin \theta \mathbf{k}^\times + (1 - \cos \theta) \mathbf{k}^\times \mathbf{k}^\times \quad (1567)$$

The rotation matrix will be the same for (θ, \mathbf{k}) and $(-\theta, -\mathbf{k})$, which both correspond to the same logarithm $\log(\mathbf{R}) = \theta \mathbf{k}^\times$. The vector form $\theta \mathbf{k}$ of the logarithm regarded as a vector in \mathbb{R}^3 with length given by the norm

$$\|\theta \mathbf{k}\| = |\theta| \|\mathbf{k}\| = |\theta| \quad (1568)$$

The matrix form $\theta \mathbf{k}^\times$ of the logarithm will have the Frobenius norm $\|\theta \mathbf{k}^\times\|_F = |\theta| \|\mathbf{k}^\times\|_F$. Here the Frobenius norm $\|\mathbf{k}^\times\|_F$ satisfies

$$\|\mathbf{k}^\times\|_F^2 = \text{tr}(\mathbf{k}^{\times T} \mathbf{k}^\times) = -\text{tr}(\mathbf{k}^\times \mathbf{k}^\times) = \text{tr}(\mathbf{k}^T \mathbf{k} \mathbf{I} - \mathbf{k} \mathbf{k}^T) = 2 \quad (1569)$$

This means that

$$\|\theta \mathbf{k}^\times\|_F = |\theta| \|\mathbf{k}^\times\|_F = \sqrt{2} |\theta| \quad (1570)$$

22.4 Angular distance

The first distance function that will be introduced is the angular distance. This distance function has a clear geometric interpretation as the smallest angle of rotation between two orientations, and is based on the angle-axis representation of a rotation matrix.

Consider two rotation matrices \mathbf{R}_1 and $\mathbf{R}_2 = \mathbf{R}_1 \mathbf{R}_e$ where $\mathbf{R}_e = \mathbf{R}_1^T \mathbf{R}_2$ is the increment in rotation from \mathbf{R}_1 to \mathbf{R}_2 . The incremental rotation (θ_e, \mathbf{k}_e) , which means that

$$\mathbf{R}_e = \mathbf{R}_1^T \mathbf{R}_2 = \exp(\theta_e \mathbf{k}_e^\times) \quad (1571)$$

The angular distance from \mathbf{R}_1 to \mathbf{R}_2 is defined as

$$d_a(\mathbf{R}_1, \mathbf{R}_2) = d_a(\mathbf{I}, \mathbf{R}_1^T \mathbf{R}_2) = d(\mathbf{I}, \mathbf{R}_e) = |\theta_e| \in [0, \pi] \quad (1572)$$

It follows that

$$d_a(\mathbf{R}_1, \mathbf{R}_2) = d_a(\mathbf{R}_1 \mathbf{R}_2^T, \mathbf{I}) = d_a(\mathbf{I}, \mathbf{R}_1^T \mathbf{R}_2) \quad (1573)$$

The angular metric is given by norm of the vector form of the logarithm as

$$d_a(\mathbf{I}, \mathbf{R}) = \|\theta \mathbf{k}\| \quad (1574)$$

Moreover it is given by the Frobenius norm of the matrix form of the logarithm as

$$d_a(\mathbf{I}, \mathbf{R}) = \frac{1}{\sqrt{2}} \|\theta \mathbf{k}^\times\|_F \quad (1575)$$

This means that the angular distance can be given by the Frobenius norm of the logarithm as

$$d_a(\mathbf{R}, \mathbf{I})^2 = \frac{1}{2} \|\log(\mathbf{R})\|_F^2 = \|\mathbf{u}\|^2 \quad (1576)$$

where $\mathbf{u}^\times = \log(\mathbf{R})$.

22.5 Chordal distance

The chordal distance between two rotation matrices \mathbf{R}_1 and \mathbf{R}_2 is defined as the Frobenius norm of the difference of the rotation matrices [29]. This is written

$$d_c(\mathbf{R}_1, \mathbf{R}_2) = \|\mathbf{R}_1 - \mathbf{R}_2\|_F \quad (1577)$$

Note that the difference $\mathbf{R}_1 - \mathbf{R}_2$ of two rotation matrices is not a rotation matrix. It follows from (2246) that

$$\|\mathbf{R}_1 - \mathbf{R}_2\|_F^2 = \text{tr}[(\mathbf{R}_1 - \mathbf{R}_2)(\mathbf{R}_1 - \mathbf{R}_2)^T] \quad (1578)$$

$$= \text{tr}(2\mathbf{I} - 2\mathbf{R}_1 \mathbf{R}_2^T) \quad (1579)$$

$$= 2\text{tr}[\mathbf{I} - \mathbf{R}_e] \quad (1580)$$

$$= 4(1 - \cos \theta_e) \quad (1581)$$

which follows from

$$\mathbf{R}_e = \mathbf{R}_1 \mathbf{R}_2^T = \mathbf{I} + \sin \theta_e \mathbf{k}_e^\times + (1 - \cos \theta_e) (\mathbf{k}_e^\times)^2$$

which means that (θ_e, \mathbf{k}_e) are the angle-axis parameters of \mathbf{R}_e , and $\text{tr}(\mathbf{R}_e) = 1 + 2 \cos \theta_e$.

Insertion of $1 - \cos \theta_e = 2 \sin^2(\theta_e/2)$ gives

$$d_c(\mathbf{R}_1, \mathbf{R}_2)^2 = 8 \sin^2 \frac{\theta_e}{2} \quad (1582)$$

It is interesting to note that although the chordal distance is defined as the Frobenius norm of the matrix $\mathbf{R}_1 - \mathbf{R}_2$, which is not a rotation matrix, the resulting Frobenius norm has a clear geometric interpretation as the scaled sine of the half angle of the angle-axis parameters. In fact, the chordal norm is proportional to the length of the quaternion vector $\mathbf{\epsilon}_e = \sin(\theta_e/2)\mathbf{k}_e$.

22.6 Quaternion distance*

The quaternion distance between two rotations \mathbf{R}_1 and \mathbf{R}_2 represented by the unit quaternions \mathbf{q}_1 and \mathbf{q}_2 is given by

$$d_q = \|\mathbf{q}_1 - \mathbf{q}_2\| \quad (1583)$$

Note that $\mathbf{q}_1 - \mathbf{q}_2$ is not a unit quaternion, but it is a quaternion, and the norm is the usual quaternion norm. This gives

$$d_q^2 = \|\mathbf{q}_1 - \mathbf{q}_2\|^2 = (\mathbf{q}_1 - \mathbf{q}_2) \cdot (\mathbf{q}_1 - \mathbf{q}_2) = \mathbf{q}_1 \cdot \mathbf{q}_1 - 2\mathbf{q}_1 \cdot \mathbf{q}_2 + \mathbf{q}_2 \cdot \mathbf{q}_2 = 2 \left(1 - \cos \frac{\theta}{2}\right) \quad (1584)$$

Insertion of $1 - \cos(\theta/2) = 2 \sin^2(\theta/4)$ gives

$$d_q^2 = 4 \sin^2 \frac{\theta}{4} \quad (1585)$$

which gives

$$d_q = 2 \left| \sin \frac{\theta}{4} \right| \quad (1586)$$

A potential problem is that \mathbf{q}_1 and $-\mathbf{q}_1$ represent the same rotation, and \mathbf{q}_2 and $-\mathbf{q}_2$ represent the same rotation. Therefore, the quaternion distance should be calculated as recommended in [29] as

$$d_q = \min(\|\mathbf{q}_1 - \mathbf{q}_2\|, \|\mathbf{q}_1 + \mathbf{q}_2\|) \quad (1587)$$

22.7 The gradient in Euclidean space

Consider the Euclidean space, where the position of a point is given by $\mathbf{x} = [x, y, z]^T \in \mathbb{R}^3$. Let $f(\mathbf{x}) \in \mathbb{R}$ be a function of the position \mathbf{x} , which means that $f(\mathbf{x})$ is a scalar field. The gradient of the function $f(\mathbf{x})$ is

$$\text{grad}f(\mathbf{x}) = \nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix} \quad (1588)$$

where

$$\nabla^T = \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \end{bmatrix} \quad (1589)$$

is a vector of partial differentiation operators. The time derivative of the function $f(\mathbf{x}(t))$ is then

$$\frac{d}{dt}f(\mathbf{x}(t)) = \nabla^T f(\mathbf{x}(t))\dot{\mathbf{x}}(t) = \frac{\partial f}{\partial x}\dot{x} + \frac{\partial f}{\partial y}\dot{y} + \frac{\partial f}{\partial z}\dot{z} \quad (1590)$$

The directional derivative of $f(\mathbf{x})$ along a vector \mathbf{a} is found by differentiating $f(\mathbf{z}(t))$ where $\mathbf{z}(t) = \mathbf{x} + t\mathbf{a}$. It is noted that $\mathbf{z}(0) = \mathbf{x}$, and that $\dot{\mathbf{z}}(0) = \mathbf{a}$. The directional derivative is then

$$\frac{d}{dt}f(\mathbf{z}(t))\Big|_{t=0} = \nabla^T f(\mathbf{x})\dot{\mathbf{z}}(0) = \nabla^T f(\mathbf{x})\mathbf{a} \quad (1591)$$

This means that the gradient can be given in terms of the directional derivative as

$$\mathbf{a}^T \nabla f(\mathbf{x}) = \frac{d}{dt}f(\mathbf{x} + t\mathbf{a})\Big|_{t=0} \quad (1592)$$

where $\mathbf{a} \in \mathbb{R}^3$ is an arbitrary vector. The definition of the gradient on a manifold with Riemannian metric is based on this formulation.

Example

The function

$$f(\mathbf{x}) = \frac{1}{2}(x^2 + y^2 + z^2) = \frac{1}{2}\mathbf{x}^T \mathbf{x} \quad (1593)$$

will have the gradient

$$\nabla f(\mathbf{x}) = [x \ y \ z]^T = \mathbf{x} \quad (1594)$$

The directional derivative of f along \mathbf{a} can be found by differentiating $f(\mathbf{z}(t))$ where $\mathbf{z}(t) = \mathbf{x} + t\mathbf{a}$. The directional derivative is then

$$\frac{d}{dt}f(\mathbf{z}(t))\Big|_{t=0} = \mathbf{z}(0)^T \dot{\mathbf{z}}(0) = \mathbf{x}^T \mathbf{a} \quad (1595)$$

This could also be found in terms of the gradient as

$$\frac{d}{dt}f(\mathbf{z}(t))\Big|_{t=0} = \nabla^T f(\mathbf{x})\dot{\mathbf{z}}(0) = \nabla^T f(\mathbf{x})\mathbf{a} \quad (1596)$$

□

22.8 The Riemannian metric on $SO(3)$

A gradient on $SO(3)$ can be defined from the Riemannian metric on $SO(3)$, which is defined in the following. The definition of a Riemannian metric on $SO(3)$ makes it possible to extend notions like norm and inner product to the tangent space of $SO(3)$. This makes it possible to define a gradient of a function on the tangent plane of $SO(3)$, which is useful in optimization problems.

In the Euclidean space \mathbb{R}^3 a metric or distance function $d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T(\mathbf{x} - \mathbf{y})}$ can be defined from the inner product $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T \mathbf{v}$ in \mathbb{R}^3 , and a norm of a vector $\mathbf{u} \in \mathbb{R}^3$ can be defined as $\|\mathbf{u}\|^2 = \langle \mathbf{u}, \mathbf{u} \rangle = \mathbf{u}^T \mathbf{u}$.

On $T_R SO(3)$ the Riemannian metric [51] is defined as the inner product

$$\langle \mathbf{A}, \mathbf{B} \rangle = \frac{1}{2} \langle \mathbf{A}, \mathbf{B} \rangle_F = \frac{1}{2} \text{tr}(\mathbf{A}^T \mathbf{B}), \quad \mathbf{A}, \mathbf{B} \in T_R SO(3) \quad (1597)$$

where $\langle \mathbf{A}, \mathbf{B} \rangle_F = \text{tr}(\mathbf{A}^T \mathbf{B})$ the Frobenius inner product.

The Riemannian metric on $so(3)$ then gives a simple expression, since for two elements \mathbf{u}^\times and \mathbf{v}^\times on the Lie algebra $so(3)$ the Riemannian metric satisfies

$$\langle \mathbf{u}^\times, \mathbf{v}^\times \rangle = \mathbf{u}^T \mathbf{v} \quad (1598)$$

This follows from the calculation

$$\langle \mathbf{u}^\times, \mathbf{v}^\times \rangle = \frac{1}{2} \text{tr}[(\mathbf{u}^\times)^T \mathbf{v}^\times] = -\frac{1}{2} \text{tr}[\mathbf{u}^\times \mathbf{v}^\times] = \frac{1}{2} \text{tr}(\mathbf{u}^T \mathbf{v} \mathbf{I} - \mathbf{u} \mathbf{v}^T) = \mathbf{u}^T \mathbf{v} \quad (1599)$$

The Riemannian metric on $T_R SO(3)$ is bi-invariant. This is verified by first showing left-invariance with the calculation

$$\langle \mathbf{R} \mathbf{u}^\times, \mathbf{R} \mathbf{v}^\times \rangle = \frac{1}{2} \text{tr}[(\mathbf{u}^\times)^T \mathbf{R}^T \mathbf{R} \mathbf{v}^\times] = \frac{1}{2} \text{tr}[(\mathbf{u}^\times)^T \mathbf{v}^\times] = \langle \mathbf{u}^\times, \mathbf{v}^\times \rangle \quad (1600)$$

Right-invariance follows from

$$\langle \mathbf{u}^\times \mathbf{R}, \mathbf{v}^\times \mathbf{R} \rangle = \frac{1}{2} \text{tr}[\mathbf{R}^T (\mathbf{u}^\times)^T \mathbf{v}^\times \mathbf{R}] = \frac{1}{2} \text{tr}[(\mathbf{u}^\times)^T \mathbf{v}^\times] = \langle \mathbf{u}^\times, \mathbf{v}^\times \rangle \quad (1601)$$

The induced norm of the Riemannian metric is given by [68]

$$\|\mathbf{u}^\times\|^2 = \langle \mathbf{u}^\times, \mathbf{u}^\times \rangle \quad (1602)$$

22.9 Distance induced by the Riemannian metric*

Consider the motion from $\mathbf{R} \in SO(3)$ to $\mathbf{Q} \in SO(3)$ described by the rotation with angular velocity $\boldsymbol{\omega}(t) = \omega \mathbf{k}$, for $0 \leq t \leq T$, where ω is constant, and \mathbf{k} is a constant unit vector. Moreover, it is assumed that $\mathbf{Q} = \mathbf{R} \exp(\theta \mathbf{k})$, which means that $\omega T = \theta$. This means that

$$\mathbf{R}(t) = \mathbf{R} \exp(\omega t \mathbf{k}), \quad 0 \leq t \leq T \quad (1603)$$

Then the Riemannian metric is

$$\langle \boldsymbol{\omega}^\times, \boldsymbol{\omega}^\times \rangle = \boldsymbol{\omega}^T \boldsymbol{\omega} = \omega^2 \mathbf{k}^T \mathbf{k} = \omega^2 \quad (1604)$$

The length of the path induced by the Riemannian metric is then

$$L = \int_{t=0}^T \sqrt{\langle \boldsymbol{\omega}^\times, \boldsymbol{\omega}^\times \rangle} dt = \int_{t=0}^T \sqrt{\omega^2} dt = \int_{t=0}^T \omega dt = \omega T = \theta \quad (1605)$$

This shows that the length induced by the Riemannian metric is the angular distance. This type of motion is called a geodesic motion, and because of that the angular metric is also called the geodesic metric.

It is noted that the corresponding length of a curve $\mathbf{r}(t) = [x(t), y(t), z(t)]^T$ in \mathbb{R}^3 with velocity $\mathbf{v}(t) = \dot{\mathbf{r}}(t) = [\dot{x}(t), \dot{y}(t), \dot{z}(t)]^T$ is given by

$$L = \int_{t=0}^T \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle} dt = \int_{t=0}^T \sqrt{\mathbf{v}^T \mathbf{v}} dt \quad (1606)$$

If the velocity is constant and given by $\mathbf{v} = v\mathbf{k}$ for a constant unit vector \mathbf{k} , then the length of the curve is $L = vT$.

A general presentation of geodesics and Riemannian geometry is found in textbooks like [11], [43] and [53], while application to $SO(3)$ is described in [51], [46] and [42].

22.10 The gradient in $SO(3)$ based on the Riemannian metric

Consider the function $f(\mathbf{R}) \in \mathbb{R}$ which maps a rotation matrix $\mathbf{R} \in SO(3)$ to a real number. Then $f(\mathbf{R})$ is a function on the Riemannian manifold $SO(3)$, and it is possible to express the gradient of f on the in $SO(3)$ based on the Riemannian metric. This is well described in [46] and [42] where this is used to calculate the mean of a set of rotation matrices by iterative optimization methods.

The gradient $\text{grad}f(\mathbf{R})$ of $f(\mathbf{R})$ is on the tangent plane at \mathbf{R} , which is written $\text{grad}f(\mathbf{R}) \in T_{\mathbf{R}}SO(3)$. The gradient can be written

$$\text{grad}f(\mathbf{R}) = \mathbf{R}\mathbf{g}^\times \in T_{\mathbf{R}}SO(3) \quad (1607)$$

where $\mathbf{g}^\times \in so(3)$. The gradient cannot be found in a straightforward manner by applying the ∇ operator as in Euclidean space. Instead the gradient is defined in terms of the Riemannian metric and the directional derivative.

The directional derivative of the function $f(\mathbf{R})$ is found by differentiating the function $f(\mathbf{P}(t))$, where

$$\mathbf{P}(t) = \mathbf{R} \exp(t\mathbf{a}^\times) \in SO(3) \quad (1608)$$

Then $\mathbf{P}(0) = \mathbf{R}$, and

$$\dot{\mathbf{P}}(0) = \mathbf{P}(0)\mathbf{a}^\times = \mathbf{R}\mathbf{a}^\times \quad (1609)$$

Moreover, $\dot{\mathbf{P}}(0) = \mathbf{P}(0)\boldsymbol{\omega}(0)^\times$ where $\boldsymbol{\omega}(t)^\times = \mathbf{P}^T \dot{\mathbf{P}}$ is the right velocity corresponding to $\mathbf{P}(t)$. From this it is seen that $\boldsymbol{\omega}(0) = \mathbf{a}$.

The gradient at \mathbf{R} is then defined in terms of the directional derivative and the Riemannian metric by

$$\langle \mathbf{R}\mathbf{a}^\times, \text{grad}f \rangle = \frac{d}{dt} f(\mathbf{P}(t)) \Big|_{t=0} \quad (1610)$$

The Riemannian metric is bi-invariant on $SO(3)$, and it follows that this can be reformulated to

$$\langle \mathbf{a}^\times, \mathbf{g}^\times \rangle = \frac{d}{dt} f(\mathbf{P}(t)) \Big|_{t=0} \quad (1611)$$

22.11 The gradient of the angular distance

In this section the gradient is found for the square of the angular distance from a rotation matrix \mathbf{R} to the identity matrix \mathbf{I} . This is extended in the next section to the angular distance of the deviation between two rotation matrices \mathbf{R} and \mathbf{Q} . The derivation is based on the kinematic differential equation of the logarithm. Consider the function

$$f(\mathbf{R}) = \frac{1}{4} \|\log(\mathbf{R})\|_F^2 = \frac{1}{2} \mathbf{u}^T \mathbf{u} = \frac{1}{2} \theta^2 \quad (1612)$$

where the logarithm $\mathbf{u}^\times = \log(\mathbf{R})$ is given in vector form by $\mathbf{u} = \theta \mathbf{k}$ where \mathbf{k} is a unit vector. Let $\mathbf{P}(t) = \mathbf{R} \exp(t \mathbf{a}^\times)$, and let $\mathbf{w}(t)^\times = \log(\mathbf{P}(t))$ be the logarithm of $\mathbf{P}(t)$. It is noted that $\mathbf{w}(0) = \mathbf{u}$. Then

$$\dot{\mathbf{P}}(0) = \mathbf{P}(0) \mathbf{a}^\times = \mathbf{R} \mathbf{a}^\times \quad (1613)$$

Moreover,

$$\dot{\mathbf{P}}(0) = \mathbf{P}(0) \boldsymbol{\omega}(0)^\times = \mathbf{R} \boldsymbol{\omega}(0)^\times \quad (1614)$$

where $\boldsymbol{\omega}^\times = \mathbf{P}^T \dot{\mathbf{P}}$ is the right angular velocity of \mathbf{P} . It is seen that $\boldsymbol{\omega}(0) = \mathbf{a}$.

The kinematic differential equation for the logarithm of \mathbf{P} is

$$\dot{\mathbf{w}}(0) = \Psi_r(\mathbf{w}(0)^\times)^{-1} \boldsymbol{\omega}(0) = \Psi_r(\mathbf{u}^\times)^{-1} \mathbf{a} \quad (1615)$$

The directional derivative of $f(\mathbf{P}(t)) = \frac{1}{2} \mathbf{w}(t)^T \mathbf{w}(t)$ along \mathbf{a} is

$$\frac{d}{dt} f(\mathbf{P}(t)) \Big|_{t=0} = \mathbf{w}(0)^T \dot{\mathbf{w}}(0) = \mathbf{u}^T \Psi_r(\mathbf{u}^\times)^{-1} \mathbf{a} = \mathbf{u}^T \mathbf{a} = \langle \mathbf{a}^\times, \mathbf{u}^\times \rangle \quad (1616)$$

where

$$\Psi_r(\mathbf{u}^\times)^{-1} = \mathbf{I} + \frac{1}{2} \mathbf{u}^\times + \frac{1 - \frac{\theta}{2} \cot(\frac{\theta}{2})}{\theta^2} (\mathbf{u}^\times)^2 \quad (1617)$$

is the inverse of the right Jacobian $\Psi_L(\mathbf{u}^\times)$, and it is used that $\mathbf{u}^T \mathbf{u}^\times = \mathbf{0}^T$ and $\mathbf{u}^T (\mathbf{u}^\times)^2 = \mathbf{0}^T$ which gives $\mathbf{u}^T \Psi_r(\mathbf{u})^{-1} = \mathbf{u}^T$. It follows that

$$\langle \mathbf{R} \mathbf{a}^\times, \mathbf{R} \mathbf{u}^\times \rangle = \langle \mathbf{a}^\times, \mathbf{u}^\times \rangle = \frac{d}{dt} f(\mathbf{P}(t)) \Big|_{t=0} \quad (1618)$$

and it can be concluded that

$$\text{grad} f(\mathbf{R}) = \mathbf{R} \mathbf{u}^\times = \mathbf{R} \log(\mathbf{R}) \quad (1619)$$

It is noted that the gradient can also be written as $\text{grad} f(\mathbf{R}) = -\mathbf{R} \log(\mathbf{R}^T)$.

22.12 The gradient of the angular distance of an error rotation

In this section the gradient of the angular distance from $\mathbf{R} \in SO(3)$ to $\mathbf{Q} \in SO(3)$ is derived as an extension to the result of the previous section. Consider the desired rotation is $\mathbf{Q} \in SO(3)$, and the actual rotation is $\mathbf{R} \in SO(3)$. The rotation error is written $\mathbf{R}_e = \mathbf{R}^T \mathbf{Q}$, which means that $\mathbf{R} \mathbf{R}_e = \mathbf{Q}$. The angle-axis representation of \mathbf{R}_e is (\mathbf{k}_e, θ_e) , which means that the logarithm of the error rotation is $\log(\mathbf{R}_e) = \mathbf{u}_e^\times = \theta_e \mathbf{k}_e^\times$.

The cost function to be minimized is

$$f(\mathbf{R}) = \frac{1}{2}d_a(\mathbf{Q}, \mathbf{R})^2 = \frac{1}{2}d_a(\mathbf{R}_e, \mathbf{I})^2 = \frac{1}{2}\theta_e^2 \quad (1620)$$

The function $f(\mathbf{R})$ can be written in terms of the Frobenius norm of the logarithm as

$$f(\mathbf{R}) = \frac{1}{4}\|\log(\mathbf{R}_e)\|_F^2 \quad (1621)$$

which follows from (1576).

Consider $\mathbf{P}(t) = \mathbf{R} \exp(t\mathbf{a}^\times)$ where \mathbf{a} is constant. The rotation error associated with $\mathbf{P}(t)$ is $\mathbf{P}_e(t) = \mathbf{P}(t)^\top \mathbf{Q}$. Let the logarithm of this rotation error be

$$\mathbf{w}(t)^\times = \log(\mathbf{P}_e(t)) \quad (1622)$$

Then $\mathbf{w}(0) = \mathbf{u}_e$. The cost function at $\mathbf{P}(t)$ is

$$f(\mathbf{P}(t)) = \frac{1}{4}\|\log(\mathbf{P}^\top \mathbf{Q})\|_F^2 = \frac{1}{4}\text{tr}[(\mathbf{w}^\times)^\top (\mathbf{w}^\times)] = \frac{1}{2}\langle \mathbf{w}^\times, \mathbf{w}^\times \rangle = \frac{1}{2}\mathbf{w}^\top \mathbf{w} \quad (1623)$$

Consider

$$\dot{\mathbf{P}}(t)\Big|_{t=0} = \frac{d}{dt}[\mathbf{R} \exp(t\mathbf{a}^\times)]\Big|_{t=0} = \mathbf{R}\mathbf{a}^\times \quad (1624)$$

Moreover, if $\boldsymbol{\omega}^\times = \mathbf{P}^\top \dot{\mathbf{P}}$ is the angular velocity corresponding to $\mathbf{P}(t)$, then

$$\dot{\mathbf{P}}(t) = \mathbf{P}(t)\boldsymbol{\omega}(t)^\times \quad (1625)$$

and it follows that $\boldsymbol{\omega}(0) = \mathbf{a}$. The time derivative of the error rotation is

$$\dot{\mathbf{P}}_e(t) = \frac{d}{dt}(\mathbf{P}^\top \mathbf{Q}) = \dot{\mathbf{P}}^\top \mathbf{Q} = (\mathbf{P}(t)\boldsymbol{\omega}^\times)^\top \mathbf{Q} = -\boldsymbol{\omega}^\times \mathbf{P}^\top \mathbf{Q} = -\boldsymbol{\omega}^\times \mathbf{P}_e \quad (1626)$$

which gives

$$\dot{\mathbf{P}}_e(t) = -\boldsymbol{\omega}(t)^\times \mathbf{P}_e(t) \quad (1627)$$

This in combination with $\mathbf{P}_e(t) = \exp(\mathbf{w}(t))$ and $\boldsymbol{\omega}(0) = \mathbf{a}$ gives

$$\dot{\mathbf{w}}(0) = -\Psi_L(\mathbf{w}(0))^{-1}\boldsymbol{\omega}(0) = -\Psi_L(\mathbf{u}_e)^{-1}\mathbf{a} \quad (1628)$$

where

$$\Psi_L(\mathbf{u})^{-1} = \mathbf{I} - \frac{1}{2}\mathbf{u}^\times + \frac{1 - \frac{\theta}{2}\cot(\frac{\theta}{2})}{\theta^2}(\mathbf{u}^\times)^2 \quad (1629)$$

is the inverse of the left Jacobian $\Psi_L(\mathbf{u})$. This gives

$$\frac{d}{dt}f(\mathbf{P}(t))\Big|_{t=0} = \mathbf{w}(0)^\top \dot{\mathbf{w}}(0) = -\mathbf{u}_e^\top \Psi_L(\mathbf{u}_e)^{-1}\mathbf{a} = -\mathbf{u}_e^\top \mathbf{a} \quad (1630)$$

This can be written in terms of the Riemannian metric as

$$\frac{d}{dt}f(\mathbf{P}(t))\Big|_{t=0} = -\langle \mathbf{a}^\times, \mathbf{u}_e^\times \rangle = -\langle \mathbf{R}\mathbf{a}^\times, \mathbf{R}\mathbf{u}_e^\times \rangle \quad (1631)$$

Then it follows from

$$\langle \mathbf{R}\mathbf{a}^\times, \text{grad}f \rangle = \frac{d}{dt}f(\mathbf{P}(t))\Big|_{t=0} \quad (1632)$$

that

$$\text{grad}f(\mathbf{R}) = -\mathbf{R}\mathbf{u}_e^\times = -\mathbf{R}\log(\mathbf{R}^\top \mathbf{Q}) \quad (1633)$$

22.13 More on the gradient of the angular distance*

In this section the gradient is found for the square of the angular distance. The derivation is based on the presentation in [46], with additional details from [42]. Consider the rotation from $\mathbf{R} \in SO(3)$ to $\mathbf{Q} \in SO(3)$, which is given by the rotation matrix $\mathbf{R}_{RQ} = \mathbf{R}^T \mathbf{Q}$, and let θ_{RQ} is the rotation angle of \mathbf{R}_{RQ} . The gradient is to be found for the function

$$f(\mathbf{R}) = \frac{1}{2} d_a(\mathbf{R}^T \mathbf{Q}, \mathbf{I})^2 = \frac{1}{2} \theta_{RQ}^2 \quad (1634)$$

The function $f(\mathbf{R})$ can be written in terms of the Frobenius norm of the logarithm as

$$f(\mathbf{R}) = \frac{1}{4} \|\log(\mathbf{R}^T \mathbf{Q})\|_F^2 \quad (1635)$$

which follows from (1576). It is noted that in [46] this expression is multiplied by a factor 2, which is believed to be a typographical error.

To find the gradient of f at \mathbf{R} , the function $f(\mathbf{P}(t))$ is introduced where $\mathbf{P}(t) = \mathbf{R} \exp(t\mathbf{A})$ is a rotation matrix, $\mathbf{P}(0) = \mathbf{R}$, and $\mathbf{A} \in so(3)$ is an arbitrary element of the Lie algebra. Let $\theta(t) \in [-\pi, \pi]$ be the angle of rotation of the rotation matrix $\mathbf{P}(t)^T \mathbf{Q}$, which means that $\theta(0) = \theta_{RQ}$ and

$$\text{tr}(\mathbf{P}(t)^T \mathbf{Q}) = 1 + 2 \cos \theta(t) \quad (1636)$$

while

$$f(\mathbf{P}(t)) = \frac{1}{2} \theta(t)^2 \quad (1637)$$

The gradient $\text{grad } f \in T_R SO(3)$ is then defined as the element in $T_R SO(3)$ which satisfies the following Riemannian metric

$$\langle \mathbf{R}\mathbf{A}, \text{grad } f \rangle = \frac{d}{dt} f(\mathbf{P}(t)) \Big|_{t=0} \quad (1638)$$

for $\mathbf{R}\mathbf{A} \in T_R SO(3)$. The definition of the gradient can therefore be written

$$\langle \mathbf{R}\mathbf{A}, \text{grad } f \rangle = \frac{d}{dt} f(\mathbf{R} \exp(t\mathbf{A})) \Big|_{t=0} \quad (1639)$$

The time derivative of $f(\mathbf{P}(t))$ is found from (1637) to be

$$\frac{d}{dt} f(\mathbf{P}(t)) \Big|_{t=0} = \theta_{RQ} \dot{\theta} \quad (1640)$$

To proceed it is noted that the time derivative of (1636) gives

$$\frac{d}{dt} \text{tr}(\mathbf{P}(t)^T \mathbf{Q}) \Big|_{t=0} = -2 \sin \theta_{RQ} \dot{\theta} \quad (1641)$$

This derivative is also given by

$$\frac{d}{dt} \text{tr}(\mathbf{P}(t)^T \mathbf{Q}) \Big|_{t=0} = \frac{d}{dt} \text{tr}(\mathbf{Q}^T \mathbf{P}(t)) \Big|_{t=0} = \text{tr}(\mathbf{Q}^T \mathbf{R}\mathbf{A}) \quad (1642)$$

where it is used that $(d/dt)\mathbf{P}(t)|_{t=0} = \mathbf{R}\mathbf{A}$. From (1640,1641,1642) it is then seen that

$$\frac{d}{dt}f(\mathbf{P}(t))\bigg|_{t=0} = -\frac{\theta_{RQ}}{2\sin\theta_{RQ}}\text{tr}(\mathbf{Q}^T\mathbf{R}\mathbf{A}) = \frac{\theta_{RQ}}{2\sin\theta_{RQ}}\text{tr}(\mathbf{Q}^T\mathbf{R}\mathbf{A}) \quad (1643)$$

The trace satisfies $\text{tr}(\mathbf{X}) = \text{tr}(\mathbf{X}^T) = \text{tr}(\mathbf{Y}\mathbf{X}^T)$ for any $\mathbf{X} \in \mathbb{R}^{3 \times 3}$. This in combination with $\mathbf{A}^T = -\mathbf{A}$ gives

$$\text{tr}(\mathbf{Q}^T\mathbf{R}\mathbf{A}) = \frac{1}{2}\text{tr}(\mathbf{Q}^T\mathbf{R}\mathbf{A} + \mathbf{R}^T\mathbf{Q}\mathbf{A}^T) = \frac{1}{2}\text{tr}[(\mathbf{Q}^T\mathbf{R} - \mathbf{R}^T\mathbf{Q})\mathbf{A}] \quad (1644)$$

Moreover, the trace satisfies $\text{tr}(\mathbf{X}, \mathbf{Y}) = \text{tr}(\mathbf{Y}, \mathbf{X})$ for any $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{3 \times 3}$, which gives

$$\text{tr}(\mathbf{Q}^T\mathbf{R}\mathbf{A}) = \frac{1}{2}\text{tr}[\mathbf{A}(\mathbf{Q}^T\mathbf{R} - \mathbf{R}^T\mathbf{Q})] \quad (1645)$$

Then from (1523) it follows that $\mathbf{Q}^T\mathbf{R} - \mathbf{R}^T\mathbf{Q} = (2\sin\theta_{RQ}/\theta_{RQ})\log(\mathbf{Q}^T\mathbf{R})$, which gives

$$\text{tr}(\mathbf{Q}^T\mathbf{R}\mathbf{A}) = \frac{\sin\theta_{RQ}}{\theta_{RQ}}\text{tr}[\mathbf{A}\log(\mathbf{Q}^T\mathbf{R})] \quad (1646)$$

This gives

$$\frac{d}{dt}f(\mathbf{P}(t))\bigg|_{t=0} = -\frac{1}{2}\text{tr}[\mathbf{A}\log(\mathbf{Q}^T\mathbf{R})] = -\frac{1}{2}\text{tr}[(\mathbf{R}\mathbf{A})^T\mathbf{R}\log(\mathbf{R}^T\mathbf{Q})] \quad (1647)$$

which is written in terms of the Riemannian metric as

$$\frac{d}{dt}f(\mathbf{P}(t))\bigg|_{t=0} = \langle \mathbf{R}\mathbf{A}, -\mathbf{R}\log(\mathbf{Q}^T\mathbf{R}) \rangle \quad (1648)$$

It follows from (1610) that the gradient is

$$\text{grad}f(\mathbf{R}) = -\mathbf{R}\log(\mathbf{R}^T\mathbf{Q}) \in T_R SO(3) \quad (1649)$$

22.14 Rotation averaging for the angular distance

Consider the situation that n rotation matrices are \mathbf{R}_i are given, and the goal is to find an average rotation matrix \mathbf{R} with respect to the angular distance measure [29, 34, 42].

Averaging in terms of the angular distance is done by minimizing the objective function

$$f(\mathbf{R}) = \frac{1}{n} \sum_{i=1}^n d_a(\mathbf{R}_i, \mathbf{R})^2 \quad (1650)$$

with respect to \mathbf{R} . This objective function has the gradient

$$\text{grad}f(\mathbf{R}) = -\frac{1}{n} \sum_{i=1}^n \mathbf{R}\log(\mathbf{R}^T\mathbf{R}_i) = -\frac{1}{n}\mathbf{R} \sum_{i=1}^n \log(\mathbf{R}^T\mathbf{R}_i) \quad (1651)$$

which follows from (1649).

The difference in rotation from \mathbf{R} to the observation \mathbf{R}_i can be modeled as $\mathbf{R}_i = \mathbf{R}\tilde{\mathbf{R}}_i$, where the deviation in rotation is given by the rotation matrix $\tilde{\mathbf{R}}_i = \mathbf{R}^T\mathbf{R}_i$, which has the logarithm

$$\mathbf{u}_i^\times = \log(\tilde{\mathbf{R}}_i) = \log(\mathbf{R}^T\mathbf{R}_i) \quad (1652)$$

which gives

$$\mathbf{R}_i = \mathbf{R} \exp(\mathbf{u}_i^\times) \quad (1653)$$

The gradient of $f(\mathbf{R})$ at \mathbf{R} is $\text{grad}f(\mathbf{R}) \in T_R SO(3)$, which is in the tangent plane of $SO(3)$ at \mathbf{R} . It is noted that the gradient has the form $\text{grad}f(\mathbf{R}) = \mathbf{R}\mathbf{g}^\times \in T_R SO(3)$ where $\mathbf{g}^\times \in so(3)$ is given by

$$\mathbf{g}^\times = -\frac{1}{n} \sum_{i=1}^n \log(\mathbf{R}^T \mathbf{R}_i) \quad (1654)$$

A gradient method with unit step-length in the negative gradient direction is then given by

$$\mathbf{R}(k+1) = \mathbf{R}(k) \exp(-\mathbf{R}(k)^T \text{grad}f(\mathbf{R}(k))) = \mathbf{R}(k) \exp(-\mathbf{g}(k)^\times) \quad (1655)$$

This gives the gradient search method

$$\mathbf{R}(k+1) = \mathbf{R}(k) \exp\left(\frac{1}{n} \sum_{i=1}^n \log(\mathbf{R}(k)^T \mathbf{R}_i)\right) \quad (1656)$$

It is shown in [46, 42] that the Hessian of $f(\mathbf{R})$ is positive with eigenvalues less than or equal to unity, and that this implies that this gradient search method will converge unless the initial rotation has an error of π , which means that in practice, the method is expected to converge to the correct rotation matrix.

Example

```
% Input data
n = 10; Rd = zeros(3,3,n);
theta = pi/8 + 0.3*(rand(1,n)-0.5); % Random angles
k = [0;0;1] + 0.3*(rand(3,n)-0.5); k = k/norm(k);% Random rotation axes
for i= 1:n
    Rd(:,:,i) = eye(3) + sin(theta(i))*skew(k(:,:,i))...
        + (1-cos(theta(i)))*skew(k(:,:,i))*skew(k(:,:,i));
end

% Average rotation from
% minimization of angular distance with gradient search
R = eye(3); r = ones(3,1);
while norm(r) > 0.001
    r = 0;
    for i = 1:n
        r = r + LogRot(R'*Rd(:,:,i));
    end
    R = R*ExpRot(r/n);
end
```

```

R_grad_search = R

% Average rotation with Procrustes minimization of chordal distance
Ce = zeros(3,3);
for i = 1:n
    Ce = Ce + Rd(:,:,i)';
end
[U,S,V] = svd(Ce);
Rsvd = V*diag([1,1,det(U*V')])*U'

```

22.15 Rotation averaging for two rotation matrices

A closed for solution is available for the average of two rotation matrices. In this case the average in terms of the angular distance is found by calculating the logarithm of the increment from

$$\mathbf{u}^\times = \log(\mathbf{R}_1^T \mathbf{R}_2) \quad (1657)$$

and then calculating the average as

$$\mathbf{R} = \mathbf{R}_1 \exp\left(\frac{1}{2} \log(\mathbf{R}_1^T \mathbf{R}_2)\right) \quad (1658)$$

using (1516).

It is noted that $\mathbf{R}_2 = \mathbf{R}_1 \exp(\mathbf{u}^\times)$, and it is seen that the matrix \mathbf{R} will be obtained by rotating half the angle between \mathbf{R}_1 and \mathbf{R}_2 about the unit vector $\mathbf{k}_e = \mathbf{u}/\|\mathbf{u}\|$. It follows that

$$d_a(\mathbf{R}_1, \mathbf{R}) = d_a(\mathbf{R}, \mathbf{R}_2) = \frac{1}{2} \|\mathbf{u}\| \quad (1659)$$

The matrix \mathbf{R} is therefore the average of the two rotation matrices \mathbf{R}_1 and \mathbf{R}_2 with respect to the angular norm.

22.16 Rotation averaging for rotation matrices with the same rotation vector

A closed for solution is also available for n rotation matrices $\mathbf{R}_i = \exp(\theta_i \mathbf{k})$ where the rotation axis \mathbf{k} is the same. If it is assumed that $\max_{i,j} |\theta_i - \theta_j| < \pi$, then the average is

$$\mathbf{R} = \exp(\theta \mathbf{k}) \quad (1660)$$

where

$$\theta = \frac{1}{n} \sum_{i=1}^n \theta_i \quad (1661)$$

is the average rotation angle. Then, since all rotation matrices has the same rotation axis,

$$f(\mathbf{R}) = \frac{1}{n} \sum_{i=1}^n d_a(\mathbf{R}_i, \mathbf{R})^2 = \frac{1}{2n} \|\log(\mathbf{R}_i, \mathbf{R})\|_F^2 = \frac{1}{n} \sum_{i=1}^n \|\theta_i - \theta\|^2 \quad (1662)$$

is the minimum value for $f(\mathbf{R})$.

23 Exponential description of displacements in $SE(3)$

23.1 Exponential description of a displacement

A displacement is given by the homogeneous transformation matrix $\mathbf{T} = \mathbf{T}_b^s \in SE(3)$ from the spatial frame s to the body frame b . The homogeneous transformation matrix is written as

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1663)$$

where $\mathbf{R} = \mathbf{R}_b^s \in SO(3)$ and $\mathbf{p} = \mathbf{p}_{sb}^s$. The time derivative of the homogeneous transformation matrix is

$$\dot{\mathbf{T}} = \bar{\mathbf{w}}_{/s} \mathbf{T} = \mathbf{T} \bar{\mathbf{w}}_{/b} \quad (1664)$$

where the twist $\bar{\mathbf{w}}_{/s} = \bar{\mathbf{w}}_{sb/s}^s$ is the spatial velocity, and the twist $\bar{\mathbf{w}}_{/b} = \bar{\mathbf{w}}_{sb/b}^b$ is the body velocity. The spatial and body velocities are given in matrix form as

$$\bar{\mathbf{w}}_{/s} = \begin{bmatrix} (\boldsymbol{\omega}_{sb}^s)^\times & (\mathbf{p}_{sb}^s)^\times \boldsymbol{\omega}_{sb}^s + \mathbf{v}_{sb}^s \\ \mathbf{0}^T & 0 \end{bmatrix}, \quad \bar{\mathbf{w}}_{/b} = \begin{bmatrix} (\boldsymbol{\omega}_{sb}^b)^\times & \mathbf{v}_{sb}^b \\ \mathbf{0}^T & 0 \end{bmatrix} \quad (1665)$$

which is verified by direct calculation of the expressions in (1664). It is seen that the spatial velocity is given in the coordinates of frame s , while the body velocity is given in the coordinates of frame b . The expressions are in agreement with the transformation

$$\bar{\mathbf{w}}_{/s} = \mathbf{T} \bar{\mathbf{w}}_{/b} \mathbf{T}^{-1} \quad (1666)$$

which follows from from (1664).

The twists $\bar{\mathbf{w}}_{/s}$ and $\bar{\mathbf{w}}_{/b}$, which are 6×6 matrices can be represented by the 6-dimensional screw vectors

$$\mathbf{w}_{/b} = \left\{ \begin{array}{c} \boldsymbol{\omega}_{sb}^b \\ \mathbf{v}_{sb}^b \end{array} \right\} \quad (1667)$$

where $\boldsymbol{\omega}_{sb}^b$ is the angular velocity of frame b relative to frame s in the coordinates of b , while \mathbf{v}_{sb}^b is the velocity of the origin of frame b relative to the origin of frame s in the coordinates of frame b . The vector form of the spatial velocity is

$$\mathbf{w}_{/s} = \left\{ \begin{array}{c} \boldsymbol{\omega}_{sb}^s \\ (\mathbf{p}_{sb}^s)^\times \boldsymbol{\omega}_{sb}^s + \mathbf{v}_{sb}^s \end{array} \right\} \quad (1668)$$

which is in agreement with the screw transformation

$$\mathbf{w}_{/s} = \mathbf{U} \mathbf{w}_{/b} \quad (1669)$$

where

$$\mathbf{U} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{p}^\times \mathbf{R} & \mathbf{R} \end{bmatrix} \quad (1670)$$

is a screw transformation matrix.

In Lie group terminology, the homogeneous transformation matrix $\mathbf{T} \in SE(3)$ is an element in the Lie group $SE(3)$, while the time derivative $\dot{\mathbf{T}} = \bar{\mathbf{w}}_{/s} \mathbf{T} = \mathbf{T} \bar{\mathbf{w}}_{/b} \in T_T SE(3)$ is an element of the tangent plane at \mathbf{T} . The twist $\bar{\mathbf{w}}_{/b} \in se(3)$ is an element in the Lie algebra $se(3) = T_I SE(3)$, which is the tangent plane at the identity element \mathbf{I} of $SE(3)$. Also the spatial velocity $\bar{\mathbf{w}}_{/s} \in se(3)$ is an element of the Lie algebra $se(3)$.

23.2 The exponential function on $SE(3)$

The logarithm in $SE(3)$ is an element of $se(3)$, and is written in matrix form as

$$\bar{\xi} = \begin{bmatrix} \mathbf{u}^\times & \mathbf{w} \\ \mathbf{0}^T & 0 \end{bmatrix} \in se(3) \quad (1671)$$

The vector form of the logarithm is the screw

$$\xi = \begin{Bmatrix} \mathbf{u} \\ \mathbf{w} \end{Bmatrix} \quad (1672)$$

An element \mathbf{T} in the Lie group is given by the exponential function as

$$\mathbf{T} = \exp(\bar{\xi}) \quad (1673)$$

The exponential function is defined as the matrix exponential function

$$\exp(\bar{\xi}) \triangleq \mathbf{I} + \bar{\xi} + \frac{1}{2!} \bar{\xi}^2 + \frac{1}{3!} \bar{\xi}^3 + \dots = \sum_{k=0}^{\infty} \frac{\bar{\xi}^k}{k!} \quad (1674)$$

To find an expression for the exponential function $\exp(\bar{\xi})$ it is used that the square of the logarithm is

$$\bar{\xi}^2 = \begin{bmatrix} \mathbf{u}^\times & \mathbf{w} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^\times & \mathbf{w} \\ \mathbf{0}^T & 0 \end{bmatrix} = \begin{bmatrix} (\mathbf{u}^\times)^2 & \mathbf{u}^\times \mathbf{w} \\ \mathbf{0}^T & 0 \end{bmatrix} \quad (1675)$$

while the logarithm raised to the third order is

$$\bar{\xi}^3 = \begin{bmatrix} \mathbf{u}^\times & \mathbf{w} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} (\mathbf{u}^\times)^2 & \mathbf{u}^\times \mathbf{w} \\ \mathbf{0}^T & 0 \end{bmatrix} = \begin{bmatrix} (\mathbf{u}^\times)^3 & (\mathbf{u}^\times)^2 \mathbf{w} \\ \mathbf{0}^T & 0 \end{bmatrix} \quad (1676)$$

Proceeding in this way, it is found that the logarithm raised to order k is

$$\bar{\xi}^k = \begin{bmatrix} (\mathbf{u}^\times)^k & (\mathbf{u}^\times)^{k-1} \mathbf{w} \\ \mathbf{0}^T & 0 \end{bmatrix} \quad (1677)$$

The exponential function is therefore

$$\exp(\bar{\xi}) = \sum_{k=0}^{\infty} \frac{\bar{\xi}^k}{k!} = \begin{bmatrix} \sum_{k=0}^{\infty} \frac{(\mathbf{u}^\times)^k}{k!} & \sum_{k=1}^{\infty} \frac{(\mathbf{u}^\times)^{k-1}}{k!} \mathbf{w} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1678)$$

This leads to the expression

$$\exp(\bar{\xi}) = \begin{bmatrix} \exp(\mathbf{u}^\times) & \Psi_L(\mathbf{u}^\times) \mathbf{w} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1679)$$

where $\exp(\mathbf{u}^\times) = \mathbf{R} \in SO(3)$ is the exponential function in $SO(3)$, and

$$\Psi_L(\mathbf{u}^\times) \triangleq \sum_{k=0}^{\infty} \frac{(\mathbf{u}^\times)^k}{(k+1)!} \quad (1680)$$

This matrix is recognized as the left Jacobian in $SO(3)$, which is given in closed form as

$$\Psi_L(\mathbf{u}^\times) = \mathbf{I} + \left(\frac{1 - \cos \|\mathbf{u}\|}{\|\mathbf{u}\|^2} \mathbf{u}^\times + \frac{\|\mathbf{u}\| - \sin \|\mathbf{u}\|}{\|\mathbf{u}\|^3} \mathbf{u}^\times \mathbf{u}^\times \right) \quad (1681)$$

23.3 Chasles' theorem

According to Chasles' theorem [49] a rigid motion can be described as a rotation about a line and a translation along the same line. This will be explored in this section by studying the rigid motion resulting from a rotation about a line through the origin, in combination with a translation along the same line.

Suppose that the logarithm is given by

$$\bar{\zeta} = \begin{bmatrix} \mathbf{u}^\times & h\mathbf{u} \\ \mathbf{0}^T & 0 \end{bmatrix} \quad (1682)$$

The corresponding vector form is

$$\zeta = \begin{Bmatrix} \mathbf{u} \\ h\mathbf{u} \end{Bmatrix} \quad (1683)$$

where the rotation part \mathbf{u} and the translation part $h\mathbf{u}$ are parallel. The scalar h is called the pitch. The logarithm corresponds to a rotation about the vector \mathbf{u} , and a translation along the same vector. This can also be seen as a screw motion with a rotation about a line ℓ , and a translation along the same line with pitch h . The line is through the origin with direction vector \mathbf{u} . The Plücker coordinates of the line is then

$$\ell = (\mathbf{k}, \mathbf{0}) \quad (1684)$$

where $\mathbf{k} = \mathbf{u}/\|\mathbf{u}\|$. It is straightforward to verify that

$$\bar{\zeta}^2 = \begin{bmatrix} (\mathbf{u}^\times)^2 & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}, \quad \bar{\zeta}^3 = \begin{bmatrix} (\mathbf{u}^\times)^3 & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}, \quad \dots \quad (1685)$$

and it follows that

$$\exp(\bar{\zeta}) = \begin{bmatrix} \mathbf{I} + \mathbf{u}^\times + \frac{1}{2!}(\mathbf{u}^\times)^2 + \dots & h\mathbf{u} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1686)$$

which gives

$$\exp(\bar{\zeta}) = \begin{bmatrix} \exp(\mathbf{u}^\times) & h\mathbf{u} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1687)$$

This is consistent with (1679), which is seen from

$$\Psi_L(\mathbf{u}^\times)h\mathbf{u} = \mathbf{I} + \left(\frac{1 - \cos \|\mathbf{u}\|}{\|\mathbf{u}\|^2} \mathbf{u}^\times + \frac{\|\mathbf{u}\| - \sin \|\mathbf{u}\|}{\|\mathbf{u}\|^3} \mathbf{u}^\times \mathbf{u}^\times \right) h\mathbf{u} = h\mathbf{u} \quad (1688)$$

□

A general rigid motion can be described based on this by displacing the line ℓ by

$$\mathbf{T}_q = \begin{bmatrix} \mathbf{I} & \mathbf{q} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1689)$$

which is a translation by a vector \mathbf{q} . The vector form ζ of the logarithm is then transformed to ξ with the screw transformation

$$\xi = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{q}^\times & \mathbf{I} \end{bmatrix} \zeta = \begin{Bmatrix} \mathbf{u} \\ \mathbf{q}^\times \mathbf{u} + h\mathbf{u} \end{Bmatrix} \quad (1690)$$

This means that the line ℓ is transformed to the line

$$\ell_q = (\mathbf{k}, \mathbf{q}^\times \mathbf{k}) \quad (1691)$$

and the motion is described as a rotation about the line ℓ_q and a translation along the same line.

The corresponding transformation in matrix form is

$$\bar{\xi} = \mathbf{T}_q \bar{\zeta} \mathbf{T}_q^{-1} = \begin{bmatrix} \mathbf{I} & \mathbf{q} \\ \mathbf{0}^T & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{u}^\times & p\mathbf{u} \\ \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{q} \\ \mathbf{0}^T & \mathbf{I} \end{bmatrix} \quad (1692)$$

which gives

$$\bar{\xi} = \begin{bmatrix} \mathbf{u}^\times & \mathbf{q}^\times \mathbf{u} + p\mathbf{u} \\ \mathbf{0}^T & 0 \end{bmatrix} \quad (1693)$$

The exponential function for the general logarithm $\bar{\xi} = \mathbf{T}_q \bar{\zeta} \mathbf{T}_q^{-1}$ is found from the series expansion through the calculation

$$\exp(\bar{\xi}) = \mathbf{I} + \mathbf{T}_q \bar{\zeta} \mathbf{T}_q^{-1} + \frac{1}{2!} (\mathbf{T}_q \bar{\zeta} \mathbf{T}_q^{-1})(\mathbf{T}_q \bar{\zeta} \mathbf{T}_q^{-1}) + \dots \quad (1694)$$

$$= \mathbf{T}_q \left(\mathbf{I} + \bar{\zeta} + \frac{1}{2!} \bar{\zeta}^2 + \dots \right) \mathbf{T}_q^{-1} \quad (1695)$$

$$= \mathbf{T}_q \begin{bmatrix} \exp(\mathbf{u}^\times) & p\mathbf{u} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{T}_q^{-1} \quad (1696)$$

which gives

$$\exp(\bar{\xi}) = \begin{bmatrix} \exp(\mathbf{u}^\times) & (\mathbf{I} - \exp(\mathbf{u}^\times))\mathbf{q} + h\mathbf{u} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1697)$$

This means that a general rigid motion

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^T & \mathbf{I} \end{bmatrix} \quad (1698)$$

is given by a rotation by angle $\|\mathbf{u}\|$ about ℓ_q and a translation $h\mathbf{u}$ along the same line, where

$$\mathbf{R} = \exp(\mathbf{u}^\times) \quad (1699)$$

$$\mathbf{p} = (\mathbf{I} - \exp(\mathbf{u}^\times))\mathbf{q} + h\mathbf{u} \quad (1700)$$

Suppose that the general logarithm is given by

$$\bar{\xi} = \begin{bmatrix} \mathbf{u}^\times & \mathbf{w} \\ \mathbf{0}^T & 0 \end{bmatrix} \quad (1701)$$

Then this can be converted to the form (1693) by setting $\mathbf{w} = \mathbf{q}^\times \mathbf{u} + h\mathbf{u}$, where $h\mathbf{u}$ is the component along \mathbf{u} , and $\mathbf{q}^\times \mathbf{u}$ is the component orthogonal to \mathbf{u} . This means that

$$h\mathbf{u} = \frac{\mathbf{w}^T \mathbf{u}}{\|\mathbf{u}\|^2} \mathbf{u} = \frac{\mathbf{u} \mathbf{u}^T}{\|\mathbf{u}\|^2} \mathbf{w}, \quad \mathbf{q} = \frac{\mathbf{u}^\times \mathbf{w}}{\|\mathbf{u}\|^2} \quad (1702)$$

The exponential function of the general logarithm is therefore

$$\exp(\bar{\xi}) = \begin{bmatrix} \exp(\mathbf{u}^\times) & \frac{1}{\|\mathbf{u}\|^2} [(\mathbf{I} - \exp(\mathbf{u}^\times))\mathbf{u}^\times + \mathbf{u} \mathbf{u}^T] \mathbf{w} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1703)$$

24 Distance metrics on $SE(3)$

24.1 Introduction

In the Euclidean space \mathbb{R}^3 the distance metric is the distance between two points \mathbf{p} and \mathbf{p}_0 is

$$d(\mathbf{p}, \mathbf{p}_0) = \sqrt{(\mathbf{p} - \mathbf{p}_0)^T(\mathbf{p} - \mathbf{p}_0)} \quad (1704)$$

This is the length of the line segment between the two points. The spaces $SO(3)$ and $SE(3)$ are different from Euclidean space, and the distance function between two configurations must be found in some other way.

Both $SO(3)$ and $SE(3)$ are Lie groups [49]. Lie groups have certain properties in common. In particular, the time derivative of a configuration \mathbf{X} in $SO(3)$ or $SE(3)$ can be written

$$\dot{\mathbf{X}} = \mathbf{X}\bar{\boldsymbol{\xi}}_b \quad (1705)$$

where $\bar{\boldsymbol{\xi}}_b = \mathbf{X}^{-1}\dot{\mathbf{X}}$ is the body velocity, which is an element in the Lie algebra. The time derivative can alternatively be written

$$\dot{\mathbf{X}} = \bar{\boldsymbol{\xi}}_s \mathbf{X} \quad (1706)$$

where $\bar{\boldsymbol{\xi}}_s = \dot{\mathbf{X}}\mathbf{X}^{-1}$ is the spatial velocity, which is also in the Lie algebra.

The distance between \mathbf{X} and \mathbf{X}_0 can be described with a distance function $d(\mathbf{X}, \mathbf{X}_0)$. A distance function is said to be left-invariant if

$$d(\mathbf{X}_l \mathbf{X}, \mathbf{X}_l \mathbf{X}_0) = d(\mathbf{X}, \mathbf{X}_0) \quad (1707)$$

for any constant \mathbf{X}_l in the Lie group, and it is said to be right-invariant if

$$d(\mathbf{X} \mathbf{X}_r, \mathbf{X}_0 \mathbf{X}_r) = d(\mathbf{X}, \mathbf{X}_0) \quad (1708)$$

for any constant \mathbf{X}_r in the Lie group.

24.2 $SO(3)$ as a Lie group

The set $SO(3)$ of rotation matrices forms a matrix Lie group where the elements are rotation matrices and the group action is matrix multiplication. For two rotation matrices $\mathbf{R}_1, \mathbf{R}_2 \in SO(3)$ the group operation is closed since $\mathbf{R} = \mathbf{R}_1 \mathbf{R}_2 \in SO(3)$. The identity element is $\mathbf{I} \in SO(3)$, and the inverse is $\mathbf{R}^{-1} = \mathbf{R}^T \in SO(3)$.

The Lie algebra $so(3)$ of the Lie group $SO(3)$ is the set of all skew-symmetric matrices $\boldsymbol{\sigma}^\times$ of dimension 3×3 . The time derivative of the rotation matrix is

$$\dot{\mathbf{R}} = \mathbf{R}\boldsymbol{\omega}_b^\times \in T_R SO(3) \quad (1709)$$

where $\boldsymbol{\omega}_b^\times = \mathbf{R}^T \dot{\mathbf{R}}$ and $T_R SO(3)$ is the tangent space of $SO(3)$ at \mathbf{R} . The tangent space at the identity $\mathbf{R} = \mathbf{I}$ is $T_I SO(3) = so(3)$. This is in agreement with

$$\dot{\mathbf{R}}|_{\mathbf{R}=\mathbf{I}} = \boldsymbol{\omega}_b^\times \in so(3) \quad (1710)$$

The time derivative of the rotation matrix can also be written

$$\dot{\mathbf{R}} = \boldsymbol{\omega}_s^\times \mathbf{R} \in T_R SO(3) \quad (1711)$$

where $\boldsymbol{\omega}_s^\times = \mathbf{R} \boldsymbol{\omega}_b^\times \mathbf{R}^T = \dot{\mathbf{R}} \mathbf{R}^T$, which at the identity gives

$$\dot{\mathbf{R}}|_{\mathbf{R}=\mathbf{I}} = \boldsymbol{\omega}_s^\times \in so(3) \quad (1712)$$

This shows that both $\boldsymbol{\omega}_b^\times$ and $\boldsymbol{\omega}_s^\times$ are in $so(3)$.

It is noted that if $\boldsymbol{\sigma}^\times \in so(3)$, then $\exp(\boldsymbol{\sigma}^\times) \in SO(3)$. Moreover, if $\mathbf{R} = \exp(\theta \mathbf{k}^\times) \in SO(3)$, then $\log(\mathbf{R}) = \theta \mathbf{k}^\times \in so(3)$.

Let \mathbf{k} be a constant unit vector. Then $\exp(\theta \mathbf{k}^\times)$ is called a one-parameter sub-group of $SO(3)$.

24.3 Bi-invariance of the Riemannian metric on $SO(3)^*$

In a previous section the Riemannian metric on $SO(3)$ was defined as

$$\langle \dot{\mathbf{R}}, \dot{\mathbf{R}} \rangle = \frac{1}{2} \text{tr}(\dot{\mathbf{R}}^T \dot{\mathbf{R}}) \quad (1713)$$

Insertion of $\dot{\mathbf{R}} = \mathbf{R} \boldsymbol{\omega}_b^\times$ then gives

$$\langle \dot{\mathbf{R}}, \dot{\mathbf{R}} \rangle = -\frac{1}{2} \text{tr}(\boldsymbol{\omega}_b^\times \mathbf{R}^T \mathbf{R} \boldsymbol{\omega}_b^\times) = -\frac{1}{2} \text{tr}(\boldsymbol{\omega}_b^\times \boldsymbol{\omega}_b^\times) = \boldsymbol{\omega}_b^T \boldsymbol{\omega}_b \quad (1714)$$

In the same way insertion of $\dot{\mathbf{R}} = \boldsymbol{\omega}_s^\times \mathbf{R}$ gives

$$\langle \dot{\mathbf{R}}, \dot{\mathbf{R}} \rangle = -\frac{1}{2} \text{tr}(\mathbf{R}^T \boldsymbol{\omega}_s^\times \boldsymbol{\omega}_s^\times \mathbf{R}) = -\frac{1}{2} \text{tr}(\boldsymbol{\omega}_s^\times \boldsymbol{\omega}_s^\times) = \boldsymbol{\omega}_s^T \boldsymbol{\omega}_s \quad (1715)$$

and it is seen that the Riemannian metric is given by

$$\langle \dot{\mathbf{R}}, \dot{\mathbf{R}} \rangle = \boldsymbol{\omega}_s^T \boldsymbol{\omega}_s = \boldsymbol{\omega}_b^T \boldsymbol{\omega}_b \quad (1716)$$

This property of $SO(3)$ where the Riemannian metric is the same when it is given in term of the spatial velocity $\boldsymbol{\omega}_s$ and in the body velocity $\boldsymbol{\omega}_b$ is referred to as the bi-invariance of the Riemannian metric for $SO(3)$ in [51].

It is noted that $\dot{\mathbf{R}} \in T_R SO(3)$ and $\boldsymbol{\omega}_s^\times, \boldsymbol{\omega}_b^\times \in so(3)$, where $so(3) = T_I SO(3)$ is the set of skew-symmetric matrices of dimension 3×3 . Let $\mathbf{W}^\times = \mathbf{W} \in so(3)$ be a skew symmetric matrix. The Riemannian metric induces the norm

$$\|\mathbf{W}\|^2 = \langle \mathbf{W}, \mathbf{W} \rangle \quad (1717)$$

This norm is bi-invariant since

$$\langle \mathbf{R}_l \mathbf{W} \mathbf{R}_r, \mathbf{R}_l \mathbf{W} \mathbf{R}_r \rangle = \frac{1}{2} \text{tr}(\mathbf{R}_r^T \mathbf{W}^T \mathbf{R}_l^T \mathbf{R}_l \mathbf{W} \mathbf{R}_r) \quad (1718)$$

$$= \frac{1}{2} \text{tr}(\mathbf{R}_r^T \mathbf{W}^T \mathbf{W} \mathbf{R}_r) \quad (1719)$$

$$= \frac{1}{2} \text{tr}(\mathbf{R}_r \mathbf{R}_r^T \mathbf{W}^T \mathbf{W}) \quad (1720)$$

$$= \frac{1}{2} \text{tr}(\mathbf{W}^T \mathbf{W}) \quad (1721)$$

$$= \langle \mathbf{W}, \mathbf{W} \rangle \quad (1722)$$

for any rotation matrices $\mathbf{R}_l, \mathbf{R}_r \in SO(3)$. Therefore, a requirement for the bi-invariance of the Riemannian metric is that

$$\omega_s^T \omega_s = \langle \dot{\mathbf{R}} \mathbf{R}^T, \dot{\mathbf{R}} \mathbf{R}^T \rangle = \langle \mathbf{R}^T \dot{\mathbf{R}}, \mathbf{R}^T \dot{\mathbf{R}} \rangle = \omega_b^T \omega_b \quad (1723)$$

A more general definition of the Riemannian metric [51, 37] includes a symmetric positive define matrix \mathbf{M} and is given by

$$\langle \dot{\mathbf{R}}, \dot{\mathbf{R}} \rangle_M = \omega_b^T \mathbf{M} \omega_b \quad (1724)$$

Note that the Riemannian metric is defined on the tangent plane $T_R SO(3)$ at \mathbf{R} , while it is computed as $\omega_b^T \mathbf{M} \omega_b$ in $so(3)$, which is the tangent plane at the identity. Moreover, this is a left invariant formulation since $\omega_b = \mathbf{R}^T \dot{\mathbf{R}}$ which is unchanged if \mathbf{R} is replaced by $\mathbf{R}_l \mathbf{R}_r$.

The Riemannian metric can also be given as

$$\langle \dot{\mathbf{R}}, \dot{\mathbf{R}} \rangle_M = \omega_s^T \mathbf{M} \omega_s \quad (1725)$$

where the angular velocity is in the spatial frame, which is a right invariant formulation since $\omega_s = \dot{\mathbf{R}} \mathbf{R}^T$ is unchanged if \mathbf{R} is replaced by $\mathbf{R} \mathbf{R}_r$. It follows that a condition for the Riemannian metric to be bi-invariant is that

$$\omega_s^T \mathbf{M} \omega_s = \omega_b^T \mathbf{M} \omega_b \quad (1726)$$

Then, since $\omega_s = \mathbf{R} \omega_b$ it follows that bi-invariance requires that

$$\mathbf{R}^T \mathbf{M} \mathbf{R} = \mathbf{M} \quad (1727)$$

which implies that the Riemannian metric on $SO(3)$ can only be bi-invariant when

$$\mathbf{M} = \alpha \mathbf{I} \quad (1728)$$

24.4 Riemannian metrics on $SE(3)$

In this section Riemannian metrics are discussed for $SE(3)$ based on [49, 51]. In $SO(3)$ a bi-invariant Riemannian metric was defined from the Frobenius norm. In $SE(3)$ this cannot be done. In fact there is no bi-invariant Riemannian metric on $SE(3)$, which is shown in the following.

A homogeneous transformation matrix $\mathbf{T} = \mathbf{T}_b^s$ is considered. The twist in the body frame is given by $\bar{\mathbf{w}}_{/b} = \mathbf{T}^{-1} \dot{\mathbf{T}}$. This twist is left invariant. This is shown by pre-multiplying \mathbf{T} by a constant matrix $\mathbf{T}_l \in SE(3)$, which gives $\mathbf{T}_l \mathbf{T} \in SE(3)$ the twist in the body frame is then the same as for \mathbf{T} , which is seen from

$$(\mathbf{T}_l \mathbf{T})^{-1} \frac{d}{dt} (\mathbf{T}_l \mathbf{T}) = \mathbf{T}^{-1} \mathbf{T}_l^{-1} \mathbf{T}_l \dot{\mathbf{T}} = \mathbf{T}^{-1} \dot{\mathbf{T}} = \bar{\mathbf{w}}_{/b} \quad (1729)$$

The twist in the body frame is not right invariant since, for constant $\mathbf{T}_r \in SE(3)$,

$$(\mathbf{T} \mathbf{T}_r)^{-1} \frac{d}{dt} (\mathbf{T} \mathbf{T}_r) = \mathbf{T}_r^{-1} \mathbf{T}^{-1} \dot{\mathbf{T}} \mathbf{T}_r = \mathbf{T}_r^{-1} \bar{\mathbf{w}}_{/b} \mathbf{T}_r \quad (1730)$$

The twist $\bar{\mathbf{w}}_{/s} = \dot{\mathbf{T}}\mathbf{T}^{-1}$ in the spatial frame is right invariant, but not left invariant which is seen from

$$\frac{d}{dt}(\mathbf{T}_l\mathbf{T})(\mathbf{T}_l\mathbf{T})^{-1} = \mathbf{T}_l\dot{\mathbf{T}}\mathbf{T}^{-1}\mathbf{T}_l^{-1} = \mathbf{T}^{-1}\dot{\mathbf{T}} = \mathbf{T}_l\bar{\mathbf{w}}_{/s}\mathbf{T}_l^{-1} \quad (1731)$$

The twist in the body frame is not right invariant since, for constant $\mathbf{T}_r \in SE(3)$,

$$\frac{d}{dt}(\mathbf{T}\mathbf{T}_r)(\mathbf{T}\mathbf{T}_r)^{-1} = \dot{\mathbf{T}}\mathbf{T}_r\mathbf{T}_r^{-1}\mathbf{T}^{-1} = \dot{\mathbf{T}}\mathbf{T}^{-1} = \bar{\mathbf{w}}_{/s} \quad (1732)$$

A left-invariant Riemannian metric on the tangent space $se(3)$ can be expressed in terms of the left invariant twist in the body frame, which in vector form is $\mathbf{w}_{/b} = [\boldsymbol{\omega}_b^T, \mathbf{v}_b^T]^T$. The left-invariant Riemannian metric is therefore given by

$$\langle \mathbf{T}^{-1}\dot{\mathbf{T}}, \mathbf{T}^{-1}\dot{\mathbf{T}} \rangle = \begin{bmatrix} \boldsymbol{\omega}_b \\ \mathbf{v}_b \end{bmatrix}^T \begin{bmatrix} \mathbf{A}_b & \mathbf{B}_b^T \\ \mathbf{B}_b & \mathbf{C}_b \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_b \\ \mathbf{v}_b \end{bmatrix} \quad (1733)$$

A right-invariant Riemannian metric must be expressed in terms of the right-invariant twist $\bar{\mathbf{w}}_{/s} = \dot{\mathbf{T}}\mathbf{T}^{-1}$, which is given in the spatial frame as $\mathbf{w}_{/s} = [\boldsymbol{\omega}_s^T, \mathbf{v}_s^T]^T$. The Riemannian metric is then

$$\langle \dot{\mathbf{T}}\mathbf{T}^{-1}, \dot{\mathbf{T}}\mathbf{T}^{-1} \rangle = \begin{bmatrix} \boldsymbol{\omega}_s \\ \mathbf{v}_s \end{bmatrix}^T \begin{bmatrix} \mathbf{A}_s & \mathbf{B}_s^T \\ \mathbf{B}_s & \mathbf{C}_s \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_s \\ \mathbf{v}_s \end{bmatrix} \quad (1734)$$

Then, for the Riemannian metric to be bi-invariant, it is required that

$$\langle \mathbf{T}^{-1}\dot{\mathbf{T}}, \mathbf{T}^{-1}\dot{\mathbf{T}} \rangle = \langle \dot{\mathbf{T}}\mathbf{T}^{-1}, \dot{\mathbf{T}}\mathbf{T}^{-1} \rangle \quad (1735)$$

which gives

$$\begin{bmatrix} \boldsymbol{\omega}_s \\ \mathbf{v}_s \end{bmatrix}^T \begin{bmatrix} \mathbf{A}_s & \mathbf{B}_s^T \\ \mathbf{B}_s & \mathbf{C}_s \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_s \\ \mathbf{v}_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega}_b \\ \mathbf{v}_b \end{bmatrix}^T \begin{bmatrix} \mathbf{A}_b & \mathbf{B}_b^T \\ \mathbf{B}_b & \mathbf{C}_b \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_b \\ \mathbf{v}_b \end{bmatrix} \quad (1736)$$

This means that the Riemannian metric can only be bi-invariant if

$$\begin{bmatrix} \mathbf{A}_s & \mathbf{B}_s^T \\ \mathbf{B}_s & \mathbf{C}_s \end{bmatrix} = \begin{bmatrix} \mathbf{A}_b & \mathbf{B}_b^T \\ \mathbf{B}_b & \mathbf{C}_b \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \quad (1737)$$

for a set of matrices \mathbf{A} , \mathbf{B} and \mathbf{C} . The screw transformation of the twist is given in vector form as

$$\begin{bmatrix} \boldsymbol{\omega}_s \\ \mathbf{v}_s \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{p}^{\times}\mathbf{R} & \mathbf{R} \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_b \\ \mathbf{v}_b \end{bmatrix} \quad (1738)$$

This means that bi-invariance leads to the condition

$$\begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{p}^{\times}\mathbf{R} & \mathbf{R} \end{bmatrix}^T \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{p}^{\times}\mathbf{R} & \mathbf{R} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{C} \end{bmatrix} \quad (1739)$$

This gives

$$\mathbf{A} = \mathbf{R}^T \mathbf{A} \mathbf{R} - \mathbf{R}^T \mathbf{p}^{\times} \mathbf{B} \mathbf{R} + \mathbf{R}^T \mathbf{B}^T \mathbf{p}^{\times} \mathbf{R} - \mathbf{R}^T \mathbf{p}^{\times} \mathbf{C} \mathbf{p}^{\times} \mathbf{R} \quad (1740)$$

$$\mathbf{B} = \mathbf{R}^T \mathbf{B} \mathbf{R} + \mathbf{R}^T \mathbf{C} \mathbf{p}^{\times} \mathbf{R} \quad (1741)$$

$$\mathbf{C} = \mathbf{R}^T \mathbf{C} \mathbf{R} \quad (1742)$$

This set of equations is only satisfied when $\mathbf{A} = \alpha\mathbf{I}$, $\mathbf{B} = \beta\mathbf{I}$ and $\mathbf{C} = \mathbf{0}$, or when $\mathbf{A} = \mathbf{0}$, $\mathbf{B} = \mathbf{0}$ and $\mathbf{C} = \gamma\mathbf{I}$, that is for

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{C} \end{bmatrix} = \begin{bmatrix} \alpha\mathbf{I} & \beta\mathbf{I} \\ \beta\mathbf{I} & \mathbf{0} \end{bmatrix} \text{ or } \begin{bmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{C} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \gamma\mathbf{I} \end{bmatrix} \quad (1743)$$

These two solutions are not positive definite, and do not provide a Riemannian metric that can be used to define a distance measure. This shows that there is no bi-invariant metric on $SE(3)$. This in turn implies that there is no bi-invariant distance function on $SE(3)$ [51].

24.5 Distance functions on $SE(3)$

A further complication for $SE(3)$ is that there is no simple way to find geodesic motions on $SE(3)$. In particular, if $\bar{\xi} \in se(3)$ is constant in the tangent plane $se(3)$, then the one-parameter subgroup $\exp(t\bar{\xi}) \in SE(3)$ will not be a geodesic on $SE(3)$, that is, the one-parameter subgroup will not be a solution of a set of equations of motion on $SE(3)$.

Instead of finding a distance function on $SE(3)$ based on geodesic motions it is proposed by [51] to find a solution by describing $SE(3)$ as the product space $\mathbb{R}^3 \times SO(3)$. Then the distance function can be found by finding the geodesics on $SO(3)$ and the shortest path in \mathbb{R}^3 , and then define a distance function based on this.

Consider two displacements $\mathbf{T}_1, \mathbf{T}_2 \in SE(3)$ where

$$\mathbf{T}_1 = \begin{bmatrix} \mathbf{R}_1 & \mathbf{p}_1 \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_2 & \mathbf{p}_2 \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1744)$$

The rotational distance is computed from

$$d_g(\mathbf{R}_1, \mathbf{R}_2) = |\log(\mathbf{R}_1^T \mathbf{R}_2)| \quad (1745)$$

Let the position metric be the usual distance function \mathbb{R}^3 which is computed from

$$d_p(\mathbf{p}_1, \mathbf{p}_2) = \|\mathbf{p}_2 - \mathbf{p}_1\| \quad (1746)$$

The a left-invariant metric on $SE(3)$ can be defined by

$$d_L(\mathbf{T}_1, \mathbf{T}_2) = \sqrt{\alpha d_g^2(\mathbf{R}_1, \mathbf{R}_2) + \beta d_p^2(\mathbf{p}_1, \mathbf{p}_2)} \quad (1747)$$

Left-invariance is established as follows: Consider $d_L(\mathbf{T}\mathbf{T}_1, \mathbf{T}\mathbf{T}_2)$ where

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1748)$$

Then

$$\mathbf{T}\mathbf{T}_1 = \begin{bmatrix} \mathbf{R}\mathbf{R}_1 & \mathbf{R}\mathbf{p}_1 + \mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{T}\mathbf{T}_2 = \begin{bmatrix} \mathbf{R}\mathbf{R}_2 & \mathbf{R}\mathbf{p}_2 + \mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1749)$$

and it follows that

$$d_L(\mathbf{T}\mathbf{T}_1, \mathbf{T}\mathbf{T}_2) = \sqrt{\alpha d_g^2(\mathbf{R}\mathbf{R}_1, \mathbf{R}\mathbf{R}_2) + \beta d_p^2(\mathbf{R}\mathbf{p}_1 + \mathbf{p}, \mathbf{R}\mathbf{p}_2 + \mathbf{p})} \quad (1750)$$

The rotation terms will not change as the rotational distance is left invariant, while the position metric is

$$d_p(\mathbf{R}\mathbf{p}_1 + \mathbf{p}, \mathbf{R}\mathbf{p}_2 + \mathbf{p}) = \|\mathbf{R}\mathbf{p}_2 + \mathbf{p} - (\mathbf{R}\mathbf{p}_1 + \mathbf{p})\| = \|\mathbf{R}(\mathbf{p}_2 - \mathbf{p}_1)\| = \|\mathbf{p}_2 - \mathbf{p}_1\| = d_p(\mathbf{p}_1, \mathbf{p}_2) \quad (1751)$$

this shows that $d_L(\mathbf{T}_1, \mathbf{T}_2)$ is left invariant.

It is noted that the left-invariance implies that $d_L(\mathbf{T}_1, \mathbf{T}_2) = d_L(\mathbf{I}, \mathbf{T}_1^{-1}\mathbf{T}_2)$, which results by using $\mathbf{T} = \mathbf{T}_1^T$. Therefore the difference between \mathbf{T}_1 and \mathbf{T}_2 is the same as the difference between \mathbf{I} and

$$\mathbf{T}_1^{-1}\mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_1^T \mathbf{R}_2 & \mathbf{R}_1^T(\mathbf{p}_2 - \mathbf{p}_1) \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1752)$$

A right-invariant metric on $SE(3)$ is given by

$$d_R(\mathbf{T}_1, \mathbf{T}_2) = \sqrt{\alpha d_g^2(\mathbf{R}_1, \mathbf{R}_2) + \beta d_p^2(\mathbf{R}_1^T \mathbf{p}_1, \mathbf{R}_2^T \mathbf{p}_2)} \quad (1753)$$

Right-invariance is shown by calculating the distance between the two displacements

$$\mathbf{T}_1\mathbf{T} = \begin{bmatrix} \mathbf{R}_1\mathbf{R} & \mathbf{p}_1 + \mathbf{R}_1\mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{T}_2\mathbf{T} = \begin{bmatrix} \mathbf{R}_2\mathbf{R} & \mathbf{p}_2 + \mathbf{R}_2\mathbf{p} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1754)$$

which is

$$d_R(\mathbf{T}_1\mathbf{T}, \mathbf{T}_2\mathbf{T}) = \sqrt{\alpha d_g^2(\mathbf{R}_1\mathbf{R}, \mathbf{R}_2\mathbf{R}) + \beta d_p^2(\mathbf{R}_1^T(\mathbf{p}_1 + \mathbf{R}_1\mathbf{p}), \mathbf{R}_2^T(\mathbf{p}_2 + \mathbf{R}_2\mathbf{p}))} \quad (1755)$$

The rotational distance is unchanged due to its right-invariance, while the position metric satisfies

$$d_p(\mathbf{R}_1^T(\mathbf{p}_1 + \mathbf{R}_1\mathbf{p}), \mathbf{R}_2^T(\mathbf{p}_2 + \mathbf{R}_2\mathbf{p})) = d_p(\mathbf{R}_1^T \mathbf{p}_1, \mathbf{R}_2^T \mathbf{p}_2) \quad (1756)$$

and it follows that $d_R(\mathbf{T}_1, \mathbf{T}_2)$ is right invariant. For the right-invariant metric it follows that $d_R(\mathbf{T}_1, \mathbf{T}_2) = d_R(\mathbf{I}, \mathbf{T}_2\mathbf{T}_1^{-1})$, where

$$\mathbf{T}_2\mathbf{T}_1^{-1} = \begin{bmatrix} \mathbf{R}_2\mathbf{R}_1^T & \mathbf{p}_2 - \mathbf{R}_2\mathbf{R}_1^T \mathbf{p}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1757)$$

It is straightforward to show that d_L is not right invariant, and that d_R is not left invariant, therefore the two distance functions d_L and d_R are not bi-invariant. This agrees with the fact that there are no bi-invariant distance functions on $SE(3)$.

Example

Let two displacements be given by

$$\mathbf{T}_1 = \begin{bmatrix} \mathbf{I} & \mathbf{e}_x \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_z(\pi/2) & \mathbf{e}_x \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1758)$$

where $\mathbf{e}_x = [1, 0, 0]^T$ is the unit vector in the x direction, and $\mathbf{R}_z(\pi/2)$ is a rotation by an angle $\pi/2$ about the z axis. Then the left-invariant distance with scaling $\alpha = 1$ and $\beta = 1$ is

$$d_L(\mathbf{T}_1, \mathbf{T}_2) = \sqrt{d_g^2(\mathbf{I}, \mathbf{R}_z(\pi/2)) + d_p^2(\mathbf{e}_x, \mathbf{e}_x)} = \sqrt{\left(\frac{\pi}{2}\right)^2} = 1.57 \quad (1759)$$

The right-invariant distance with the same scaling is

$$d_R(\mathbf{T}_1, \mathbf{T}_2) = \sqrt{d_g^2(\mathbf{I}, \mathbf{R}_z(\pi/2)) + d_p^2(\mathbf{e}_x, \mathbf{R}_z(-\pi/2)\mathbf{e}_x)} = \sqrt{\left(\frac{\pi}{2}\right)^2 + (\sqrt{2})^2} = 2.11 \quad (1760)$$

It is seen that the right-invariant distance has a contribution from the position difference, whereas the left-invariant distance has not.

To gain further insight, it is noted that the left-invariant distance can be written in the form $d_L(\mathbf{I}, \mathbf{T}_L)$ where

$$\mathbf{T}_L = \mathbf{T}_1^{-1}\mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_1^T \mathbf{R}_2 & \mathbf{R}_1^T(\mathbf{p}_2 - \mathbf{p}_1) \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1761)$$

Here the translations are in the same frame. Then the translations cancel out, and there is no contribution to the distance function from translation $\mathbf{R}_1^T(\mathbf{p}_2 - \mathbf{p}_1) = \mathbf{0}$.

The right-invariant distance can be written $d_R(\mathbf{I}, \mathbf{T}_R)$ where

$$\mathbf{T}_R = \mathbf{T}_2\mathbf{T}_1^{-1} = \begin{bmatrix} \mathbf{R}_2\mathbf{R}_1^T & \mathbf{p}_2 - \mathbf{R}_2\mathbf{R}_1^T\mathbf{p}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1762)$$

Here the translations \mathbf{p}_2 and $\mathbf{R}_2\mathbf{R}_1^T\mathbf{p}_1$ will have a difference in rotation of $\pi/2$, and a difference in the distance function appears due to the translation $\mathbf{p}_2 - \mathbf{R}_2\mathbf{R}_1^T\mathbf{p}_1 = [1, -1, 0]^T$. \square

24.6 A closer look at the left-invariant distance metric on $SE(3)$

Consider two displacements $\mathbf{T}_{b_1}^a, \mathbf{T}_{b_2}^a \in SE(3)$ where

$$\mathbf{T}_{b_1}^a = \begin{bmatrix} \mathbf{R}_{b_1}^a & \mathbf{p}_{ab_1}^a \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{T}_{b_2}^a = \begin{bmatrix} \mathbf{R}_{b_2}^a & \mathbf{p}_{ab_2}^a \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1763)$$

In this description the two displacements are from a common frame a . The difference between the two displacements is described in terms of the different displaced frames, which is b_1 for $\mathbf{T}_{b_1}^a$ and b_2 for $\mathbf{T}_{b_2}^a$.

The distance of the displacement in $SO(3)$ is computed from

$$d_g(\mathbf{R}_{b_1}^a, \mathbf{R}_{b_2}^a) = |\log((\mathbf{R}_{b_1}^a)^T \mathbf{R}_{b_2}^a)| = |\log \mathbf{R}_{b_2}^{b_1}| \quad (1764)$$

The distance in \mathbb{R}^3 is given by

$$d_p(\mathbf{p}_{ab_1}^a, \mathbf{p}_{ab_2}^a) = \|\mathbf{p}_{ab_2}^a - \mathbf{p}_{ab_1}^a\| = \|\mathbf{p}_{b_1 b_2}^a\| = \|\mathbf{p}_{b_1 b_2}^b\| \quad (1765)$$

where it is used that the length of a vector is unchanged under a coordinate transformation.

A left-invariant metric on $SE(3)$ is then given by

$$d_L(\mathbf{T}_{b_1}^a, \mathbf{T}_{b_2}^a) = \sqrt{\alpha d_g^2(\mathbf{R}_{b_1}^a, \mathbf{R}_{b_2}^a) + \beta d_p^2(\mathbf{p}_{ab_1}^a, \mathbf{p}_{ab_2}^a)} \quad (1766)$$

It is noted that

$$d_L(\mathbf{T}_{b_1}^a, \mathbf{T}_{b_2}^a) = \sqrt{\alpha |\log(\mathbf{R}_{b_2}^{b_1})|^2 + \beta \|\mathbf{p}_{b_1 b_2}^b\|^2} \quad (1767)$$

It is seen that the left-invariant distance is independent of the starting frame of the two displacements, that is, it is invariant to a change in the starting frame from a to a_0 by a left multiplication $\mathbf{T}_{b_i}^{a_0} = \mathbf{T}_a^{a_0} \mathbf{T}_{b_i}^a$, $i = 1, 2$, which would give

$$d_L(\mathbf{T}_{b_1}^{a_0}, \mathbf{T}_{b_2}^{a_0}) = d_L(\mathbf{T}_a^{a_0} \mathbf{T}_{b_1}^a, \mathbf{T}_a^{a_0} \mathbf{T}_{b_2}^a) = d_L(\mathbf{T}_{b_1}^a, \mathbf{T}_{b_2}^a) \quad (1768)$$

This is also in agreement with

$$d_L(\mathbf{T}_{b_1}^a, \mathbf{T}_{b_2}^a) = d_L(\mathbf{I}, \mathbf{T}_{b_2}^{b_1}) \quad (1769)$$

which is invariant to a change of the starting frame by a left multiplication.

24.7 A closer look at the right-invariant distance metric on $SE(3)$

Consider two displacements

$$\mathbf{T}_b^{a_1} = \begin{bmatrix} \mathbf{R}_b^{a_1} & \mathbf{p}_{a_1 b}^{a_1} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{T}_b^{a_2} = \begin{bmatrix} \mathbf{R}_b^{a_2} & \mathbf{p}_{a_2 b}^{a_2} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1770)$$

In this case the two displacements has the same displaced frame, which is b . The difference of the two displacements is described in terms of the different starting frames, which is a_1 for $\mathbf{T}_b^{a_1}$ and a_2 for $\mathbf{T}_b^{a_2}$. The rotational distance of the displacement in $SO(3)$ is computed from

$$d_g(\mathbf{R}_b^{a_1}, \mathbf{R}_b^{a_2}) = |\log(\mathbf{R}_b^{a_1} (\mathbf{R}_b^{a_2})^T)| = |\log \mathbf{R}_{a_2}^{a_1}| \quad (1771)$$

The distance in \mathbb{R}^3 is computed from

$$d_p((\mathbf{R}_b^{a_1})^T \mathbf{p}_{a_1 b}^{a_1}, (\mathbf{R}_b^{a_2})^T \mathbf{p}_{a_2 b}^{a_2}) = d_p(\mathbf{p}_{a_1 b}^b, \mathbf{p}_{a_2 b}^b) \quad (1772)$$

The right-invariant metric on $SE(3)$ is the given by

$$d_R(\mathbf{T}_b^{a_1}, \mathbf{T}_b^{a_2}) = \sqrt{\alpha d_g^2(\mathbf{R}_b^{a_1}, \mathbf{R}_b^{a_2}) + \beta d_p^2(\mathbf{p}_{a_1 b}^b, \mathbf{p}_{a_2 b}^b)} \quad (1773)$$

It is noted that in the same way as for the left-invariant distance, the right-invariant distance will satisfy

$$d_R(\mathbf{T}_b^{a_1}, \mathbf{T}_b^{a_2}) = \sqrt{\alpha |\log(\mathbf{R}_{a_2}^{a_1})|^2 + \beta \|\mathbf{p}_{a_1 a_2}^a\|^2} \quad (1774)$$

which which is independent of the common displaced frame b . It follows that the right-invariant distance is invariant to a change in the displaced frame by a right multiplication $\mathbf{T}_{b_0}^{a_i} = \mathbf{T}_b^{a_i} \mathbf{T}_{b_0}^b$, which would give

$$d_R(\mathbf{T}_{b_0}^{a_1}, \mathbf{T}_{b_0}^{a_2}) = d_R(\mathbf{T}_b^{a_1} \mathbf{T}_{b_0}^b, \mathbf{T}_b^{a_2} \mathbf{T}_{b_0}^b) = d_R(\mathbf{T}_b^{a_1}, \mathbf{T}_b^{a_2}) \quad (1775)$$

This result is also seen from

$$d_R(\mathbf{T}_b^{a_1}, \mathbf{T}_b^{a_2}) = d_R(\mathbf{I}, \mathbf{T}_b^{a_2} (\mathbf{T}_b^{a_1})^{-1}) = d_R(\mathbf{I}, \mathbf{T}_{a_1}^{a_2}) \quad (1776)$$

which is invariant to a change of the displaced frame by a right multiplication.

25 Determination of rotations and displacements from measurements

25.1 Calculation of rotation matrices in the orthogonal Procrustes problem

The orthogonal Procrustes problem [27] can be used to find a rotation matrix based on observations in the form of point mappings. A solution to this problem was presented by [4] and [75] for rotation matrices, and in [32] where unit quaternions were used. This problem is also referred to as Wahba's problem [77], which was formulated for the determination of the rotation of a satellite from star observations. Early solutions to this problem are the TRIAD and QUEST solutions which were developed in [66] for space applications.

Suppose that the model data is given in the form of vectors $\mathbf{a}_i \in \mathbb{R}^3$, while the observation data is given by $\mathbf{b}_i \in \mathbb{R}^3$ for $i = 1, \dots, n$, where it is assumed that the observation data is a result of the same rotation from the model data, so that

$$\mathbf{a}_i = \mathbf{R}\mathbf{b}_i, \quad i = 1, \dots, n \quad (1777)$$

where \mathbf{R} is a rotation matrix. The rotation matrix is found through the minimization problem

$$\min_{\mathbf{R}} L = \sum_{i=1}^n \|\mathbf{a}_i - \mathbf{R}\mathbf{b}_i\|^2 \quad (1778)$$

The cost function L can also be written in matrix form, so that the minimization problem can be written

$$\min_{\mathbf{R}} L = \|\mathbf{A} - \mathbf{R}\mathbf{B}\|_F^2 \quad (1779)$$

where

$$\mathbf{A} = (\mathbf{a}_1, \dots, \mathbf{a}_n) \in \mathbb{R}^{3 \times n} \quad \text{and} \quad \mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n) \in \mathbb{R}^{3 \times n}$$

and $\|\cdot\|_F$ is the Frobenius norm.

The optimal solution as presented in [4] is then found by computing the matrix

$$\mathbf{H} = \mathbf{B}\mathbf{A}^T \quad (1780)$$

and computing the singular value decomposition

$$\mathbf{H} = \mathbf{U}\Sigma\mathbf{V}^T \quad (1781)$$

The optimal rotation matrix is then

$$\mathbf{R} = \mathbf{V}\mathbf{S}\mathbf{U}^T \quad (1782)$$

where

$$\mathbf{S} = \text{diag}[1, 1, \det(\mathbf{V}\mathbf{U}^T)] \quad (1783)$$

Example

```

% Optimal Procrustes Problem: Script that solves for the optimal
% R when datapoints ai and bi are give so that ai = R*bi.
% Datapoints are arranged in matrices
% A = [a1,...,an] and B = [b1,...,bn] so that A = R*B

% Input: Data points
B = [1 0 0; 1 1 0; 1 1 1; 0 1 0; 0 1 1; 0 0 1]';
% Input: Actual rotation matrix
Ra = Rotz(pi/9)*Roty(pi/4)*Rotx(pi/6)
A = Ra*B; % Mapping of input data points
%A(1,1) = A(1,1) + 0.01; % Noise

H = B*A';
[U,S,V] = svd(H);

Rc = V*U' % Optimal least-squares solution

% Test: Identity matrix expected
deltaRtest = Rc*Ra';

```

□

25.2 Derivation of the solution of the Procrustes problem

The solution to the Procrustes problem is derived in this section. First it is noted that the Frobenius norm of a matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ satisfies $\|\mathbf{M}\|_F^2 = \text{tr}(\mathbf{M}^T \mathbf{M}) = \text{tr}(\mathbf{M} \mathbf{M}^T)$. Therefore, the cost function can be written

$$L = \|\mathbf{A} - \mathbf{R}\mathbf{B}\|_F^2 \quad (1784)$$

$$= \text{tr}((\mathbf{A} - \mathbf{R}\mathbf{B})(\mathbf{A} - \mathbf{R}\mathbf{B})^T) \quad (1785)$$

$$= \text{tr}(\mathbf{A}\mathbf{A}^T) + \text{tr}(\mathbf{B}\mathbf{B}^T) - 2\text{tr}(\mathbf{R}\mathbf{B}\mathbf{A}^T) \quad (1786)$$

This shows that the minimization of L with respect to \mathbf{R} , where \mathbf{R} is an orthogonal matrix, is equivalent to the maximization problem

$$\max_{\mathbf{R} \in O(3)} \text{tr}(\mathbf{R}\mathbf{H}) \quad (1787)$$

where $\mathbf{H} = \mathbf{B}\mathbf{A}^T$, which has the singular value decomposition $\mathbf{H} = \mathbf{U}\Sigma\mathbf{V}^T$.

The first step in the determination of the optimal $\mathbf{R} \in O(3)$ is to define the orthogonal matrix $\mathbf{Z} = \mathbf{V}^T \mathbf{R} \mathbf{U}$ where $\mathbf{R} \in O(3)$. The orthogonality of \mathbf{Z} is verified by

$$\mathbf{Z}^T \mathbf{Z} = \mathbf{U}^T \mathbf{R}^T \mathbf{V} \mathbf{V}^T \mathbf{R} \mathbf{U} = \mathbf{I} \quad (1788)$$

It follows that $\mathbf{R} = \mathbf{V} \mathbf{Z} \mathbf{U}^T$, and therefore

$$\text{tr}(\mathbf{R}\mathbf{H}) = \text{tr}(\mathbf{V} \mathbf{Z} \mathbf{U}^T \mathbf{U} \Sigma \mathbf{V}^T) = \text{tr}(\mathbf{V} \mathbf{Z} \Sigma \mathbf{V}^T) = \text{tr}(\mathbf{Z} \Sigma) \quad (1789)$$

The next step is to find the \mathbf{Z} that maximizes $\text{tr}(\mathbf{RH})$. To investigate this further, define the matrix $\mathbf{N} = (\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3)$ by the Cholesky decomposition $\mathbf{\Sigma} = \mathbf{NN}^T$. Then

$$\text{tr}(\mathbf{Z}\mathbf{\Sigma}) = \text{tr}(\mathbf{Z}\mathbf{NN}^T) = \text{tr}(\mathbf{N}^T\mathbf{ZN}) = \sum_{i=1}^3 \mathbf{n}_i^T \mathbf{Z} \mathbf{n}_i \quad (1790)$$

From Schwarz' inequality it follows that

$$(\mathbf{n}_i^T \mathbf{Z} \mathbf{n}_i)^2 = (\mathbf{n}_i^T (\mathbf{Z} \mathbf{n}_i))^2 \leq \|\mathbf{n}_i\|^2 \|\mathbf{Z} \mathbf{n}_i\|^2 = (\mathbf{n}_i^T \mathbf{n}_i)(\mathbf{n}_i^T \mathbf{Z}^T \mathbf{Z} \mathbf{n}_i) = (\mathbf{n}_i^T \mathbf{n}_i)^2 \quad (1791)$$

where it is used that $\mathbf{Z}^T \mathbf{Z} = \mathbf{I}$. This means that the maximum value for $\sum_{i=1}^3 \mathbf{n}_i^T \mathbf{Z} \mathbf{n}_i$ is achieved with $\mathbf{Z} = \mathbf{I}$, which therefore gives the maximum value for $\text{tr}(\mathbf{Z}\mathbf{\Sigma})$ as $\text{tr}(\mathbf{\Sigma})$. It follows that $\mathbf{R} = \mathbf{V}\mathbf{U}^T$, which is orthogonal, is the optimum value.

There remains a problem with this solution, since the solution is a rotation matrix if $\det(\mathbf{V}\mathbf{U}^T) = 1$, while it is a reflection matrix if $\det(\mathbf{V}\mathbf{U}^T) = -1$. There is a surprisingly simple solution to this problem due to [75], which is to use the solution

$$\mathbf{R} = \mathbf{V}\mathbf{S}\mathbf{U}^T \quad (1792)$$

where

$$\mathbf{S} = \text{diag}[1, 1, \det(\mathbf{V}\mathbf{U}^T)] \quad (1793)$$

This solution will be the optimal rotation matrix.

This is shown as follows based on the proof in [75]. The maximization problem is

$$\max_{\mathbf{R} \in SO(3)} \text{tr}(\mathbf{RH}) \quad (1794)$$

Suppose that $\mathbf{R} = \mathbf{V}\mathbf{S}\mathbf{U}^T$ where $\mathbf{S} = \text{diag}(s_1, s_2, s_3)$. It is noted that $\det(\mathbf{S}) = s_1 s_2 s_3$, and that

$$\det(\mathbf{R}) = \det(\mathbf{S}) \det(\mathbf{V}\mathbf{U}^T) \quad (1795)$$

Note that $|\det(\mathbf{S})| = 1$ to ensure that \mathbf{R} is orthogonal. This gives

$$\text{tr}(\mathbf{RH}) = \text{tr}(\mathbf{V}\mathbf{S}\mathbf{U}^T \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T) = \text{tr}(\mathbf{S}\mathbf{\Sigma}) = s_1 \sigma_1 + s_2 \sigma_2 + s_3 \sigma_3 \quad (1796)$$

where $\mathbf{\Sigma} = (\sigma_1, \sigma_2, \sigma_3)$ and $\sigma_1 \geq \sigma_2 \geq \sigma_3 > 0$. The maximization problem is then

$$\max_{\mathbf{R}} \text{tr}(\mathbf{RH}) = \max_{\mathbf{R}} (s_1 \sigma_1 + s_2 \sigma_2 + s_3 \sigma_3) \quad (1797)$$

Therefore, when $\det(\mathbf{V}\mathbf{U}^T) = 1$ the optimal solution is found for $s_1 = s_2 = s_3 = 1$. If $\det(\mathbf{V}\mathbf{U}^T) = -1$, then \mathbf{R} will be the optimal rotation matrix if $s_1 = s_2 = 1$ and $s_3 = -1$. The optimal solution is therefore found for

$$\mathbf{S} = (\text{diag}(1, 1, \det(\mathbf{V}\mathbf{U}^T))) \quad (1798)$$

and the rotation matrix is

$$\mathbf{R} = \mathbf{V}\mathbf{S}\mathbf{U}^T \quad (1799)$$

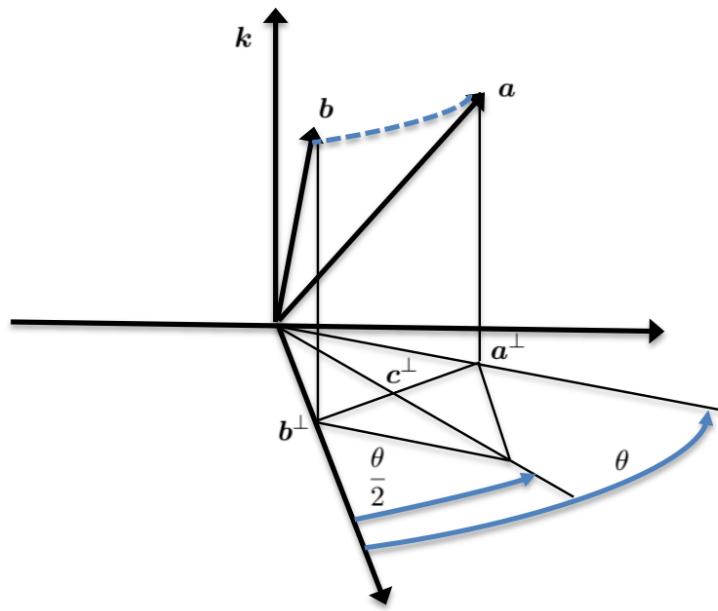


Figure 84: The vector \mathbf{b} is rotated to the vector $\mathbf{a} = \mathbf{R}\mathbf{b}$ by an angle θ about a rotation axis given by the unit vector \mathbf{k} through the origin. The projection of the vectors \mathbf{a} and \mathbf{b} to the plane normal to the rotation axis \mathbf{k} is denoted \mathbf{a}^\perp and \mathbf{b}^\perp , respectively. The mid-point between \mathbf{a}^\perp and \mathbf{b}^\perp is \mathbf{c}^\perp . The vector \mathbf{k} is normal to the line between \mathbf{a}^\perp and \mathbf{b}^\perp , and since $\mathbf{a} - \mathbf{b} = \mathbf{a}^\perp - \mathbf{b}^\perp$ it follows that \mathbf{k} is normal to $\mathbf{a} - \mathbf{b}$.

25.3 Number point correspondences in the Procrustes problem

In this section it is shown that at least two point correspondences $\mathbf{a}_i = \mathbf{R}\mathbf{b}_i$ are required to determine the rotation matrix. The presentation is based on [44].

Consider the rotation of a vector \mathbf{b} to a vector \mathbf{a} given by $\mathbf{a} = \mathbf{R}\mathbf{b}$. The rotation matrix \mathbf{R} can be represented by a rotation by an angle θ about a unit vector \mathbf{k} through the origin. In this derivation the conditions on the rotation angle θ and the rotation vector \mathbf{k} will be established.

Let θ and \mathbf{k} be any combination of rotation angle and rotation vector so that $\mathbf{a} = \mathbf{R}\mathbf{b}$ as shown in Figure 84. The plane through the origin which is normal to the rotation axis \mathbf{k} be called the rotation plane. Let \mathbf{a}^\perp and \mathbf{b}^\perp be the projection of the points \mathbf{a} and \mathbf{b} to the rotation plane. Let $\mathbf{c} = \frac{1}{2}(\mathbf{a} + \mathbf{b})$ be the midpoint between \mathbf{a} and \mathbf{b} , and let

$$\mathbf{c}^\perp = \frac{1}{2}(\mathbf{a}^\perp + \mathbf{b}^\perp) \quad (1800)$$

be the projection of \mathbf{c} to the rotation plane. Then from the geometry in the rotation plane it is seen that

$$\tan \frac{\theta}{2} = \frac{|\mathbf{a}^\perp - \mathbf{c}^\perp|}{|\mathbf{c}^\perp|} \quad (1801)$$

Moreover, $|\mathbf{k} \times \mathbf{c}^\perp| = |\mathbf{c}^\perp|$ since \mathbf{k} is normal to \mathbf{c}^\perp , and since $\mathbf{a}^\perp - \mathbf{c}^\perp$ is normal to \mathbf{c}^\perp , it follows that

$$\mathbf{a}^\perp - \mathbf{c}^\perp = \tan \frac{\theta}{2} \mathbf{k} \times \mathbf{c}^\perp \quad (1802)$$

Then it is used that

$$\mathbf{k} \times \mathbf{c}^\perp = \frac{1}{2} \mathbf{k} \times (\mathbf{a}^\perp + \mathbf{b}^\perp) = \frac{1}{2} \mathbf{k} \times (\mathbf{a} + \mathbf{b}) \quad (1803)$$

and that

$$\mathbf{a}^\perp - \mathbf{c}^\perp = \mathbf{a} - \mathbf{c} = \frac{1}{2}(\mathbf{a} - \mathbf{b}) \quad (1804)$$

From these two equations the condition

$$(\mathbf{a} - \mathbf{b}) = \tan \frac{\theta}{2} \mathbf{k} \times (\mathbf{a} + \mathbf{b}) \quad (1805)$$

is found for θ and \mathbf{k} given \mathbf{a} and \mathbf{b} . It is noted that any combination of θ and \mathbf{k} which satisfies this condition will give a rotation of \mathbf{b} to \mathbf{a} .

To characterize the possible rotations which satisfies the condition (1805) it is noted that any vector \mathbf{k} which is normal to $\mathbf{a} - \mathbf{b}$ is a valid solution of (1805). The corresponding rotation angle θ is determined by

$$\tan \frac{|\theta|}{2} = \frac{|\mathbf{a} - \mathbf{b}|}{|\mathbf{k} \times (\mathbf{a} + \mathbf{b})|} \quad (1806)$$

where $0 < \theta < \pi$ if $(\mathbf{k} \times (\mathbf{a} + \mathbf{b})) \cdot (\mathbf{a} - \mathbf{b}) > 0$ and $-\pi < \theta < 0$ if $(\mathbf{k} \times (\mathbf{a} + \mathbf{b})) \cdot (\mathbf{a} - \mathbf{b}) < 0$.

This means that \mathbf{k} can be any unit vector in the plane π through the origin which is normal to the vector $\mathbf{a} - \mathbf{b}$. The angle θ can be found when a vector \mathbf{k} in the plane π has been selected. It is noted that if \mathbf{k} is selected to be normal to $\mathbf{a} + \mathbf{b}$, then $\tan \frac{\theta}{2} = \frac{|\mathbf{a}-\mathbf{b}|}{|\mathbf{a}+\mathbf{b}|}$. If \mathbf{k} is selected to have a direction that tends to $\mathbf{a} + \mathbf{b}$, then $\tan \frac{\theta}{2}$ will tend to infinity, which means that $|\theta|$ will tend to π .

Note that given one correspondence \mathbf{a} and \mathbf{b} , the vector \mathbf{k} is only determined as a vector in the plane through the origin which is normal to $\mathbf{a} - \mathbf{b}$. If two correspondences $(\mathbf{a}_1, \mathbf{b}_1)$ and $(\mathbf{a}_2, \mathbf{b}_2)$ are given, then \mathbf{k} must be in the plane π_1 determined by $(\mathbf{a}_1, \mathbf{b}_1)$, and in the plane π_2 determined by $(\mathbf{a}_2, \mathbf{b}_2)$. This means that the rotation vector \mathbf{k} must be along the line at the intersection of the two planes π_1 and π_2 , which means that \mathbf{k} is uniquely determined under the condition that the two planes π_1 and π_2 are distinct, which will be the case if $\mathbf{a}_1 - \mathbf{b}_1 \neq \mathbf{a}_2 - \mathbf{b}_2$.

The conclusion is that the Procrustes problem can be solved if there are at least $n = 2$ point correspondences.

25.4 A vector formulation of the Procrustes problem

It is noted that if the vector description is used, then the same optimization problem is developed as follows:

$$\min_{\mathbf{R}} L = \sum_{i=1}^n \|\mathbf{a}_i - \mathbf{R}\mathbf{b}_i\|^2 = \sum_{i=1}^n (\mathbf{a}_i - \mathbf{R}\mathbf{b}_i)^T (\mathbf{a}_i - \mathbf{R}\mathbf{b}_i) \quad (1807)$$

$$= \sum_{i=1}^n (\mathbf{a}_i^T \mathbf{a}_i + \mathbf{b}_i^T \mathbf{R}^T \mathbf{R} \mathbf{b}_i - 2\mathbf{a}_i^T \mathbf{R} \mathbf{b}_i) \quad (1808)$$

$$= \sum_{i=1}^n (\mathbf{a}_i^T \mathbf{a}_i + \mathbf{b}_i^T \mathbf{b}_i - 2\mathbf{a}_i^T \mathbf{R} \mathbf{b}_i) \quad (1809)$$

which is equivalent to

$$\max_{\mathbf{R}} \sum_{i=1}^n \mathbf{a}_i^T \mathbf{R} \mathbf{b}_i = \text{tr} \left(\sum_{i=1}^n \mathbf{R} \mathbf{b}_i \mathbf{a}_i^T \right) = \text{tr}(\mathbf{R} \mathbf{H}) \quad (1810)$$

where

$$\mathbf{H} = \sum_{i=1}^n \mathbf{b}_i \mathbf{a}_i^T = \mathbf{B} \mathbf{A}^T \quad (1811)$$

□

25.5 The weighted Procrustes problem

In the weighted Procrustes problem the cost function is

$$L = \sum_{i=1}^n w_i \|\mathbf{a}_i - \mathbf{R}\mathbf{b}_i\|^2 \quad (1812)$$

where $w_i \geq 0$ is the weight of point mapping i . The minimization problem is then written

$$\min_{\mathbf{R}} L = \sum_{i=1}^n w_i \|\mathbf{a}_i - \mathbf{R}\mathbf{b}_i\|^2 \quad (1813)$$

$$= \sum_{i=1}^n w_i (\mathbf{a}_i - \mathbf{R}\mathbf{b}_i)^T (\mathbf{a}_i - \mathbf{R}\mathbf{b}_i) \quad (1814)$$

$$= \sum_{i=1}^n w_i (\mathbf{a}_i^T \mathbf{a}_i + \mathbf{b}_i^T \mathbf{b}_i - 2\mathbf{a}_i^T \mathbf{R} \mathbf{b}_i) \quad (1815)$$

This is equivalent to

$$\max_{\mathbf{R}} \sum_{i=1}^n w_i \mathbf{a}_i^T \mathbf{R} \mathbf{b}_i = \text{tr} \left(\sum_{i=1}^n \mathbf{R} \mathbf{b}_i w_i \mathbf{a}_i^T \right) = \text{tr}(\mathbf{R} \mathbf{H}) \quad (1816)$$

where

$$\mathbf{H} = \sum_{i=1}^n \mathbf{b}_i w_i \mathbf{a}_i^T = \mathbf{B} \mathbf{W} \mathbf{A}^T \quad (1817)$$

where

$$\mathbf{W} = \text{diag}(w_1, w_2, \dots, w_n) \quad (1818)$$

This is the same maximization problem $\max_{\mathbf{R}} \text{tr}(\mathbf{R} \mathbf{H})$ as in the usual Procrustes problem, except for the expression for \mathbf{H} . The solution is then found as in the usual case as $\mathbf{R} = \mathbf{V} \mathbf{S} \mathbf{U}^T$ where \mathbf{U} and \mathbf{V} are given by $\mathbf{H} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$.

25.6 Reflection and rotation matrices in the optimal Procrustes problem

A rotation matrix \mathbf{R} will satisfy the condition $\mathbf{R} \mathbf{R}^T = \mathbf{I}$, which ensures that the matrix is orthogonal and a member of the orthogonal group $O(3)$, and in addition, it will satisfy $\det \mathbf{R} = 1$, which makes it a member of the special orthogonal group $SO(3)$, which is the set of rotation matrices.

A reflection matrix \mathbf{Z} will satisfy the condition $\mathbf{Z} \mathbf{Z}^T = \mathbf{I}$, which makes it a member of the orthogonal group $O(3)$, however, the determinant for the reflection matrix is $\det \mathbf{Z} = -1$.

The row vectors \mathbf{r}_1 , \mathbf{r}_2 and \mathbf{r}_3 of a rotation matrix

$$\mathbf{R} = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{bmatrix} \quad (1819)$$

will form a right-hand system of orthogonal vectors where $\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$. In contrast to this the row vectors \mathbf{z}_1 , \mathbf{z}_2 and \mathbf{z}_3 of a reflection matrix

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1^T \\ \mathbf{z}_2^T \\ \mathbf{z}_3^T \end{bmatrix} \quad (1820)$$

will form a left-hand system of orthogonal vectors where $\mathbf{z}_3 = -\mathbf{z}_1 \times \mathbf{z}_2$.

To investigate this further, consider the situation where the rotation matrix \mathbf{R} and the reflection matrix \mathbf{Z} have the same row vectors in rows 1 and 2, so that $\mathbf{z}_1 = \mathbf{r}_1$ and $\mathbf{z}_2 = \mathbf{r}_2$ which implies that the last row of the matrices have opposite signs according to $\mathbf{z}_3 = -\mathbf{r}_3$. The reflection matrix is then

$$\mathbf{Z} = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ -\mathbf{r}_3^T \end{bmatrix} \quad (1821)$$

In the orthogonal Procrustes problem the transformation that is studied is $\mathbf{a} = \mathbf{R}\mathbf{b}$. In terms of the row vectors of the rotation matrix, this is written

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{bmatrix} \mathbf{b} = \begin{bmatrix} \mathbf{r}_1^T \mathbf{b} \\ \mathbf{r}_2^T \mathbf{b} \\ \mathbf{r}_3^T \mathbf{b} \end{bmatrix} \quad (1822)$$

where $\mathbf{a} = (a_1, a_2, a_3)^T$. The mapping $\mathbf{a} = \mathbf{Z}\mathbf{b}$ with a reflection matrix \mathbf{Z} gives

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ -\mathbf{r}_3^T \end{bmatrix} \mathbf{b} = \begin{bmatrix} \mathbf{r}_1^T \mathbf{b} \\ \mathbf{r}_2^T \mathbf{b} \\ -\mathbf{r}_3^T \mathbf{b} \end{bmatrix} \quad (1823)$$

The only difference between the two mappings is in the third coordinate a_3 of \mathbf{a} .

In the case of noisy data it is possible that the optimization may result the reflection matrix \mathbf{Z} if the data tends to give small a_3 component that is dominated by noise. To be precise, suppose that the point mappings are given by

$$\mathbf{a}_i = \mathbf{R}\mathbf{b}_i + \mathbf{n}_i, \quad i = 1, \dots, n \quad (1824)$$

where $\mathbf{a}_i = (a_{i1}, a_{i2}, a_{i3})^T$, and the vector $\mathbf{n}_i = (n_{i1}, n_{i2}, n_{i3})^T$ represents measurement noise. Then if the data gives small values a_{i3} that are dominated by the noise n_{i3} in the same coordinate, it is possible that the last coordinate will give a better fit to the reflection values $-\mathbf{r}_3^T \mathbf{b}_i$ than the rotation values $\mathbf{r}_3^T \mathbf{b}_i$ of the last coordinate, and a reflection matrix will result from the optimization. This analysis can be used to construct data that would lead to a reflection matrix as the result in the orthogonal Procrustes problem.

Example

```
% Optimal Procrustes Problem where noise gives a reflection matrix.
% The optimal orthogonal matrix R is found from datapoints ai and bi
% where ai = R*bi. The solution of Umeyama is demonstrated for data that
% gives a reflection matrix with the Procrustes solution.
% Datapoints are arranged in matrices A = [a1, ..., an] and
% B = [b1, ..., bn] so that A = R*B

% Input: 2 data points, rank(B*A') = 2
B = [0 1 0; 1 0 0.01]';
% Input: Noise
N = -[0 0 0; 0 0 0.02]'; %; 0 0 0; 0 0 0; 0 0 0.02; 0 0 0]';
% Input: Actual rotation matrix
Ra = Rotz(pi/9)
A = Ra*B + N;
H = B*A';
[U,S,V] = svd(H);
Rc = V*U' % Procrustes solution
detRc = det(Rc);
deltaRc = Rc*Ra'; % Test: Identity matrix expected
```

```

Rc_umeyama = V*diag([1,1,det(V*U')])*U' % Umeyama solution
detR_umeyama = det(Rc_umeyama);
deltaR_umeyama = Rc_umeyama*Ra'; % Test: Identity matrix expected

% Input: 6 data points, rank(B*A') = 3
B6 = [0 1 0.01; 0 0 0.01; 1 0 0; 1 1 0; 1 1 0.01; 0 1 0]';
% Input: Noise
N6 = -[0 0 0.02; 0 0 0.02; 0 0 0; 0 0 0; 0 0 0.02; 0 0 0]';
% Input: Actual rotation matrix
A6 = Ra*B6 + N6;

H6 = B6*A6';
[U6,S6,V6] = svd(H6);
Rc = V6*U6' % Procrustes solution
detRc = det(Rc);
deltaRc = Rc*Ra'; % Test: Identity matrix expected

Rc_umeyama = V6*diag([1,1,det(V6*U6')])*U6' % Umeyama solution
detR_umeyama = det(Rc_umeyama);
deltaR_umeyama = Rc_umeyama*Ra'; % Test: Identity matrix expected

```

□

25.7 Calculation of rigid displacement from two sets of 3D data

Vision systems can be used to calculate the displacement \mathbf{T}_b^a of a frame b with respect to a frame a by measuring the position of N points fixed in the rigid body with respect to a reference frame a , and then comparing this with the position of the same N points in the object frame b fixed in the rigid body. Then the displacement

$$\mathbf{T}_b^a = \begin{bmatrix} \mathbf{R}_b^a & \mathbf{t}_{ab}^a \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1825)$$

from a to b can be determined from the position data of a sufficient number of points in general position. Here $\mathbf{R} = \mathbf{R}_b^a$ is the rotation matrix from a to b , $\mathbf{t} = \mathbf{t}_{ab}^a$ is the translation from the origin of frame a to the origin of frame b in the coordinates of a . This means that the frame b results from the displacement \mathbf{T}_b^a of frame a . A fixed point on the rigid object will then have the same position in reference frame a before the displacement as it has in b after the displacement. The method in this section is based on [4, 75].

It is shown in [44] that the displacement \mathbf{T}_b^a can be determined with 2 point correspondences. This is done with screw theory as an extension of the result for rotations. In the case of noisy measurements the addition of more point mappings will improve accuracy.

Let \mathbf{x}_i denote the position of point i in object frame b , and let the same point have position

$$\mathbf{y}_i = \mathbf{R}\mathbf{x}_i + \mathbf{t} \quad (1826)$$

in reference frame a .

The problem to be solved is to find \mathbf{T} when the point positions \mathbf{x}_i and \mathbf{y}_i are given for $i = 1, \dots, N$. This is done in the minimization problem

$$\min_{\mathbf{T}} L_r = \sum_{i=1}^N \|\mathbf{y}_i - (\mathbf{R}\mathbf{x}_i + \mathbf{t})\|^2 \quad (1827)$$

The positions are written

$$\mathbf{y}_i = \bar{\mathbf{y}} + \delta\mathbf{y}_i, \quad \mathbf{x}_i = \bar{\mathbf{x}} + \delta\mathbf{x}_i \quad (1828)$$

where $\bar{\mathbf{y}} = \frac{1}{N} \sum_{i=1}^N \mathbf{y}_i$ and $\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i$ are the centroids of the two point sets. It is noted that

$$\frac{1}{N} \sum_{i=1}^N \delta\mathbf{y}_i = \mathbf{0}, \quad \frac{1}{N} \sum_{i=1}^N \delta\mathbf{x}_i = \mathbf{0} \quad (1829)$$

It follows that

$$\begin{aligned} \|\mathbf{y}_i - (\mathbf{R}\mathbf{x}_i + \mathbf{t})\|^2 &= \|\bar{\mathbf{y}} + \delta\mathbf{y}_i - \mathbf{R}(\bar{\mathbf{x}} + \delta\mathbf{x}_i) - \mathbf{t}\|^2 \\ &= \|(\bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}} - \mathbf{t}) + (\delta\mathbf{y}_i - \mathbf{R}\delta\mathbf{x}_i)\|^2 \\ &= \|\bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}} - \mathbf{t}\|^2 + \|\delta\mathbf{y}_i - \mathbf{R}\delta\mathbf{x}_i\|^2 \\ &\quad + 2(\bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}} - \mathbf{t})^T(\delta\mathbf{y}_i - \mathbf{R}\delta\mathbf{x}_i) \end{aligned} \quad (1830)$$

From (1829) it follows that

$$\sum_{i=1}^N \|\mathbf{y}_i - (\mathbf{R}\mathbf{x}_i + \mathbf{t})\|^2 = N\|\bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}} - \mathbf{t}\|^2 + \sum_{i=1}^N \|\delta\mathbf{y}_i - \mathbf{R}\delta\mathbf{x}_i\|^2 \quad (1831)$$

This can be minimized by minimizing each of the terms on the right hand side independently. The last term on the right hand side, which is

$$\sum_{i=1}^N \|\delta\mathbf{y}_i - \mathbf{R}\delta\mathbf{x}_i\|^2 = \sum_{i=1}^N (\delta\mathbf{y}_i^T \delta\mathbf{y}_i + \delta\mathbf{x}_i^T \delta\mathbf{x}_i - 2\delta\mathbf{y}_i^T \mathbf{R}\delta\mathbf{x}_i) \quad (1832)$$

is minimized with respect to \mathbf{R} . This is a standard Procrustes problem. The optimal value of \mathbf{R} is then used to compute the optimal translation from

$$\mathbf{t} = \bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}} \quad (1833)$$

which minimizes the first term on the right hand side of (1831) by setting it to zero. In this minimization procedure the minimum value is found for the second term, and the first term is set to zero, which gives an optimal solution.

The minimization problem for the rotation is therefore the usual Procrustes problem

$$\min_{\mathbf{R}} L_r = \sum_{i=1}^n \|\delta\mathbf{y}_i - \mathbf{R}\delta\mathbf{x}_i\|^2 = \text{tr}((\mathbf{A} - \mathbf{R}\mathbf{B})(\mathbf{A} - \mathbf{R}\mathbf{B})^T) \quad (1834)$$

where $\mathbf{A} = (\delta\mathbf{y}_1, \dots, \delta\mathbf{y}_n)$ and $\mathbf{B} = (\delta\mathbf{x}_1, \dots, \delta\mathbf{x}_n)$. The minimization of L_r is then equivalent to the maximization of $\text{tr}(\mathbf{R}\mathbf{H})$ where

$$\mathbf{H} = \mathbf{B}\mathbf{A}^T = \mathbf{U}\Sigma\mathbf{V}^T \quad (1835)$$

This is the same equation as in the optimal Procrustes problem, and the optimal solution is

$$\mathbf{R} = \mathbf{V}\mathbf{S}\mathbf{U}^T \quad (1836)$$

where the \mathbf{U} and \mathbf{V} matrices are found by the singular value decomposition of the 3×3 matrix, and the Umeyama correction

$$\mathbf{S} = \text{diag}[1, 1, \det(\mathbf{V}\mathbf{U}^T)] \quad (1837)$$

ensures that \mathbf{R} is a rotation matrix.

The translation is then found from

$$\mathbf{t} = \bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}} \quad (1838)$$

To sum up, the solution is

$$\mathbf{R} = \mathbf{V}\mathbf{S}\mathbf{U}^T \quad (1839)$$

$$\mathbf{t} = \bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}} \quad (1840)$$

It is shown with screw theory in [44] that displacement $(\mathbf{R}, \mathbf{t}) \in SE(3)$ is determined by two point correspondences. as in the case of a pure rotation.

Example

```
import numpy as np

def skewm(r):
    return np.array([[0,-r[2],r[1]], [r[2],0,-r[0]], [-r[1],r[0],0]])
def expso3(u):
    S = skewm(u); un = np.linalg.norm(u)
    return np.eye(3) + np.sinc(un/np.pi)*S \
        + 0.5*(np.sinc(un/(2*np.pi)))**2 * S@S

# Preparations
np.set_printoptions(formatter={'float': '{: 0.4f}'.format})
ex = np.array([1.,0.,0.]); ey = np.array([0.,1.,0.]); ez = np.array([0.,0.,1.])

# Generation of input data
Y = np.random.uniform(-1,1,[3, 200])
n_points = Y.shape[1]
R_a = expso3(ez*np.pi*0.3); pos_a = 2.* (ex - ey + 2*ez)
X = R_a.T @ (Y - pos_a.reshape(3,1))
yb = np.average(Y, axis = 1)
xb = np.average(X, axis = 1)
Yr = Y - yb.reshape(3,1); Xr = X - xb.reshape(3,1)
```

```

# Weighted Procrustes
H = Xr @ Yr.T
U, S, Vt = np.linalg.svd(H)
R = Vt.T @ np.diag([1,1,np.linalg.det(Vt.T @ U.T)]) @ U.T

p = yb - R @ xb

print('R = \n{}'.format(R_a))
print('p = {}'.format(pos_a))
print('R estimate = \n{}'.format(R))
print('p estimate = \n{}'.format(p))

```

25.8 Calculation of displacement with weighting

Consider the weighted least-squares problem

$$\min_{\mathbf{R}} L_r = \sum_{i=1}^N w_i \|\mathbf{y}_i - (\mathbf{R}\mathbf{x}_i + \mathbf{t})\|^2 \quad (1841)$$

Define the the weighted centroids of the two point sets as

$$\bar{\mathbf{y}}_w = \frac{\sum_{i=1}^N w_i \mathbf{y}_i}{\sum_{i=1}^N w_i}, \quad \bar{\mathbf{x}}_w = \frac{\sum_{i=1}^N w_i \mathbf{x}_i}{\sum_{i=1}^N w_i} \quad (1842)$$

The positions are then written

$$\mathbf{y}_i = \bar{\mathbf{y}}_w + \tilde{\mathbf{y}}_i, \quad \mathbf{x}_i = \bar{\mathbf{x}}_w + \tilde{\mathbf{x}}_i \quad (1843)$$

where

$$\sum_{i=1}^N w_i \tilde{\mathbf{y}}_i = \mathbf{0}, \quad \sum_{i=1}^N w_i \tilde{\mathbf{x}}_i = \mathbf{0} \quad (1844)$$

The terms of the least-squares cost function can be written

$$\begin{aligned} w_i \|\mathbf{y}_i - (\mathbf{R}\mathbf{x}_i + \mathbf{t})\|^2 &= w_i \|\bar{\mathbf{y}}_w + \tilde{\mathbf{y}}_i - \mathbf{R}(\bar{\mathbf{x}}_w + \tilde{\mathbf{x}}_i) - \mathbf{t}\|^2 \\ &= w_i \|(\bar{\mathbf{y}}_w - \mathbf{R}\bar{\mathbf{x}}_w - \mathbf{t}) + (\tilde{\mathbf{y}}_i - \mathbf{R}\tilde{\mathbf{x}}_i)\|^2 \\ &= w_i \|\bar{\mathbf{y}}_w - \mathbf{R}\bar{\mathbf{x}}_w - \mathbf{t}\|^2 + w_i \|\tilde{\mathbf{y}}_i - \mathbf{R}\tilde{\mathbf{x}}_i\|^2 \\ &\quad + 2(\bar{\mathbf{y}}_w - \mathbf{R}\bar{\mathbf{x}}_w - \mathbf{t})^T (\tilde{\mathbf{y}}_i - \mathbf{R}\tilde{\mathbf{x}}_i) \end{aligned} \quad (1845)$$

From (1842) and (1844) it follows that

$$\sum_{i=1}^N w_i \|\mathbf{y}_i - (\mathbf{R}\mathbf{x}_i + \mathbf{t})\|^2 = \left(\sum_{i=1}^N w_i \right) \|\bar{\mathbf{y}}_w - \mathbf{R}\bar{\mathbf{x}}_w - \mathbf{t}\|^2 + \sum_{i=1}^N w_i \|\tilde{\mathbf{y}}_i - \mathbf{R}\tilde{\mathbf{x}}_i\|^2 \quad (1846)$$

This can be minimized by minimizing the two terms on the right hand side independently. First

$$\sum_{i=1}^N w_i \|\tilde{\mathbf{y}}_i - \mathbf{R}\tilde{\mathbf{x}}_i\|^2 \quad (1847)$$

is minimized with respect to \mathbf{R} , which is the usual weighted Procrustes problem, and then using the optimal value of \mathbf{R} to compute the translation

$$\mathbf{t} = \bar{\mathbf{y}}_w - \mathbf{R}\bar{\mathbf{x}}_w \quad (1848)$$

This minimizes the first term on the right hand side by setting it to zero.

Example

```
import numpy as np

def skewm(r):
    return np.array([[0,-r[2],r[1]], [r[2],0,-r[0]], [-r[1],r[0],0]])
def expso3(u):
    S = skewm(u); un = np.linalg.norm(u)
    return np.eye(3) + np.sinc(un/np.pi)*S \
        + 0.5*(np.sinc(un/(2*np.pi)))**2 * S@S

# Preparations
np.set_printoptions(formatter={'float': '{: 0.4f}'.format})
ex = np.array([1.,0.,0.]); ey = np.array([0.,1.,0.]); ez = np.array([0.,0.,1.])

# Generation of input data
Y = np.random.uniform(-1,1,[3, 200])
n_points = Y.shape[1]
R_a = expso3(ez*np.pi*0.3); pos_a = 2.* (ex - ey + 2*ez)
X = R_a.T @ (Y - pos_a.reshape(3,1))
w = 0.1*np.arange(n_points)
ybw = np.average(Y, axis = 1, weights=w)
xbw = np.average(X, axis = 1, weights=w)
Yr = Y - ybw.reshape(3,1); Xr = X - xbw.reshape(3,1)

# Weighted Procrustes
H = Xr @ np.diag(w) @ Yr.T
U, S, Vt = np.linalg.svd(H)
R = Vt.T @ np.diag([1,1,np.linalg.det(Vt.T @ U.T)]) @ U.T

p = ybw - R @ xbw

print('R = \n{}'.format(R_a))
print('p = {}'.format(pos_a))
print('R estimate = \n{}'.format(R))
print('p estimate = \n{}'.format(p))
```

25.9 Calculation of a similarity transform from two sets of 3D data

The Procrustes problem can be modified to include scaling, as shown in [75]. A scaling factor $c > 0$ is included in the model, so that the position is given by

$$\mathbf{y}_i = c\mathbf{R}\mathbf{x}_i + \mathbf{t} \quad (1849)$$

This is a similarity transformation with homogeneous transformation matrix

$$\mathbf{T}_s = \begin{bmatrix} c\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1850)$$

Note that this matrix is not an element of $SE(3)$ when $c \neq 1$. In this case there are 7 unknowns, which are 3 degrees of freedom in translation, 3 degrees of freedom in rotation and one scaling parameter. This means that at least 3 point mappings are required, as this gives 9 equations.

The problem to be solved is to find \mathbf{T}_s when the point positions \mathbf{x}_i and \mathbf{y}_i are given for $i = 1, \dots, N$. This is done by minimizing the quadratic error

$$\min_{\mathbf{R}} L_r = \sum_{i=1}^n \|\mathbf{y}_i - (c\mathbf{R}\mathbf{x}_i + \mathbf{t})\|^2 \quad (1851)$$

The positions are written

$$\mathbf{y}_i = \bar{\mathbf{y}} + \delta\mathbf{y}_i, \quad \mathbf{x}_i = \bar{\mathbf{x}} + \delta\mathbf{x}_i \quad (1852)$$

where $\bar{\mathbf{y}} = \frac{1}{N} \sum_{i=1}^N \mathbf{y}_i$ and $\bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i$ are the centroids of the two point sets. The cost function is written

$$\sum_{i=1}^N \|\mathbf{y}_i - (c\mathbf{R}\mathbf{x}_i + \mathbf{t})\|^2 = N\|\bar{\mathbf{y}} - c\mathbf{R}\bar{\mathbf{x}} - \mathbf{t}\|^2 + \sum_{i=1}^N \|\delta\mathbf{y}_i - c\mathbf{R}\delta\mathbf{x}_i\|^2 \quad (1853)$$

This is minimized by minimizing $\sum_{i=1}^N \|\delta\mathbf{y}_i - c\mathbf{R}\delta\mathbf{x}_i\|^2$ with respect to \mathbf{R} and c , and finally \mathbf{t} is found from

$$\mathbf{t} = \bar{\mathbf{y}} - c\mathbf{R}\bar{\mathbf{x}} \quad (1854)$$

The cost function

$$\delta L_r = \sum_{i=1}^N \|\delta\mathbf{y}_i - c\mathbf{R}\delta\mathbf{x}_i\|^2 \quad (1855)$$

is written in terms of the Frobenius norm as

$$\delta L_r = \|\mathbf{A} - c\mathbf{R}\mathbf{B}\|_F^2 \quad (1856)$$

where $\mathbf{A} = (\delta\mathbf{y}_1, \dots, \delta\mathbf{y}_n)$ and $\mathbf{B} = (\delta\mathbf{x}_1, \dots, \delta\mathbf{x}_n)$. This can be written

$$\delta L_r = \|\mathbf{A}\|_F^2 + c^2\|\mathbf{B}\|_F^2 - 2\text{ctr}(\mathbf{R}\mathbf{B}\mathbf{A}^T) \quad (1857)$$

This is minimized with respect to \mathbf{R} by maximizing $\text{tr}(\mathbf{R}\mathbf{B}\mathbf{A}^T)$, which is done with the singular value decomposition

$$\mathbf{H} = \mathbf{B}\mathbf{A}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad (1858)$$

The solution is given by $\mathbf{R} = \mathbf{V} \mathbf{S} \mathbf{U}^T$. It is used that $\mathbf{R} \mathbf{B} \mathbf{A}^T = \mathbf{V} \mathbf{S} \mathbf{U}^T \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T = \mathbf{V} \mathbf{S} \mathbf{\Sigma} \mathbf{V}^T$, and that $\text{tr}(\mathbf{V} \mathbf{S} \mathbf{\Sigma} \mathbf{V}^T) = \text{tr}(\mathbf{S} \mathbf{\Sigma})$ since \mathbf{V} is orthogonal. This gives

$$\delta L_r = \|\mathbf{A}\|_F^2 + c^2 \|\mathbf{B}\|_F^2 - 2c \text{tr}(\mathbf{S} \mathbf{\Sigma}) \quad (1859)$$

The c that minimizes the cost function δL_r is found by differentiation with respect to c and setting the derivative to zero, which gives

$$2c \|\mathbf{B}\|_F^2 - 2\text{tr}(\mathbf{S} \mathbf{\Sigma}) = 0 \quad (1860)$$

This gives the optimal c as

$$c = \frac{\text{tr}(\mathbf{S} \mathbf{\Sigma})}{\|\mathbf{B}\|_F^2} \quad (1861)$$

To sum up, the solution is

$$\mathbf{R} = \mathbf{V} \mathbf{S} \mathbf{U}^T \quad (1862)$$

$$c = \frac{\text{tr}(\mathbf{S} \mathbf{\Sigma})}{\|\mathbf{B}\|_F^2} \quad (1863)$$

$$\mathbf{t} = \bar{\mathbf{y}} - c \mathbf{R} \bar{\mathbf{x}} \quad (1864)$$

$$\mathbf{S} = \text{diag}[1, 1, \det(\mathbf{V} \mathbf{U}^T)] \quad (1865)$$

Comment:

The parameter c can also be found from the observation that $(\delta \mathbf{x}_i, \delta \mathbf{y}_i)$ are of equal length, which implies that $\|\delta \mathbf{y}_i\|^2 = c^2 \|\delta \mathbf{x}_i\|^2$. An estimate for c is then found from

$$c^2 = \frac{\sum_{i=1}^N \|\delta \mathbf{x}_i\|^2}{\sum_{i=1}^N \|\delta \mathbf{y}_i\|^2} \quad (1866)$$

Example

```
import numpy as np

def skewm(r):
    return np.array([[0,-r[2],r[1]], [r[2],0,-r[0]], [-r[1],r[0],0]])
def expso3(u):
    S = skewm(u); un = np.linalg.norm(u)
    return np.eye(3) + np.sinc(un/np.pi)*S \
        + 0.5*(np.sinc(un/(2*np.pi)))**2 * S@S

# Preparations
np.set_printoptions(formatter={'float': '{: 0.4f}'.format})
ex = np.array([1.,0.,0.]); ey = np.array([0.,1.,0.]); ez = np.array([0.,0.,1.])

# Generation of input data
#Y = np.random.uniform(-1,1,[3, 200])
n_points = Y.shape[1]
```

```

R_a = expso3(ez*np.pi*0.3); pos_a = 2.* (ex - ey + 2*ez)
c_a = 0.55;
X = (1/c_a)*R_a.T @ ( Y - pos_a.reshape(3,1))
# Centroids
yb = np.average(Y, axis = 1); xb = np.average(X, axis = 1)
Yr = Y - yb.reshape(3,1); Xr = X - xb.reshape(3,1)

# Procrustes
H = Xr @ Yr.T
U, Sigma, Vt = np.linalg.svd(H)
S = np.diag([1,1,np.linalg.det(Vt.T @ U.T)])
R = Vt.T @ np.diag([1,1,np.linalg.det(Vt.T @ U.T)]) @ U.T
#c = np.sqrt(np.sum(np.linalg.norm(Yr, axis=0)**2) \
#           / np.sum(np.linalg.norm(Xr, axis=0)**2))
c = np.sum(S*Sigma) / np.linalg.norm(Xr, ord='fro')**2
p = yb - c* R @ xb

print('R = \n{}\n'.format(R_a))
print('p = {}\n'.format(pos_a))
print('R estimate = \n{}\n'.format(R))
print('p estimate = \n{}\n'.format(p))

```

□

25.10 Rotation averaging for the chordal distance

Consider the situation that n rotation matrices \mathbf{R}_i are given, and the goal is to find an average rotation matrix \mathbf{R} with respect to the chordal distance measure [29]. Define the matrix

$$\mathbf{H} = \sum_{i=1}^n \mathbf{R}_i^T \in \mathbb{R}^{3 \times 3} \quad (1867)$$

The cost function to be minimized is

$$\sum_{i=1}^n d_c(\mathbf{R}_i, \mathbf{R})^2 = \sum_{i=1}^n \|\mathbf{R}_i - \mathbf{R}\|_F^2 \quad (1868)$$

$$= \sum_{i=1}^n \text{tr}[(\mathbf{R}_i - \mathbf{R})(\mathbf{R}_i - \mathbf{R})^T] \quad (1869)$$

$$= \sum_{i=1}^n \text{tr}[(2\mathbf{I} - 2\mathbf{R}\mathbf{R}_i^T)] \quad (1870)$$

This gives

$$\sum_{i=1}^n d_c(\mathbf{R}_i, \mathbf{R})^2 = 2n\text{tr}(\mathbf{I}) - 2\text{tr}(\mathbf{RH}) \quad (1871)$$

and it is seen that the minimization of $\sum_{i=1}^n d_c(\mathbf{R}_i, \mathbf{R})^2$ is equivalent to the maximization of $\text{tr}(\mathbf{R}\mathbf{H})$, which is the Procrustes problem. Then the optimal solution is found by calculating the singular value decomposition $\mathbf{H} = \mathbf{U}\Sigma\mathbf{V}^T$. The optimal solution is then

$$\mathbf{R} = \mathbf{V}\mathbf{S}\mathbf{U}^T, \quad \mathbf{S} = \text{diag}(1, 1, \det(\mathbf{V}\mathbf{U}^T)) \quad (1872)$$

25.11 Orthogonalization of an approximation of a rotation matrix

Suppose that the matrix $\mathbf{Q} \in \mathbb{R}^{3 \times 3}$ has been estimated as a rotation matrix by some method. Then if the estimation method is not accurate, the matrix \mathbf{Q} will not be orthogonal, and it will be of interest to orthogonalize the matrix, which is to find the rotation matrix $\mathbf{R} \in SO(3)$ that is closest to the estimate \mathbf{Q} . This can be done by minimizing the chordal distance

$$\|\mathbf{R} - \mathbf{Q}\|_F^2 = \text{tr}((\mathbf{R} - \mathbf{Q})(\mathbf{R} - \mathbf{Q})^T) \quad (1873)$$

$$= \text{tr}(\mathbf{R}\mathbf{R}^T - 2\mathbf{R}\mathbf{Q}^T + \mathbf{Q}\mathbf{Q}^T) \quad (1874)$$

$$= \text{tr}(\mathbf{I} - 2\mathbf{R}\mathbf{Q}^T + \mathbf{Q}\mathbf{Q}^T) \quad (1875)$$

Minimization of the chordal distance is seen to be equivalent to the maximization of $\mathbf{R}\mathbf{Q}^T$, which leads to the maximization problem

$$\max_{\mathbf{R} \in SO(3)} \text{tr}(\mathbf{R}\mathbf{Q}^T) \quad (1876)$$

This is again the Procrustes problem with $\mathbf{H} = \mathbf{Q}^T$. Then if the SVD of \mathbf{Q} is

$$\mathbf{Q} = \mathbf{U}\Sigma\mathbf{V}^T \quad (1877)$$

which means that the SVD of \mathbf{H} is $\mathbf{H} = \mathbf{V}\Sigma\mathbf{U}^T$, then it follows that

$$\mathbf{R} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \quad \mathbf{S} = \text{diag}(1, 1, \det(\mathbf{U}\mathbf{V}^T)) \quad (1878)$$

is the rotation matrix that minimizes the chordal distance to the estimate \mathbf{Q} .

25.12 Recovery of rotation matrix from an estimate of two columns

In machine learning the use of a 6-parameter description of rotation is recommended in [83].

25.13 Gradient search for the Procrustes problem

The Procrustes problem can also be solved with a gradient search method. This is not recommended, since there is a closed form solution that should be used. Still, the gradient search method is presented in the following to gain more insight into gradient search methods on $SO(3)$. The objective function of the Procrustes problem is written

$$h(\mathbf{R}) = \frac{1}{n} \sum_{i=1}^n \mathbf{a}_i^T \mathbf{R} \mathbf{b}_i \quad (1879)$$

The rotation matrix $\mathbf{P}(t) \in SO(3)$ is defined by $\mathbf{P}(t) = \mathbf{R} \exp(t\mathbf{U})$ where $\mathbf{U} = \mathbf{u}^\times \in so(3)$. Then $\mathbf{P}(0) = \mathbf{R}$ and $\dot{\mathbf{P}}(0) = \mathbf{R}\mathbf{U} = \mathbf{R}\mathbf{u}^\times$. The gradient $\text{grad}h(\mathbf{R}) \in T_{\mathbf{R}}SO(3)$ is then found from

$$\langle \mathbf{R}\mathbf{u}^\times, \text{grad}h \rangle = \frac{d}{dt} \bigg|_{t=0} h(\mathbf{P}(t)) \quad (1880)$$

It is found that

$$\frac{d}{dt} h(\mathbf{P}(t)) \Big|_{t=0} = \frac{1}{n} \sum_{i=1}^n \frac{d}{dt} \Big|_{t=0} \mathbf{a}_i^T \mathbf{P}(t) \mathbf{b}_i = \frac{1}{n} \sum_{i=1}^n \mathbf{a}_i^T \mathbf{R} \mathbf{u}^\times \mathbf{b}_i \quad (1881)$$

This expression can be reformulated using the rules for the triple scalar product, which gives

$$\mathbf{a}_i^T \mathbf{R} \mathbf{u}^\times \mathbf{b}_i = (\mathbf{R}^T \mathbf{a}_i)^T \mathbf{u}^\times \mathbf{b}_i = \mathbf{u}^T (\mathbf{b}_i^\times \mathbf{R}^T \mathbf{a}_i) = -\mathbf{u}^T ((\mathbf{R}^T \mathbf{a}_i)^\times \mathbf{b}_i) \quad (1882)$$

The scalar triple product can be written in terms of the Riemannian metric as

$$\mathbf{u}^T ((\mathbf{R}^T \mathbf{a}_i)^\times \mathbf{b}_i) = \langle \mathbf{u}^\times, ((\mathbf{R}^T \mathbf{a}_i)^\times \mathbf{b}_i)^\times \rangle = \langle \mathbf{R} \mathbf{u}^\times, \mathbf{R} ((\mathbf{R}^T \mathbf{a}_i)^\times \mathbf{b}_i)^\times \rangle \quad (1883)$$

where it is used that the Riemannian metric is unchanged when both factors are premultiplied by a rotation matrix. This gives

$$\frac{d}{dt} h(\mathbf{P}(t)) \Big|_{t=0} = -\frac{1}{n} \sum_{i=1}^n \langle \mathbf{R} \mathbf{u}^\times, \mathbf{R} ((\mathbf{R}^T \mathbf{a}_i)^\times \mathbf{b}_i)^\times \rangle \quad (1884)$$

and it follows that the gradient is

$$\text{grad} h = -\mathbf{R} \frac{1}{n} \sum_{i=1}^n ((\mathbf{R}^T \mathbf{a}_i)^\times \mathbf{b}_i)^\times \in T_R SO(3) \quad (1885)$$

The can be written $\text{grad} h = \mathbf{R} \mathbf{g}^\times$ where

$$\mathbf{g}^\times = -(1/n) \sum_{i=1}^n ((\mathbf{R}^T \mathbf{a}_i)^\times \mathbf{b}_i)^\times \quad (1886)$$

A gradient method with unit step-length in the negative gradient direction is then given by

$$\mathbf{R}(k+1) = \mathbf{R}(k) \exp(-\mathbf{R}(k)^T \text{grad} f(\mathbf{R}(k))) = \mathbf{R}(k) \exp(-\mathbf{g}(k)^\times) \quad (1887)$$

This gives the gradient search method

$$\mathbf{R}(k+1) = \mathbf{R}(k) \exp \left(\frac{1}{n} \sum_{i=1}^n ((\mathbf{R}^T \mathbf{a}_i)^\times \mathbf{b}_i)^\times \right) \quad (1888)$$

The vector form of \mathbf{g}^\times is $\mathbf{g} = -(1/n) \sum_{i=1}^n (\mathbf{R}^T \mathbf{a}_i)^\times \mathbf{b}_i$. The vector $(\mathbf{R}^T \mathbf{a}_i)^\times \mathbf{b}_i$ is normal to both vectors $\mathbf{R}^T \mathbf{a}_i$ and \mathbf{b}_i , and the length is equal to $\sin \theta_i$, where θ_i is the angle between $\mathbf{R}^T \mathbf{a}_i$ and \mathbf{b}_i . This means that $\exp[-(\mathbf{R}^T \mathbf{a}_i)^\times \mathbf{b}_i]^\times$ will rotate $\mathbf{R}^T \mathbf{a}_i$ in the direction of \mathbf{b}_i .

Example

An related objective function which was proposed by Taylor and Kriegman in [72] is

$$h(\mathbf{R}) = \frac{1}{2} \sum_{i=1}^n (\mathbf{a}_i^T \mathbf{R} \mathbf{b}_i)^2 \quad (1889)$$

Then the gradient is

$$\text{grad}h(\mathbf{R}) = -(\mathbf{a}_i^T \mathbf{R} \mathbf{b}_i) \mathbf{R} ((\mathbf{R}^T \mathbf{a}_i)^\times \mathbf{b}_i)^\times \quad (1890)$$

This objective function is not to be recommended for two reasons. Firstly, the closed form solution of the Procrustes problem should be used, and secondly, if a gradient search method is to be used, then the squaring of the objective function leads to a gradient that becomes small close to the optimal solution, which is not good for convergence. \square

Example

```
% Script that solves for the optimal
% R when datapoints ai and bi are given so that ai = R*bi.
% Datapoints are arranged in matrices
% A = [a1, ..., an] and B = [b1, ..., bn] so that A = R*B
skew =@(u) [0 -u(3) u(2); u(3) 0 -u(1);-u(2) u(1) 0];
expso3 =@(u) eye(3)+sinc(norm(u)/pi)*skew(u)+0.5*(sinc(norm(u)/(2*pi)))^2*(skew(u))^2;
% Input: Data points
B = [1 0 0; 1 1 0; 1 1 1; 0 1 0; 0 1 1; 0 0 1]';
% Input: Actual rotation matrix
Ra = expso3(pi/6*[0 0 1]');%*expso3(pi/4*[0 1 0]')
logRa = logS03(Ra);
A = Ra*B;

% Procrustes solution using SVD
H = B*A';
[U,S,V] = svd(H);
Rc = V*diag([1 1 det(U*V')])*U' % Computed optimal least-squares solution
deltaRtest = Rc*Ra'; % Test: Identity matrix expected
logSVD = logS03(deltaRtest)

% Gradient search using h(R) = sum ai^T*R*bi
Ri = eye(3);
for i = 1:10
    u = -sum(cross(Ri'*A,B),2)/size(B,2);
    Ri = Ri*expso3(u);
end
deltaRi1 = Ri'*Ra;
logGS = logS03(deltaRi1)

% Gradient search using h(R) = sum (ai^T*R*bi)^2
Ri = eye(3);
for i = 1:10
    u = -0.5*sum(dot(A,Ri*B)) * sum(cross(Ri'*A,B),2)/size(B,2)^2;
    Ri = Ri*expso3(u);
end
deltaRi2 = Ri'*Ra;
logTK = logS03(deltaRi2)
```

```

%%%%%
function z = sinc(x)
    if abs(x) >= 0.000001
        z = sin(pi*x)/(pi*x);
    else
        z = 1 - pi*pi*x*x/6;
    end
end

%%%%%
function u = logSO3(R)
% The vector form of the logarithm in SO(3) according to Iserles(2006)
vex3 = @(u) [u(3,2); u(1,3); u(2,1)];
eh = 0.5*(R-R'); % eh = sin(theta)kh
en = norm(vex3(eh)); % en = |sin(theta)|
if en < 0.000001
    g = 1 + en^2/6;
else
    g = (asin(en)/en);
end
u = vex3(g*eh);
end

```

25.14 Kinematics of the hand-eye calibration problem

The hand-eye calibration problem is to find the displacement from the end-effector frame to the camera frame for a manipulator with a camera mounted on the end effector. The problem was first formulated in [63] and [74]. The motivation of the problem is the following application. A robot has a camera mounted on the end-effector to make it possible to do advanced assembly tasks where the camera is used to determine the position and orientation of the object frame o of a workpiece to be assembled with another workpiece held by the robot. It is then necessary to know the exact position and orientation \mathbf{T}_c^e of the camera frame c relative to the end-effector frame e , so that it can be computed where the end-effector should move to assemble the two workpieces. The process of determining \mathbf{T}_c^e is called hand-eye calibration.

The homogeneous transformation matrix \mathbf{T}_e^0 from the base frame 0 to the end-effector frame e is found from the forward kinematics of the robot given the joint variables \mathbf{q} . To determine \mathbf{T}_c^e a calibration rig with frame w is used. The position and orientation \mathbf{T}_w^c of the frame w of the calibration rig with respect to the camera frame c is found with pose estimation techniques. Then the position and orientation of the calibration frame w with respect to the base frame 0 is found to be given by

$$\mathbf{T}_w^0 = \mathbf{T}_e^0 \mathbf{T}_c^e \mathbf{T}_w^c \quad (1891)$$

The calibration problem is to find the unknown transformation

$$\mathbf{X} = \mathbf{T}_c^e \quad (1892)$$

To find the unknown transformation \mathbf{X} the robot is moved to a pair of configurations i and j . In configuration i the joint variables are \mathbf{q}_i , and the end-effector is at $\mathbf{A}_i = (\mathbf{T}_e^0)_i$, while the workpiece is found to be at $\mathbf{B}_i = (\mathbf{T}_w^c)_i$ relative to the camera. In configuration j the joint variables are \mathbf{q}_j , and the end-effector is at $\mathbf{A}_j = (\mathbf{T}_e^0)_j$ while the workpiece is found to be at $\mathbf{B}_j = (\mathbf{T}_w^c)_j$.

The workpiece position and orientation \mathbf{T}_w^0 is the same during this procedure. This means that

$$\mathbf{T}_w^0 = \mathbf{A}_i \mathbf{X} \mathbf{B}_i = \mathbf{A}_j \mathbf{X} \mathbf{B}_j \quad (1893)$$

The next step is to define the matrices

$$\mathbf{A}_1 = \mathbf{A}_j^{-1} \mathbf{A}_i, \quad \mathbf{B}_1 = \mathbf{B}_j \mathbf{B}_i^{-1} \quad (1894)$$

Then the basic equation of the hand-eye calibration problem is found in the form

$$\mathbf{A}_1 \mathbf{X} = \mathbf{X} \mathbf{B}_1 \quad (1895)$$

where \mathbf{A}_1 and \mathbf{B}_1 are known, and the calibration task is to find \mathbf{X} . It is noted that the equation results from measurements at a pair i and j of configurations for the robot.

To determine the unknown transformation \mathbf{X} uniquely, it is required to have at least two pairs of configurations. The second pair of configurations can be selected as i and k , which means that the robot must be moved to 3 different configurations i , j and k . Then $\mathbf{A}_2 = \mathbf{A}_k^{-1} \mathbf{A}_i$ and $\mathbf{B}_2 = \mathbf{B}_k \mathbf{B}_i^{-1}$ can be computed, which leads to the two equations

$$\mathbf{A}_1 \mathbf{X} = \mathbf{X} \mathbf{B}_1 \quad (1896)$$

$$\mathbf{A}_2 \mathbf{X} = \mathbf{X} \mathbf{B}_2 \quad (1897)$$

which leads to a unique solution for \mathbf{X} . It is to be expected that there will be some noise in the determination of the matrices, therefore it may be advantageous to have $n > 2$ pairs of configurations, which can be achieved by combining, giving n equations

$$\mathbf{A}_1 \mathbf{X} = \mathbf{X} \mathbf{B}_1 \quad (1898)$$

$$\mathbf{A}_2 \mathbf{X} = \mathbf{X} \mathbf{B}_2 \quad (1899)$$

$$\dots \quad (1900)$$

$$\mathbf{A}_n \mathbf{X} = \mathbf{X} \mathbf{B}_n \quad (1901)$$

The solution \mathbf{X} can be found as a least-squares solution.

A method for solving the calibration equation for \mathbf{X} is given in the following.

Comment

The notation used here is based on the article of Shiu and Ahmad from 1989 [63]. In the article of Daniilidis from 1999 [18] the definition of \mathbf{X} is the inverse of what is used here, and the matrices \mathbf{A} and \mathbf{B} have been interchanged. The equation is still $\mathbf{A}\mathbf{X} = \mathbf{X}\mathbf{B}$. The paper of [18] is based on the use of dual quaternions, and it is argued that method is an improvement since it is based on a simultaneous optimization of position and rotation in the calibration problem. This claim is not clearly justified, in fact, it seems clear from the analysis of the Park-Martin method that optimization of translation and rotation in sequence is the optimal solution. \square

25.15 The Park-Martin solution to the hand-eye calibration problem

In this section the Park-Martin solution from 1994 [52] to the hand-eye calibration problem is presented. This is a widely used solution to the calibration problem where two homogeneous transformation matrices \mathbf{A} and \mathbf{B} are given, and the problem is to find a homogeneous transformation matrix \mathbf{X} so that

$$\mathbf{AX} = \mathbf{XB} \quad (1902)$$

The equation $\mathbf{AX} = \mathbf{XB}$ is written

$$\begin{bmatrix} \mathbf{R}_a & \mathbf{t}_a \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_x & \mathbf{t}_x \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_x & \mathbf{t}_x \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_b & \mathbf{t}_b \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (1903)$$

which shows that the rotation part and the translation part can be written as two equations according to

$$\mathbf{R}_a \mathbf{R}_x = \mathbf{R}_x \mathbf{R}_b \quad (1904)$$

$$\mathbf{R}_a \mathbf{t}_x + \mathbf{t}_a = \mathbf{R}_x \mathbf{t}_b + \mathbf{t}_x \quad (1905)$$

It is interesting to note that the rotation equation can be written

$$\mathbf{R}_a = \mathbf{R}_x \mathbf{R}_b \mathbf{R}_x^T \quad (1906)$$

which can be interpreted as a transformation of the rotation matrix \mathbf{R}_b by \mathbf{R}_x to \mathbf{R}_a .

The problem is solved by first finding the rotation \mathbf{R}_x from (1904) and then to use this solution in (1905) to solve for the translation \mathbf{t}_x .

First the rotational problem is considered. Let θ_a, \mathbf{k}_a be the angle-axis representation of \mathbf{R}_a and let θ_b, \mathbf{k}_b be the angle-axis representation of \mathbf{R}_b . The rotation matrices can then be written in exponential form as

$$\mathbf{R}_a = \exp(\theta_a \mathbf{k}_a^\times) = \mathbf{I} + \sin \theta_a \mathbf{k}_a^\times + (1 - \cos \theta_a) (\mathbf{k}_a^\times)^2 \quad (1907)$$

$$\mathbf{R}_b = \exp(\theta_b \mathbf{k}_b^\times) = \mathbf{I} + \sin \theta_b \mathbf{k}_b^\times + (1 - \cos \theta_b) (\mathbf{k}_b^\times)^2 \quad (1908)$$

where \mathbf{k}_a^\times is the skew-symmetric form of the vector \mathbf{k}_a , and \mathbf{k}_b^\times is the skew-symmetric form of the vector \mathbf{k}_b .

The matrix exponential function of a skew-symmetric matrix $\mathbf{u}^\times = \theta \mathbf{k}^\times$ is defined as

$$\exp(\mathbf{u}^\times) = \mathbf{I} + \mathbf{u}^\times + \frac{1}{2!} (\mathbf{u}^\times)^2 + \frac{1}{3!} (\mathbf{u}^\times)^3 + \dots \quad (1909)$$

Therefore, since a rotation matrix \mathbf{R} satisfies $\mathbf{R}^T \mathbf{R} = \mathbf{I}$, it follows that

$$\begin{aligned} \mathbf{R} \exp(\mathbf{u}^\times) \mathbf{R}^T &= \mathbf{R} \mathbf{R}^T + \mathbf{R} \mathbf{u}^\times \mathbf{R}^T + \frac{1}{2!} \mathbf{R} (\mathbf{u}^\times)^2 \mathbf{R}^T + \frac{1}{3!} \mathbf{R} (\mathbf{u}^\times)^3 \mathbf{R}^T + \dots \\ &= \mathbf{I} + \mathbf{R} \mathbf{u}^\times \mathbf{R}^T + \frac{1}{2!} \mathbf{R} \mathbf{u}^\times \mathbf{R}^T \mathbf{R} \mathbf{u}^\times \mathbf{R}^T + \frac{1}{3!} \mathbf{R} \mathbf{u}^\times \mathbf{R}^T \mathbf{R} \mathbf{u}^\times \mathbf{R}^T \mathbf{R} \mathbf{u}^\times \mathbf{R}^T + \dots \\ &= \mathbf{I} + \mathbf{R} \mathbf{u}^\times \mathbf{R}^T + \frac{1}{2!} (\mathbf{R} \mathbf{u}^\times \mathbf{R}^T)^2 + \frac{1}{3!} (\mathbf{R} \mathbf{u}^\times \mathbf{R}^T)^3 + \dots \end{aligned} \quad (1910)$$

According to the definition (1909) of the matrix exponential this gives the result

$$\mathbf{R} \exp(\mathbf{u}^\times) \mathbf{R}^T = \exp(\mathbf{R} \mathbf{u}^\times \mathbf{R}^T) \quad (1911)$$

From (1907, 1908, 1911) it follows that (1906) can be written

$$\exp(\theta_a \mathbf{k}_a^\times) = \mathbf{R}_x \exp(\theta_b \mathbf{k}_b^\times) \mathbf{R}_x^T = \exp[\mathbf{R}_x(\theta_b \mathbf{k}_b^\times) \mathbf{R}_x^T] \quad (1912)$$

and it follows that

$$\theta_a \mathbf{k}_a^\times = \mathbf{R}_x(\theta_b \mathbf{k}_b^\times) \mathbf{R}_x^T \quad (1913)$$

The corresponding vector form of this is

$$\theta_a \mathbf{k}_a = \mathbf{R}_x \theta_b \mathbf{k}_b \quad (1914)$$

The problem of finding \mathbf{R}_x has then been reformulated as the orthogonal Procrustes problem.

The solution for the rotation matrix can be found from the minimization problem

$$\begin{aligned} \min_{\mathbf{R}} L_r &= \sum_{i=1}^n \|(\theta_a \mathbf{k}_a)_i - \mathbf{R}_x(\theta_b \mathbf{k}_b)_i\|^2 = \text{tr}(\mathbf{A} - \mathbf{R}_x \mathbf{B})(\mathbf{A} - \mathbf{R}_x \mathbf{B})^T \\ &= \text{tr}(\mathbf{A} \mathbf{A}^T + \mathbf{B} \mathbf{B}^T - 2 \mathbf{R}_x \mathbf{B} \mathbf{A}^T) \end{aligned} \quad (1915)$$

where $\mathbf{A} = [(\theta_a \mathbf{k}_a)_1, \dots, (\theta_a \mathbf{k}_a)_n]$ and $\mathbf{B} = [(\theta_b \mathbf{k}_b)_1, \dots, (\theta_b \mathbf{k}_b)_n]$ are matrices with the observations as column vectors. This is an orthogonal Procrustes problem that can be solved by the equivalent maximization problem

$$\max_{\mathbf{R}_x} \text{tr}(\mathbf{R}_x \mathbf{H}) \quad (1916)$$

where

$$\mathbf{H} = \mathbf{B} \mathbf{A}^T = \mathbf{U} \Sigma \mathbf{V}^T \quad (1917)$$

The optimal solution is given by

$$\mathbf{R}_x = \mathbf{V} \mathbf{S} \mathbf{U}^T \quad (1918)$$

where

$$\mathbf{S} = \text{diag}[1, 1, \det(\mathbf{V} \mathbf{U}^T)] \quad (1919)$$

is the Umeyama correction to ensure that \mathbf{R}_x is a rotation matrix.

It is shown in [52] that it is required that $n \geq 2$ to find a unique solution. This also follows directly from the orthogonal Procrustes problem.

When the solution \mathbf{R}_x is available, the translational equation can be written

$$(\mathbf{R}_a - \mathbf{I}) \mathbf{t}_x = \mathbf{R}_x \mathbf{t}_b - \mathbf{t}_a \quad (1920)$$

The solution in translation can then be found by minimizing the loss function

$$L_t = \sum_{i=1}^n \|(\mathbf{R}_a - \mathbf{I}) \mathbf{t}_x - \mathbf{R}_x \mathbf{t}_{bi} + \mathbf{t}_{ai}\|^2 = \|\mathbf{C} \mathbf{t}_x - \mathbf{d}\|^2 \quad (1921)$$

where

$$\mathbf{C} = \begin{bmatrix} \mathbf{R}_{a1} - \mathbf{I} \\ \vdots \\ \mathbf{R}_{an} - \mathbf{I} \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} \mathbf{R}_x \mathbf{t}_{b1} - \mathbf{t}_{a1} \\ \vdots \\ \mathbf{R}_x \mathbf{t}_{bn} - \mathbf{t}_{an} \end{bmatrix} \quad (1922)$$

The solution is the usual least-squares solution

$$\mathbf{t}_x = (\mathbf{C}^T \mathbf{C})^{-1} \mathbf{C}^T \mathbf{d} \quad (1923)$$

Also in this case, $n \geq 2$ combinations of \mathbf{A}_i and \mathbf{B}_i are needed to find a unique \mathbf{t}_x given \mathbf{R}_x .

Example:

```
% Calculation of unknown X in AX = XB based on (Park and Martin, 1994).
%
% Input: Unknown transformation
X = [Rotx(pi/6) [1;2;1]; 0 0 0 1]
% Input: Point data
B1 = [Rotz(pi/8) [0;0;1]; 0 0 0 1];
B2 = [Roty(pi/7) [0;1;1]; 0 0 0 1];
B3 = [Roty(pi/6)* Rotz(-pi/6)*Rotx(pi/9) [0;1;1]; 0 0 0 1];
B = [B1 B2 B3];
A1 = X*B1*inv(X);
A2 = X*B2*inv(X);
A3 = X*B3*inv(X);
A = [A1 A2 A3];

[m,N] = size(A); n = N/4;

Ka = zeros(3,n); Kb = zeros(3,n);
for i = 1:n
    Ka(:,i) = R2Log(A(:,4*i-3:4*i-1));
    Kb(:,i) = R2Log(B(:,4*i-3:4*i-1));
end
H = Kb*Ka';
[U,S,V] = svd(H);
R = V*U'; % Optimal R

C = zeros(3*n,3); d = zeros(3*n,1);
for i = 1:n
    C(3*i-2:3*i,:) = A(1:3,4*i-3:4*i-1) - eye(3);
    d(3*i-2:3*i) = R*B(1:3,4*i) - A(1:3,4*i);
end
t = (C'*C)\C'*d; % Optimal t

X = [R t; 0 0 0 1] % Output: Calculated transformation matrix
```

Comment 1:

In [52] the solution in rotation is expressed as

$$\mathbf{R}_x = (\mathbf{H}^T \mathbf{H})^{-1/2} \mathbf{H}^T \quad (1924)$$

This is consistent with the solution based on the singular value decomposition, which is seen from

$$(\mathbf{H}^T \mathbf{H})^{-1/2} \mathbf{H}^T = (\mathbf{V} \boldsymbol{\Sigma} \mathbf{U}^T \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T)^{-1/2} \mathbf{V} \boldsymbol{\Sigma} \mathbf{U}^T \quad (1925)$$

$$= (\mathbf{V} \boldsymbol{\Sigma}^2 \mathbf{V}^T)^{-1/2} \mathbf{V} \boldsymbol{\Sigma} \mathbf{U}^T \quad (1926)$$

$$= \mathbf{V} \boldsymbol{\Sigma}^{-1} \mathbf{V}^T \mathbf{V} \boldsymbol{\Sigma} \mathbf{U}^T \quad (1927)$$

$$= \mathbf{V} \mathbf{U}^T \quad (1928)$$

Here the square root is defined for a positive definite symmetric matrix $\mathbf{M} = \mathbf{V} \boldsymbol{\Sigma} \mathbf{V}^T$ as $\mathbf{M}^{1/2} = \mathbf{V} \boldsymbol{\Sigma}^{1/2} \mathbf{V}^T$ where \mathbf{V} is orthogonal and $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, \dots, \sigma_n)$, and $\boldsymbol{\Sigma}^{1/2} = \text{diag}(\sigma_1^{1/2}, \dots, \sigma_n^{1/2})$. \square

Comment 2:

Suppose that \mathbf{R}_p is a solution to the calibration problem so that $\theta_a \mathbf{k}_a = \mathbf{R}_p(\theta_b \mathbf{k}_b)$. Then also

$$\mathbf{R}_x = \exp(\phi_a \mathbf{k}_a^\times) \mathbf{R}_p \exp(\phi_b \mathbf{k}_b^\times) \quad (1929)$$

will be a solution for $0 \leq \phi_a \leq 2\pi$ and $0 \leq \phi_b \leq 2\pi$. This is verified by showing that this solution satisfies $\theta_a \mathbf{k}_a = \mathbf{R}_x(\theta_b \mathbf{k}_b)$, which is seen from

$$\exp(\phi_a \mathbf{k}_a^\times) \mathbf{R}_p \exp(\phi_b \mathbf{k}_b^\times)(\theta_b \mathbf{k}_b) = \exp(\phi_a \mathbf{k}_a^\times) \mathbf{R}_p(\theta_b \mathbf{k}_b) = \exp(\phi_a \mathbf{k}_a^\times) \theta_a \mathbf{k}_a = \theta_a \mathbf{k}_a \quad (1930)$$

where it is used that $\exp(\theta \mathbf{k}^\times) \mathbf{k} = \mathbf{k}$ since it is a rotation of the vector \mathbf{k} about the vector itself. The expression in (1929) can be further developed as

$$\exp(\phi_a \mathbf{k}_a^\times) \mathbf{R}_p \exp(\phi_b \mathbf{k}_b^\times) = \exp(\phi_a \mathbf{k}_a^\times) \mathbf{R}_p \exp(\phi_b \mathbf{k}_b^\times) \mathbf{R}_p^T \mathbf{R}_p \quad (1931)$$

$$= \exp(\phi_a \mathbf{k}_a^\times) \exp(\phi_b \mathbf{R}_p \mathbf{k}_b^\times \mathbf{R}_p^T) \mathbf{R}_p \quad (1932)$$

$$= \exp(\phi \mathbf{k}_a^\times) \mathbf{R}_p \quad (1933)$$

where $\phi = \phi_a + \phi_b$, or as

$$\exp(\phi_a \mathbf{k}_a^\times) \mathbf{R}_p \exp(\phi_b \mathbf{k}_b^\times) = \mathbf{R}_p \mathbf{R}_p^T \exp(\phi_a \mathbf{k}_a^\times) \mathbf{R}_p \exp(\phi_b \mathbf{k}_b^\times) \quad (1934)$$

$$= \mathbf{R}_p \exp(\phi_a \mathbf{R}_p^T \mathbf{k}_a^\times \mathbf{R}_p) \exp(\phi_b \mathbf{k}_b^\times) \quad (1935)$$

$$= \mathbf{R}_p \exp(\phi \mathbf{k}_b^\times) \quad (1936)$$

This shows that the solution has one free parameter ϕ so that if \mathbf{R}_p is one particular solution, then also

$$\mathbf{R}_x = \exp(\phi \mathbf{k}_a^\times) \mathbf{R}_p = \mathbf{R}_p \exp(\phi \mathbf{k}_b^\times) \quad (1937)$$

is a solution. In Theorem 1 of [52] the rotation matrix is transposed in $\exp(\phi \mathbf{k}_a^\times) \mathbf{R}_p$, which must be a mistake.

To have a unique solution it is necessary to have n combinations, where $n \geq 2$, of \mathbf{A}_i and \mathbf{B}_i . In particular, if $n = 2$ it is required that the axes of rotation \mathbf{k}_{ai} of the rotation matrices

\mathbf{R}_{ai} do not coincide, which is equivalent to the condition that the axes of rotation \mathbf{k}_{bi} of the rotation matrices \mathbf{R}_{bi} do not coincide. \square

Comment 3: It is noted that since \mathbf{R}_x is a rotation matrix it follows from (1914) that

$$\|\theta_a \mathbf{k}_a\| = \|\theta_b \mathbf{k}_b\| \quad (1938)$$

and since \mathbf{k}_a and \mathbf{k}_b are unit vectors this implies that the magnitude of the rotation angles must be the same, that is,

$$|\theta_a| = |\theta_b| \quad (1939)$$

In [18] it is argued that because of this the angle can be left out and the solution can be based on the optimal Procrustes problem with

$$\mathbf{k}_a = \mathbf{R}_x \mathbf{k}_b \quad (1940)$$

\square

26 Point clouds

A point cloud is a set of points $\mathbf{p}_i = (x_i, y_i, z_i)^T$ in combination with image intensity I_i or RGB intensity r_i, g_i, b_i , and possibly with surface normal $\mathbf{n}_i = (n_x, n_y, n_z)^T$. A point cloud is recorded with a 3D camera which will typically have a range finder in combination with a camera to detect points. A significant advantage to 2D camera data is that a point cloud has depth data, and there is less problems with perspective effects.

26.1 Computation of surface normals

The computation of surface normals from a point cloud is important in many 3D vision applications. The usual method for doing this is to use a least-squares method [5, 31]. Then the surface normal for a point is found by fitting the neighboring k points with a plane $\pi = (\mathbf{n}^T, d)^T$, where \mathbf{n} is the surface normal. The number k of points used is a design parameter of the method which will depend on the noise characteristics, and the shape of the surface.

In the literature this problem is formulated as follows: The optimization is done by minimizing the cost function

$$\min_{\mathbf{n}} L = \sum_{i=1}^k (\mathbf{p}_i^T \mathbf{n} + d)^2 \quad \text{when } |\mathbf{n}| = 1 \quad (1941)$$

with respect to the surface normal \mathbf{n} . The solution is found from the covariance matrix

$$\mathbf{M} = \frac{1}{k} \sum_{i=1}^k (\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^T \quad (1942)$$

where

$$\bar{\mathbf{p}} = \frac{1}{k} \sum_{i=1}^k \mathbf{p}_i \quad (1943)$$

is the centroid of the points \mathbf{p}_i , $i = 1, \dots, k$. The eigenvalues λ_1 , λ_2 and λ_3 will describe the shape of the ellipsoid defined by \mathbf{M} , and the smallest eigenvalue, say λ_3 , will be associated with the eigenvector \mathbf{m}_3 in the direction that with the shortest principal axis. Therefore, the idea is to identify the normal of the plane the direction of the eigenvector \mathbf{m}_3 , that is,

$$\mathbf{n} = \pm \mathbf{m}_3 \quad (1944)$$

where the direction is selected depending on neighboring surface normals.

It is noted that since the matrix \mathbf{M} is symmetric, the solution can be found with the singular value decomposition $\mathbf{M} = \mathbf{U}_m \boldsymbol{\Sigma}_m \mathbf{U}_m^T$, and the normal is found to be in the direction of the last column in \mathbf{U}_m , that is, $\mathbf{n} = c_m \mathbf{u}_3$ where c_m is a nonzero constant.

It is interesting to note that this problem is more efficiently formulated in homogeneous coordinates. Then this is the problem of fitting a set of points to k points. The least-squares solution is found from

$$\mathbf{A}\boldsymbol{\pi} = \begin{bmatrix} \mathbf{p}_1^T & 1 \\ \vdots & \vdots \\ \mathbf{p}_k^T & 1 \end{bmatrix} \boldsymbol{\pi} = 0 \quad (1945)$$

where $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$. The solution is $\boldsymbol{\pi} = (\mathbf{n}^T, d)^T = c\mathbf{v}_4$ where c is a nonzero constant and \mathbf{v}_4 is the last column in \mathbf{V} .

Example

```
% 10 points are generated to be on the plane x + y + z = 3 with
% the addition of noise with a standard deviation of 0.01.
X = repmat([1 1 1]',1,10); % Input: The centroid of the 10 points

X(1:2,:) = X(1:2,:) + [[0.1 0.1]; [0.1 0.5]; [-0.1 0.1]; [0.2 -0.2]; ...
    [-0.1 -0.1]; [-0.1 -0.5]; [0.1 -0.1]; [-0.2 -0.2]; [0.2 0.2]; [0.2 0.2]]';
X(3,:) = 3 - (X(1,:)+X(2,:));
% Add normally distributed noise with sigma = 0.01
X = X + 0.01*randn(size(X,1),size(X,2));
k = 10;

% Calculation in Euclidean coordinates with M matrix
Xc = mean(X,2); % Centroid: Mean over the points
M = (X-repmat(Xc,1,10))*(X-repmat(Xc,1,10))'/k;
% Calculation of surface normal with eigenvalues of M
[m,lambda] = eig(M);
[lambda_min,i_min] = min(diag(lambda,0));
n_eig = m(:,i_min)
% Alternative calculation of surface normal with svd of M
[u,s,v] = svd(M);
n_svd = v(:,3);

% Alternative calculation of surface normal in homogeneous coordinates
% by fitting a plane to k points. M need not be calculated
A = [X' ones(size(X',1),1)];
[U,S,V] = svd(A);
n = V(1:3,4); n_hom = sign(n(1))*n/norm(n)
```

26.2 Iterative closest point

3D cameras are used to determine the position and orientation of a known object. The geometry of the object will typically be known from a CAD model, or as a point cloud of the model. Then, the point cloud data will be compared to the model geometry to find matching points so that the relative position and orientation can be found. This matching of points is called the registration problem, or the closest point problem. If matching points can be determined in the model and data point clouds, then the orthogonal Procrustes method can be used to find the rigid displacement between the two point clouds. There are also alternative methods where the distance from the data points to the model planes are computed. This may be an advantage when the model has clearly defined planes. Registration is also used when point cloud data has been captured from different views of the object. Then, the overlapping points of the different views must be identified or registered, so that the images can be combined into a single 3D point cloud of the object.

A usual situation is that the correspondence between the data points and the model points is unknown. The standard method in this case is the iterative closest point method. In that case, the data point cloud is moved so that it has the same centroid as the model point cloud. Then

for each data point the closest model point is found, and is considered to be the corresponding point. The rigid displacement for this association of model and data points is computed using, e.g, the Procrustes method, and the data point cloud is adjusted according to this displacement. Then the closest point for each data point is found in the new configuration, the displacement is found, and the data point cloud is again adjusted for this displacement. This iteration is done until a sufficiently good match is found and the scheme has converged so that the data point cloud and the model point cloud coincides with sufficient accuracy. Convergence is not guaranteed, and will depend on the initial condition and the geometry of the point clouds.

The iterative closest point method was introduced in [8], where a point-to-point distance was used, and in [14], where a point-to-plane distance was used, which means that surface normals were used in the computations. The iterative closest point method is implemented in the MATLAB Vision Toolbox, and in open-source libraries like OpenCV and the Point-Cloud Library. The method is further discussed in [59], and in [25] where the solution is found using Levenberg-Marquardt optimization in place of the Procrustes method.

Let the model be given by the point cloud $\mathbf{b}_i, i = 1, \dots, n_m$. The measurements are given by the data point cloud $\mathbf{a}_i, i = 1, \dots, n_d$. It is assumed that for corresponding points \mathbf{a}_i and $\mathbf{b}_{j(i)}$, the data point \mathbf{a}_i results form a rigid displacement

$$\mathbf{a}_i = \mathbf{R}\mathbf{b}_{j(i)} + \mathbf{t} \quad (1946)$$

of the model point $\mathbf{b}_{j(i)}$ corresponding to \mathbf{a}_i . The goal is to find \mathbf{R} and \mathbf{t} .

In the point-to-point formulations, an iterative scheme is used, where $k = 1, 2, \dots, n_k$ is the iteration number. Initialization is done by setting the initial value of the estimated rotation matrix

$$\hat{\mathbf{R}}^{(1)} = \mathbf{I} \quad (1947)$$

calculating the centroids

$$\bar{\mathbf{a}} = \frac{1}{n_d} \sum_{i=1}^{n_d} \mathbf{a}_i, \quad \bar{\mathbf{b}} = \frac{1}{n_m} \sum_{i=1}^{n_m} \mathbf{b}_i \quad (1948)$$

and the deviations

$$\delta\mathbf{a}_i = \mathbf{a}_i - \bar{\mathbf{a}} \quad (1949)$$

from a data points to the data centroid.

The first step at iteration k is to calculate the model deviation at step k from

$$\delta\mathbf{b}_{j(i)}^{(k)} = \mathbf{b}_{j(i)}^{(k)} - \bar{\mathbf{b}} \quad (1950)$$

The next step is for each data point \mathbf{a}_i , find the model point $\mathbf{b}_j^{(k)}$ so that $\delta\mathbf{a}_i - \delta\mathbf{b}_j^{(k)}$ is minimized. Then the loss function

$$L^{(k)} = \frac{1}{n_d} \sum_{i=1}^{n_d} \|\delta\mathbf{a}_i - \mathbf{R}^{(k)} \delta\mathbf{b}_{j(i)}^{(k)}\|^2 \quad (1951)$$

is minimized with respect to \mathbf{R} . This is done with the Procrustes method where $\mathbf{H} = \mathbf{B}\mathbf{A}^T$ is calculated for $\mathbf{A} = [\delta\mathbf{a}_1, \dots, \delta\mathbf{a}_{n_d}]$ and $\mathbf{B} = [\delta\mathbf{b}_{j(1)}, \dots, \delta\mathbf{b}_{j(n_d)}]$. The SVD us used to find

$\mathbf{H} = \mathbf{U}\Sigma\mathbf{V}^T$, which gives the optimal rotation matrix $\mathbf{R}^{(k)} = \mathbf{V}\text{diag}(1, 1, \det(\mathbf{V}\mathbf{U}^T))\mathbf{U}^T$ at iteration k . The model point cloud is updated according to

$$\delta\mathbf{b}_i^{(k+1)} = \mathbf{R}^{(k)}\delta\mathbf{b}_i^{(k)} \quad (1952)$$

and the rotation matrix is updated by

$$\hat{\mathbf{R}}^{(k+1)} = \hat{\mathbf{R}}^{(k)}\mathbf{R}^{(k)} \quad (1953)$$

Then the iteration is repeated until the rotation matrix has converged at $k + 1 = k_c$. The translation is then found from

$$\mathbf{t} = \bar{\mathbf{a}} - \hat{\mathbf{R}}^{(k_c)}\bar{\mathbf{b}} \quad (1954)$$

Example 1

In this example a point cloud of four points is given a rigid displacement. The solution is found by finding the closest neighbor for each point, and then solving the Procrustes problem. The solution is found in one iteration for this special problem.

```

clear;
dim = 4;
M = [1 0 0; 0 1 0 ; 0 0 1; 0 0 -1]';
Ra = Rotx(0.2)*Roty(0.3)*Rotz(0.4); ta = [1,2,3]';
Di = Ra*M + repmat(ta,1,4);
D(:,1) = Di(:,4); D(:,2) = Di(:,3); D(:,3) = Di(:,2); D(:,4) = Di(:,1);

Dc = sum(D,2)/dim;
t = zeros(3,1);
T = eye(4);
for i = 1:dim
    Dcc(:,i) = D(:,i)-Dc;
end

Mc = sum(M,2)/dim
for i = 1:dim
    Mcc(:,i) = M(:,i)-Mc;
end
mcorr = closestPoint(Dcc, Mcc)

W = zeros(3,3);
for i = 1:dim
    W = W+(Mcc(:,mcorr(i))*(Dcc(:,i))');
end

[U,S,V] = svd(W);

R = V*U';
t = Dc - R*Mc;

```

```

T = [R t; 0 0 0 1]
Ta = [Ra ta; 0 0 0 1]
deltaT = Ta\T

%%%%%%%%%%%%%
function mcorr = closestPoint(D, M)
% For each data point D(:,j), fin the closest point M(:,i)),
% and store the index i in mcorr(j)
nd = size(D,2);
mcorr = zeros(1,nd);
for j = 1:nd % for D(:,j)
    mcorr(j) = 1; dmin = norm(D(:,j) - M(:,1));
    for i = 2:nd
        if norm(D(:,j) - M(:,i)) < dmin
            dmin = norm(D(:,j) - M(:,i));
            mcorr(j) = i;
        end
    end
end
end

```

□

Example 2

In this example a point cloud of 10 points is given a rigid displacement. The solution is found by the iterative closest point method. At each step the Procrustes problem is solved.

```

clear;
dim = 10;
M = rand(3,dim)-0.5;
skew =@(u) [0 -u(3) u(2); u(3) 0 -u(1);-u(2) u(1) 0];
expso3 =@(u) eye(3)+sinc(norm(u)/pi)*skew(u) ...
+0.5*(sinc(norm(u)/(2*pi)))^2*(skew(u))^2;

Ra = expso3(0.2*[1;0;0])*expso3(0.3*[0;1;0])*expso3(0.4*[0;0;1]);
ta = [1,2,3]';
D = Ra*M + repmat(ta,1,dim) + 0.02*(rand(3,dim)-0.5);

Dc = mean(D,2);
Mc = mean(M,2);
for i = 1:dim
    Dcc(:,i) = D(:,i)-Dc;
    Mcc(:,i) = M(:,i)-Mc;
end
R = eye(3);
for j = 1:4

```

```

mcorr = closestPoint(Dcc, Mcc)
W = zeros(3,3);
for i = 1:dim
    W = W+(Mcc(:,mcorr(i))*(Dcc(:,i))');
end
[U,S,V] = svd(W);
Rj = V*diag([1,1,det(V*U')])*U';
Mcc = Rj*Mcc;
R = R*Rj;
end
t = Dc - R*mean(M,2);
T = [R t; 0 0 0 1]
Ta = [Ra ta; 0 0 0 1]
deltaT = Ta\T

```

□

26.3 Point-Feature Histogram (PFH)

The point-feature histogram (PFH) is a local descriptor for point clouds. The descriptor for a point \mathbf{p} is based on the deflections from the surface normal \mathbf{n}_i at \mathbf{p}_i to the surface normals of a set of adjacent points, and the angle between the normal vector and the unit vector

$$\mathbf{d} = \frac{\mathbf{p}_j - \mathbf{p}_i}{|\mathbf{p}_j - \mathbf{p}_i|} \quad (1955)$$

along the vector from \mathbf{p}_i to \mathbf{p}_j . This is described with three angles α , ϕ and θ .

Let frame i with orthogonal unit vectors $\mathbf{i}_x, \mathbf{i}_y, \mathbf{i}_z$ be defined with the origin in the point \mathbf{p}_i , and with $\mathbf{i}_z = \mathbf{n}_i$, which means that the z_i axis is along the surface normal \mathbf{n}_i . A neighboring point \mathbf{p}_j with surface normal \mathbf{n}_j is considered. Let the vector \mathbf{d} be given in frame i as

$$\mathbf{d} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}, \quad \mathbf{d}_p = \begin{bmatrix} d_x \\ d_y \\ 0 \end{bmatrix} \quad (1956)$$

The frame i is rotated by an angle ψ_x about the z_i axis to a frame a with orthogonal unit vectors $\mathbf{a}_x, \mathbf{a}_y, \mathbf{a}_z$ so that the \mathbf{a}_x axis is along \mathbf{d}_p . This is achieved with

$$\psi_x = \text{Atan2}(d_y, d_x) \quad (1957)$$

Next, frame a is rotated to a frame b with orthogonal unit vectors $\mathbf{b}_x, \mathbf{b}_y, \mathbf{b}_z$ by an angle ψ_y about the current y_a axis where

$$\psi_y = \text{Atan2}(\mathbf{a}_x^T \mathbf{n}_j, \mathbf{a}_z^T \mathbf{n}_j) \quad (1958)$$

Finally, the b frame is rotated to the c frame by an angle ψ_x about the \mathbf{x}_b axis, where

$$\psi_x = \text{Atan2}(-\mathbf{b}_y^T \mathbf{n}_j, \mathbf{b}_z^T \mathbf{n}_j) \quad (1959)$$

In the presentation of the descriptor the vectors $\mathbf{u} = -\mathbf{a}_z$, $\mathbf{v} = -\mathbf{a}_y$ and $\mathbf{w} = \mathbf{a}_x$ are used, and the angles of the descriptor are given by

$$\alpha = \arccos(\mathbf{v}^T \mathbf{n}_j) \quad (1960)$$

$$\theta = \text{Atan2}(\mathbf{a}_x^T \mathbf{n}_j, \mathbf{a}_z^T \mathbf{n}_j) \quad (1961)$$

$$\phi = \arccos(\mathbf{u}^T \mathbf{d}) \quad (1962)$$

where the angles α and θ describes the orientation of \mathbf{n}_j relative to \mathbf{n}_i , and are given in terms of the Euler angles as by

$$\alpha = -\psi_x + \frac{\pi}{2} \quad (1963)$$

$$\theta = \psi_y \quad (1964)$$

The last angle ϕ is the angle between \mathbf{n}_i and $\mathbf{p}_j - \mathbf{p}_i$, which is $\pi/2$ when the surface is flat.

26.4 Viewpoint feature histogram (VFH)

In the viewpoint feature histogram (Rusu 2010) the centroid \mathbf{p}_c of all the points \mathbf{p}_i is computed, and this centroid is assigned a unit normal vector \mathbf{n}_c which is referred to as the average of the normal vectors \mathbf{n}_i . Then the angles α , θ and ϕ of the PFH are computed between each point \mathbf{p}_i with normal \mathbf{n}_i and the centroid \mathbf{p}_c with normal \mathbf{n}_c . In addition the angle β_i is calculated from

$$\cos \beta_i = \mathbf{n}_i \cdot \frac{\mathbf{p}_c}{|\mathbf{p}_c|} \quad (1965)$$

The viewpoint feature histogram then has 45 bins for each of the angles α , θ and ϕ , and 128 bins for β .

26.5 Radius-based Surface Descriptor (RSD)

Consider a point \mathbf{p}_1 with surface normal \mathbf{n}_1 and a point \mathbf{p}_2 with surface normal \mathbf{n}_2 . Let the angle between the surface normals be α . The radius-based surface descriptor is based on the idea that a circle with radius r is fitted to the two points.

The center of the circle is denoted \mathbf{c} . The vector from the circle center to \mathbf{p}_1 is denoted \mathbf{u}_1 and the vector from the origin to \mathbf{p}_2 is \mathbf{u}_2 . The angle between \mathbf{u}_1 and \mathbf{u}_2 is denoted α . Let the vector from \mathbf{p}_1 to \mathbf{p}_2 be denoted $\mathbf{v} = \mathbf{p}_2 - \mathbf{p}_1$ with length $d = |\mathbf{v}|$. The perpendicular bisector \mathbf{u} of \mathbf{v} will be a line from the origin to the midpoint of \mathbf{v} so that the angle between \mathbf{u}_1 and \mathbf{u} is $\alpha/2$.

Then it follows that

$$\frac{d}{2} = r \sin \frac{\alpha}{2} \quad (1966)$$

The Taylor expansion

$$\sin x = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 + \dots \quad (1967)$$

of the sine function then gives

$$d = r\alpha - \frac{1}{24}r\alpha^3 + \frac{1}{1920}r\alpha^5 + \dots \quad (1968)$$

This can be approximated with $d = r\alpha$, and the radius is found from

$$r = \frac{d}{\alpha} \quad (1969)$$

In the original formulation the expression is given as

$$d = \sqrt{2}r\sqrt{1 - \cos \alpha} \quad (1970)$$

Squaring of both sides gives

$$d^2 = 2r^2(1 - \cos \alpha) = 4r^2 \sin^2 \frac{\alpha}{2} \quad (1971)$$

which leads to (1966).

27 PnP and PnL

27.1 P4P with 4 points in a plane

A well known method is the case where 4 model points have a known position in a plane. This technique is described in textbooks like [30, 67].

The object frame o is then assigned so that the model points are in the $x_o y_o$ plane. The pose of frame o relative to the camera frame c is given by

$$\mathbf{T}_o^c = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1972)$$

where $\mathbf{R} = \mathbf{R}_o^c$ and $\mathbf{t} = \mathbf{t}_{oc}^c$. Four model $\mathbf{p}_1^o, \mathbf{p}_2^o, \mathbf{p}_3^o, \mathbf{p}_4^o$ fixed in the object frame is known with homogeneous coordinates $\tilde{\mathbf{p}}_i^o = (x_i, y_i, 0, 1)^T$, $i = 1, 2, 3, 4$. This is shown in Figure 85.

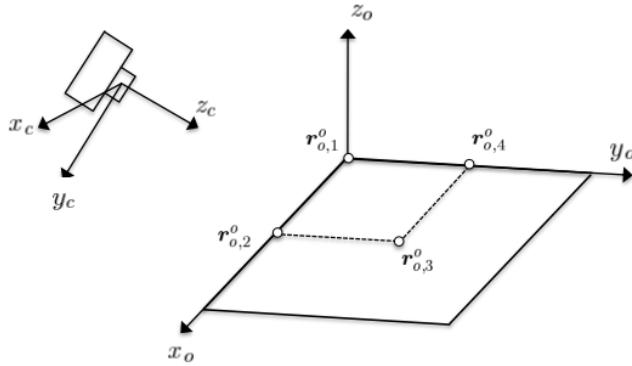


Figure 85: Four points in the xy plane of the object frame.

The normalized image coordinates of the 4 points are i are

$$\tilde{\mathbf{s}}_i = [\mathbf{R} \quad \mathbf{t}] \tilde{\mathbf{p}}_i^o = [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{r}_3 \quad \mathbf{t}] \tilde{\mathbf{p}}_i^o = \mathbf{H} \tilde{\mathbf{x}}_i$$

where the homography is

$$\mathbf{H} = [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}]$$

and $\tilde{\mathbf{x}}_i = [x_i, y_i, 1]^T$. Then, with 4 point mappings $\tilde{\mathbf{s}}_i = \mathbf{H} \tilde{\mathbf{x}}_i$ the linear homogeneous system

$$\mathbf{A} \mathbf{h} = 0 \quad (1973)$$

can be established where \mathbf{h} are the stacked columns of \mathbf{H} , and the solution is found in the usual way using SVD. The pose is then found as

$$\mathbf{T}_o^c = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_1^\times \mathbf{r}_2 & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1974)$$

27.2 PnP with control points: EPnP

The EPnP method [47], [36] is an efficient implementation of the PnP problem, where a set of model points are given, and the a set of corresponding image coordinates are measured with a

camera, and where the objective is to find the rigid body displacement from the model points to the points corresponding to the image points.

In the EPnP formulation there are n homogeneous model points which are given in the world frame w as $\tilde{\mathbf{p}}_i^w = [\mathbf{p}_i^{wT}, 1]^T$ and n corresponding normalized image points $\tilde{\mathbf{s}}_i$ for $i = 1, \dots, n$. The problem is to find the homogeneous transformation matrix \mathbf{T}_w^c from the camera frame c to the world frame w .

The main idea of the EPnP method is to select 4 homogeneous control points $[\tilde{\mathbf{c}}_j^w = \mathbf{c}_j^{wT}, 1]^T$ for $j = 1, 2, 3, 4$ and represent the model points in terms of the control points as

$$\tilde{\mathbf{p}}_i^w = \sum_{j=1}^4 \alpha_{ij} \tilde{\mathbf{c}}_j^w \quad (1975)$$

where α_{ij} are scalar coefficients. Then the point correspondences $(\tilde{\mathbf{s}}_i, \tilde{\mathbf{p}}_i^w)$ are used to find the control points as $\tilde{\mathbf{c}}_j^c = \mathbf{T}_w^c \tilde{\mathbf{c}}_j^w$ in the camera frame, and finally the displacement \mathbf{T}_w^c is calculated from the 4 point correspondences $(\tilde{\mathbf{c}}_j^w, \tilde{\mathbf{c}}_j^c)$, which can be done, e.g., with the Procrustes method.

The coefficients satisfy $\sum_{j=1}^4 \alpha_{ij} = 1$, which follows from the last row of (1975). Moreover,

$$\mathbf{p}_i^w = \sum_{j=1}^4 \alpha_{ij} \mathbf{c}_j^w \quad (1976)$$

which follows from the 3 first lines of (1975). The n model points are imaged to the normalized image coordinates as

$$\lambda_i \tilde{\mathbf{s}}_i = \mathbf{p}_i^c = \mathbf{\Pi} \tilde{\mathbf{p}}_i^c = \mathbf{\Pi} \mathbf{T}_w^c \tilde{\mathbf{p}}_i^w \quad (1977)$$

where λ_i is the depth coordinate and

$$\mathbf{\Pi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (1978)$$

is the projection matrix which recovers the non-homogeneous part from a homogeneous vector when the homogeneous coordinate is unity. The problem is then to calculate the transformation $\mathbf{T}_w^c \in SE(3)$ given the known model points $\tilde{\mathbf{p}}_i^w$, the control points $\tilde{\mathbf{c}}_j^w$, the coefficients α_{ij} and the image points $\tilde{\mathbf{s}}_i$.

The homogeneous model points are given in the camera frame as

$$\tilde{\mathbf{p}}_i^c = \mathbf{T}_w^c \tilde{\mathbf{p}}_i^w = \mathbf{T}_w^w \sum_{j=1}^4 \alpha_{ij} \tilde{\mathbf{c}}_j^w = \sum_{j=1}^4 \alpha_{ij} \mathbf{T}_w^w \tilde{\mathbf{c}}_j^w = \sum_{j=1}^4 \alpha_{ij} \tilde{\mathbf{c}}_j^c \quad (1979)$$

which gives

$$\mathbf{p}_i^c = \sum_{j=1}^4 \alpha_{ij} \mathbf{c}_j^c \quad (1980)$$

It follows that the normalized image coordinates are given by

$$\lambda_i \tilde{\mathbf{s}}_i = \sum_{j=1}^4 \alpha_{ij} \mathbf{c}_j^c \quad (1981)$$

where $\tilde{\mathbf{s}}_i$ are known image coordinates. The unknown control points in the camera frame can then be calculated from the linear equation

$$\mathbf{A}_i \mathbf{x} = \mathbf{0} \quad (1982)$$

where

$$\mathbf{x} = \begin{bmatrix} \mathbf{c}_1^c \\ \vdots \\ \mathbf{c}_4^c \end{bmatrix} \quad (1983)$$

and

$$\mathbf{A}_i = \begin{bmatrix} \alpha_{i1} & \alpha_{i2} & \alpha_{i3} & \alpha_{i4} \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & -s_{xi} \\ 0 & 1 & -s_{yi} \end{bmatrix} \quad (1984)$$

where \otimes denotes the Kronecker product. With n points this gives

$$\begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_n \end{bmatrix} \mathbf{x} = \mathbf{0} \quad (1985)$$

and \mathbf{x} can be found with SVD [23]. The homogeneous transformation \mathbf{T}_w^c is then found with Procrustes method from the 4 point correspondences

$$\tilde{\mathbf{c}}_{c,j}^c = \mathbf{T}_w^c \tilde{\mathbf{c}}_{w,j}^w \quad (1986)$$

The expression for the \mathbf{A}_i matrix is derived as follows. First it is observed that (1981) can be written in the form

$$\lambda_i \tilde{\mathbf{s}}_i = \mathbf{B}_i \mathbf{x} \quad (1987)$$

where

$$\mathbf{B}_i = \begin{bmatrix} \alpha_{i1} \mathbf{I}_3 & \alpha_{i2} \mathbf{I}_3 & \alpha_{i3} \mathbf{I}_3 & \alpha_{i4} \mathbf{I}_3 \end{bmatrix} \quad (1988)$$

The unknown λ_i is eliminated with the third row. Let $\tilde{\mathbf{s}}_i = [s_{xi}, s_{yi}, 1]^T$ and $\mathbf{B}_i \mathbf{x} = [\beta_{i1}, \beta_{i2}, \beta_{i3}]^T$. Then (1981) can be written in component form as

$$\begin{bmatrix} \lambda_i s_{xi} \\ \lambda_i s_{yi} \\ \lambda_i \end{bmatrix} = \begin{bmatrix} \beta_{i1} \\ \beta_{i2} \\ \beta_{i3} \end{bmatrix} \quad (1989)$$

Insertion of $\lambda_i = \beta_{i3}$ in the two first equations gives

$$\begin{bmatrix} \beta_{i3} s_{xi} \\ \beta_{i3} s_{yi} \end{bmatrix} = \begin{bmatrix} \beta_{i1} \\ \beta_{i2} \end{bmatrix} \quad (1990)$$

which is written in the form

$$\begin{bmatrix} 1 & 0 & -s_{xi} \\ 0 & 1 & -s_{yi} \end{bmatrix} \begin{bmatrix} \beta_{i1} \\ \beta_{i2} \\ \beta_{i3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (1991)$$

This can be written

$$\begin{bmatrix} 1 & 0 & -s_{xi} \\ 0 & 1 & -s_{yi} \end{bmatrix} \begin{bmatrix} \alpha_{i1} \mathbf{I}_3 & \alpha_{i2} \mathbf{I}_3 & \alpha_{i3} \mathbf{I}_3 & \alpha_{i4} \mathbf{I}_3 \end{bmatrix} \mathbf{x} = \mathbf{0} \quad (1992)$$

which gives

$$\mathbf{A}_i = \begin{bmatrix} 1 & 0 & -s_{xi} \\ 0 & 1 & -s_{yi} \end{bmatrix} \begin{bmatrix} \alpha_{i1} \mathbf{I}_3 & \alpha_{i2} \mathbf{I}_3 & \alpha_{i3} \mathbf{I}_3 & \alpha_{i4} \mathbf{I}_3 \end{bmatrix} \quad (1993)$$

$$= \begin{bmatrix} \alpha_{i1} & \alpha_{i2} & \alpha_{i3} & \alpha_{i4} \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & -s_{xi} \\ 0 & 1 & -s_{yi} \end{bmatrix} \quad (1994)$$

27.3 DLT-Plücker-Lines

Suppose that m model lines are given in the world frame w in Plücker coordinates as the screw

$$\mathbf{L}_{i/w}^w = \begin{bmatrix} \mathbf{a}_i^w \\ \mathbf{m}_{i/w}^w \end{bmatrix} \quad (1995)$$

where \mathbf{a}_i^w is the unit direction vector of the line, and $\mathbf{m}_{i/w}^w = \mathbf{q}_{i/w}^{w \times} \mathbf{a}_i^w$ is the moment, where $\mathbf{q}_{i/w}^w$ is the vector from the origin of frame w to a point on the line. If $\mathbf{q}_{i/w}^w = \mathbf{a}_i^{w \times} \mathbf{m}_{i/w}^w$, then $\mathbf{q}_{i/w}^w$ is the point on the line that is closest to the reference point. The line is given in the camera frame by

$$\mathbf{L}_{i/c}^c = \begin{bmatrix} \mathbf{a}_i^c \\ \mathbf{m}_{i/c}^c \end{bmatrix} \quad (1996)$$

where $\mathbf{m}_{i/c}^c = \mathbf{q}_{i/c}^{c \times} \mathbf{a}_i^c$ is the moment, where $\mathbf{q}_{i/c}^c$ is the vector from the origin of frame c to a point on the line. The displacement from the c frame to the w frame is given by

$$\mathbf{T}_w^c = \begin{bmatrix} \mathbf{R}_w^c & \mathbf{p}_{cw}^c \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3) \quad (1997)$$

The simplified notation $\mathbf{T} = \mathbf{T}_w^c$, $\mathbf{R} = \mathbf{R}_w^c$, $\mathbf{p} = \mathbf{p}_{cw}^c$ is used, and the displacement is written $\mathbf{T} = (\mathbf{R}, \mathbf{p}) \in SE(3)$. The screw transformation is

$$\mathbf{L}_{i/c}^c = \mathbf{U} \mathbf{L}_{i/w}^w \quad (1998)$$

where

$$\mathbf{U} = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{p}^{c \times} \mathbf{R} & \mathbf{R} \end{bmatrix} \quad (1999)$$

Let π be the plane which contains the origin of frame c and the line $\mathbf{L}_{i/c}^c$. Then the plane π will contain the vectors $\mathbf{q}_{i/c}^c$ and \mathbf{a}_i^c . This means that the moment $\mathbf{m}_{i/c}^c = \mathbf{q}_{i/c}^{c \times} \mathbf{a}_i^c$ must be normal to the plane. It follows that the plane can be given in dual Plücker coordinates in frame c as

$$\pi = (0, \mathbf{m}_{i/c}^c) \quad (2000)$$

where it is used that the plane contains the origin of c .

The calibrated camera model is assumed to be

$$\tilde{\mathbf{s}} = [\mathbf{I} \mid \mathbf{0}] \tilde{\mathbf{r}}_c \quad (2001)$$

where $\tilde{\mathbf{r}}$ is the homogeneous position of a 3D world point in the camera frame, and $\tilde{\mathbf{s}} = [s_x, s_y, 1]^T$ is the homogeneous 2D point in the normalized image plane. If two points on the

line are imaged to the points \tilde{s}_1 and \tilde{s}_2 , then \tilde{s}_1 and \tilde{s}_2 will be in the plane π , and the line $\ell_i = \tilde{s}_1 \times \tilde{s}_2$ in the normalized image plane must be normal to the plane π . This means that $\ell_i = \mathbf{m}_{i/c}^c$ up to a scaling factor.

This means that if ℓ_i is the line in the image that corresponds to the 3D line L_i . It follows that the lines will satisfy

$$\ell_i = \mathbf{P} L_{i/w}^w \quad (2002)$$

where

$$\mathbf{P} = [\mathbf{p}^\times \mathbf{R} \quad \mathbf{R}] \quad (2003)$$

Then, with a set of line correspondences (ℓ_i, L_i) , the elements of the matrix \mathbf{P} can be found. This is done in the method DLT-Plücker-Lines proposed by [56, 57].

Suppose that the matrix \mathbf{P} is computed as \mathbf{P}_c , and the rotation matrix \mathbf{R}_c is found as the last 3 columns of \mathbf{P} . First a scaling is done so to have a determinant equal to unity. This is done by computing $\mathbf{R}_c = \mathbf{U}_c \Sigma_c \mathbf{V}_c^T$, and setting the scaling factor s which is the inverse of the mean of the singular values, which can be found as $s = [\text{tr}(\Sigma_c)/3]^{-1}$, or equivalently, $s = [\det(\mathbf{R}_c)]^{-1}$. The matrix \mathbf{P}_c is then scaled according to

$$\mathbf{P} = s \mathbf{P}_c \quad (2004)$$

The submatrix defined by the first 3 columns of \mathbf{P} is denoted \mathbf{Q} , which means that $\mathbf{Q} = \mathbf{p}^\times \mathbf{R}$. This matrix has the same structure as the essential matrix. This means that \mathbf{p} and \mathbf{R} can be computed from \mathbf{Q} with the same technique as for the reconstruction from the essential matrix.

The solution is

$$\mathbf{t}_1^\times = \mathbf{U} \mathbf{R}_z \left(+\frac{\pi}{2} \right) \Sigma \mathbf{U}^T, \quad \mathbf{R}_1 = \mathbf{U} \mathbf{R}_z^T \left(+\frac{\pi}{2} \right) \mathbf{V}^T \quad (2005)$$

$$\mathbf{t}_2^\times = \mathbf{U} \mathbf{R}_z \left(-\frac{\pi}{2} \right) \Sigma \mathbf{U}^T, \quad \mathbf{R}_2 = \mathbf{U} \mathbf{R}_z^T \left(-\frac{\pi}{2} \right) \mathbf{V}^T \quad (2006)$$

where $\mathbf{Q} = \mathbf{U} \Sigma \mathbf{V}^T$, $\Sigma = \frac{\sigma_1 + \sigma_2}{2} \text{diag}[1, 1, 0]$ and

$$\mathbf{R}_z \left(\pm \frac{\pi}{2} \right) = \begin{bmatrix} 0 & \mp 1 & 0 \\ \pm 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2007)$$

The orthogonal matrices \mathbf{U} and \mathbf{V} will have determinant equal to unity in the ideal case. In the noisy case, this may not be the case. Then, if both \mathbf{U} and \mathbf{V} have determinants of -1 , the sign is changed for both with $\mathbf{U} := -\mathbf{U}$ and $\mathbf{V} := -\mathbf{V}$. If only \mathbf{U} has negative determinant, then $\mathbf{U} := -\mathbf{U}$, which is the same as setting $\mathbf{Q} := -\mathbf{Q}$, which is the same as changing the sign of the scaling of the homogeneous matrix \mathbf{Q} . In the same way, if only \mathbf{V} has a negative determinant, then $\mathbf{V} := -\mathbf{V}$ is used.

28 Graduated non-convexity for outlier rejection

28.1 Introduction

The registration problem is the problem of finding the transformation between a model set of points and a data set of points. One example is point-cloud registration where a point cloud

from a camera is matched with a point cloud from model to the transformation between the point clouds. This includes the Procrustes problem where only rotation is unknown. Another example is absolute pose estimation where a set of image points are matched with a mode point cloud. This includes the Perspective-n-Point (PnP) problem. A major challenge in registration is the outlier problem due to false point correspondences. The usual method for handling this problem is to use the RANSAC method [24]. An alternative method is Graduated Non-Convexity (GNC) which is described in the following. GNC was proposed by Blake and Zisserman in 1987 [10], and further developed in [9], and it was recently used with good results in, e.g., [81] for outlier rejection.

28.2 Robust cost function

In this section the concept of a robust cost function is introduced for the Procrustes problem. In this problem rotation $\mathbf{R} \in SO(3)$ is found as the least squares solution to $\mathbf{b}_i = \mathbf{R}\mathbf{a}_i$ for two sets of points $\{\mathbf{a}_1, \dots, \mathbf{a}_N\}$ and $\{\mathbf{b}_1, \dots, \mathbf{b}_N\}$. If there are no outliers, then this problem can be solved in closed form using the Procrustes solution as in [4]. This approach minimizes the least-squares cost function

$$f_P(\mathbf{a}_i, \mathbf{y}_i, \mathbf{R}) = \sum_{i=1}^N \|\mathbf{b}_i - \mathbf{R}\mathbf{a}_i\|^2 \quad (2008)$$

This problem is a special case of the general least-squares problem given as the minimization of the least-squares cost function

$$f_{LS}(\mathbf{y}_i, \mathbf{x}) = \sum_{i=1}^N r^2(\mathbf{y}_i, \mathbf{x}) \quad (2009)$$

where \mathbf{y}_i is measurement i , $r(\mathbf{y}_i, \mathbf{x})$ is the residual of measurement i and \mathbf{x} is the variable to be determined. In the case of Procrustes problem with point correspondences $\mathbf{a}_i = \mathbf{R}\mathbf{b}_i$ the measurements are $\mathbf{y}_i = (\mathbf{a}_i, \mathbf{b}_i)$ and the variable to be determined is $\mathbf{x} = \mathbf{R}$.

A potential problem with the least-squares method is that it is sensitive to outliers. There are methods to remove outliers from the data-set, but such methods may remove inliers instead of outliers as shown in the introductory regression example of [24]. The usual solution to outlier rejection has been to use RANSAC.

A recent alternative to RANSAC is outlier-robust estimation techniques [3], [81]. The squared residual $r^2(\mathbf{y}_i, \mathbf{x})$ is then replaced by a robust cost $\rho(r(\mathbf{R}\mathbf{a}_i, \mathbf{b}_i))$ for measurement i , which gives the cost function

$$f_\rho(\mathbf{y}_i, \mathbf{x}) = \sum_{i=1}^N \rho(r(\mathbf{y}_i, \mathbf{x})) \quad (2010)$$

The robust cost $\rho(r)$ can be the Huber loss, a Geman-McClure cost or a truncated least squares cost [9].

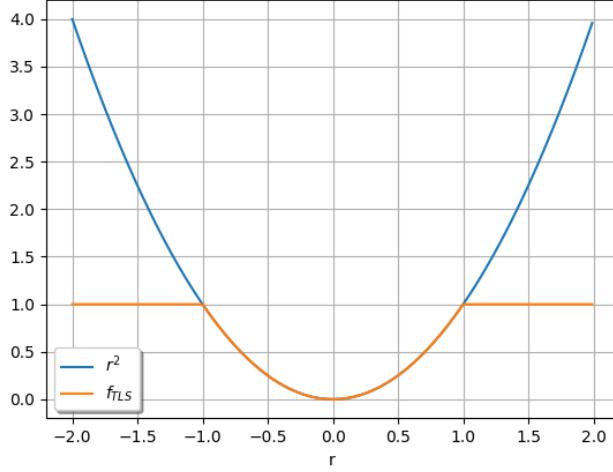


Figure 86: The least-squares cost function and the truncated least-squares cost function f_{TLS} with $\epsilon = 1$.

28.3 Truncated least squares

In the following, the truncated least squares cost used as in [81]. The robust cost of the truncated least squares is given by

$$\rho(r) = \min(r^2, \epsilon^2) = \begin{cases} r^2, & r^2 \leq \epsilon^2 \\ \epsilon^2, & \epsilon^2 \leq r^2 \end{cases} \quad (2011)$$

where ϵ is the truncation threshold. The cost function of the truncated least squares problem is then

$$f_{\text{TLS}}(\mathbf{y}_i, \mathbf{x}) = \sum_{i=1}^N \min(r^2(\mathbf{y}_i, \mathbf{x}), \epsilon^2) \quad (2012)$$

The function is shown in Figure 86.

In this formulation a measurement i is referred to as an inlier if $r^2 \leq \epsilon^2$, while it is an outlier if $r^2 \geq \epsilon^2$. Let the set of indices for the measurements be denoted by \mathcal{M} . In this case, $\mathcal{M} = \{1, \dots, N\}$. The set of outliers is denoted by \mathcal{O} , while the set of inliers is $\mathcal{I} = \mathcal{M} \setminus \mathcal{O}$, which is the set of points that are contained in \mathcal{M} but not in \mathcal{O} . The cost function in the truncated least squares problem can be written in terms of inliers and outliers as

$$f_{\text{TLS}}(\mathbf{y}_i, \mathbf{x}) = \sum_{i \in \mathcal{I}} r^2(\mathbf{y}_i, \mathbf{x}) + \sum_{i \in \mathcal{O}} \epsilon^2 \quad (2013)$$

Note that the cost of the outliers is independent from the variable \mathbf{x} . This means that only the inliers are used in the minimization of the truncated least-squares cost with respect to \mathbf{x} .

The truncated least-squares cost can also be written in terms of the binary weighting variables $w_i \in \{0, 1\}$. The cost function is the written

$$f_{\text{TLS}}(\mathbf{y}_i, \mathbf{x}) = \sum_{i \in \mathcal{M}} \min_{w_i \in \{0, 1\}} (w_i r^2(\mathbf{y}_i, \mathbf{x}) + (1 - w_i) \epsilon^2) \quad (2014)$$

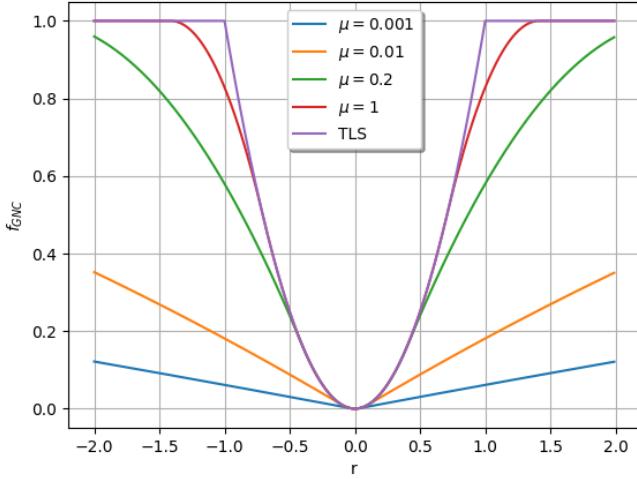


Figure 87: The surrogate cost function f_{GNC} for the graduated non-convex formulation of the truncated least-squares problem for different values of the control parameter μ .

where $w_i = 1$ for inliers and $w_i = 0$ for outliers. It is seen that this is equivalent to (2013)

28.4 Graduated non-convexity cost function for truncated least squares

A major problem with the minimization of robust cost functions like the truncated least squares function is that the minimization problem is not convex, since the truncated least squares function $\rho(r)$ is not convex. This is solved in the graduated non-convexity method with a regularization of the $(1 - w_i)\epsilon^2$ term. This is done so that the resulting regularized cost function is convex, and a global minimum can be found. Then the regularized cost is gradually changed so that it tends to the truncated least-squares cost. This type of approach is called a continuation method, which is widely used in optimization of non-convex functions.

This is done in the graduated non-convexity method by minimization of the surrogate cost function [3]

$$f_{\text{GNC}}(\mathbf{y}_i, \mathbf{x}) = \sum_{i \in \mathcal{M}} \min_{w_i \in [0,1]} \left(w_i r^2(\mathbf{y}_i, \mathbf{x}) + \frac{\mu(1-w_i)}{\mu + w_i} \epsilon^2 \right) \quad (2015)$$

The function is shown in Figure 87 for different values of the control parameter μ .

Note that the minimization of the weighting factors w_i for each i in (2015) is taken for the interval $0 \leq w_i \leq 1$. It is seen that the surrogate function f_{GNC} tends to the original least-squares cost function when $\mu \rightarrow 0$, while the truncated least squares function is recovered when $\mu \rightarrow \infty$.

The minimization of (2014) is then performed a sequence of minimization problems where the surrogate cost function (2015) is minimized for a sequence of increasing values of the variable μ . The first minimization problem is solved for a small $\mu = \mu_0$, which gives a convex problem. Then μ is increased to $\mu_1 = \gamma \mu_0$, for some $\gamma > 1$, and a new minimization problem is solved

with $\mu = \mu_1$. Minimization problem i is solved for $\mu_i = \gamma\mu_{i-1}$, and the process is terminated when the surrogate cost function is sufficiently close to the truncated least-squares cost (2014).

The formulation of the graduated non-convexity method is simplified by noticing that the minimization of the surrogate cost function with respect to $w_i \in [0, 1]$ can be solved with a closed form solution of w_i [81]. The minimization of the surrogate cost function can then be replaced by the minimization problem

$$\min_{\mathbf{x}} \sum_{i \in \mathcal{M}} w_i r_i^2(\mathbf{y}_i, \mathbf{x}) \quad (2016)$$

where the weights are given by

$$w_i = \begin{cases} 1 & \text{when } |r_i| \leq \epsilon \sqrt{\frac{\mu}{\mu+1}} \\ 0, & \text{when } |r_i| \geq \epsilon \sqrt{\frac{\mu+1}{\mu}} \\ \frac{\epsilon}{|r_i|} \sqrt{\mu(\mu+1)} - \mu, & \text{otherwise} \end{cases} \quad (2017)$$

as shown in Figure 88.

The graduated non-convexity problem is then solved as a sequence of minimization problems (2016) where minimization problem j has the control parameter $\mu_j = \gamma\mu_{j-1}$, and where w_i is given by (2017).

In the first iteration the weights are set to $w_i = 1$ for all i . This means that the first iteration is a minimization of the least-squares cost function $\sum_{i \in \mathcal{M}} r_i^2(\mathbf{y}_i, \mathbf{x})$, which is convex. The parameter μ is then gradually increased, and the cost function will approach the truncated least-squares cost. In this process measurements with large residuals will have decreasing costs, and will have $w_i = 0$ when μ becomes sufficiently large, which means that these measurements will not be included in the optimization for large μ . Points with small residuals will have $w_i = 1$, and will be included in the optimization. The method is described in detail in [3], where also Matlab code is included.

28.5 Minimization of the surrogate cost function with respect to the weights

This closed-form solution for w_i is found through the minimization

$$\min_{w_i \in [0,1]} f(w_i) = \min_{w_i \in [0,1]} \left(w_i r_i^2 + \frac{\mu(1-w_i)}{\mu+w_i} \epsilon^2 \right) \quad (2018)$$

The derivative with respect to w_i is

$$g_i(w_i) = \frac{d}{dw_i} f_i(w_i) = r_i^2 - \frac{\mu(\mu+1)}{(\mu+w_i)^2} \epsilon^2 \quad (2019)$$

when $0 \leq w_i \leq 1$. It is seen that $g_i(w_i) = 0$ gives

$$r_i^2 = \frac{\mu(\mu+1)}{(\mu+w_i)^2} \epsilon^2 \quad (2020)$$

It follows from (2020) that $g_i = 0$ when

$$(\mu+w_i)^2 r_i^2 = \epsilon^2 \mu(\mu+1) \quad (2021)$$

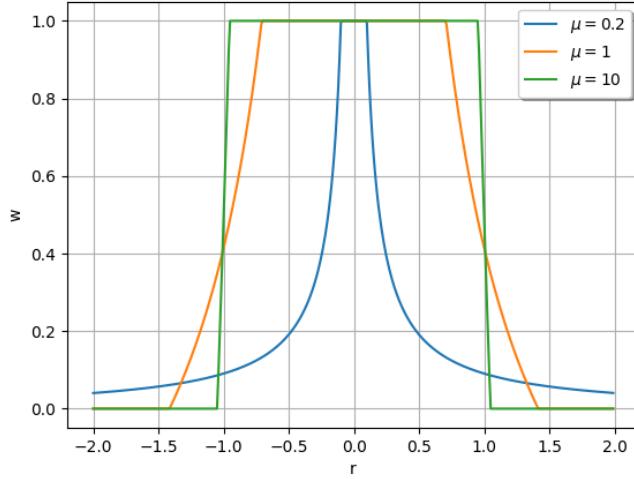


Figure 88: Optimal weights in the graduated non-convexity method.

This can be solved for w_i , which gives

$$w_i = \frac{\epsilon}{|r_i|} \sqrt{\mu(\mu+1)} - \mu \quad (2022)$$

From (2021) it is seen that $g_i(w_i) = 0$ in combination with $0 \leq w_i \leq 1$ implies that

$$\frac{\mu}{\mu+1} \epsilon^2 \leq r_i^2 \leq \frac{\mu+1}{\mu} \epsilon^2 \quad (2023)$$

Here the lower bound corresponds to $w_i = 1$ and the upper bound corresponds to $w_i = 0$. When r_i^2 does not satisfy (2023), then the minimization with respect to w_i is done by inspection of (2018). It is seen that when r_i^2 is larger than the upper bound in (2023), then $r_i^2 > \epsilon^2$ and $w_i = 0$ is the optimal value, while for r_i^2 smaller than the lower bound in (2023), then $r_i^2 < \epsilon^2$ and $w_i = 1$ is the optimal value.

This means that the optimal solution for w_i is

$$w_i = \begin{cases} 1 & \text{when } |r_i| \leq \epsilon \sqrt{\frac{\mu}{\mu+1}} \\ 0, & \text{when } |r_i| \geq \epsilon \sqrt{\frac{\mu+1}{\mu}} \\ \frac{\epsilon}{|r_i|} \sqrt{\mu(\mu+1)} - \mu, & \text{otherwise} \end{cases} \quad (2024)$$

It is noted that when $\mu \rightarrow \infty$ then $\epsilon \sqrt{\mu/(\mu+1)} \rightarrow \epsilon$ and $\epsilon \sqrt{(\mu+1)/\mu} \rightarrow \epsilon$. It follows that

$$\mu \rightarrow \infty \Rightarrow w_i \rightarrow \begin{cases} 1 & \text{when } r_i < \epsilon \\ 0, & \text{when } r_i > \epsilon \end{cases} \quad (2025)$$

which ensures that w_i converges to $\{0, 1\}$ when $\mu \rightarrow \infty$.

To sum up, minimization of the surrogate function f_{GNC} with respect to \mathbf{x} be written as

$$\min_{\mathbf{x}} f_{\text{GNC}}(\mathbf{y}_i, \mathbf{x}) = \min_{\mathbf{x}} \sum_{i \in \mathcal{M}} \left(w_i r_i^2(\mathbf{y}_i, \mathbf{x}) + \frac{\mu(1 - w_i)}{\mu + w_i} \epsilon^2 \right) \quad (2026)$$

where w_i is a fixed variable given by (2017).

It is seen that the second term in the bracket of the cost function does not depend on \mathbf{x} . This means that solution \mathbf{x} of the minimization problem (2026) is the same as for the minimization problem

$$\min_{\mathbf{x}} \sum_{i \in \mathcal{M}} w_i r_i^2(\mathbf{y}_i, \mathbf{x}) \quad (2027)$$

28.6 Graduated non-convexity for the average of a set of real numbers

A simple example of the use of GNC is to study the problem of finding the average of a set of real numbers, where the set contains outliers that should be removed before the average is calculated. A set of real numbers y_1, \dots, y_n are given, and the GNC problem is to solve a sequence of problems indexed by $j = 0, \dots, N$ where the average x minimizes

$$L(x) = \sum_{i=1}^n w_i r^2(y_i, x) = \sum_{i=1}^n w_i (y_i - x)^2 \quad (2028)$$

for the weighs are given by

$$w_i = \begin{cases} 1 & \text{when } |r_i| \leq \epsilon \sqrt{\frac{\mu_j}{\mu_j + 1}} \\ 0, & \text{when } |r_i| \geq \epsilon \sqrt{\frac{\mu_j + 1}{\mu_j}} \\ \frac{\epsilon}{|r_i|} \sqrt{\mu_j(\mu_j + 1)} - \mu_j, & \text{otherwise} \end{cases} \quad (2029)$$

for $j = 1, \dots, N$. The first problem with iteration number $j = 0$ is solved for $w_i = 1$ for all i .

Then, according to the procedure in [81] the initial value for the control parameter μ_j is set to

$$\mu_0 = \frac{\epsilon^2}{2r_{\max}^2 - \epsilon^2} \quad (2030)$$

where r_{\max}^2 is the largest value of $r^2(y_i, x)$ for $i = 1, \dots, n$. The a sequence of problems $j = 1, \dots, N$ is solved with

$$\mu_j = \gamma \mu_{j-1} \quad (2031)$$

where $\gamma = 1.4$.

The minimum of the cost function is found by calculation the derivative of the cost function with respect to x , which gives

$$\frac{\partial}{\partial x} r^2(y_i, x) = \sum_{i=1}^n w_i \frac{\partial}{\partial x} (y_i^2 - 2y_i x + x^2) \quad (2032)$$

$$= 2 \sum_{i=1}^n w_i (y_i - x) \quad (2033)$$

$$= 2 \sum_{i=1}^n w_i y_i - x \sum_{i=1}^n w_i \quad (2034)$$

The minimum is found by setting the derivative to zero, which gives

$$x = \frac{\sum_{i=1}^n w_i y_i}{\sum_{i=1}^n w_i} \quad (2035)$$

which is known as the weighted average. The weighted average x is calculated for each iteration j . Then the residuals $r(y_i, x)$ are found, and the weights w_i for the next iteration $j + 1$ are calculated from the residuals $r(y_i, x)$. The iteration is terminated when convergence is achieved, or when the maximum number of iterations has been run.

Example

```
import numpy as np

def w_from_r(r, eps, mu):
    if abs(r) < eps * np.sqrt(mu/(mu+1)):
        w = 1
    elif abs(r) > eps * np.sqrt((mu+1)/mu):
        w = 0
    else:
        w = (eps * np.sqrt(mu*(mu+1)))/abs(r) - mu
    return w

# Preparations
np.set_printoptions(formatter={'float': '{: 0.4f}'.format})

# Generation of input data
y = np.array([2.0, 2.1, 1.9, 1.95, 2.07, 1.93, 2.01, 5, 6, -5, -6, 100, 101])
n = y.shape[0]

# initial value
x0 = 1.0
# Loss function intial data
r = np.zeros(n)
for i in range(n):
    r[i] = np.linalg.norm(y[i] - x0)
r_0_max = np.max(r)
print('\n r0 =\n {}'.format(r))

# GNC initialization
eps = 0.2; mu_update_factor = 1.4
max_iterations = 1000
w = np.ones(n)
mu = eps**2 / (2*r_0_max**2 - eps**2)
print('\n mu =\n {}'.format(mu))

for i in range(max_iterations):
    # Weighted average
```

```

x = np.dot(y,w)/np.sum(w)

# Loss function
for i in range(n):
    r[i] = np.linalg.norm(y[i] - x)
    w[i] = w_from_r(r[i], eps, mu)

mu = mu_update_factor * mu

print('\n x =\n {}'.format(x))
print('\n r =\n {}'.format(r))
print('\n w =\n {}'.format(w))

```

28.7 Graduated non-convexity for the Procrustes problem

In the Procrustes problem there are point correspondences $\mathbf{a}_i = \mathbf{R}\mathbf{b}_i$ for $i = 1, \dots, n$ where \mathbf{a}_i are model points, and \mathbf{b}_i are the corresponding measured data points. The cost function is

$$\min_{\mathbf{R}} L = \sum_{i=1}^n r(\mathbf{b}_i, \mathbf{R})^2 \quad (2036)$$

where the residuals are given by

$$r(\mathbf{b}_i, \mathbf{R}) = \|\mathbf{a}_i - \mathbf{R}\mathbf{b}_i\| \quad (2037)$$

The graduated non-convexity method is then given by

$$\min_{\mathbf{R}} \sum_{i=1}^n w_i r^2(\mathbf{b}_i, \mathbf{R}) \quad (2038)$$

where w_i is given by (2017). This is seen to be the weighted Procrustes problem.

The solution to the weighted Procrustes problem is calculated from –

$$\mathbf{H} = \sum_{i=1}^n w_i \mathbf{b}_i \mathbf{a}_i^T \quad (2039)$$

$$\mathbf{H} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T \quad (2040)$$

$$\mathbf{R} = \mathbf{V} \text{diag}[1, 1, \det(\mathbf{V}\mathbf{U}^T)] \mathbf{U}^T \quad (2041)$$

Example

```

import numpy as np

def skewm(r):
    return np.array([[0, -r[2], r[1]], [r[2], 0, -r[0]], [-r[1], r[0], 0]])
def expso3(u):
    S = skewm(u); un = np.linalg.norm(u)
    return np.eye(3) + np.sinc(un/np.pi)*S \

```

```

+ 0.5*(np.sinc(un/(2*np.pi)))**2 * S@S

def w_from_r(r, eps, mu):
    if abs(r) < eps * np.sqrt(mu/(mu+1)):
        w = 1
    elif abs(r) > eps * np.sqrt((mu+1)/mu):
        w = 0
    else:
        w = (eps * np.sqrt(mu*(mu+1)))/abs(r) - mu
    return w

# Preparations
np.set_printoptions(formatter={'float': '{: 0.4f}'.format})
ex = np.array([1.,0.,0.]); ey = np.array([0.,1.,0.]); ez = np.array([0.,0.,1.])

# Generation of input data
B = np.block([[ex], [ex], [ex], [ex], [ex], [ey], [ex]]).T
n = B.shape[1]
A = np.zeros((3, n))
Ra = np.stack((expso3(np.pi/6*ez),
               expso3((np.pi/6*ez - 0.01*ex)),
               expso3((np.pi/6*ez + 0.01*ex)),
               expso3((np.pi/6*ez - 0.1*ex)),
               expso3((np.pi/6*ez + 0.1*ex)),
               expso3((np.pi/6 + 0.5*np.pi/2)*ez),
               expso3((np.pi/6 + 0.5*np.pi/2)*ey)),
               axis = 0)
for i in range(n):
    A[:,i] = Ra[i]@B[:,i]

R0 = np.identity(3)
# Loss function intial data
r = np.zeros(n)
for i in range(n):
    r[i] = np.linalg.norm(A[:,i] - R0@B[:,i])
r_0_max = np.max(r)
print('\n r0 = {}\n r_0_max = {}'.format(r,r_0_max))

# GNC initialization
eps = 0.011; mu_update_factor = 1.4
max_iterations = 1000
w = np.ones(n)
mu = eps**2 / (2*r_0_max**2 - eps**2)
print('\n mu = {}'.format(mu))

for i in range(max_iterations):

```

```

# Weighted Procrustes
H = B @ np.diag(w) @ A.T
U, S, Vt = np.linalg.svd(H)
R = Vt.T @ np.diag([1,1,np.linalg.det(Vt.T @ U.T)]) @ U.T

# Loss function
for i in range(n):
    r[i] = np.linalg.norm(A[:,i] - R@B[:,i])
    w[i] = w_from_r(r[i], eps, mu)

mu = mu_update_factor * mu

print('R = {}'.format(R))
print('r = {}'.format(r))
print('w = {}'.format(w))

```

29 Latent space

29.1 Introduction

Reduction of dimension can be useful in learning. This is used to represent high-dimensional data $\mathbf{x} \in \mathbb{R}^D$ to a low-dimensional latent space with variables $\mathbf{z} \in \mathbb{R}^L$. This reduction is called an embedding \mathbf{z}_n of a data input \mathbf{x}_n . Then learning and data processing can be done in the latent space.

29.2 Autoencoder

A good reference for autoencoders is the book of Murphy [48]. An autoencoder is an encoding mapping from a data vector $\mathbf{x} \in \mathbb{R}^D$ to a latent variable $\mathbf{z} = \mathbf{f}_e(\mathbf{x}) \in \mathbb{R}^L$, and then a decoding mapping from the latent variable \mathbf{z} to a reconstructed data vector $\hat{\mathbf{x}} = \mathbf{f}_d(\mathbf{z}) \in \mathbb{R}^D$. The reconstructed vector is then given by

$$\hat{\mathbf{x}} = \mathbf{f}_d(\mathbf{f}_e(\mathbf{x})) \quad (2042)$$

The encoding function \mathbf{f}_e and the decoding function \mathbf{f}_d are then optimized to minimize a cost function L .

A bottleneck autoencoder is an autoencoder where the dimension of the latent space is smaller than the dimension of the data space, which means that $L < D$.

A linear autoencoder is given by the encoding $\mathbf{z} = \mathbf{W}_e \mathbf{x}$ and the decoder $\hat{\mathbf{x}} = \mathbf{W}_d \mathbf{z} = \mathbf{W}_d \mathbf{W}_e \mathbf{x}$. A linear autoencoder will be a bottleneck autoencoder, where the dimension of the latent space is less than the dimension of the data space.

29.3 Principal Component Analysis (PCA)

29.3.1 Introduction

Principal component analysis can be used to design a linear autoencoder. The idea is to identify the directions in the data space which has the most information, and then project the

data to these directions, which forms the latent space.

29.3.2 The empirical covariance matrix

Consider N samples of a data vector $\mathbf{x}_n \in \mathbb{R}^D$, $n = \{1, \dots, N\}$, and let the data be organized in a data matrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix} \in \mathbb{R}^{N \times D} \quad (2043)$$

It is assumed that the number of samples is larger than the dimension of the data vector, which means that $N > D$.

The empirical mean of the data vectors is

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \quad (2044)$$

and the centered data matrix is

$$\mathbf{X}_c = \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_N - \bar{\mathbf{x}})^T \end{bmatrix} \in \mathbb{R}^{N \times D} \quad (2045)$$

The empirical covariance matrix is then

$$\mathbf{S} = \frac{1}{N-1} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T = \frac{1}{N-1} \mathbf{X}_c^T \mathbf{X}_c \in \mathbb{R}^{D \times D} \quad (2046)$$

where division by $N-1$ is used instead of division by N to ensure that the empirical covariance matrix is unbiased. It is assumed that the input data is sufficiently varied in the sense that \mathbf{S} is full rank.

29.3.3 Singular value decomposition

The singular value decomposition (SVD) of the centered data matrix is given by

$$\mathbf{X}_c = \sum_{i=1}^D \sigma_i \mathbf{u}_i \mathbf{v}_i^T \in \mathbb{R}^{N \times D} \quad (2047)$$

where $\mathbf{u}_i \in \mathbb{R}^N$ is a set of orthogonal output unit vectors, and $\mathbf{v}_i \in \mathbb{R}^D$ is a set of orthogonal input unit vectors. It is assumed that $\text{rank}(\mathbf{X}_c) = D$, which means that $\sigma_1 \geq \dots \geq \sigma_D > 0$.

The empirical covariance matrix is then

$$\mathbf{S} = \frac{1}{N-1} \sum_{i=1}^D \sigma_i \mathbf{v}_i \mathbf{u}_i^T \sum_{j=1}^D \sigma_j \mathbf{u}_j \mathbf{v}_j^T = \frac{1}{N-1} \sum_{i=1}^D \sum_{j=1}^D \sigma_i \sigma_j \mathbf{v}_i \mathbf{v}_j^T \delta_{ij} \quad (2048)$$

which gives

$$\mathbf{S} = \frac{1}{N-1} \sum_{i=1}^D \sigma_i^2 \mathbf{v}_i \mathbf{v}_i^T \quad (2049)$$

This shows that the covariance in the direction of \mathbf{v}_i is σ_i^2 . It follows that the directions \mathbf{v}_i with the largest covariance are associated with the largest singular values σ_i . The L directions with the largest covariance are therefore given by $i = 1, \dots, L$. These directions are called the principal directions. The data matrix for the principal direction i is

$$\mathbf{X}_c \mathbf{v}_i = (\sigma_i \mathbf{u}_i \mathbf{v}_i^T) \mathbf{v}_i = \sigma_i \mathbf{u}_i \quad (2050)$$

which is the principal component in direction i .

The empirical covariance matrix \mathbf{S} can then be truncated to \mathbf{S}_L , which includes only the first L principal directions, by setting

$$\mathbf{S}_L = \frac{1}{N-1} \sum_{i=1}^L \sigma_i^2 \mathbf{v}_i \mathbf{v}_i^T \quad (2051)$$

This truncated matrix includes only the terms with the L largest singular values, and is the empirical covariance matrix of the truncated data matrix

$$\mathbf{X}_L = \sum_{i=1}^L \sigma_i \mathbf{u}_i \mathbf{v}_i^T \quad (2052)$$

where only the components of the L largest singular values are included.

29.3.4 Singular value decomposition in matrix form

The singular decomposition of the centered data matrix can be written in matrix form with the thin SVD as

$$\mathbf{X}_c = \mathbf{U}_D \mathbf{\Sigma}_D \mathbf{V}^T \in \mathbb{R}^{N \times D} \quad (2053)$$

Here

$$\mathbf{U}_D = (\mathbf{u}_1, \dots, \mathbf{u}_D) \in \mathbb{R}^{N \times D} \quad (2054)$$

$$\mathbf{\Sigma}_D = \text{diag}(\sigma_1, \dots, \sigma_D) \in \mathbb{R}^{D \times D} \quad (2055)$$

$$\mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_D) \in \mathbb{R}^{D \times D} \quad (2056)$$

The empirical covariance matrix is then

$$\mathbf{S} = \frac{1}{N-1} \mathbf{X}_c^T \mathbf{X}_c = \frac{1}{N-1} \mathbf{V} \mathbf{\Sigma}_D \mathbf{U}_D^T \mathbf{U}_D \mathbf{\Sigma}_D \mathbf{V}^T = \frac{1}{N-1} \mathbf{V} \mathbf{\Sigma}_D^2 \mathbf{V}^T \quad (2057)$$

The truncated empirical covariance matrix is then

$$\mathbf{S}_L = \frac{1}{N-1} \mathbf{V} \tilde{\mathbf{\Sigma}}_L^2 \mathbf{V}^T \quad (2058)$$

where

$$\tilde{\Sigma}_L = \begin{bmatrix} \Sigma_L & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{D \times D} \quad (2059)$$

and $\Sigma_L = \text{diag}(\sigma_1, \dots, \sigma_L) \in \mathbb{R}^{L \times L}$. This is the empirical covariance matrix of the truncated data matrix

$$\mathbf{X}_L = \mathbf{U}_D \tilde{\Sigma}_L \mathbf{V}^T = \mathbf{U}_D \Sigma_D \mathbf{W}^T \in \mathbb{R}^{N \times D} \quad (2060)$$

where

$$\mathbf{W} = [\mathbf{v}_1 \ \dots \ \mathbf{v}_L] \in \mathbb{R}^{D \times L} \quad (2061)$$

is a matrix of the L first columns of \mathbf{V} , which means that $\tilde{\Sigma}_L \mathbf{V}^T = \tilde{\Sigma}_W \mathbf{W}^T$. Moreover, the columns of \mathbf{W} are orthogonal so that $\mathbf{W}^T \mathbf{W} = \mathbf{I}$.

29.3.5 Latent space from principal components analysis

The matrix of principal components is defined in terms of the truncated data matrix \mathbf{X}_L as

$$\mathbf{Z} = \mathbf{X}_L \mathbf{W} \in \mathbb{R}^{N \times L} \quad (2062)$$

The data vectors are recovered with

$$\hat{\mathbf{X}} = \mathbf{Z} \mathbf{W}^T \in \mathbb{R}^{N \times D} \quad (2063)$$

The matrix of principal components can be written

$$\mathbf{Z} = \mathbf{U}_D \Sigma_D \mathbf{V}^T \mathbf{W} = \mathbf{U}_D \tilde{\Sigma}_D \begin{bmatrix} \mathbf{I}_L \\ \mathbf{0} \end{bmatrix} = \mathbf{U}_D \tilde{\Sigma}_W \quad (2064)$$

where

$$\tilde{\Sigma}_W = \begin{bmatrix} \Sigma_L \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{D \times L} \quad (2065)$$

This means that the matrix of principal components \mathbf{Z} has the empirical covariance matrix

$$\mathbf{S}_Z = \frac{1}{N-1} \tilde{\Sigma}_W^T \mathbf{U}_D^T \mathbf{U}_D \tilde{\Sigma}_W = \frac{1}{N-1} \tilde{\Sigma}_W^T \tilde{\Sigma}_W = \frac{1}{N-1} \Sigma_L^2 \in \mathbb{R}^{L \times L} \quad (2066)$$

which is a rotated version of the upper left $L \times L$ submatrix of \mathbf{S}_L .

The recovered data matrix is the truncated data matrix, which is seen from

$$\hat{\mathbf{X}} = \mathbf{Z} \mathbf{W}^T = \mathbf{U}_D \Sigma_D \mathbf{W}^T = \mathbf{X}_L \quad (2067)$$

The matrix of principal components is written in the form

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1^T \\ \vdots \\ \mathbf{z}_N^T \end{bmatrix} \in \mathbb{R}^{N \times L} \quad (2068)$$

which defines the vectors $\mathbf{z}_n \in \mathbb{R}^L$, which are reduced order representations of the data vectors. From $\mathbf{Z}^T = \mathbf{W}^T \mathbf{X}^T$ it is seen that the encoding from \mathbf{x}_n to \mathbf{z}_n is given by

$$\mathbf{z}_n = \mathbf{W}^T \mathbf{x}_n \in \mathbb{R}^L \quad (2069)$$

The variables \mathbf{z}_n are said to belong to a latent space of lower dimension. From $\hat{\mathbf{X}}^T = \mathbf{W} \mathbf{Z}^T$ it follows that the decoding from \mathbf{z}_n to $\hat{\mathbf{x}}_n$ is given by

$$\hat{\mathbf{x}}_n = \mathbf{W} \mathbf{z}_n = \mathbf{W} \mathbf{W}^T \mathbf{x}_n \in \mathbb{R}^D \quad (2070)$$

It is seen that

$$\mathbf{z}_n = \begin{bmatrix} \mathbf{v}_1^T \mathbf{x}_n \\ \vdots \\ \mathbf{v}_L^T \mathbf{x}_n \end{bmatrix} \quad (2071)$$

which are the components of \mathbf{x}_n along \mathbf{v}_i , while

$$\hat{\mathbf{x}}_n = \sum_{i=1}^L \mathbf{v}_i (\mathbf{v}_i^T \mathbf{x}_n) \quad (2072)$$

which is the projection of \mathbf{x}_n to the space spanned by $\mathbf{v}_1, \dots, \mathbf{v}_L$.

29.3.6 PCA in 3-dimensional space

Suppose that a set of points $\mathbf{x}_n \in \mathbb{R}^3$ are given for $n = 1, \dots, N$. Each point is given by

$$\mathbf{x}_n = \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} \quad (2073)$$

If all the points are in the xy plane, then the empirical covariance matrix \mathbf{S} will have no covariance in the z direction. The singular value decomposition will then be given by

$$\mathbf{S} = \frac{1}{N-1} \sum_{i=1}^3 \sigma_i \mathbf{v}_i \mathbf{v}_i^T \quad (2074)$$

where $\mathbf{v}_3 = [0, 0, 1]^T$ and $\sigma_3 = 0$. A truncated representation \mathbf{z}_n of dimension 2 will then be given by

$$\mathbf{z}_n = \begin{bmatrix} \mathbf{v}_1^T \mathbf{x}_n \\ \mathbf{v}_2^T \mathbf{x}_n \end{bmatrix} \quad (2075)$$

where \mathbf{v}_1 and \mathbf{v}_2 are vectors in the xy plane, and the recovered data vector is

$$\hat{\mathbf{x}}_n = \mathbf{v}_1 (\mathbf{v}_1^T \mathbf{x}_n) + \mathbf{v}_2 (\mathbf{v}_2^T \mathbf{x}_n) \quad (2076)$$

A Numerical optimization

Numerical optimization is used in many applications to minimize an objective function. The techniques are based on iterative improvement of the solution to a minimization problem by computing increments in the solution based on gradient information. A basic reference on methods for numerical optimization is [50], while the least squares problem is described in [69]. In robot kinematics numerical optimization methods are used in iterative inverse kinematics, where a solution to the inverse kinematics is computed using the Jacobian in an incremental solution technique instead of the direct computation used in analytic inverse kinematics. Numerical optimization is also used in robot vision, where it is one of the main tools for finding solutions. In this section least squares techniques for numerical optimization are presented. First the linear least squares is presented, where the solution is calculated directly from the normal equations. Then it is explained how linear least squares can be extended to nonlinear problems, where iterative methods must be used. This includes gradient search methods, also known as hill-climbing or steepest descent, and Newton methods where both the gradient and the second order derivative of the objective function are used.

A.1 Linear least-squares in the underdetermined case

Consider a system of m linear equations in n unknowns given by

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad (2077)$$

Here \mathbf{x} is a vector of n unknowns, \mathbf{A} is an $m \times n$ matrix where $m < n$, and \mathbf{b} is a vector of dimension m . It is assumed that there are $m < n$ equations, which means that the system of equations is underdetermined, as there are less equations than unknown variables.

Suppose that $\text{rank} \mathbf{A} = m$. Then $\mathbf{A}\mathbf{A}^T$ is an $m \times m$ matrix with full rank. It follows that

$$\mathbf{x}^* = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} \quad (2078)$$

is a solution as $\mathbf{A}\mathbf{x}^* = \mathbf{A}\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} = \mathbf{b}$. Moreover, this solution is the least-squares solution that minimizes $(\mathbf{A}\mathbf{x} - \mathbf{b})^T(\mathbf{A}\mathbf{x} - \mathbf{b})$. This can be verified by solving the minimization problem of minimizing $\mathbf{x}^T\mathbf{x}$ with the constraint that $\mathbf{A}\mathbf{x} = \mathbf{b}$. This is done with a Lagrangian multiplier $\boldsymbol{\lambda}$ in the minimization problem

$$\frac{1}{2}\mathbf{x}^T\mathbf{x} + \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \quad (2079)$$

The first order conditions are found by differentiation with respect to \mathbf{x} and $\boldsymbol{\lambda}$ to be

$$\mathbf{x} + \mathbf{A}^T\boldsymbol{\lambda} = \mathbf{0} \quad (2080)$$

$$\mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0} \quad (2081)$$

Substitution of \mathbf{x} from the first equation into the second equation gives

$$\mathbf{A}\mathbf{A}^T\boldsymbol{\lambda} = \mathbf{b} \quad \Rightarrow \quad \boldsymbol{\lambda} = (\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} \quad (2082)$$

where it is used that $\mathbf{A}\mathbf{A}^T$ will have full rank. This gives the least-squares solution (2078).

The matrix

$$\mathbf{A}^+ = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1} \quad (2083)$$

is called the Moore-Penrose pseudo inverse. It is noted that the general solution of (2077) is

$$\mathbf{x} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} + (\mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A})\mathbf{y} \quad (2084)$$

where \mathbf{y} is arbitrary. This is verified by the direct calculation

$$\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} + (\mathbf{A} - \mathbf{A}\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A})\mathbf{y} = \mathbf{b} \quad (2085)$$

Moreover, the two terms of (2084) are orthogonal, which is seen from

$$\mathbf{b}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}(\mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A})\mathbf{y} = 0 \quad (2086)$$

It follows that the least-squares solution is obtained with $\mathbf{y} = \mathbf{0}$, which shown that (2078) is the optimal solution.

A.2 Linear least-squares in the overdetermined case

Consider a system of m linear equations in n unknowns given by

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad (2087)$$

Here \mathbf{x} is a vector of n unknowns, \mathbf{A} is an $m \times n$ matrix where $m > n$, and \mathbf{b} is a vector of dimension m . It is assumed that there are $m > n$ equations, which means that the system of equations is overdetermined, as there are more equations than unknown variables. This type of formulation results when there are m observations that are used to find n unknown variables. In general there will be no solution \mathbf{x} that satisfies all the m equations due to inaccuracies and noise in the measurements.

In applications this problem is usually solved by finding the least-squares solution \mathbf{x}^* to the problem. This can be done by minimizing the quadratic objective function

$$f(\mathbf{x}) = \frac{1}{2}(\mathbf{A}\mathbf{x} - \mathbf{b})^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \quad (2088)$$

with respect to \mathbf{x} . The gradient of the cost function $f(\mathbf{x})$ with respect to \mathbf{x} is found from

$$f(\mathbf{x}) = \frac{1}{2}(\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - 2\mathbf{x}^T\mathbf{A}^T\mathbf{b} + \mathbf{b}^T\mathbf{b}) \quad (2089)$$

to be

$$\nabla f(\mathbf{x}) = \mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{A}^T\mathbf{b} \quad (2090)$$

where

$$\nabla = \left[\frac{\partial}{\partial x_1}, \dots, \frac{\partial}{\partial x_n} \right]^T$$

is a column vector of differentiation operators. The gradient is the column vector

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f_L(\mathbf{x})}{\partial x_1}, \dots, \frac{\partial f_L(\mathbf{x})}{\partial x_n} \right]^T \quad (2091)$$

The gradient must be zero at the optimal solution \mathbf{x}^* , and it follows from (2090) that the optimal solution can be found from

$$\mathbf{A}^T\mathbf{A}\mathbf{x}^* = \mathbf{A}^T\mathbf{b} \quad (2092)$$

The equation (2092) is the normal equation of the problem. The matrix $\mathbf{A}^T \mathbf{A}$ appearing in the normal equation is of dimension $n \times n$, and will have full rank whenever \mathbf{A} is full rank, which is the case if there are at least n linearly independent equations in (2088). In terms of matrix rank this means that $\text{rank}(\mathbf{A}^T \mathbf{A}) = n$ whenever $\text{rank}(\mathbf{A}) = n$.

The solution can be written

$$\mathbf{x}^* = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} \quad (2093)$$

which shows that $\mathbf{A}^T \mathbf{A}$ must be full rank for the solution to be well-defined. In practice, the solution is found from the normal equation using some numerical method like Cholesky decomposition [27] without having to invert $\mathbf{A}^T \mathbf{A}$.

A.3 Weighted least-squares

Consider the weighted least-squares problem with quadratic objective function

$$f(\mathbf{x}) = \frac{1}{2} (\mathbf{A}\mathbf{x} - \mathbf{b})^T \mathbf{Q} (\mathbf{A}\mathbf{x} - \mathbf{b}) \quad (2094)$$

where the weighting matrix \mathbf{Q} can be factored as $\mathbf{Q} = \mathbf{W}^T \mathbf{W}$ where \mathbf{W} is square and invertible. The optimal solution is the least squares solution of $\mathbf{W}\mathbf{A}\mathbf{x} = \mathbf{W}\mathbf{b}$, which has normal equations

$$\mathbf{A}^T \mathbf{Q} \mathbf{A} \mathbf{x} = \mathbf{A}^T \mathbf{Q} \mathbf{b} \quad (2095)$$

The solution can be written

$$\mathbf{x}^* = (\mathbf{A}^T \mathbf{Q} \mathbf{A})^{-1} \mathbf{A}^T \mathbf{Q} \mathbf{b} \quad (2096)$$

Example

Consider the cost function

$$f(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^n \frac{1}{\sigma_i^2} r_i(\mathbf{x})^2 \quad (2097)$$

where r_i are the components of the residual vector $\mathbf{r} = [r_1, \dots, r_N]^T$ which is given by

$$\mathbf{r} = \mathbf{A}\mathbf{x} - \mathbf{b} \quad (2098)$$

Then the cost function can be written as the weighted least-squares problem

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{r}^T \mathbf{Q} \mathbf{r} = \frac{1}{2} (\mathbf{A}\mathbf{x} - \mathbf{b})^T \mathbf{Q} (\mathbf{A}\mathbf{x} - \mathbf{b}) \quad (2099)$$

where

$$\mathbf{Q} = \text{diag}(\sigma_1^2, \dots, \sigma_N^2) \quad (2100)$$

A.4 Damped linear least-squares

If the matrix $\mathbf{A}^T \mathbf{A}$ is rank deficient, there is no solution to the normal equation (2092). Then an approximate solution can be found using the damped least squares solution. The damped least squares solution is found by minimizing the damped objective function

$$f_d = \frac{1}{2} (\mathbf{A}\mathbf{x} - \mathbf{b})^T (\mathbf{A}\mathbf{x} - \mathbf{b}) + \frac{1}{2} \lambda^2 \mathbf{x}^T \mathbf{x} \quad (2101)$$

which is a modification of the original objective function, where a quadratic damping term has been added. Obviously, the solution to this modified problem will minimize the damped objective function instead of the original objective function. The motivation for using this technique is that it may be better to have a solution to a modified objective function than to have no solution. The optimal solution of the linear system is denoted \mathbf{x}_λ^* , where it is indicated that the solution will depend on the damping factor λ .

The normal equations for the damped least-squares problem is seen to be

$$(\mathbf{A}^T \mathbf{A} + \lambda^2 \mathbf{I}) \mathbf{x}_\lambda^* = \mathbf{A}^T \mathbf{b} \quad (2102)$$

The advantage of this solution is that it can be computed even when \mathbf{A} is not full rank. Moreover, the damped-least squares solution is well-conditioned when \mathbf{A} is close to being rank deficient, which means that $\mathbf{A}^T \mathbf{A}$ is close to being singular, whereas the solution \mathbf{x}^* without damping can tend to infinity when \mathbf{A} is close to being rank deficient.

To investigate the solution of the damped least-squares solution further the singular value decomposition is used. The singular value decomposition of \mathbf{A} is

$$\mathbf{A} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T = \sum_{i=1}^n \sigma_i \mathbf{u}_i \mathbf{v}_i^T \quad (2103)$$

where $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$, where r is the rank of \mathbf{A} , and $\sigma_{r+1} = \dots = \sigma_n = 0$ if $r < n$.

Then

$$\mathbf{A}^T \mathbf{A} + \lambda^2 \mathbf{I} = \mathbf{V} \boldsymbol{\Sigma} \mathbf{U}^T \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T + \lambda^2 \mathbf{I} = \mathbf{V} \boldsymbol{\Sigma}^2 \mathbf{V}^T + \lambda^2 \mathbf{I} = \mathbf{V} (\boldsymbol{\Sigma}^2 + \lambda^2 \mathbf{I}) \mathbf{V}^T \quad (2104)$$

where it is used that $\mathbf{U} \mathbf{U}^T = \mathbf{I}$ and $\mathbf{V} \mathbf{V}^T = \mathbf{I}$. This can also be written

$$\mathbf{A}^T \mathbf{A} + \lambda^2 \mathbf{I} = \sum_{i=1}^n (\sigma_i^2 + \lambda^2) \mathbf{v}_i \mathbf{v}_i^T \quad (2105)$$

The inverse is

$$(\mathbf{A}^T \mathbf{A} + \lambda^2 \mathbf{I})^{-1} = \mathbf{V} (\boldsymbol{\Sigma}^2 + \lambda^2 \mathbf{I})^{-1} \mathbf{V}^T = \sum_{i=1}^n \frac{1}{\sigma_i^2 + \lambda^2} \mathbf{v}_i \mathbf{v}_i^T \quad (2106)$$

The solution to the normal equation is then

$$\mathbf{x}_\lambda^* = (\mathbf{A}^T \mathbf{A} + \lambda^2 \mathbf{I})^{-1} \mathbf{A}^T \mathbf{b} = \mathbf{V} (\boldsymbol{\Sigma}^2 + \lambda^2 \mathbf{I})^{-1} \boldsymbol{\Sigma} \mathbf{U}^T \mathbf{b} \quad (2107)$$

which can be written

$$\mathbf{x}_\lambda^* = \sum_{i=1}^n \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \mathbf{v}_i \mathbf{u}_i^T \mathbf{b} \quad (2108)$$

It is noted that in the original problem without damping the optimal solution is

$$\mathbf{x}^* = \sum_{i=1}^n \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T \mathbf{b} \quad (2109)$$

Here $1/\sigma_n$ will tend to infinity when σ_n tends to zero. The solution is undefined when $\sigma_n = 0$.

In the damped solution \mathbf{x}_λ^* the potentially problematic factor $1/\sigma_n$ is replaced with the factor $g(\sigma_n) = \sigma_n/(\sigma_n^2 + \lambda^2)$. This factor will be well-behaved even when σ_n tends to zero. When $\sigma_n \gg \lambda$ the solution $\mathbf{x}_\lambda^* \approx \mathbf{x}^*$ since

$$\frac{\sigma_n}{\sigma_n^2 + \lambda^2} \approx \frac{1}{\sigma_n}, \quad \text{when } \sigma_n \gg \lambda \quad (2110)$$

Moreover,

$$\frac{\sigma_n}{\sigma_n^2 + \lambda^2} \approx \frac{\sigma_n}{\lambda^2}, \quad \text{when } \sigma_n \ll \lambda \quad (2111)$$

The maximum value for $g(\sigma_n)$ is found from

$$\frac{dg(\sigma_n)}{d\sigma_n} = \frac{\lambda^2 - \sigma_n^2}{(\sigma_n^2 + \lambda^2)^2} = 0 \quad (2112)$$

which gives the maximum value $g(\lambda) = 1/(2\lambda)$ at $\sigma_n = \lambda$.

A.5 Formulation of the nonlinear least squares problem

In a nonlinear least-squares problem the objective function is nonlinear and is given in the form

$$f(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^N r_i(\mathbf{x})^2 \quad (2113)$$

This objective function is a nonlinear function of the vector $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$. The N functions $r_i(\mathbf{x})$ are called the residuals of the objective function. The residuals can be arranged in a residual vector

$$\mathbf{r}(\mathbf{x}) = [r_1(\mathbf{x}), r_2(\mathbf{x}), \dots, r_N(\mathbf{x})]^T \quad (2114)$$

which is a column vector of dimension N , and the objective function can be written

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{r}(\mathbf{x})^T \mathbf{r}(\mathbf{x}) \quad (2115)$$

A nonlinear least-squares solution will in general not have a closed-form solution. Instead the solution must be found by iteration using a numerical optimization method. This is done in an iterative scheme where the solution at step $k + 1$ is found from the solution at step k by the computation

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k \quad (2116)$$

where \mathbf{p}_k is the increment of the solution. This increment is typically calculated from the first and second order derivatives of the objective function $f(\mathbf{x})$.

The gradient of the objective function can be written in either of the two forms

$$\nabla f(\mathbf{x}) = \sum_{i=1}^N r_i \nabla r_i \quad (2117)$$

where the gradient ∇r_i of a residual r_i is

$$\nabla r_i(\mathbf{x}) = \left[\frac{\partial r_i}{\partial x_1}, \dots, \frac{\partial r_i}{\partial x_n} \right]^T$$

The notation

$$\nabla^2 = \nabla \nabla^T = \left\{ \frac{\partial^2}{\partial x_i \partial x_j} \right\}$$

is used in the second order derivative, which is the Hessian

$$\nabla^2 f(\mathbf{x}) = \nabla \nabla^T f(\mathbf{x}) = \left\{ \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} \right\} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix} \quad (2118)$$

The Hessian can be given in terms of the residuals and the gradients of the residuals by insertion of (2117), which gives

$$\nabla^2 f(\mathbf{x}) = \nabla \nabla^T f(\mathbf{x}) = \sum_{i=1}^N \nabla(r_i \nabla^T r_i) = \sum_{i=1}^N \nabla r_i \nabla^T r_i + \sum_{i=1}^N r_i \nabla^2 r_i$$

The terms $\nabla^2 r_i$ will normally be somewhat complicated to compute. It is noted that these terms appear in multiplication with r_i , which will be close to zero when \mathbf{x} approaches the optimal solution. Therefore, the Hessian will often be approximated by

$$\nabla^2 f(\mathbf{x}) \approx \sum_{i=1}^N \nabla r_i \nabla^T r_i \quad (2119)$$

This is easy to compute, which will be seen in the following.

The formulation can be made more efficient by introducing the Jacobian of the problem, which is the matrix of first-order derivatives of the residuals given by

$$\mathbf{J}(\mathbf{x}) = \left\{ \frac{\partial r_i}{\partial x_j} \right\} = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_N}{\partial x_1} & \cdots & \frac{\partial r_N}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla^T r_1 \\ \vdots \\ \nabla^T r_N \end{bmatrix} \quad (2120)$$

The gradient of the objective function can be expressed in terms of the Jacobian as

$$\nabla f(\mathbf{x}) = \mathbf{J}(\mathbf{x})^T \mathbf{r}(\mathbf{x}) \quad (2121)$$

Moreover, the Hessian can be written in terms of the Jacobian as

$$\nabla^2 f(\mathbf{x}) \approx \sum_{i=1}^N \nabla r_i \nabla^T r_i = \mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}) \quad (2122)$$

where the second order terms $\nabla^2 r_i$ has been left out.

A.6 Gradient search

The gradient search method for minimization of a function $f(\mathbf{x})$ is also called steepest descent. In this method the increment \mathbf{p}_k is in the negative direction of the gradient $\nabla f(\mathbf{x})$. The intuitive idea is that if the increment is in the direction where the decrease in the function is largest, then the solution will progress towards the minimum. The method is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}) \quad (2123)$$

where α is the step length. The step length α_k can be set to some constant value, or it can be found at each step by a line search along the negative direction of the gradient. In terms for the residual the method is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{J}(\mathbf{x}_k)^T \mathbf{r}(\mathbf{x}_k) \quad (2124)$$

A.7 The Gauss-Newton method for nonlinear least squares

The nonlinear least-squares optimization problem is solved through the iteration $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k$ where the solution \mathbf{x}_k at iteration k is updated to the solution \mathbf{x}_{k+1} at iteration $k+1$ by solving a linear least squares problem. This is done by defining a quadratic approximation of the objective function in the form (2088) based on the Taylor series expansion of the objective function about the current solution \mathbf{x}_k . This Taylor series expansion is given by

$$f(\mathbf{x}_k + \mathbf{p}) = f(\mathbf{x}_k) + \nabla^T f(\mathbf{x}_k) \mathbf{p} + \frac{1}{2} \mathbf{p}^T \nabla^2 f(\mathbf{x}_k) \mathbf{p} + \dots \quad (2125)$$

$$= f(\mathbf{x}_k) + \mathbf{r}_k^T \mathbf{J}_k \mathbf{p} + \frac{1}{2} \mathbf{p}^T \mathbf{J}_k^T \mathbf{J}_k \mathbf{p} + \dots \quad (2126)$$

where the free variable is \mathbf{p} , which is the increment from \mathbf{x}_k , and the notation $\mathbf{J}_k = \mathbf{J}(\mathbf{x}_k)$ and $\mathbf{r}_k = \mathbf{r}(\mathbf{x}_k)$ is used.

Typically, the quadratic approximation of the objective function at step k will be selected as the two first terms of the Taylor series, which gives the local objective function

$$m_k(\mathbf{p}) = \frac{1}{2} \mathbf{r}_k^T \mathbf{r}_k + \mathbf{r}^T \mathbf{J}_k \mathbf{p} + \frac{1}{2} \mathbf{p}^T \mathbf{J}_k^T \mathbf{J}_k \mathbf{p} \quad (2127)$$

to be minimized at step $k+1$. The local objective function can be written

$$m_k(\mathbf{p}) = \frac{1}{2} (\mathbf{J}_k \mathbf{p} + \mathbf{r}_k)^T (\mathbf{J}_k \mathbf{p} + \mathbf{r}_k) \quad (2128)$$

which is in the same form as the objective function (2088) of the linear least squares problem. Given the current residual \mathbf{r}_k and the current Jacobian \mathbf{J}_k , this can be optimized with respect to \mathbf{p} as in the linear least-squares problem. The optimal solution for \mathbf{p} is found by setting the gradient of $m_k(\mathbf{p})$ to zero, which gives

$$\nabla m_k(\mathbf{p}) = \mathbf{J}_k^T \mathbf{r}_k + \mathbf{J}_k^T \mathbf{J}_k \mathbf{p} = \mathbf{0} \quad (2129)$$

The optimal solution \mathbf{p}_k that minimizes the local objective function (2128) is then found from the normal equation

$$\mathbf{J}_k^T \mathbf{J}_k \mathbf{p}_k = -\mathbf{J}_k^T \mathbf{r}_k \quad (2130)$$

which can be written

$$\mathbf{p}_k = -(\mathbf{J}_k^T \mathbf{J}_k)^{-1} \mathbf{J}_k^T \mathbf{r}_k \quad (2131)$$

The solution \mathbf{x}_{k+1} at iteration $k + 1$ is then calculated from the update equation

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\mathbf{J}_k^T \mathbf{J}_k)^{-1} \mathbf{J}_k^T \mathbf{r}_k \quad (2132)$$

It is noted that this update gives the Taylor series expansion

$$f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k) - \frac{1}{2} \mathbf{p}_k^T \mathbf{J}_k^T \mathbf{J}_k \mathbf{p}_k + \dots \quad (2133)$$

which shows that the objective function decreases whenever the Hessian approximation $\mathbf{J}_k^T \mathbf{J}_k$ is positive definite.

The optimization is then repeated at \mathbf{x}_{k+1} as a linear least-squares problem with local objective function $m_{k+1}(\mathbf{p}) = \frac{1}{2}(\mathbf{J}_{k+1} \mathbf{p} - \mathbf{r}_{k+1})^T (\mathbf{J}_{k+1} \mathbf{p} - \mathbf{r}_{k+1})$ to calculate \mathbf{x}_{k+2} , and so on. This iteration is continued until the solution has converged according to some convergence criteria like

$$\frac{\|\mathbf{p}_{k+1} - \mathbf{p}_k\|}{\|\mathbf{p}_k\|} < 0.001 \quad (2134)$$

or

$$\frac{\|\mathbf{f}(\mathbf{x}_{k+1}) - \mathbf{f}(\mathbf{x}_k)\|}{\|\mathbf{f}(\mathbf{x}_k)\|} < 0.0001 \quad (2135)$$

A.8 Gauss-Newton method for a quadratic objective function

To investigate the properties of nonlinear least squares methods a quadratic objective function is considered. The objective function is written

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x} \quad (2136)$$

where $\mathbf{Q} = \mathbf{Q}^T$ is symmetric and positive definite, which means that $\mathbf{x}^T \mathbf{Q} \mathbf{x} > 0$ for all $\mathbf{x} \neq 0$. The matrix \mathbf{Q} can then be factored, e.g., using the Cholesky decomposition, as $\mathbf{Q} = \mathbf{C}^T \mathbf{C}$, where \mathbf{C} is a upper triangular matrix. The objective function can then be written in terms of residual vectors as

$$f(\mathbf{x}) = \mathbf{r}^T \mathbf{r}, \quad \mathbf{r} = \mathbf{C} \mathbf{x} \quad (2137)$$

The Jacobian is $\mathbf{J} = \mathbf{C}$, and the Hessian can be approximated by $\nabla^2 f(\mathbf{x}) \approx \mathbf{J}^T \mathbf{J} = \mathbf{C}^T \mathbf{C} = \mathbf{Q}$. The gradient of the objective function will be

$$\nabla f(\mathbf{x}) = \mathbf{C}^T \mathbf{r}(\mathbf{x}) = \mathbf{C}^T \mathbf{C} \mathbf{x}_k = \mathbf{Q} \mathbf{x} \quad (2138)$$

The normal equations at step k is

$$\mathbf{Q} \mathbf{p}_k = -\mathbf{Q} \mathbf{x}_k \quad (2139)$$

The optimal step as used in the Gauss-Newton method is then seen to be

$$\mathbf{p}_k = -\mathbf{x}_k \quad (2140)$$

which gives $\mathbf{x}_{k+1} = \mathbf{0}$. This shows that the Gauss-Newton method converges in one step if the objective function is quadratic.

A.9 Gradient method for a quadratic objective function

If a gradient descent method is used the step will be

$$\mathbf{p}_k = -\alpha_k \nabla f(\mathbf{x}) = -\alpha_k \mathbf{Q} \mathbf{x}_k \quad (2141)$$

where the step length α_k is determined with a line search. If $\mathbf{Q} = \mathbf{I}$, then the step is in the direction of the solution, and the step direction of the gradient method will be the same as for the Gauss-Newton method.

The step length is found by minimizing $f(\mathbf{x}_{k+1})$ with respect to the step length α_k . From

$$f(\mathbf{x}_{k+1}) = \mathbf{x}_{k+1}^T \mathbf{Q} \mathbf{x}_{k+1} \quad (2142)$$

$$= (\mathbf{x}_k - \alpha_k \nabla f_k)^T \mathbf{Q} (\mathbf{x}_k - \alpha_k \nabla f_k) \quad (2143)$$

$$= \mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k - 2\alpha_k \nabla f_k^T \nabla f_k + \alpha_k^2 \nabla f_k^T \mathbf{Q} \nabla f_k \quad (2144)$$

it is seen that the minimum can be found from

$$\frac{\partial f(\mathbf{x}_{k+1})}{\partial \alpha_k} = -2 \nabla f_k^T \nabla f_k + 2\alpha_k \nabla f_k^T \mathbf{Q} \nabla f_k = 0 \quad (2145)$$

which gives the optimum step length

$$\alpha_k = \frac{\nabla f_k^T \nabla f_k}{\nabla f_k^T \mathbf{Q} \nabla f_k} \quad (2146)$$

To investigate the convergence speed of the gradient method for a quadratic objective function, it is noted that the change in the objective function from iteration k to $k+1$ is given by

$$\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k - \mathbf{x}_{k+1}^T \mathbf{Q} \mathbf{x}_{k+1} = 2\alpha_k \nabla f_k^T \nabla f_k - \alpha_k^2 \nabla f_k^T \mathbf{Q} \nabla f_k \quad (2147)$$

$$= \frac{(\nabla f_k^T \nabla f_k)^2}{\nabla f_k^T \mathbf{Q} \nabla f_k} \quad (2148)$$

where the optimal value of α_k has been inserted from (2146). This result can be combined with the expression $\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k = \nabla f_k^T \mathbf{Q}^{-1} \nabla f_k$, and it is seen that the relative change in the objective function from iteration k to $k+1$ is

$$\frac{\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k - \mathbf{x}_{k+1}^T \mathbf{Q} \mathbf{x}_{k+1}}{\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k} = \frac{(\nabla f_k^T \nabla f_k)^2}{[\nabla f_k^T \mathbf{Q} \nabla f_k][\nabla f_k^T \mathbf{Q}^{-1} \nabla f_k]} \quad (2149)$$

From linear algebra, it is known that

$$\nabla f_k^T \mathbf{Q} \nabla f_k \leq \sigma_1 \nabla f_k^T \nabla f_k \quad (2150)$$

and

$$\nabla f_k^T \mathbf{Q}^{-1} \nabla f_k \leq \frac{1}{\sigma_n} \nabla f_k^T \nabla f_k \quad (2151)$$

where σ_1 is the largest singular value of \mathbf{Q} , and σ_n is the smallest singular value of \mathbf{Q} , which implies that $1/\sigma_n$ is the largest singular value of \mathbf{Q}^{-1} .

It follows that

$$\frac{\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k - \mathbf{x}_{k+1}^T \mathbf{Q} \mathbf{x}_{k+1}}{\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k} \geq \frac{\sigma_n}{\sigma_1} \quad (2152)$$

This leads to the result [38]

$$\frac{\mathbf{x}_{k+1}^T \mathbf{Q} \mathbf{x}_{k+1}}{\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k} \leq 1 - \frac{\sigma_n}{\sigma_1} = \frac{\sigma_1 - \sigma_n}{\sigma_1} \quad (2153)$$

It is seen that if the singular values are equal so that $\sigma_1 = \sigma_n$ then the gradient method will converge in one iteration. In the case that $n = 2$, this means that the level curves of the objective function given by $f(\mathbf{x}) = c$ are circles.

If the matrix \mathbf{Q} is ill conditioned, so that $\sigma_n \ll \sigma_1$, then the gradient method will converge slowly. If $\sigma_n = \sigma_i/1000$, then the relative decrease in the objective function from iteration k to $k + 1$ is only $\sigma_n/\sigma_i = 0.001$. In the case that $n = 2$ this means that the level curves $f(\mathbf{x}) = c$ are highly elliptic.

Example

Consider the function

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} \quad (2154)$$

where $\mathbf{x} = [x_1, x_2]^T$ and

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 16 \end{bmatrix} \quad (2155)$$

The gradient is

$$\nabla f(\mathbf{x}) = \mathbf{Q} \mathbf{x} = \begin{bmatrix} x_1 \\ 16x_2 \end{bmatrix} \quad (2156)$$

A gradient search will then be slow since the gradient does not point at the minimum in the first steps of the optimization. Moreover, the step length must be small. The resulting convergence is shown in Figure 89. A Gauss-Newton search will converge in one step, as seen in Figure 90.

```
import numpy as np
import matplotlib.pyplot as plt

# Initial condition
Q = np.array([[1, 0], [0, 16]])
x = np.array([1, 1])
y = np.array([2, 1])
xx = x; yy = y
N = 100
alpha = 0.01
for i in range(1, N+1):
    x = x - alpha * Q @ x
    y = y - alpha * Q @ y
    xx = np.block([[xx], [x]])
    yy = np.block([[yy], [y]])
```

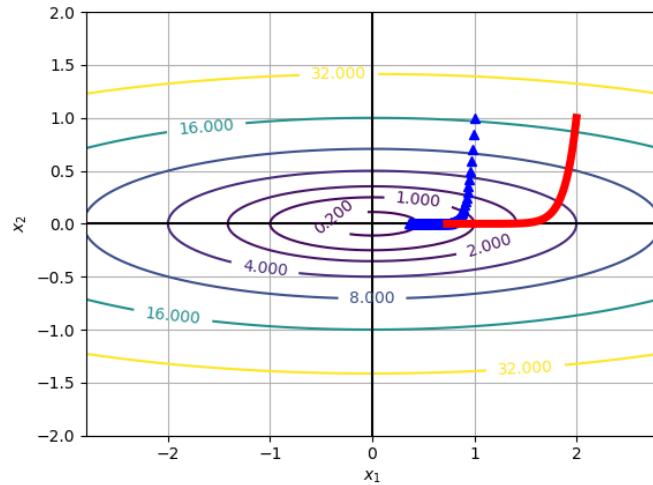


Figure 89: Gauss-Newton minimization of the quadratic function. The curves tends towards the valley of the minimum and the progresses slowly towards the minimum at the origin along the valley.

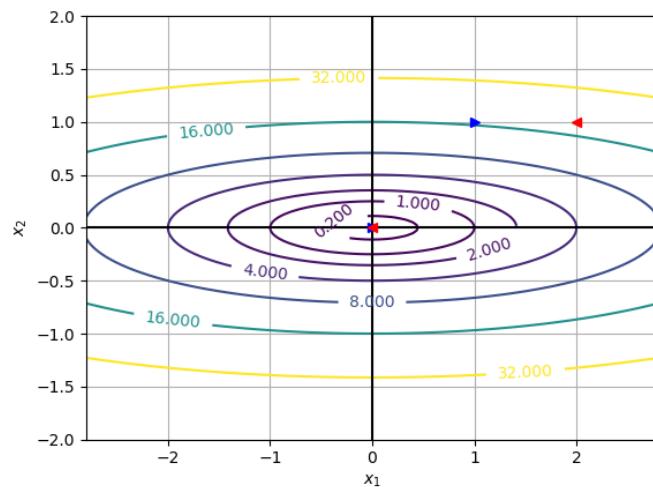


Figure 90: Gauss-Newton minimization of the quadratic function. The curves reaches the minimum at the origin in one step.

```

# Plotting of gradient search
# Genereate data for contour plots
delta = 0.025
xm = np.arange(-3.2, 3.2, delta)
ym = np.arange(-3.2, 3.2, delta)
X, Y = np.meshgrid(xm, ym)
n = X.shape[0]
Z = np.zeros((n,n))
for i in range(0, n):
    for j in range(0, n):
        Z[i,j] = (Q[0,0]*X[i,j]**2 + Q[0,1]*X[i,j]*Y[i,j]
                   + Q[1,0]*X[i,j]*Y[i,j] + Q[1,1]*Y[i,j]**2)
# Plot
x_axis = np.block([[[-1.4*2, 1.4*2], [0, 0]]])
y_axis = np.block([[0, 0], [-2, 2]])
plt.figure(1)
plt.figure(1).clear()
CS = plt.contour(X, Y, Z, levels = [0.2, 1, 2, 4, 8, 16, 32])
plt.clabel(CS, inline=1, fontsize=10)
plt.plot(x_axis[0,:], x_axis[1,:], 'k', y_axis[0,:], y_axis[1,:], 'k')
plt.plot(xx[:,0], xx[:,1], 'b^', yy[:,0], yy[:,1], 'r', linewidth=5)
plt.axis([-1.4*2, 1.4*2, -2, 2])
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.grid()

# Gauss-Newton search
x = np.array([1,1]); y = np.array([2,1])
xx = x; yy = y
N = 1
for i in range(1,N+1):
    x = x - x
    y = y - y
    xx = np.block([[xx], [x]])
    yy = np.block([[yy], [y]])
# Plotting of Gauss-Newton search
plt.figure(2)
plt.figure(2).clear()
CS = plt.contour(X, Y, Z, levels = [0.2, 1, 2, 4, 8, 16, 32])
plt.clabel(CS, inline=1, fontsize=10)
plt.plot(x_axis[0,:], x_axis[1,:], 'k', y_axis[0,:], y_axis[1,:], 'k')
plt.plot(xx[:,0], xx[:,1], 'b>', yy[:,0], yy[:,1], 'r<', linewidth=5)
plt.axis([-1.4*2, 1.4*2, -2, 2])
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.grid()

```

A.10 The Levenberg-Marquardt method for nonlinear least squares

There are many methods that are based on modifications of the basic Gauss-Newton method. One modification is to use damped least squares at each iteration, where the local objective function is changed to

$$m_k(\mathbf{p}) = \frac{1}{2}(\mathbf{J}_k \mathbf{p} + \mathbf{r}_k)^T(\mathbf{J}_k \mathbf{p} + \mathbf{r}_k) + \frac{1}{2}\lambda^2 \mathbf{p}^T \mathbf{p} \quad (2157)$$

This provides a certain robustness in problems where the Jacobian may become ill-conditioned, as the normal equation will be

$$(\mathbf{J}_k^T \mathbf{J}_k + \lambda^2 \mathbf{I}) \mathbf{p}_k = -\mathbf{J}_k^T \mathbf{r}_k \quad (2158)$$

It is noted that this reduces to the Gauss-Newton method when $\lambda = 0$. When λ is large, the normal equation tends to $\lambda^2 \mathbf{p}_k = \mathbf{J}_k^T \mathbf{r}_k = \nabla f_k$, which means that the step will be in the gradient direction.

A.11 The Rosenbrock function

A benchmark in optimization is the Rosenbrock function. The minimum is in a long valley which is curved and almost flat. Moreover, the function is non-convex. The function illustrates some of the problems with a pure gradient method. The objective function is

$$f(\mathbf{x}) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 \quad (2159)$$

The gradient is found to be

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 400x_1^3 - 400x_1x_2 + 2x_1 - 2 \\ 200(-x_1^2 + x_2) \end{bmatrix} \quad (2160)$$

It is noted that the Hessian matrix is

$$\nabla^2 f(\mathbf{x}) = \left\{ \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} \right\} = \begin{bmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix} \quad (2161)$$

A straightforward gradient search

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k) \quad (2162)$$

gives the result in Figure 91. The step length is $\alpha = 0.0001$ and there are 10000 iterations.

In terms of residuals the objective function can be written

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{r}^T \mathbf{r} \quad (2163)$$

where the residual vector is

$$\mathbf{r} = \sqrt{2} \begin{bmatrix} 10(x_1^2 - x_2) \\ x_1 - 1 \end{bmatrix} \quad (2164)$$

It follows that the minimum is at $\mathbf{x} = [0, 0]^T$. The gradient is $\nabla f(\mathbf{x}) = \mathbf{J}(\mathbf{x})^T \mathbf{r}(\mathbf{x})$, where the Jacobian is given by

$$\mathbf{J}(\mathbf{x}) = \left\{ \frac{\partial r_i}{\partial x_j} \right\} = \sqrt{2} \begin{bmatrix} 20x_1 & -10 \\ 1 & 0 \end{bmatrix} \quad (2165)$$

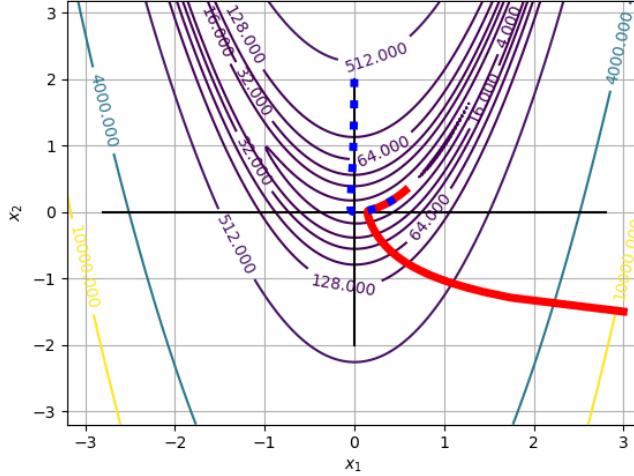


Figure 91: Gradient minimization of the Rosenbrock function for one curve with initial value $(3, -1.5)$ and one curve with initial value $(0, 2)$. The curves enter the valley of the minimum close to the origin, and progress slowly along the valley towards the minimum at $(1, 1)$.

It follows that

$$\mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}) = \begin{bmatrix} 800x_1^2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix} \quad (2166)$$

It is noted that the relation the Hessian is

$$\nabla^2 f(\mathbf{x}) = \mathbf{J}(\mathbf{x})^T \mathbf{J}(\mathbf{x}) + \frac{\partial^2 r_1}{\partial x_1 \partial x_2} r_1 + \frac{\partial^2 r_2}{\partial x_1 \partial x_2} r_2 \quad (2167)$$

$$= \begin{bmatrix} 800x_1^2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix} + 2 \begin{bmatrix} 20 & 0 \\ 0 & 0 \end{bmatrix} 10(x_1^2 - x_2) \quad (2168)$$

$$= \begin{bmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix} \quad (2169)$$

The Gauss-Newton method is given with the approximate Hessian as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha (\mathbf{J}(\mathbf{x}_k)^T \mathbf{J}(\mathbf{x}_k))^{-1} \mathbf{J}(\mathbf{x}_k) r(\mathbf{x}_k) \quad (2170)$$

where a step-length factor α is used. It is straightforward to verify that [80]

$$(\mathbf{J}(\mathbf{x}_k)^T \mathbf{J}(\mathbf{x}_k))^{-1} \mathbf{J}(\mathbf{x}_k) = -2 \begin{bmatrix} x_1 - 1 \\ x_1^2 - 2x_1 + x_2 \end{bmatrix} \quad (2171)$$

This gives the result shown in Figure 92. The step length is $\alpha = 0.01$ and there are 1000 iterations.

```
# Script for gradient and Gauss-Newton optimization of the Rosenbrock function
import numpy as np
import matplotlib.pyplot as plt
```

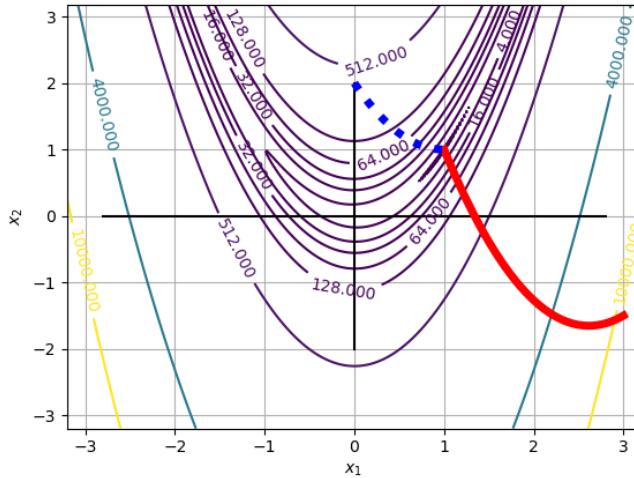


Figure 92: Gauss-Newton minimization of the Rosenbrock function. The curves approach the minimum more directly and with fewer iterations than for the gradient method.

```

# Initial condition
#x = np.array([1,1])
def Minv_gradL(x):
    return 2*np.array([x[0]-1,
                      x[0]**2 - 2*x[0] + x[1]])
def gradL(x):
    return np.array([ 400*x[0]**3 - 400*x[0]*x[1] + 2*x[1] + 2*x[0] - 2,
                     -200*x[0]**2 +200*x[1] ])
# L = 100*(x[0]**2 - x[1])**2 + (x[0] - 1)**2
x = np.array([3,-1.5])
y = np.array([0, 2])
xx = x; yy = y

# Gradient search
#N = 10000
#alpha = 0.0001
#for i in range(1,N):
#    x = x - alpha*gradL(x)
#    y = y - alpha*gradL(y)
#    xx = np.block([[xx], [x]])
#    yy = np.block([[yy], [y]])

# Newton search
N = 1000
alpha = 0.01
for i in range(1,N):

```

```

x = x - alpha*Minv_gradL(x)
y = y - alpha*Minv_gradL(y)
xx = np.block([[xx], [x]])
yy = np.block([[yy], [y]])

# Plotting
delta = 0.025
x = np.arange(-3.2, 3.2, delta)
y = np.arange(-3.2, 3.2, delta)
X, Y = np.meshgrid(x, y)
Z = 100*(X**2 - Y)**2 + (X - 1)**2

x_axis = np.block([[-1.4*2, 1.4*2], [0, 0]])
y_axis = np.block([[0, 0], [-2, 2]])

plt.figure(1)
plt.figure(1).clear()
CS = plt.contour(X, Y, Z, levels = [0.1, 4, 16, 32, 64, 128, 512,
                                      4000, 10000])
plt.clabel(CS, inline=1, fontsize=10)
plt.plot(x_axis[0,:], x_axis[1,:], 'k', y_axis[0,:], y_axis[1,:], 'k')
plt.plot(xx[:,0], xx[:,1], 'r', yy[:,0], yy[:,1], 'b:', linewidth=5)
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.grid()

```

A.12 Optimization of Rosenbrock's function with scipy

```

import numpy as np
from scipy.optimize import least_squares

def residual_rosenbrock(x):
    # The residual of the rosenbrock function
    print(x)
    return np.array([10 * (x[1] - x[0]**2),
                    (1 - x[0])])

def jac_rosenbrock(x):
    return np.array([
        [-20 * x[0], 10],
        [-1, 0]])

#x0_rosenbrock = np.array([3, -1.5])
x0_rosenbrock = np.array([0, 2])
#res_1 = least_squares(residual_rosenbrock, x0_rosenbrock)
print('Iterations:')
res_1 = least_squares(residual_rosenbrock, x0_rosenbrock, jac_rosenbrock)

```

```

print('\n Optimal solution: x = ', res_1.x)
print('\n cost = ', res_1.cost)
print('\n optimality = ', res_1.optimality)

```

B Results from linear algebra

This section presents a useful background in linear algebra. The presentation is based on [27] and [69].

B.1 Singular value decomposition

The singular value decomposition (SVD) of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is given by

$$\mathbf{A} = \sum_{i=1}^p \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \quad \mathbf{u}_i \in \mathbb{R}^m, \quad \mathbf{v}_i \in \mathbb{R}^n \quad (2172)$$

where $p = \min(m, n)$, $\mathbf{u}_i \in \mathbb{R}^m$ and $\mathbf{v}_i \in \mathbb{R}^n$ are orthogonal unit vectors, which means that $\mathbf{u}_i^T \mathbf{u}_j = \mathbf{v}_i^T \mathbf{v}_j = \delta_{ij}$, which is the Kronecker delta, which is given by $\delta_{ij} = 1, i = j$ and $\delta_{ij} = 0, i \neq j$. The scalars σ_i are called the singular values. The singular values are arranged so that

$$\sigma_1 \geq \sigma_2 \geq \dots \sigma_r > \sigma_{r+1} = \dots = \sigma_p = 0$$

where r is the rank of the matrix \mathbf{A} . It is seen that the first r of the singular values are greater than zero, and the remaining $p - r$ singular values are equal to zero.

The singular value decomposition can also be written

$$\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T \quad (2173)$$

where the matrices \mathbf{U} and \mathbf{V} are orthogonal matrices given by

$$\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_m) \in \mathbb{R}^{m \times m} \quad \text{and} \quad \mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_n) \in \mathbb{R}^{n \times n} \quad (2174)$$

while $\Sigma \in \mathbb{R}^{m \times n}$ is a rectangular matrix with the singular values σ_i on the diagonal.

If $n = m$, the matrix Σ is the square diagonal matrix

$$\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n) \in \mathbb{R}^{n \times n}, \quad m = n \quad (2175)$$

with the singular values on the diagonal.

If $m > n$, the matrix will in addition have $m - n$ rows with zeros as the last rows so that

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_n \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} \in \mathbb{R}^{m \times n} \quad (2176)$$

In this case $\Sigma^T \Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_n^2) \in \mathbb{R}^{n \times n}$.

If $m < n$, then

$$\Sigma = \begin{bmatrix} \sigma_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_m & 0 & \dots & 0 \end{bmatrix} \in \mathbb{R}^{m \times n} \quad (2177)$$

Example

Consider the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{bmatrix} \in \mathbb{R}^{2 \times 3} \quad (2178)$$

The singular value decomposition is

$$\mathbf{A} = \underbrace{\begin{bmatrix} -0.7071 & 0.7071 \\ -0.7071 & -0.7071 \end{bmatrix}}_U \underbrace{\begin{bmatrix} 5.0000 & 0 & 0 \\ 0 & 1.7321 & 0 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} -0.4243 & -0.4082 & -0.8083 \\ -0.7071 & -0.4082 & 0.5774 \\ -0.5657 & 0.8165 & -0.1155 \end{bmatrix}}_V^T \quad (2179)$$

It is easy to verify that the columns of \mathbf{U} and \mathbf{V} are orthogonal and of unit length.

Next, consider the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 4 & 1 \\ 4 & 5 & 1 \end{bmatrix} \in \mathbb{R}^{4 \times 3} \quad (2180)$$

The singular value decomposition is $\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$ where

$$\mathbf{U} = \begin{bmatrix} -0.3088 & 0.9372 & 0.1619 & 0.0000 \\ -0.3945 & 0.0122 & -0.8231 & 0.4082 \\ -0.5367 & -0.1509 & -0.1500 & -0.8165 \\ -0.6789 & -0.3141 & 0.5232 & 0.4082 \end{bmatrix} \quad (2181)$$

$$\Sigma = \begin{bmatrix} 9.4744 & 0 & 0 \\ 0 & 2.4885 & 0 \\ 0 & 0 & 0.2078 \\ 0 & 0 & 0 \end{bmatrix} \quad (2182)$$

$$\mathbf{V} = \begin{bmatrix} -0.5725 & -0.3003 & 0.7630 \\ -0.7750 & -0.1056 & -0.6231 \\ -0.2677 & 0.9480 & 0.1723 \end{bmatrix} \quad (2183)$$

Note that the rank of \mathbf{A} is 3. □

B.2 Nullspace and range space

In this section the system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ with $\mathbf{A} \in \mathbb{R}^{m \times n}$ will be analyzed in terms of the singular value decomposition. Using the singular value decomposition of (2172), this

equation can be written

$$\sum_{i=1}^p \sigma_i \mathbf{u}_i (\mathbf{v}_i^T \mathbf{x}) = \mathbf{b} \quad (2184)$$

where $p = \min(m, n)$. It is seen that \mathbf{b} is a linear combination of the vectors \mathbf{u}_i , $i = 1, \dots, p$. Moreover, it is seen that the component of \mathbf{b} along \mathbf{u}_i has magnitude which is the product of σ_i and $\mathbf{v}_i^T \mathbf{x}$, which is the magnitude of the component of \mathbf{x} along \mathbf{v}_i .

Suppose that the rank of \mathbf{A} is $r \leq p$. Then

$$\mathbf{A}\mathbf{x} = \sum_{i=1}^r \sigma_i (\mathbf{v}_i^T \mathbf{x}) \mathbf{u}_i \quad (2185)$$

Then $\mathbf{A}\mathbf{x}$ is a linear combination of \mathbf{u}_i , $i = 1, \dots, r$, which is written $\mathbf{A}\mathbf{x} \in \mathcal{R}(\mathbf{A})$, where

$$\mathcal{R}(\mathbf{A}) \stackrel{\text{def}}{=} \text{span}\{\mathbf{u}_1, \dots, \mathbf{u}_r\} \quad (2186)$$

Here $\mathcal{R}(\mathbf{A})$ us called the range space of \mathbf{A} . It is seen that if \mathbf{x} is a linear combination of $\mathbf{v}_{r+1}, \dots, \mathbf{v}_m$, then $\mathbf{b} = \mathbf{0}$. Therefore, the space spanned by these vectors is called the nullspace of \mathbf{A} , which is written

$$\mathcal{N}(\mathbf{A}) \stackrel{\text{def}}{=} \text{span}\{\mathbf{v}_{r+1}, \dots, \mathbf{v}_n\} \quad (2187)$$

It follows that $\mathbf{x} \in \mathcal{N}(\mathbf{A}) \Rightarrow \mathbf{A}\mathbf{x} = \mathbf{0}$.

The transpose of the singular value decomposition of \mathbf{A} is

$$\mathbf{A}^T = \sum_{i=1}^p \sigma_i \mathbf{v}_i \mathbf{u}_i^T \quad (2188)$$

This implies that

$$\mathcal{R}(\mathbf{A}^T) = \text{span}\{\mathbf{v}_1, \dots, \mathbf{v}_r\} \quad (2189)$$

$$\mathcal{N}(\mathbf{A}^T) = \text{span}\{\mathbf{u}_{r+1}, \dots, \mathbf{u}_m\} \quad (2190)$$

It is seen that vectors in the nullspace $\mathcal{N}(\mathbf{A}^T)$ of \mathbf{A}^T are orthogonal to vectors in the range space $\mathcal{R}(\mathbf{A})$ of \mathbf{A} , and that vectors in range space $\mathcal{R}(\mathbf{A}^T)$ of \mathbf{A}^T are orthogonal to vectors in the nullspace $\mathcal{N}(\mathbf{A})$ of \mathbf{A} . This is written

$$\mathcal{R}(\mathbf{A}) = \mathcal{N}(\mathbf{A}^T)^\perp, \quad \mathcal{N}(\mathbf{A}^T) = \mathcal{R}(\mathbf{A})^\perp \quad (2191)$$

$$\mathcal{R}(\mathbf{A}^T) = \mathcal{N}(\mathbf{A})^\perp, \quad \mathcal{N}(\mathbf{A}) = \mathcal{R}(\mathbf{A}^T)^\perp \quad (2192)$$

where the superscript \perp denotes the orthogonal complement of a set. The orthogonal complement V^\perp of a set V is the set of all vectors that are orthogonal to all the vectors in the set V .

B.3 The singular value decomposition and matrix inversion

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a square matrix with singular value decomposition $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$.

If it is assumed that \mathbf{A} is nonsingular, the singular values will all be greater than zero, that is, $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n > 0$. Then the inverse of \mathbf{A} is

$$\mathbf{A}^{-1} = \mathbf{V}\Sigma^{-1}\mathbf{U}^T = \sum_{i=1}^n \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T \quad (2193)$$

This is verified by

$$(\mathbf{U}\Sigma\mathbf{V}^T)(\mathbf{V}\Sigma^{-1}\mathbf{U}^T) = \mathbf{I} \quad (2194)$$

The solution of the equation $\mathbf{A}\mathbf{x} = \mathbf{b}$ is then

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} = \sum_{i=1}^n \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T \mathbf{b} = \sum_{i=1}^n \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i \quad (2195)$$

This shows that the solution \mathbf{x} is the sum of vectors $\alpha_i \mathbf{v}_i$, where the length α_i is the component of \mathbf{b} along \mathbf{u}_i scaled with $1/\sigma_i$. When the matrix \mathbf{A} is close to being singular, σ_n will be close to zero, and the scaling $1/\sigma_n$ will tend to infinity.

B.4 The thin SVD

The thin SVD of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ where $m \geq n$ is given by

$$\mathbf{A} = \mathbf{U}_n \Sigma_n \mathbf{V}^T \quad (2196)$$

where $\mathbf{V} = (\mathbf{v}_1, \dots, \mathbf{v}_n) \in \mathbb{R}^{n \times n}$ as in the usual SVD, while

$$\mathbf{U}_n = (\mathbf{u}_1, \dots, \mathbf{u}_n) \in \mathbb{R}^{m \times n} \quad (2197)$$

only includes the first n columns, which are the columns that correspond to the singular values. It is straightforward to verify that

$$\mathbf{U}_n^T \mathbf{U}_n = \mathbf{I} \in \mathbb{R}^{n \times n} \quad (2198)$$

The matrix \mathbf{U}_n has orthogonal columns, but not orthogonal rows, and $\mathbf{U}_n \mathbf{U}_n^T$ will in general not be the identity matrix. Finally, the matrix

$$\Sigma_n = \text{diag}(\sigma_1, \dots, \sigma_n) \in \mathbb{R}^{n \times n} \quad (2199)$$

of singular values is square.

In the case that $m < n$ the thin SVD is

$$\mathbf{A} = \mathbf{U} \Sigma_m \mathbf{V}_m^T \quad (2200)$$

where $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_m) \in \mathbb{R}^{m \times m}$ as in the usual SVD, while

$$\mathbf{V}_m = (\mathbf{v}_1, \dots, \mathbf{v}_m) \in \mathbb{R}^{n \times m} \quad (2201)$$

only includes the first m columns, which are the columns that correspond to the singular values. The matrix \mathbf{V}_m has orthogonal columns, but not orthogonal rows. This means that

$$\mathbf{V}_m^T \mathbf{V}_m = \mathbf{I} \in \mathbb{R}^{m \times m} \quad (2202)$$

Finally, the matrix

$$\boldsymbol{\Sigma}_m = \text{diag}(\sigma_1, \dots, \sigma_m) \in \mathbb{R}^{m \times m} \quad (2203)$$

of singular values is square.

The thin SVD is the same as the SVD when $m = n$.

The thin SVD is called the economical SVD in MATLAB.

Example

Let

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix} \quad (2204)$$

Then the singular value decomposition is $\mathbf{A} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T$ where

$$\mathbf{U} = \begin{bmatrix} -0.1525 & -0.8226 & -0.3945 & -0.3800 \\ -0.3499 & -0.4214 & 0.2428 & 0.8007 \\ -0.5474 & -0.0201 & 0.6979 & -0.4614 \\ -0.7448 & 0.3812 & -0.5462 & 0.0407 \end{bmatrix} \quad (2205)$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} 14.2691 & 0 \\ 0 & 0.6268 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (2206)$$

$$\mathbf{V} = \begin{bmatrix} -0.6414 & 0.7672 \\ -0.7672 & -0.6414 \end{bmatrix} \quad (2207)$$

The thin SVD, which is found in Matlab with `[U,Sigma,V] = svd(A,'econ')` is $\mathbf{A} = \mathbf{U}_1 \boldsymbol{\Sigma}_1 \mathbf{V}^T$ where the matrix \mathbf{V} is the same, and

$$\mathbf{U}_1 = \begin{bmatrix} -0.1525 & -0.8226 \\ -0.3499 & -0.4214 \\ -0.5474 & -0.0201 \\ -0.7448 & 0.3812 \end{bmatrix} \quad (2208)$$

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 14.2691 & 0 \\ 0 & 0.6268 \end{bmatrix} \quad (2209)$$

□

B.5 Singular values and eigenvalues for a symmetric matrix

Consider the $n \times n$ matrix \mathbf{P} . If the matrix \mathbf{P} is symmetric so that $\mathbf{P} = \mathbf{P}^T$, then all eigenvalues are real. If the matrix \mathbf{P} is symmetric and positive definite so that $\mathbf{P} = \mathbf{P}^T$ and $\mathbf{x}^T \mathbf{P} \mathbf{x} > 0$ whenever $\mathbf{x} \neq 0$, then all the eigenvalues of \mathbf{P} are real and positive.

It is assumed that the matrix \mathbf{P} is symmetric and positive definite. This the matrix can be written

$$\mathbf{P} = \mathbf{M}^T \mathbf{M} \quad (2210)$$

where $\mathbf{M} \in \mathbf{R}^{n \times n}$ is a full rank matrix.

The singular value decomposition of \mathbf{M} is

$$\mathbf{M} = \mathbf{U} \Sigma \mathbf{V}^T = \sum_{i=1}^p \sigma_i \mathbf{u}_i \mathbf{v}_i^T \quad (2211)$$

This gives

$$\mathbf{P} = \mathbf{V} \Sigma \mathbf{U}^T \mathbf{U} \Sigma \mathbf{V}^T = \mathbf{V} \Sigma^2 \mathbf{V}^T = \sum_{i=1}^p \sigma_i^2 \mathbf{v}_i \mathbf{v}_i^T \quad (2212)$$

This shows that the singular values of \mathbf{P} are the square of the singular values of \mathbf{M} . Since the vectors \mathbf{v}_i are orthogonal, it follows that

$$\mathbf{P} \mathbf{v}_i = \sigma_i^2 \mathbf{v}_i \quad (2213)$$

This shows that the singular values of a positive definite matrix \mathbf{P} are equal to its eigenvalues, and that the eigenvectors are equal to the vectors \mathbf{v}_i of the SVD. Moreover, the singular values of $\mathbf{P} = \mathbf{M}^T \mathbf{M}$ are the square of the singular values of \mathbf{M} .

Comment

Suppose that the SVD $\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$ is used to calculate the \mathbf{V} matrix when $m \gg n$. This can be done with the thin SVD. An alternative is to calculate the SVD of $\mathbf{A}^T \mathbf{A}$ which is

$$\mathbf{A}^T \mathbf{A} = \mathbf{V} \Sigma \mathbf{U}^T \mathbf{U} \Sigma \mathbf{V}^T = \mathbf{V} \Sigma^2 \mathbf{V}^T \quad (2214)$$

Then the singular value decomposition is done for the $n \times n$ matrix $\mathbf{A}^T \mathbf{A}$. This gives a faster SVD than the SVD of the $m \times n$ matrix \mathbf{A} at the cost of calculating $\mathbf{A}^T \mathbf{A}$. It is not clear that this will lead to a reduced computational cost compared to computing the SVD of \mathbf{A} .

□

B.6 Polar decomposition

Consider the matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ where $m > n$. Let the thin SVD be $\mathbf{A} = \mathbf{U}_1 \Sigma_1 \mathbf{V}^T$. Then, using $\mathbf{V}^T \mathbf{V} = \mathbf{I}$, this can be written

$$\mathbf{A} = (\mathbf{U}_1 \mathbf{V}^T) (\mathbf{V} \Sigma_1 \mathbf{V}^T) = \mathbf{Z} \mathbf{P} \quad (2215)$$

The matrix $\mathbf{Z} = \mathbf{U}_1 \mathbf{V}^T$ has orthonormal columns, which follows from $\mathbf{Z}^T \mathbf{Z} = \mathbf{V} \mathbf{U}_1^T \mathbf{U}_1 \mathbf{V}^T = \mathbf{V} \mathbf{V}^T = \mathbf{I}$, and $\mathbf{P} = \mathbf{V} \Sigma_1 \mathbf{V}^T$ is symmetric and positive semidefinite.

In the case of a matrix $\mathbf{X} \in \mathbb{R}^{3 \times 3}$ with SVD $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$ the polar decomposition is

$$\mathbf{X} = \mathbf{R}\mathbf{P} \quad (2216)$$

where $\mathbf{R} = \mathbf{U}\mathbf{V}^T$ is an orthogonal matrix, which may be a rotation matrix or a reflection matrix, and $\mathbf{P} = \mathbf{V}\Sigma\mathbf{V}^T$ is symmetric and positive semidefinite. If \mathbf{X} is nonsingular, then \mathbf{P} is positive definite.

B.7 The square root of a matrix

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a positive semidefinite symmetric matrix with SVD given by $\mathbf{A} = \mathbf{U}\Sigma\mathbf{U}^T$ where $\Sigma = \text{diag}\{\sigma_1, \dots, \sigma_n\}$. Then the matrix

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{U}^T \quad (2217)$$

is said to be the square root of \mathbf{A} if $\mathbf{S}^2 = \Sigma$. This follows from

$$\mathbf{X}^2 = (\mathbf{U}\mathbf{S}\mathbf{U}^T)(\mathbf{U}\mathbf{S}\mathbf{U}^T) = \mathbf{U}\mathbf{S}^2\mathbf{U}^T = \mathbf{U}\Sigma\mathbf{U}^T = \mathbf{A} \quad (2218)$$

It is seen that

$$\mathbf{S} = \text{diag}\{\sqrt{\sigma_1}, \dots, \sqrt{\sigma_n}\} \quad (2219)$$

B.8 The trace of a matrix

The trace of a matrix product is useful in numerical calculations of rotation matrices, where expressions involving scalar products of vectors can be reformulated in terms of the trace of matrix products.

The trace of a $n \times n$ matrix $\mathbf{A} = \{a_{ij}\}$ is defined as

$$\text{tr}(\mathbf{A}) = \sum_{i=1}^n a_{ii} \quad (2220)$$

It is straightforward to verify that that

$$\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{A}^T) \quad (2221)$$

Let $\mathbf{X} = \{x_{ij}\}$ and $\mathbf{Y} = \{y_{ij}\}$ be $m \times n$ matrices. Then the diagonal element i of the matrix product $\mathbf{X}^T\mathbf{Y}$ is $\sum_{i=1}^n x_{ij}y_{ij}$ and

$$\text{tr}(\mathbf{X}^T\mathbf{Y}) = \sum_{j=1}^m \sum_{i=1}^n x_{ij}y_{ij} \quad (2222)$$

Then it follows that

$$\text{tr}(\mathbf{X}^T\mathbf{Y}) = \text{tr}(\mathbf{XY}^T) = \text{tr}(\mathbf{Y}^T\mathbf{X}) = \text{tr}(\mathbf{YX}^T) \quad (2223)$$

This again implies that two $n \times n$ matrices \mathbf{A} and \mathbf{B} the trace satisfies

$$\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA}) \quad (2224)$$

For the vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, this gives

$$\mathbf{a}^T \mathbf{b} = \text{tr}(\mathbf{a}^T \mathbf{b}) = \text{tr}(\mathbf{b} \mathbf{a}^T) = \text{tr}(\mathbf{a} \mathbf{b}^T) \quad (2225)$$

which shows that the inner product of two vectors is equal to the trace of the outer product of the two vectors. The trace of a similarity transform $\mathbf{N}^{-1} \mathbf{A} \mathbf{N}$ of a matrix \mathbf{A} is equal to the trace of the matrix, because

$$\text{tr}(\mathbf{N}^{-1} \mathbf{A} \mathbf{N}) = \text{tr}[\mathbf{N}^{-1}(\mathbf{A} \mathbf{N})] = \text{tr}[(\mathbf{A} \mathbf{N}) \mathbf{N}^{-1}] = \text{tr}(\mathbf{A} \mathbf{N} \mathbf{N}^{-1}) = \text{tr}(\mathbf{A}) \quad (2226)$$

A matrix \mathbf{A} can be diagonalized with the similarity transform

$$\mathbf{Q} \mathbf{A} \mathbf{Q}^{-1} = \text{diag}(\lambda_1, \dots, \lambda_n) \quad (2227)$$

where $\lambda_1, \dots, \lambda_n$ are the eigenvalues of \mathbf{A} . This diagonalization is a similarity transform, and it follows that

$$\text{tr}(\mathbf{A}) = \text{tr}(\mathbf{Q} \mathbf{A} \mathbf{Q}^{-1}) = \text{tr}(\text{diag}(\lambda_1, \dots, \lambda_n)) \quad (2228)$$

and it is concluded that the trace of a matrix is equal to the sum of its eigenvalues, that is,

$$\text{tr}(\mathbf{A}) = \lambda_1 + \dots + \lambda_n \quad (2229)$$

In particular this is the case for the similarity transform with an orthogonal matrix \mathbf{U} , where $\mathbf{U}^{-1} = \mathbf{U}^T$. Then

$$\text{tr}(\mathbf{U}^T \mathbf{A} \mathbf{U}) = \text{tr}[\mathbf{U}^T(\mathbf{A} \mathbf{U})] = \text{tr}[(\mathbf{A} \mathbf{U}) \mathbf{U}^T] = \text{tr}(\mathbf{A} \mathbf{U} \mathbf{U}^T) = \text{tr}(\mathbf{A}) \quad (2230)$$

Example 1 The trace of a rotation matrix $\mathbf{R} = \cos \theta \mathbf{I} + \sin \theta \mathbf{k}^\times + (1 - \cos \theta) \mathbf{k} \mathbf{k}^T$ is found by noting that $\text{tr}(\mathbf{k}^\times) = 0$, while $\text{tr}(\mathbf{k} \mathbf{k}^T) = \mathbf{k}^T \mathbf{k} = 1$. This gives

$$\text{tr}(\mathbf{R}) = \text{tr}(\cos \theta \mathbf{I} + (1 - \cos \theta) \mathbf{k} \mathbf{k}^T) = 3 \cos \theta + (1 - \cos \theta) = 2 \cos \theta + 1 \quad (2231)$$

In terms of quaternions the rotation matrix can be written $\mathbf{R} = (2\eta^2 - 1) \mathbf{I} + 2\boldsymbol{\epsilon} \boldsymbol{\epsilon}^T + \eta \boldsymbol{\epsilon}^\times$. The trace is then

$$\text{tr}(\mathbf{R}) = 3(2\eta^2 - 1) + 2\boldsymbol{\epsilon}^T \boldsymbol{\epsilon} = 6\eta^2 - 3 + 2(1 - \eta^2) = 4\eta^2 - 1 \quad (2232)$$

The two expressions are consistent as $4\eta^2 - 1 = 4 \cos^2(\theta/2) - 1 = 2(\cos \theta + 1) - 1 = 2 \cos \theta + 1$. \square

B.9 The trace of a quadratic form

The trace of a scalar is the scalar itself. This means that the trace of a quadratic form $\mathbf{x}^T \mathbf{Q} \mathbf{x}$ is equal to its own trace, which gives

$$\mathbf{x}^T \mathbf{Q} \mathbf{x} = \text{tr}(\mathbf{x}^T \mathbf{Q} \mathbf{x}) \quad (2233)$$

The trace of an inner product $\mathbf{x}^T \mathbf{y}$ is equal to the trace of the outer product $\mathbf{x} \mathbf{y}^T$, which is verified by straightforward calculation. It follows that

$$\text{tr}(\mathbf{x}^T \mathbf{y}) = \text{tr}(\mathbf{x} \mathbf{y}^T) = \text{tr}(\mathbf{y} \mathbf{x}^T) \quad (2234)$$

Then, by inserting $\mathbf{y} = \mathbf{Q}\mathbf{x}$ it follows that

$$\text{tr}(\mathbf{x}^T(\mathbf{Q}\mathbf{x})) = \text{tr}((\mathbf{Q}\mathbf{x})\mathbf{x}^T) = \text{tr}(\mathbf{Q}\mathbf{x}\mathbf{x}^T) \quad (2235)$$

It is concluded that

$$\mathbf{x}^T\mathbf{Q}\mathbf{x} = \text{tr}(\mathbf{Q}\mathbf{x}\mathbf{x}^T) \quad (2236)$$

This is a special case of the cyclic property of the trace, which is written

$$\text{tr}(\mathbf{ABC}) = \text{tr}(\mathbf{CAB}) = \text{tr}(\mathbf{BCA}) \quad (2237)$$

for matrices \mathbf{A} , \mathbf{B} and \mathbf{C} of dimensions so that the matrix products makes sense.

B.10 The Frobenius inner product

Let $\mathbf{A} = \{a_{ij}\} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} = \{b_{ij}\} \in \mathbb{R}^{m \times n}$. The Frobenius inner product is defined as

$$\langle \mathbf{A}, \mathbf{B} \rangle_F = \sum_{i=1}^m \sum_{j=1}^n a_{ij}b_{ij} \quad (2238)$$

The inner product can be expressed in terms of the trace as

$$\langle \mathbf{A}, \mathbf{B} \rangle_F = \text{tr}(\mathbf{A}^T \mathbf{B}) \quad (2239)$$

This is verified by calculation the elements c_{ij} of $\mathbf{C} = \mathbf{A}^T \mathbf{B} \in \mathbb{R}^{n \times n}$, which are

$$c_{ij} = \sum_{k=1}^m a_{ki}b_{kj} \quad \Rightarrow \quad c_{ii} = \sum_{k=1}^m a_{ki}b_{ki} \quad (2240)$$

It follows that

$$\text{tr}(\mathbf{A}^T \mathbf{B}) = \sum_{i=1}^n c_{ii} = \sum_{i=1}^n \sum_{k=1}^m a_{ki}b_{ki} = \sum_{i=1}^m \sum_{j=1}^n a_{ij}b_{ij} = \langle \mathbf{A}, \mathbf{B} \rangle_F \quad (2241)$$

It can also be given in terms of the vectorized forms

$$\text{vec}(\mathbf{A}) = [a_{1,1}, a_{1,2}, \dots, a_{1,n}, a_{2,1}, \dots, a_{m,n}]^T \quad (2242)$$

$$\text{vec}(\mathbf{B}) = [b_{1,1}, b_{1,2}, \dots, b_{1,n}, b_{2,1}, \dots, b_{m,n}]^T \quad (2243)$$

of the matrices \mathbf{A} and \mathbf{B} , where $\text{vec}(\mathbf{M})$ is a vector where the columns of \mathbf{M} are stacked. Then the Frobenius inner product is

$$\langle \mathbf{A}, \mathbf{B} \rangle_F = \text{vec}(\mathbf{A})^T \text{vec}(\mathbf{B}) \quad (2244)$$

B.11 The Frobenius norm

The Frobenius norm for a matrix $\mathbf{A} = \{a_{ij}\} \in \mathbb{R}^{m \times n}$ is defined as

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2} \quad (2245)$$

The Frobenius norm of a matrix \mathbf{A} is related to the trace by

$$\text{tr}(\mathbf{A}^T \mathbf{A}) = \text{tr}(\mathbf{A} \mathbf{A}^T) = \sum_{i=1}^m \sum_{j=1}^n a_{ij}^2 = \|\mathbf{A}\|_F^2 \quad (2246)$$

which is verified by

$$(\mathbf{A}^T \mathbf{A})_{ij} = \sum_{k=1}^m a_{ki} a_{kj} \Rightarrow (\mathbf{A}^T \mathbf{A})_{ii} = \sum_{k=1}^m a_{ki}^2 \Rightarrow \text{tr}(\mathbf{A}^T \mathbf{A}) = \sum_{i=1}^n \sum_{k=1}^m a_{ki}^2 \quad (2247)$$

and

$$(\mathbf{A} \mathbf{A}^T)_{ij} = \sum_{k=1}^n a_{ik} a_{jk} \Rightarrow (\mathbf{A} \mathbf{A}^T)_{ii} = \sum_{k=1}^n a_{ik}^2 \Rightarrow \text{tr}(\mathbf{A} \mathbf{A}^T) = \sum_{i=1}^m \sum_{k=1}^n a_{ik}^2 \quad (2248)$$

The Frobenius inner product satisfies

$$\langle \mathbf{A}, \mathbf{A} \rangle_F = \|\mathbf{A}\|_F^2 \quad (2249)$$

Let the column vectors of \mathbf{A} be $\mathbf{a}_i \in \mathbb{R}^m$, $i = 1, \dots, n$ so that $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_n]$, and the vectorized form of \mathbf{A} is $\text{vec}(\mathbf{A}) = [\mathbf{a}_1^T, \dots, \mathbf{a}_n^T]^T$. Then

$$\|\mathbf{A}\|_F^2 = \sum_{i=1}^n \|\mathbf{a}_i\|^2 = \sum_{i=1}^n \mathbf{a}_i^T \mathbf{a}_i = \text{vec}(\mathbf{A})^T \text{vec}(\mathbf{A}) \quad (2250)$$

In applications this is often used in the following form

$$\sum_{i=1}^n \|\mathbf{y}_i - \mathbf{R}\mathbf{x}_i\|^2 = \sum_{i=1}^n (\mathbf{y}_i - \mathbf{R}\mathbf{x}_i)^T (\mathbf{y}_i - \mathbf{R}\mathbf{x}_i) = \|\mathbf{Y} - \mathbf{R}\mathbf{X}\|_F^2 \quad (2251)$$

where $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ and $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_n)$ are $m \times n$ matrices and \mathbf{R} is a $m \times m$ matrix. Then

$$\|\mathbf{Y} - \mathbf{R}\mathbf{X}\|_F^2 = \text{tr}[(\mathbf{Y} - \mathbf{R}\mathbf{X})(\mathbf{Y} - \mathbf{R}\mathbf{X})^T] \quad (2252)$$

$$= \text{tr}(\mathbf{Y}\mathbf{Y}^T) + \text{tr}(\mathbf{R}\mathbf{X}\mathbf{X}^T\mathbf{R}^T) - \text{tr}(\mathbf{R}\mathbf{X}\mathbf{Y}^T) - \text{tr}[\mathbf{Y}(\mathbf{R}\mathbf{X})^T] \quad (2253)$$

$$= \text{tr}(\mathbf{Y}\mathbf{Y}^T) + \text{tr}(\mathbf{X}\mathbf{X}^T) - 2\text{tr}(\mathbf{R}\mathbf{X}\mathbf{Y}^T) \quad (2254)$$

Let $\mathbf{B} \in \mathbb{R}^{n \times n}$ be a square matrix, and let $\mathbf{U} \in \mathbb{R}^{n \times n}$ be an orthogonal matrix which satisfies $\mathbf{U}^{-1} = \mathbf{U}^T$. Then the Frobenius norm of \mathbf{B} , $\mathbf{U}\mathbf{B}$ and $\mathbf{B}\mathbf{U}$ will be equal. This is seen from

$$\|\mathbf{U}\mathbf{B}\|_F^2 = \text{tr}((\mathbf{U}\mathbf{B})^T \mathbf{U}\mathbf{B}) = \text{tr}(\mathbf{B}^T \mathbf{U}^T \mathbf{U}\mathbf{B}) = \text{tr}(\mathbf{B}^T \mathbf{B}) = \|\mathbf{B}\|_F^2 \quad (2255)$$

In the same way it is shown that

$$\|\mathbf{B}\mathbf{U}\|_F^2 = \text{tr}(\mathbf{B}\mathbf{U}(\mathbf{B}\mathbf{U}^T)) = \text{tr}(\mathbf{B}\mathbf{U}\mathbf{U}^T\mathbf{B}^T) = \text{tr}(\mathbf{B}\mathbf{B}^T) = \|\mathbf{B}\|_F^2 \quad (2256)$$

Moreover, if $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$ is the singular value decomposition of \mathbf{A} , then the Frobenius norm of Σ , $\mathbf{U}\Sigma$ and $\mathbf{U}\Sigma\mathbf{V}^T$ is the same since \mathbf{U} and \mathbf{V} are orthogonal. It follows that the Frobenius norm satisfies

$$\|\mathbf{A}\|_F^2 = \|\mathbf{U}\Sigma\mathbf{V}^T\|_F^2 = \|\Sigma\|_F^2 = \sigma_1^2 + \dots + \sigma_p^2, \quad p = \min(m, n) \quad (2257)$$

B.12 Vector norm

The usual norm for a vector $\mathbf{x} = [x_1, \dots, x_n]^T \in \mathbb{R}^n$ is the 2-norm

$$\|\mathbf{x}\| = \sqrt{|x_1|^2 + \dots + |x_n|^2} = \sqrt{\mathbf{x}^T \mathbf{x}} \quad (2258)$$

which will be termed the vector norm. A vector \mathbf{e} with unit vector norm is called a unit vector and satisfies $\mathbf{e}^T \mathbf{e} = 1$.

The Cauchy-Schwarz inequality states that for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

$$|\mathbf{x}^T \mathbf{y}| \leq \|\mathbf{x}\| \|\mathbf{y}\| \quad (2259)$$

B.13 The induced matrix 2-norm

The induced 2-norm of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is defined as

$$\|\mathbf{A}\|_2 = \max \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \max_{\|\mathbf{x}\|_2=1} \|\mathbf{A}\mathbf{x}\|_2 \quad (2260)$$

where $\|\mathbf{A}\mathbf{x}\|_2$ is the vector norm of $\mathbf{A}\mathbf{x}$. This type of norm is called an induced norm as the norm is found from the vector $\mathbf{A}\mathbf{x}$ instead of the matrix itself.

Using the singular value decomposition of \mathbf{A} , it is found that

$$\mathbf{A}\mathbf{x} = \sum_{i=1}^p \sigma_i \mathbf{u}_i \mathbf{v}_i^T \mathbf{x} \quad (2261)$$

where σ_1 is the largest singular value. It follows that when $\|\mathbf{x}\|_2 = 1$, the maximum value of $\|\mathbf{A}\mathbf{x}\|_2$ is found for $\mathbf{x} = \mathbf{v}_1$, which gives $\mathbf{A}\mathbf{x} = \sigma_1 \mathbf{u}_1$, which has norm $\|\mathbf{A}\mathbf{x}\|_2 = \sigma_1$. It follows that

$$\|\mathbf{A}\|_2 = \sigma_1 \quad (2262)$$

which means that the induced 2-norm of a matrix equals the largest singular value.

It is recalled that if $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T$, then $\mathbf{A}\mathbf{A}^T = \mathbf{U}\Sigma^2\mathbf{U}^T$, which means that σ_1^2 is the largest singular value of $\mathbf{A}\mathbf{A}^T$. Moreover, the largest eigenvalue $\lambda_{\max}(\mathbf{A}\mathbf{A}^T)$ of $\mathbf{A}\mathbf{A}^T$ is equal to the largest singular value σ_1^2 . This gives the alternative expression

$$\|\mathbf{A}\|_2^2 = \lambda_{\max}(\mathbf{A}\mathbf{A}^T) \quad (2263)$$

for the induced 2-norm of \mathbf{A} . It is also possible to use $\|\mathbf{A}\|_2^2 = \lambda_{\max}(\mathbf{A}^T \mathbf{A})$, which is seen from $\mathbf{A}^T \mathbf{A} = \mathbf{V} \boldsymbol{\Sigma}^2 \mathbf{V}^T$.

It is noted that the Frobenius norm satisfies $\|\mathbf{A}\|_F^2 = \sigma_1^2 + \dots + \sigma_r^2$, where r is the rank of \mathbf{A} . This implies that the Frobenius norm is related to the induced 2-norm by

$$\|\mathbf{A}\|_2 \leq \|\mathbf{A}\|_F \leq \sqrt{r} \|\mathbf{A}\|_2 \quad (2264)$$

Example

Let $\mathbf{k} = [k_x, k_y, k_z]^T \in \mathbb{R}^3$ be a unit vector. The 2-norm of the skew symmetric matrix

$$\mathbf{k}^\times = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix} \quad (2265)$$

is found by considering the vector $\mathbf{k}^\times \mathbf{x}$ where \mathbf{x} is a unit vector. Then the maximum norm of $\mathbf{k}^\times \mathbf{x}$ is unity, which is achieved when \mathbf{k} and \mathbf{x} are orthogonal. This means that

$$\|\mathbf{k}^\times\|_2 = 1 \quad (2266)$$

The Frobenius norm of \mathbf{k}^\times is

$$\|\mathbf{k}^\times\|_F = \sqrt{2(k_x^2 + k_y^2 + k_z^2)} = \sqrt{2} \quad (2267)$$

It is seen that

$$\|\mathbf{k}^\times\|_F = \sqrt{2} \|\mathbf{k}^\times\|_2 \quad (2268)$$

The function `norm` in MATLAB returns the largest singular value when it is applied to a matrix. This means that $\text{norm}(\mathbf{k}^\times) = 1$, as the singular values of \mathbf{k}^\times are $\{1, 1, 0\}$. \square

B.14 Quadratic forms and positive definite matrices

Let $\mathbf{x} = (x_1, \dots, x_n)^T$ be a vector and let $\mathbf{P} = \mathbf{P}^T = \{p_{ij}\} \in \mathbb{R}^{n \times n}$ be a symmetric square matrix. Then the quadratic form

$$q = \mathbf{x}^T \mathbf{P} \mathbf{x} = \sum_{j=1}^m \sum_{i=1}^n p_{ij} x_i x_j \quad (2269)$$

is a scalar function. The matrix \mathbf{P} is said to be positive definite if $\mathbf{x}^T \mathbf{P} \mathbf{x} > 0$ for all $\mathbf{x} \neq 0$, and it is said to be positive semidefinite if $\mathbf{x}^T \mathbf{P} \mathbf{x} \geq 0$ for all $\mathbf{x} \neq 0$.

B.15 Linear least-squares solution in terms of the normal equations for the full rank case

Linear least-squares is widely used in applications involving the solution of equations of the form

$$\mathbf{A} \mathbf{x} = \mathbf{b} \quad (2270)$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $m > n$. This is a set of m equations in n unknowns, which means that the problem is overdetermined, and an exact solution is in general not available. Then a least-squares solution is sought so the the error $\|\mathbf{Ax} - \mathbf{b}\|$ is minimized.

In the full rank case the matrix \mathbf{A} has rank n . Then the least-squares solution \mathbf{x}_{LS} is normally found from the normal equations

$$\mathbf{A}^T \mathbf{Ax}_{LS} = \mathbf{A}^T \mathbf{b} \quad (2271)$$

which is obtained by left multiplication the system of equations by \mathbf{A}^T . Since \mathbf{A} has rank n , the $n \times n$ matrix $\mathbf{A}^T \mathbf{A}$ will be of full rank, and a solution can be found from

$$\mathbf{x}_{LS} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} \quad (2272)$$

To verify that this is the least-squares solution the following expression is considered:

$$\|\mathbf{A}(\mathbf{x} + \alpha \mathbf{z}) - \mathbf{b}\|^2 = \|\mathbf{Ax} - \mathbf{b}\|^2 - 2\alpha \mathbf{z}^T \mathbf{A}^T (\mathbf{Ax} - \mathbf{b}) + \alpha^2 \|\mathbf{z}\|^2 \quad (2273)$$

where $\alpha > 0$ is a positive scalar. If $\mathbf{A}^T(\mathbf{Ax} - \mathbf{b}) \neq \mathbf{0}$, then $\|\mathbf{A}(\mathbf{x} + \alpha \mathbf{z}) - \mathbf{b}\|^2 < \|\mathbf{Ax} - \mathbf{b}\|^2$ for $\mathbf{z} = -\mathbf{A}^T(\mathbf{Ax} - \mathbf{b})$ for sufficiently small α . This means that the least-squares solution must satisfy $\mathbf{A}^T(\mathbf{Ax}_{LS} - \mathbf{b}) = \mathbf{0}$ and it follows that \mathbf{x}_{LS} must satisfy the normal equations.

B.16 The normal equations in terms of the singular value decomposition for the full rank case

In terms of the singular value decomposition $\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$ the left side of the normal equation can be written

$$\mathbf{A}^T \mathbf{Ax} = \mathbf{V} \mathbf{\Sigma}^T \mathbf{U}^T \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T \mathbf{x} = \mathbf{V} \mathbf{\Sigma}^T \mathbf{\Sigma} \mathbf{V}^T \mathbf{x} = \sum_{i=1}^n \sigma_i^2 \mathbf{v}_i (\mathbf{v}_i^T \mathbf{x}) \quad (2274)$$

The left side of the normal equation is

$$\mathbf{A}^T \mathbf{b} = \mathbf{V} \mathbf{\Sigma}^T \mathbf{U}^T \mathbf{b} = \sum_{i=1}^n \sigma_i \mathbf{v}_i (\mathbf{u}_i^T \mathbf{b}) \quad (2275)$$

It follows that the solution of the normal equation $\mathbf{A}^T \mathbf{Ax} = \mathbf{A}^T \mathbf{b}$ is found from

$$\mathbf{x} = (\mathbf{V} \mathbf{\Sigma}^T \mathbf{V}^T)^{-1} \mathbf{V} \mathbf{\Sigma}^T \mathbf{U}^T \mathbf{b} = \mathbf{V} \mathbf{\Sigma}^{-1} \mathbf{\Sigma}^{-T} \mathbf{V}^T \mathbf{V} \mathbf{\Sigma}^T \mathbf{U}^T \mathbf{b} = \mathbf{V} \mathbf{\Sigma}^{-1} \mathbf{U}^T \mathbf{b} \quad (2276)$$

In vector form the solution satisfies

$$\sum_{j=1}^n \sigma_j^2 \mathbf{v}_j (\mathbf{v}_j^T \mathbf{x}) = \sum_{k=1}^n \sigma_k \mathbf{v}_k (\mathbf{u}_k^T \mathbf{b}) \quad (2277)$$

The solution is

$$\mathbf{x} = \sum_{i=1}^n \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i \quad (2278)$$

which is verified by inserting the solution in the expression for the left side, and using that $\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij}$.

B.17 The Rayleigh quotient

Consider the minimization of the function

$$f(\mathbf{x}) = \frac{\mathbf{x}^T \mathbf{P} \mathbf{x}}{\mathbf{x}^T \mathbf{Q} \mathbf{x}} \quad (2279)$$

where \mathbf{P} and \mathbf{Q} are positive definite symmetric matrices, which means that $\mathbf{x}^T \mathbf{P} \mathbf{x} > 0$ and $\mathbf{x}^T \mathbf{Q} \mathbf{x} > 0$ for all $\mathbf{x} \neq 0$.

The condition for optimality is

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \frac{2\mathbf{P}\mathbf{x}(\mathbf{x}^T \mathbf{Q}\mathbf{x}) - \mathbf{Q}\mathbf{x}(\mathbf{x}^T \mathbf{P}\mathbf{x})}{(\mathbf{x}^T \mathbf{Q}\mathbf{x})^2} = \mathbf{0} \quad (2280)$$

which gives

$$\mathbf{P}\mathbf{x} = \frac{\mathbf{x}^T \mathbf{P} \mathbf{x}}{\mathbf{x}^T \mathbf{Q} \mathbf{x}} \mathbf{Q} \mathbf{x} = f(\mathbf{x}) \mathbf{Q} \mathbf{x} \quad (2281)$$

This is the generalized eigenvalue problem

$$\mathbf{P}\mathbf{x} = \lambda \mathbf{Q}\mathbf{x} \quad (2282)$$

where $\lambda = f(\mathbf{x})$. The optimal solution is then $\mathbf{x}_* = k\mathbf{v}_{i^*}$ where $k \neq 0$ is a scaling factor that can be selected freely, and \mathbf{v}_{i^*} is the eigenvector corresponding to the smallest eigenvalue λ_{i^*} . Moreover, the optimal solution is $f(\mathbf{x}_*) = \lambda_{i^*}$.

Comment:

The minimization of $f(\mathbf{x})$ can be reformulated as the minimization of

$$g(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x} \quad (2283)$$

subject to the constraint

$$\mathbf{x}^T \mathbf{Q} \mathbf{x} = 1 \quad (2284)$$

This is verified by noting that

$$f(\mathbf{x}) = \frac{\mathbf{x}^T \mathbf{P} \mathbf{x}}{\mathbf{x}^T \mathbf{Q} \mathbf{x}} = \frac{\mathbf{x}'^T \mathbf{x}'}{\mathbf{x}'^T \mathbf{Q} \mathbf{x}'} \quad (2285)$$

where $\mathbf{x}' = \gamma \mathbf{x}$ and $\gamma \neq 0$. Therefore, \mathbf{x} may be selected so that $\mathbf{x}^T \mathbf{Q} \mathbf{x} = 1$. The new constrained problem can be solved using Lagrange multipliers

$$L(\mathbf{x}) = \mathbf{x}^T \mathbf{P} \mathbf{x} + \lambda(1 - \mathbf{x}^T \mathbf{Q} \mathbf{x}) \quad (2286)$$

and the condition for optimality is

$$\mathbf{P}\mathbf{x} = \lambda \mathbf{Q}\mathbf{x} \quad (2287)$$

which is a generalized eigenvalue problem. \square

B.18 Upper and lower triangular matrices

A lower triangular matrix $\mathbf{L} \in \mathbb{R}^{n \times n}$ is simply a matrix with nonzero elements only on the diagonal and below the diagonal. An upper triangular matrix $\mathbf{U} \in \mathbb{R}^{n \times n}$ is a matrix with nonzero elements only on the diagonal and above the diagonal. This means that

$$\mathbf{L} = \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{22} & \dots & l_{nn} \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix} \quad (2288)$$

Because of the special structure of such triangular matrices it is straightforward and efficient to solve equations of the form $\mathbf{L}\mathbf{x} = \mathbf{b}$ and $\mathbf{U}\mathbf{x} = \mathbf{b}$. In the case of the lower triangular matrix the system of equations can be written

$$l_{11}x_1 = b_1 \quad (2289)$$

$$l_{21}x_1 + l_{22}x_2 = b_2 \quad (2290)$$

$$\vdots \quad (2291)$$

$$l_{n1}x_1 + l_{n2}x_2 + \dots + l_{nn}x_n = b_n \quad (2292)$$

The solution is then computed as

$$x_1 = b_1/l_{11} \quad (2293)$$

$$x_2 = (b_2 - l_{21}x_1)/l_{22} \quad (2294)$$

$$\vdots \quad (2295)$$

$$x_n = \left(b_n - \sum_{j=1}^{n-1} l_{nj}x_j \right) / l_{nn} \quad (2296)$$

This procedure is called a forward substitution, and the solution for \mathbf{x}_i is

$$x_i = \left(b_i - \sum_{j=1}^{i-1} l_{ij}x_j \right) / l_{ii} \quad (2297)$$

The computation of the equation $\mathbf{U}\mathbf{x} = \mathbf{b}$ is done in a similar way by starting from the last equation. This procedure is called back substitution, and the solution for \mathbf{x}_i is then

$$x_i = \left(b_i - \sum_{j=i+1}^n u_{ij}x_j \right) / u_{ii} \quad (2298)$$

B.19 The Cholesky decomposition

The Cholesky decomposition of a symmetric positive definite matrix is presented and analyzed in [27]. Let the matrix $\mathbf{A} = \mathbf{A}^T \in \mathbb{R}^{n \times n}$ be symmetric and positive definite. Then there exists a unique lower triangular matrix \mathbf{G} with positive entries on the diagonal so that

$$\mathbf{A} = \mathbf{G}\mathbf{G}^T \quad (2299)$$

This is useful in applications. As an example, the equation $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be solved by solving the two equations

$$\mathbf{G}\mathbf{y} = \mathbf{b}, \quad \mathbf{G}^T\mathbf{x} = \mathbf{y} \quad (2300)$$

which is very efficient since $\mathbf{G}\mathbf{y} = \mathbf{b}$ is solved by forward substitution and $\mathbf{G}^T\mathbf{x} = \mathbf{y}$ is solved by back substitution. It is seen that the two equations imply that $\mathbf{G}\mathbf{G}^T\mathbf{x} = \mathbf{b}$. The Cholesky decomposition is available as a library function in MATLAB, C++ and Python.

Example The MATLAB function $\mathbf{U} = \text{chol}(\mathbf{A})$ computes the upper triangular matrix \mathbf{U} in the decomposition $\mathbf{A} = \mathbf{U}^T\mathbf{U}$. This means that $\mathbf{G} = \mathbf{U}^T$. \square

B.20 QR decomposition

An important decomposition used in linear algebra is the QR decomposition. Consider a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ where $m \geq n$. Then the QR decomposition is

$$\mathbf{A} = \mathbf{Q}\mathbf{R} \quad (2301)$$

where $\mathbf{Q} \in \mathbb{R}^{m \times m}$ is an orthogonal matrix and $\mathbf{R} \in \mathbb{R}^{m \times n}$ is the upper triangular matrix

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0}_{m-n,n} \end{bmatrix} \quad (2302)$$

where $\mathbf{R}_1 \in \mathbb{R}^{n \times n}$ is upper triangular. The orthogonal matrix \mathbf{Q} satisfies $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}$.

The QR decomposition may be used in the least squares solution of the equation $\mathbf{A}\mathbf{x} = \mathbf{b}$ where \mathbf{A} is full rank. Premultiplication of the equation by \mathbf{Q}^T gives for the left side

$$\mathbf{Q}^T\mathbf{A}\mathbf{x} = \mathbf{Q}^T\mathbf{Q}\mathbf{R}\mathbf{x} = \mathbf{R}\mathbf{x} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0}_{m-n,n} \end{bmatrix} \mathbf{x} \quad (2303)$$

The right side is written

$$\mathbf{Q}^T\mathbf{b} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix} \quad (2304)$$

where $\mathbf{c} \in \mathbb{R}^n$ and $\mathbf{d} \in \mathbb{R}^{m-n}$. This gives

$$\begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0}_{m-n,n} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix} \quad (2305)$$

The least-squares norm of the error is

$$\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 = \|\mathbf{Q}^T\mathbf{A}\mathbf{x} - \mathbf{Q}^T\mathbf{b}\|^2 = \|\mathbf{R}_1\mathbf{x} - \mathbf{c}\|^2 + \|\mathbf{d}\|^2 \quad (2306)$$

The minimum of the norm $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$ is found with the solution \mathbf{x}_{LS} satisfying $\mathbf{R}_1\mathbf{x}_{LS} - \mathbf{c} = \mathbf{0}$, which minimizes the term $\|\mathbf{R}_1\mathbf{x} - \mathbf{c}\|^2$ to zero, while the term $\|\mathbf{d}\|^2$ cannot be influenced by the solution of \mathbf{x} .

The least-squares solution is therefore found by first computing the QR decomposition $\mathbf{A} = \mathbf{Q}\mathbf{R}$, then computing $\mathbf{c} = \mathbf{Q}^T\mathbf{b}$, and finally solving $\mathbf{R}_1\mathbf{x}_{LS} = \mathbf{c}$ by back substitution.

Example Consider the case $n = m = 3$. The \mathbf{Q} is a 3×3 orthogonal matrix which satisfies $\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$. This means that \mathbf{Q} is a rotation matrix if $\det \mathbf{Q} = 1$ and a reflection matrix if $\det \mathbf{Q} = -1$. The matrix $\mathbf{R} = \mathbf{R}_1$ is then an upper triangular matrix with dimension 3×3 . The notation may cause confusion as we normally use denote a rotation matrix by \mathbf{R} . \square

B.21 Cross product relations

The following relations are found in [20].

$$\mathbf{a}^\times \mathbf{a} = \mathbf{0} \quad (2307)$$

$$\mathbf{a}^\times \mathbf{b} = -\mathbf{b}^\times \mathbf{a} \quad (2308)$$

$$\mathbf{a}^\times \mathbf{b}^\times = \mathbf{b}\mathbf{a}^T - (\mathbf{a} \cdot \mathbf{b})\mathbf{I} \quad (2309)$$

$$\mathbf{a}^\times (\mathbf{b}^\times \mathbf{c}) = (\mathbf{a} \cdot \mathbf{c})\mathbf{b} - (\mathbf{a} \cdot \mathbf{b})\mathbf{c} \quad (2310)$$

$$(\mathbf{a}^\times \mathbf{b})^\times = \mathbf{b}\mathbf{a}^T - \mathbf{a}\mathbf{b}^T \quad (2311)$$

$$(\mathbf{a}^\times \mathbf{b})^\times = \mathbf{a}^\times \mathbf{b} - \mathbf{b}^\times \mathbf{a} \quad (2312)$$

In addition, these relations are found in [26].

$$(\mathbf{K}\mathbf{a})^\times = \det(\mathbf{K})\mathbf{K}^{-T}\mathbf{a}^\times \mathbf{K}^{-1} \quad (2313)$$

$$(\mathbf{K}\mathbf{a})^\times (\mathbf{K}\mathbf{b}) = \det(\mathbf{K})\mathbf{K}^{-T}(\mathbf{a}^\times \mathbf{b}) \quad (2314)$$

$$((\mathbf{K}\mathbf{a})^\times (\mathbf{K}\mathbf{b}))^\times = \mathbf{K}(\mathbf{a}^\times \mathbf{b})^\times \mathbf{K}^T \quad (2315)$$

$$\mathbf{a}^\times \mathbf{K} + \mathbf{K}^T \mathbf{a}^\times = \text{tr}(\mathbf{K})\mathbf{a}^\times - (\mathbf{K}\mathbf{a})^\times \quad (2316)$$

If $\mathbf{K} = \mathbf{R}$ where \mathbf{R} is a rotation matrix, then $\det(\mathbf{R}) = 1$ and $\mathbf{R}^{-T} = \mathbf{R}$, and this is simplified to

$$(\mathbf{R}\mathbf{a})^\times = \mathbf{R}\mathbf{a}^\times \mathbf{R}^{-1} \quad (2317)$$

$$(\mathbf{R}\mathbf{a})^\times (\mathbf{R}\mathbf{b}) = \mathbf{R}(\mathbf{a}^\times \mathbf{b}) \quad (2318)$$

$$((\mathbf{R}\mathbf{a})^\times (\mathbf{R}\mathbf{b}))^\times = \mathbf{R}(\mathbf{a}^\times \mathbf{b})^\times \mathbf{R}^T \quad (2319)$$

$$\mathbf{a}^\times \mathbf{R} + \mathbf{R}^T \mathbf{a}^\times = \text{tr}(\mathbf{R})\mathbf{a}^\times - (\mathbf{R}\mathbf{a})^\times \quad (2320)$$

References

- [1] R. J. Adcock. A problem in least squares. *The Analyst*, 5(2):53–54, 1878.
- [2] A. Al-Sharadqah and N. Chernov. Error analysis for circle fitting algorithms. *Electronic Journal of Statistics*, 3:886–911, 2009.
- [3] P. Antonante, V. Tzoumas, H. Yang, and L. Carlone. Outlier-robust estimation: Hardness, minimally tuned algorithms, and applications. *IEEE Transactions on Robotics (T-RO)*, 2021.
- [4] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares filtering of two 3-D point sets. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-9(5):698 – 700, 1987.

- [5] H. Badino, D. Huber, and Y. Park. Fast and accurate computation of surface normals from range images. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3084–3091. IEEE, 2011.
- [6] A. Bartoli and P. Sturm. The 3D line motion matrix and alignment of line reconstructions. *International Journal of Computer Vision*, 57(3):159–178, 2004.
- [7] A. Bartoli and P. Sturm. Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. *Computer Vision and Image Understanding*, 100(3):416–441, 2005.
- [8] P. J. Besl and N. D. McCay. A method for registration of 3-D shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.
- [9] M. J. Black and A. Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *Intl. J. of Computer Vision*, 19(1):57–91, 1996.
- [10] A. Blake and A. Zisserman. *Visual Reconstruction*. The MIT Press, 09 1987.
- [11] W. M. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry*. Academic Press, 2nd edition, 2002.
- [12] J.-Y. Bouguet. *Visual methods for three-dimensional modeling*. PhD thesis, California Institute of Technology, 1999.
- [13] F. Bullo and R. M. Murray. Proportional Derivative (PD) Control on the Euclidean Group. CDS Technical Report 95-010, California Institute of Technology, 1995.
- [14] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, Apr. 1992.
- [15] N. Chernov and C. Lesort. Least squares fitting of circles. *Journal of Mathematical Imaging and Vision*, 23:239–252, 2005.
- [16] G. S. Chirikjian. *Stochastic Models, Information Theory, and Lie Groups, Volume 2*. Birkhäuser, 2012.
- [17] O. Chum, T. Pajdla, and P. Sturm. The geometric error for homographies. *Comput. Vis. Image Underst.*, 97(1):86–102, Jan. 2005.
- [18] K. Daniilidis. Hand-eye calibration using dual quaternions. *International Journal of Robotics Research*, 18(3):286–298, Mar. 1999.
- [19] O. Egeland and J.-M. Godhavn. Passivity-based adaptive attitude control of a rigid spacecraft. *IEEE Transactions on Automatic Control*, 39(4):842–845, 1994.
- [20] O. Egeland and J. T. Gravdahl. *Modeling and Simulation for Automatic Control*. Marine Cybernetics, 2002.
- [21] O. Faugeras. Stratification of three-dimensional vision: projective, affine, and metric representations. *Journal of the Optical Society of America A*, 12:465–484, 1995.
- [22] O. Faugeras and L. Roberts. What can two images tell us about a third one? *Int. J. of Computer Vision*, 18(1):5–19, 1996.

- [23] L. Ferraz, X. Binefa, and F. Moreno-Noguer. Very fast solution to the pnp problem with algebraic outlier rejection. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 501–508, 2014.
- [24] 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 ACM*, 24(6):381–395, 1981.
- [25] A. W. Fitzgibbon. Robust registration of 2D and 3D point sets. In *British Machine Vision Conference*, pages 662–670, 2001.
- [26] G. Gallego and A. Yezzi. A compact formula for the derivative of a 3-D rotation in exponential coordinates. *Journal of Mathematical Imaging and Vision*, 3:378–384, 2015.
- [27] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.
- [28] B. C. Hall. *Lie Groups, Lie Algebras, and Representations. An Elementary Introduction*. Graduate Texts in Mathematics. Springer, Berlin, Heidelberg, New York, 2003.
- [29] R. Hartley, J. Trumpf, Y. Dai, and H. Li. Rotation averaging. *International Journal of Computer Vision*, 103(3):267–305, 2013.
- [30] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2nd edition, 2004.
- [31] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph.*, 26(2):71–78, July 1992.
- [32] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4):629–642, 1987.
- [33] A. Iserles, H. Munthe-Kaas, S. Nørsett, and A. Zanna. Lie-group methods. *Acta Numerica*, 2005.
- [34] H. Karcher. Riemannian center of mass and mollifier smoothing. *Communications of Pure and Applied Mathematics*, 30:509–541, 1977.
- [35] I. Kåsa. A curve fitting procedure and its error analysis. *IEEE Trans. Inst. Meas.*, 25(1):8–14, 1976.
- [36] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*, 81(5):155–166, 2009.
- [37] P. Y. Li and R. Horowitz. Passive velocity field control (PVFC). Part I. Geometry and robustness. *IEEE Trans. Automat. Contr.*, 46(9):1346–1359, 2001.
- [38] D. G. Luenberger. *Optimization by vector space methods*. John Wiley & Sons, 1969.
- [39] J. Luh, M. Walker, and R. Paul. Resolved-acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, 25(3):468 – 474, 1980.
- [40] K. Lynch and F. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge Univeristy Press, 2017.

- [41] Y. Ma, S. Soatto, J. Košecká, and S. S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer Verlag, 2003.
- [42] J. H. Manton. A globally convergent numerical algorithm for computing the centre of mass on compact Lie groups. In *Proceedings IEEE ICARCV*, pages 2211–2216. IEEE, 2004.
- [43] J. E. Marsden and T. S. Ratiu. *Introduction to mechanics and symmetry*. Springer, 1994.
- [44] J. M. McCarthy and G. S. Soh. *Geometric design of linkages*. Springer Verlag, 2011.
- [45] C. D. Meyer. *Matrix analysis and applied linear algebra*. SIAM, 2010.
- [46] M. Moakher. Means and averaging in the group of rotations. *SIAM Journal on Matrix Analysis and Applications*, 24(1):1–16, 2002.
- [47] F. Moreno-Noguer, V. Lepetit, and P. Fua. Accurate non-iterative $o(n)$ solution to the PnP problem. In *Proceedings of the 2007 IEEE 11th International Conference on Computer Vision*, 2007.
- [48] K. P. Murphy. *Probabilistic Machine Learning: An Introduction*. The MIT Press, 2022.
- [49] R. M. Murray, S. S. Sastry, and L. Zexiang. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1994.
- [50] J. Nocedal and S. J. Wright. *Numerical Optimization*. World Scientific, 2nd edition, 2006.
- [51] F. C. Park. Distance metrics on the rigid-body motions with applications to mechanism design. *J. Mechanical Design*, 117(1):48–54, 1995.
- [52] F. C. Park and B. J. Martin. Robot sensor calibration: Solving $AX = XB$ on the Euclidean group. *IEEE Trans. Robotics and Automation*, 10(5):717–721, 1994.
- [53] P. Petersen. *Riemannian Geometry*. Springer-Verlag, Berlin, 2nd edition, 1998.
- [54] H. Pottmann and J. Wallner. *Computational Line Geometry*. Springer-Verlag, Berlin, 2001.
- [55] V. Pratt. Direct least-squares fitting of algebraic surfaces. *Computer Graphics*, 21(4):145–152, 1987.
- [56] B. Přibyl, P. Zemčík, and M. Čadík. Camera pose estimation from lines using plücker coordinates. In *Proceedings of the British Machine Vision Conference (BMVC 2015)*, pages 1–12. The British Machine Vision Association and Society for Pattern, 2015.
- [57] B. Přibyl, P. Zemčík, and M. Čadík. Absolute pose estimation from line correspondences using direct linear transformation. *Computer Vision and Image Understanding*, 161(4):130–144, 2017.
- [58] J. Rothenflue, N. Gordillo-Herrejon, and R. S. Aygün. A comparison between camera calibration software toolboxes. In *Proceedings of the IEEE International Conference on Computational Science and Computational Intelligence*, pages 772–776. IEEE, 2016.
- [59] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, June 2001.

- [60] P. D. Sampson. Fitting conic sections to very scattered data: An iterative refinement of the Bookstein algorithm. *Computer Graphics and Image Processing*, 18(1):97–108, 1982.
- [61] J. G. Semple and G. T. Kneebone. *Algebraic projective geometry*. Oxford Classic Series. Clarendon Press, Oxford, 1952.
- [62] S. W. Shepperd. Quaternion from rotation matrix. *J. Guidance and Control*, 1(3):223–224, 1978.
- [63] Y. Shiu and S. Ahmad. Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form $AX = XB$. *IEEE Trans. Robotics and Automation*, 5(1):16–29, 1989.
- [64] K. Shoemake. Animating rotation with quaternion curves. In *Proceedings of SIGGRAPH*, 1985.
- [65] K. Shoemake. Plücker coordinate tutorial. *Ray Tracing News*, 11(1), 1998.
- [66] M. D. Shuster and S. D. Oh. Three-axis attitude determination from vector observations. *Journal of Guidance and Control*, 4(1):70–77, 1981.
- [67] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer, 2008.
- [68] J. W. Simpson-Porco and F. Bullo. Contraction theory on Riemannian manifolds. *Systems & Control Letters*, 65:74–80, 2014.
- [69] G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 5th edition, 2016.
- [70] A. Sveier, A. M. Sjøberg, and O. Egeland. Applied Runge–Kutta–Munthe-Kaas integration for the quaternion kinematics. *Journal of Guidance Control and Dynamics*, 42(12):2747–2754, 2019.
- [71] G. Taubin. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138, 1991.
- [72] C. J. Taylor and D. J. Kriegman. Minimization on the Lie Group $SO(3)$ and Related Manifolds. Technical Report 9405, Yale University, 1994.
- [73] D. Tazartes. An historical perspective on inertial navigation systems. In *2014 International Symposium on Inertial Sensors and Systems (ISISS)*, 2014.
- [74] R. K. Tsai and R. K. Lenz. A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *IEEE Trans. Robotics and Automation*, 5(3):345–358, 1989.
- [75] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.
- [76] B. L. Van der Waerden. Hamilton’s discovery of quaternions. *Mathematics Magazine*, 49(5):227–234, 1976.
- [77] G. Wahba. A least squares estimate of satellite attitude. *SIAM Review*, 7(3):409, 1965.

- [78] J. T. Wen and K. Kreutz-Delgado. The attitude control problem. *IEEE Transactions on Automatic Control*, 36(10):1148–1162, 1991.
- [79] J. Weng, T. S. Huang, and T. S. Huang. *Motion and Structure from Image Sequences*. Springer-Verlag, Berlin, 1993.
- [80] P. M. Wensing and J.-J. E. Slotine. Beyond convexity – contraction and global convergence of gradient descent. arXiv:1806.06655 [math.OC], 2018.
- [81] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone. Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection. *IEEE Robotics and Automation Letters (RA-L)*, 2020.
- [82] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.
- [83] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019.