Point cloud reconstruction from image sequences — Calibrated versus uncalibrated

Samuel Carlsson

8th July 2005
Master's Thesis in Computing Science, 20 credits
Supervisor at CS-UmU: Niclas Börlin
Examiner: Per Lindström

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Abstract

Methods for 3D reconstruction from image squences fall into two classes: Methods using camera knowledge that requires a separate calibration and methods using self-calibration that do not need any information in addition to the images.

Two versions of a reconstruction algorithms has been implemented. One that requires a separate calibration and one that does not. A series of tests were conducted on synthetic data and on real images.

The execution speed of the two methods are similar. However, the results show that the calibrated case generates more point matches since the lens distortion is known. The advantages of the calibrated method is that it does not require self-calibration implementation and it generates more point matches and hence object points, while the uncalibrated method is more flexible, requires no calibration, and allows varible focal length.

Contents

List of Figures

1 Introduction

This paper tries to investigate the effects of using calibrated cameras rather than uncalibrated ones when doing structure and motion recovery.

1.1 Background and purpose

The fields of photogrammetry and computer vision develops rapidly and has now come to the stage where an image sequence can in principle be automatically turned into a 3D point cloud. Lately new algorithms using so called self calibration allows to to this reconstruction without any knowledge about the camera used.

How does those self-calibration algorithms compare to the older algorithms that depend heavily on knowledge about the camera used? Are there situations where camera knowledge have an impact? This paper tries to investigate this matter in terms of precision, computation speed and matching quality.

1.2 Problem description

Implement an algorithm to reconstruct cameras and object point from image sequences in two versions, one that requires information about the camera calibration and one that does not. Furthermore, compare the two algorithms with respect to execution speed and matching quality.

1.3 Related work

Some research groups are working on systems for automatic reconstruction from image sequences with an uncalibrated camera. In a tutorial [14] Pollefeys presents a for structure and motion recovery (SMR) method similar to the one described in this report. Another SMR system is described in [4]. A method for self-calibration with variable focal length is found in [15]. The VANGUARD project[18] that aims to create a fully automated system for converting an image sequence of an object into a model file.

1.4 Prerequisties

The reader is is assumed to be familiar with linear algebra. Although the focus is not on image processing, some basic knowledge about image filtering is needed.

2 Theory

In this chapter various methods and mathematics used will be explained and described.

2.1 Projective geometry

Projective geometry plays a central role in 3D reconstruction algorithms and will be covered first. There are lots of good reading on projective geometry, e.g. [11], so only a brief description is given here.

Much of the work in this paper is based on the projective geometry. It differs from the euclidean geometry in that the coordinates of a projective space is extended with an extra variable w. So that a point in $\mathbb{P}^n$ is expressed as an point in $\mathbb{R}^{n+1}$. For example a point in two-dimensional projective space, $\mathbb{P}^2$, is defined as $\mathbf{x} = (x, y, w)^T$. The corresponding point in euclidean space is $\left(\frac{x}{w}, \frac{y}{w}\right)^T$. Coordinates in projective space are called *homogeneous coordinates*.

There are of course many values for x, y and w that corresponds to a particular point $\mathbf{x}$ in $\mathbb{P}^2$. In fact all points $k\mathbf{x}$ maps to the same point in $\mathbb{P}^2$, for any non-zero constant k, since

$$\left(\frac{x}{w}, \frac{y}{w}\right)^T = \left(\frac{kx}{kw}, \frac{ky}{kw}\right)^T. \tag{1}$$

The strength of this is we can still express linear relations with the powerful vector/matrix notation, while there is also the possibility to express relations including a division very neatly. The most important aspect of this is the ability to express the projection transformation as a left side matrix multiplication.

A homogeneous point with coordinates $\mathbf{x}^T = [x, y, 0]$ do not represent any finite point in $\mathbb{R}^2$. Trying to convert it into a inhomogeneous vector would give us $\left(\frac{x}{0}, \frac{y}{0}\right)^T$, which makes no sense. These points are called *ideal points*.

2.1.1 Lines in $\mathbb{P}^2$

As known from elementary geometry a line in $\mathbb{R}^2$ can be expressed as $Ax + By + C = 0$. However this vector is not unique for a particular line. The whole equation can be multiplied by a scalar and yield the same line since $kAx + kBy + kC = 0$ still holds for any non-zero k. A line can thus be represented as a homogeneous vector $\mathbf{l}^T = [A, B, C]$. Now we can write the equation as $\mathbf{l}^T\mathbf{x} = 0$ for a point $\mathbf{x}$ on the line $\mathbf{l}$.

But since the vector inner product is commutative we can also write $\mathbf{x}^T\mathbf{l} = 0$. The line and the point have been exchanged. Since both are represented as a homogeneous vector, a equation involving a vector in $\mathbb{P}^2$ can be seen either as a point or as a line. So for any theorem regarding points in $\mathbb{P}^2$ there is a *dual* theorem where the points are treated as lines, and vice versa. This is stated in the *duality principle*.

All ideal points lie on a common line, the *line at infinity*. This line is denoted $\mathbf{l}_\infty = [0, 0, 1]^T$. It can be verified easily. Any ideal point $\mathbf{x} = [x, y, 0]^T$ will lie on this line if $\mathbf{l}_\infty^T\mathbf{x} = 0$, which is the case.

Two lines $\mathbf{l}_1$ and $\mathbf{l}_2$ intersects in a single point. This point can be computed as $\mathbf{x} = \mathbf{l}_1 \times \mathbf{l}_2$. One can verify that this point indeed lies on both lines by using the triple scalar product identity $\mathbf{l}_1^T(\mathbf{l}_1 \times \mathbf{l}_2) = \mathbf{l}_2^T(\mathbf{l}_2 \times \mathbf{l}_1) = 0$.

A dual to this result is the equation for finding the line joining two points. The line $\mathbf{l}$ passing through two point $\mathbf{x}_1$ and $\mathbf{x}_2$ is found by $\mathbf{l} = \mathbf{x}_1 \times \mathbf{x}_2$. This can be verified in a similar way.

2.1.2 Transformations in the projective space

The most general transformation in projective space is called a homography. It is a $n+1 \times n+1$ matrix in $\mathbb{P}^n$ with no restriction on the elements in the matrix. A homography $\mathtt{H}$ transforms a point $\mathbf{x}$ into a new point $\mathbf{x}'$ like

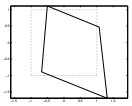
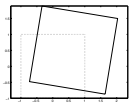
$$\mathbf{x}' = \mathtt{H}\mathbf{x} \tag{2}$$

where $\mathtt{H}$ is any invertible matrix.

Overall scale on the matrix has no effect other than changing the size of the $\mathbf{x}$ vector and all vectors $k\mathbf{x}$ represents the same point in $\mathbb{P}^n$. So the degrees of freedom for a homography is $(n+1)^2 - 1$, e.g. a homography in $\mathbb{P}^3$ has 15 DOF.

A more restricted type of transformation called the *similarity transform* allows transformations consisting of a *overall scale*[1], a rotation, and a translation. It has $\binom{n}{2} + n + 1$ DOF, for the rotation, translation, and overall scale respectively. So for example a similarity transform in $\mathbb{P}^3$ has 7 DOF.

Table 1: The differences between the projective transform and the similarity transform summarized.

Transform	Projective	Similarity
Degrees of freedom	$(n+1)^2 - 1$	$\binom{n}{2} + n + 1$
Matrix	Any invertible $\mathtt{H}$	$\begin{bmatrix} r\mathtt{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$
Distortion		
Some invariant properties	Cross-ratio, intersections.	Angles, length ratios.

In table 1 the properties of those two types of transformations are summarized.

Throughout the paper the term "up to an unknown homography" (projective transform) refers to the fact that entities, i.e. point and cameras, differ from their true values by an unknown projective transformation.

Although not entirely correct, the terms euclidean and similarity are sometimes used as synonyms in the literature. In this report the term similarity is used, e.g. a "similarity reconstruction" is a reconstruction known up to an unknown similarity transform.

[1] Also called a uniform scale indicating that it is equal in all directions

2.2 The camera model

A camera model is used to model the way a regular camera projects a motif into a picture. It is defined as an operator that projects points from the 3D world into points in the 2D image plane. The camera parameters are grouped into internal and external parameters. The internal parameters describes the properties of the actual camera taking the picture such as focal length. External parameters describes how the camera is oriented in object space.

2.2.1 Internal parameters

Assume that the camera center is located at the origin of a world coordinate frame pointing in the Z-direction we want to find where a 3D-point projects down onto the image.

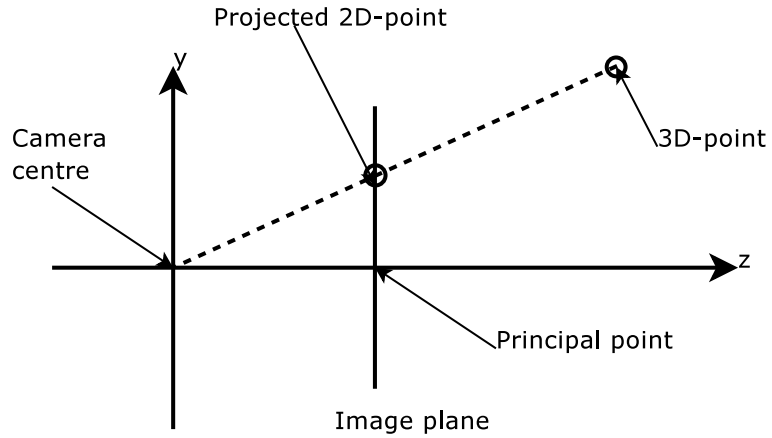


Figure 1: Side view of the camera. How a 3D point is projected onto the image plane. In a real camera the image plane is behind the camera center, but in this picture it is shown in front.

Assume the image plane is located at $Z = f$. The projection of a 3D-point is where a line between the camera center and the 3D-point intersects the image plane $Z = f$ as illustrated in figure 1 and 2.

The line from the camera center perpendicular to the image plane is called the *principal axis*. The point in the image plane where the principal axis passes through is called the principal point.

By using length ratios from similar triangles we can see that the projection of a 3D point $[x, y, z]^T$ can be expressed as $[fx/z, fy/z]^T$. If the 3D point is represented as a homogeneous 4-vector the mapping can be written as as

$$\begin{bmatrix} fx \\ fy \\ Z \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \tag{3}$$

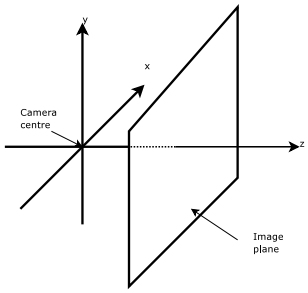


Figure 2: Overview of the camera projection.

or just $\mathbf{x} = \mathtt{P}\mathbf{X}$ where $\mathbf{X}$ is the 3D point and $\mathbf{x}$ is the projected 2D-point in the image. The matrix $\mathtt{P}$ is called a camera projection matrix.

The coordinate frame in the image plane do usually *not* coincide with the principal point. The principal point is usually close to the center of the image, while the typical coordinate frame for an image is left-handed with the origin in the top-left corner. The principal point is located at $(p_x, p_y)^T$ and in the image coordinate frame the projected point is $(fx/z + p_z, fy/z + p_y)^T$. By using homogeneous coordinates we get

$$\begin{bmatrix} fx + Zp_x \\ fy + Zp_y \\ Z \end{bmatrix} = \begin{bmatrix} f & & p_x 0 \\ & f & p_y 0 \\ & & 10 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \tag{4}$$

where the leftmost 3-by-3 matrix

$$\mathtt{K} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} \tag{5}$$

is called the *camera calibration matrix*. The camera calibration matrix contain the *internal parameters*.

2.2.2 External parameters

If the camera center is not located at the origin is not aligned with the world coordinate system, we need to rotate and translate points before projecting,

$$\mathtt{R}\tilde{\mathbf{X}} + \mathbf{t}, \tag{6}$$

where $\tilde{\mathbf{X}}$ is the inhomogeneous representation of the 3D-point. The 3-by-3 matrix $\mathtt{R}$ is the rotation which re-orients the camera so that the principal axis is parallel to the Z-axis, and the translation vector $\mathbf{t}$ is a translation that moves the camera center to the origin.

Rotation and translation and then projection can be expressed in one equation,

$$\mathbf{x} = \mathtt{K}[\mathtt{R} \mid \mathbf{t}]\mathbf{X}. \tag{7}$$

The matrix $\mathtt{K}[\mathtt{R} \mid \mathbf{t}]$ is identified as $\mathtt{P}$.

The calibration matrix can be extended to look like

$$\mathtt{K} = \begin{bmatrix} m_x f & s & p_x \\ & m_y f & p_y \\ & & 1 \end{bmatrix} \tag{8}$$

where s is the skew, it tells us how far the pixel axes are from being perpendicular. This is often zero or very close to zero. m_x and m_y are the resolutions in the x-, and y-directions.

The calibration matrix has 5 degrees of freedom. Together with the 3 degrees of freedom for the rotation and 3 for the translation it leave us with a total of 11 degrees of freedom for the camera projection matrix $\mathtt{P}$.

2.2.3 Lens distortion

All cameras have a lens that focuses the incoming light on the detector. Another effect of the lens is that it distorts the image because the incoming light is bent differently in different parts of the image [1, 7]. This results in straight lines in object space will appear as slightly curved lines in the image. An example of how this typically look is shown in figure 3 where a synthetic image is applied a simulated lens distortion.

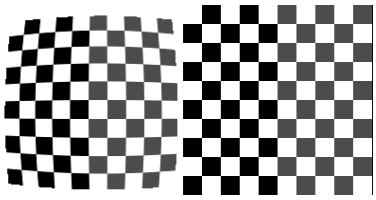


Figure 3: A checkerboard image has gone through a simulated lens distortion. The principal point is in the middle of the image. The distorted image is to the left.

The effect of lenses has been of interest for a long period of time. As early as 1919 Conrady studied decentered lenses [2].

The distortion shown here is called a *radial* distortion. One point is not distorted at all. This point is called the *center of distortion*. The amount each point is offset depends on the distance to the center of distortion. There are other types of distortions as well, but those gives rise to very subtle changes in the image and can in our case be discarded. The radial distortion however must be dealt with in some way if meaningful measures are to be made from the image.

The radial distortion is can be modeled as

$$\mathbf{x}_d = \mathbf{c} + L(r)(\mathbf{x} - \mathbf{c}) \tag{9}$$

where $\mathbf{x}_d$ is the position of the distorted point and $\mathbf{x}$ is the original point [11]. The function $L(\cdot)$ depends only on the distance to the center of distortion, $\mathbf{c}$. This distance is $r = ||\mathbf{x} - \mathbf{c}||$. The effect of this distortion can be undone by applying a similar function

on $\mathbf{x}_d$, which is the actual point measured in the image.

The function $L(\cdot)$ can be approximated by a Taylor expansion around $r = 0$. In [11] the function $L(\cdot)$ is assumed to be on the form

$$L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \dots \tag{10}$$

The first κ-values are often dominant, therefor its common to only use the two first, κ_1 and κ_2.

The coefficients of the L-function can be computed in a variety of ways. One is to measure lines in the image that is known to be straight in object space. Then compute the κ-values that best turns the curved lines in the image into straight ones. This is done by numerically minimizing a cost function that describes the "straightness" of the corrected lines. This is extra useful when the motif for the images are buildings since they usually contain lots of straight lines.

2.3 Estimating the camera

If the coordinates are known for some points in object space and it is known where they appear in an image the camera matrix P can be computed [11]. Each such correspondence gives

$$\mathbf{x} = \mathtt{P}\mathbf{X}. \tag{11}$$

The equality is defined up to an unknown scale, i.e the two vectors need not be equal, but need to have the same direction.

The camera matrix has 11 degrees of freedom and each point correspondence gives rise to two independent equations. So we need at least 6 correspondences to compute P in the general case.

2.3.1 The DLT algorithm

Equation 11 is equivalent with the expression

$$\mathbf{x} \times \mathtt{P}\mathbf{X} = \mathbf{0} \tag{12}$$

The right hand side is zero so the non-zero scale factor have no effect.

By denoting the j-th row in P by $\mathbf{p}^{jT}$ we can write the matrix multiplication as

$$\mathtt{P}\mathbf{X}_i = \begin{pmatrix} \mathbf{p}^{1T}\mathbf{x}_i \\ \mathbf{p}^{2T}\mathbf{x}_i \\ \mathbf{p}^{3T}\mathbf{x}_i \end{pmatrix} \tag{13}$$

Using the properties of the cross product we can write equation 12 like this

$$\mathbf{x}_i \times \mathtt{P}\mathbf{X}_i = \begin{pmatrix} y\mathbf{p}^{3T} - w\mathbf{p}^{2T} \\ w\mathbf{p}^{1T} - x\mathbf{p}^{3T} \\ x\mathbf{p}^{2T} - y\mathbf{p}^{1T} \end{pmatrix} \tag{14}$$

where $\mathbf{x} = (x, y, w)^T$. This matrix can be written as

$$\begin{pmatrix} \mathbf{0}^T & -w_i\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ w_i\mathbf{X}_i^T & \mathbf{0}^T & -x_i\mathbf{X}_i^T \\ -y_i\mathbf{X}_i^T & x_i\mathbf{X}_i^T & \mathbf{0}^T \end{pmatrix} \begin{pmatrix} \mathbf{p}^1 \\ \mathbf{p}^2 \\ \mathbf{p}^3 \end{pmatrix} = \mathbf{0}. \tag{15}$$

The last step can be verified by performing the multiplication. This matrix has rank two so only two rows are linearly independent. For example the last row can be removed. This is what we know holds for one point correspondence. The right hand vector containing the entries in P is to be multiplied with *all* points from the left, so all equations can be stacked on top of each other. The equation

$$\begin{pmatrix} \mathbf{0}^T & -w_1\mathbf{X}_1^T & y_1\mathbf{X}_1^T \\ w_1\mathbf{X}_1^T & \mathbf{0}^T & -x_1\mathbf{X}_1^T \\ \mathbf{0}^T & -w_2\mathbf{X}_2^T & y_2\mathbf{X}_2^T \\ w_2\mathbf{X}_2^T & \mathbf{0}^T & -x_2\mathbf{X}_2^T \\ \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \mathbf{p}^1 \\ \mathbf{p}^2 \\ \mathbf{p}^3 \end{pmatrix} = \mathbf{0} \tag{16}$$

is on the form $\mathtt{A}\mathbf{p} = \mathbf{0}$ where A is a $2n \times 12$ matrix and $\mathbf{p}$ is a vector containing the entries of P, the camera matrix. The vector $\mathbf{p}$ contains the rows of P stacked in a column vector.

$$\mathbf{p}^T = (p_1 \quad p_2 \quad p_3 \quad p_4 \quad p_5 \quad p_6 \quad p_7 \quad p_8 \quad p_9 \quad p_{10} \quad p_{11} \quad p_{12}) \tag{17}$$

$$\mathtt{P} = \begin{pmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{pmatrix} \tag{18}$$

If the matrix A has 11 rows an exact solution can be obtained by finding the right null space of A. In order to get 11 rows one coordinate has to be discarded from the last point. The points will project exactly on their supposed positions in the image.

If A has more than 11 rows and the points are disturbed only a small bit there will be no $\mathbf{p}$ that satisfies $\mathtt{A}\mathbf{p} = \mathbf{0}$ exactly, except $\mathbf{p} = \mathbf{0}$. Thus, we have to find some $\mathbf{p}$ that minimizes some cost function. A cost function that is easy to implement is

$$f = ||\mathtt{A}\mathbf{p}||^2. \tag{19}$$

Since $\mathbf{P}$ and thus $\mathbf{p}$ is only determined up to scale there has to be a restriction on the norm of $\mathbf{p}$, e.g. $||\mathbf{p}|| = 1$. Then the cost function will look like

$$f = \frac{||\mathtt{A}\mathbf{p}||}{||\mathbf{p}||}. \tag{20}$$

The solution to equation 20 may be calculated with the Singular value factorization (see appendix D). Let $\mathtt{A} = \mathtt{U}\mathtt{D}\mathtt{V}^T$. Then $\mathbf{p}$ is the last column of the V matrix.

2.3.2 Cameras with known calibration

If the calibration matrix is known only the external orientation has to be determined. It has six degrees of freedom (3 for orientation and another 3 for translation). Each

point correspondence give rise to two equations, hence three points are needed in order to compute the camera.

The method presented here was first developed by the German mathematician Grunert (1841) and has later been improved by various people. The origin and different variants of the algorithm as well as some interesting stability results are found in [9]. The version presented here is based on [12, Chapter 22.2.4.5].

Suppose we have three points in object space, in this section called A, B and C. To each of those corresponds a image point $\mathbf{x}_i$. The three points in object space form a triangle. The length of the sides of the triangle are denoted a, b, c where a opposites the corner A, etc. The triangle and camera center forms a tetrahedron. The distances from the camera center $\mathbf{c}$ to the object points are called $s_i = ||\mathbf{c} - \mathbf{X}_i||$. Three angles form at the camera center, α, β, γ where α opposites the a side of the triangle. An overview of this configuration is shown in figure 4. The plane formed by the three object points need not be parallel with the image plane.

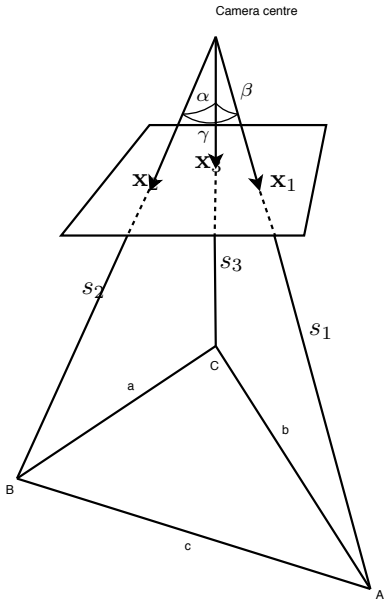


Figure 4: The setup for the three point algorithm.

In a coordinate frame aligned with the camera we form three direction vectors pointing outward from the camera center;

$$\mathbf{m}_i = N \begin{pmatrix} \mathbf{x}_i \\ f \end{pmatrix} \tag{21}$$

where f is the focal length of the camera. All image points can conveniently be multiplied with $\mathtt{K}^{-1}$ and then use $f = 1$ here. The N operator normalizes the 3-vector to unit length.

The following expression relating object points in camera coordinate frame and world coordinate frame holds for all three points.

$$s_i \mathbf{m}_i = \mathtt{R}(\mathbf{x}_i - \mathbf{t}) \qquad i = 1, 2, 3 \tag{22}$$

There are 9 equations and 9 unknown parameters; three in $\mathbf{t}$ and $\mathtt{R}$ respectively and three in s_i. First the distances s_i are determined.

From the law of cosines we know

$$a^2 = s_2^2 + s_3^2 - 2s_2s_3\cos\alpha \tag{23}$$
$$b^2 = s_3^2 + s_1^2 - 2s_3s_1\cos\beta \tag{24}$$
$$c^2 = s_1^2 + s_2^2 - 2s_1s_2\cos\gamma \tag{25}$$

With the substitutions $s_1u = s_2$ and $s_1v = s_3$ we can solve for s_1^2 to get

$$\begin{aligned} s_1^2 &= \frac{a^2}{u^2 + v^2 - 2uv\cos\alpha} \\ &= \frac{b^2}{1 + v^2 - 2v\cos\beta} \\ &= \frac{c^2}{1 + u^2 - 2u\cos\gamma}. \end{aligned} \tag{26}$$

Using those two identities we can either solve for v or u to get a fourth order polynomial. The coefficients for this polynomial are rather long and are not repeated here, but can be found in [16]. They depend on the known values $a, b, c, \alpha, \beta, \gamma$. When v is found, u can be computed from equation26

There can exist up to four solutions for v. The computation hereafter proceeds in parallel with the four values for v finally ending up in four different cameras.

The next step is to extract distances s_1, s_2, s_3. First s_1 is computed from equation 26, the other two distances can be found by inserting s_1, v, and u into the substitutions $s_1u = s_2$ and $s_1v = s_3$.

By using the the vectors $\overrightarrow{AC}$ and $\overrightarrow{AB}$ we can establish a coordinate frame for the three points. The three points form a plane, this coordinate frame can be thought of as a representation of the orientation of that plane. Since we know both the positions of the points in object space and in camera space we have two coordinate frames and the relation between those is the rotation required to rotate the object space points into camera space, which is what is in the projection matrix.

First the vectors $\overrightarrow{AC}$ and $\overrightarrow{AB}$ expressed in in object space coordinates.

$$\mathbf{b}^o = \mathbf{x}_3 - \mathbf{x}_1 \tag{27}$$
$$\mathbf{c}^o = \mathbf{x}_2 - \mathbf{x}_1 \tag{28}$$

and then in camera aligned coordinates

$$\mathbf{b}^c = s_3\mathbf{m}_3 - s_1\mathbf{m}_1 \tag{29}$$
$$\mathbf{c}^c = s_2\mathbf{m}_2 - s_1\mathbf{m}_1. \tag{30}$$

The two coordinate frames can be obtained with cross product so that we are sure to

get orthonogal matrices.

$$\mathtt{R}_o \quad = \quad [N(\mathbf{b}^o) \quad N(\mathbf{b}^o \times \mathbf{c}^o) \quad N(\mathbf{b}^o \times (\mathbf{b}^o \times \mathbf{c}^o))] \,. \tag{31}$$

The coordinate frame for the camera aligned point triple $\mathtt{R}_c$ is acquired in a similar manner using the vectors $\mathbf{b}^c$ and $\mathbf{c}^c$ instead. The wanted rotation can be found using this formula.

$$\mathtt{R} = \mathtt{R}_c \mathtt{R}_o^T. \tag{32}$$

When the rotation is know the camera center can be found from

$$\mathbf{t} = \mathbf{x}_i - \mathtt{R}^T \mathbf{x}_i \tag{33}$$

by using one of the points, e.g. the first. The full camera projection matrix is as usual

$$\mathtt{P} = \mathtt{K} \, [\mathtt{R} \mid \mathbf{t}] \,. \tag{34}$$

2.4 Two view geometry

When two cameras picture the same object some geometries can be identified. The two cameras are called $\mathtt{P}$ and $\mathtt{P}'$ with projection centers $\mathbf{C}$ and $\mathbf{C}'$. The line between the two camera centers are called the *base line*. The intersection of the baseline with one of the image plane is called a epipole. The epipole may or may not be visible in the image. The image produced by the first camera, $\mathtt{P}$, is sometimes referred to as the left image while the other image is called the right image. The situation is depicted in figure 5.

A 2D point in the first image can be back projected into a 3D line. The 2D point corresponds to some 3D point on this line. Since we do not know how deep a point in the image is, the exact 3D position can not be determined. So any 3D point on the line would be projected onto the original 2D point. If this 3D line is pictured in the second image it will project as a 2D line. This line is called an epipolar line. Epipolar lines will always intersect the epipole.

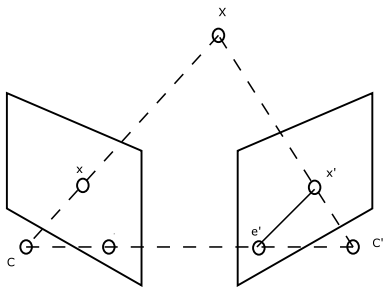


Figure 5: Two cameras captures the same 3D point.

For a particular point in the first image, where is this epipolar line in the second image? The following discussion will try to resolve this question.

A point $\mathbf{x}$ would be back projected by solving for $\mathbf{X}$ in $\mathtt{P}\mathbf{X} = \mathbf{x}$. This holds for all points

on the line, $\mathbf{X}(\lambda)$, parameterized by the scalar λ. The line has the form

$$\mathbf{X}(\lambda) = \mathtt{P}^{+}\mathbf{x} + \lambda\mathbf{C} \tag{35}$$

where $\mathtt{P}^{+}$ us the pseudo-inverse of $\mathtt{P}$ and $\mathbf{C}$. This can be verified easily since by definition we have $\mathtt{P}\mathtt{P}^{+} = \mathtt{I}$ and $\mathtt{P}\mathbf{C} = \mathbf{0}$. Two points on the line is projected by the second camera, $\mathtt{P}'$. The first point is $\mathtt{P}^{+}\mathbf{x}$ for $\lambda = 0$ and the other point is the camera center of the first camera ($\mathtt{P}$). The projection in the other image is $\mathtt{P}'\mathtt{P}^{+}\mathbf{x}$ and $\mathtt{P}'\mathbf{C}$. The line joining these two must be the the epipolar line in the second image ($\mathbf{l}'$) corresponding to the point $\mathbf{x}$ of the first image. The two points are joined with the cross product

$$\mathbf{l}' = \underbrace{[\mathtt{P}'\mathbf{C}]_{\times}\mathtt{P}'\mathtt{P}^{+}}_{\mathtt{F}}\mathbf{x}. \tag{36}$$

This is essentially a multiplication from the left on the point $\mathbf{x}$ by a 3-by-3 matrix. This matrix is called the *fundamental matrix* and is denoted by $\mathtt{F}$. As it name suggest it is an important entity in the field of 3D reconstruction.

Rewriting equation 36 in a way that states more clearly that $\mathtt{F}$ map's a point in one image into the corresponding epipolar line in the other image. Using $\mathtt{F} = [\mathtt{P}'\mathbf{C}]_{\times}\mathtt{P}'\mathtt{P}^{+}$ yields

$$\mathtt{F}\mathbf{x} = \mathbf{l}'. \tag{37}$$

This is however a one-way map since a line in one image do not correspond to a point in the other image. So naturally the $\mathtt{F}$ matrix do not have an inverse, in fact it can be shown that the $\mathtt{F}$ matrix must have rank 2. The one dimensional (right) nullspace is the image coordinate of the epipole so that $\mathtt{F}\mathbf{e} = \mathbf{0}$. This feels natural since the only point in the left image that do not have a unique corresponding epipolar line is the epipole itself.

Figure 6: Left: One of the test images with a few interest points plotted. Right: The corresponding epipolar lines. We can clearly see how the epipolar lines meet in a point a bit off the left edge of the image.

In figure 6 some points in the first image are plotted and the corresponding epipolar lines are plotted in the second image.

The fundamental matrix is a 3-by-3 homogeneous matrix and should have 8 degrees of freedom (nine for the elements in the matrix, subtract one for the overall scale), but there is also the extra requirement that the matrix must be of rank 2. This reduces the DOF by one to 7.

The transpose of the fundamental matrix $\mathtt{F}^T$ maps points in the right image into epipolar lines in the left image so that

$$\mathtt{F}^T\mathbf{x}' = \mathbf{l}. \tag{38}$$

Another way to interpret this is that $\mathtt{F}^T$ is the fundamental matrix corresponding to the camera pair $(\mathtt{P}', \mathtt{P})$, rather than the camera pair $(\mathtt{P}, \mathtt{P}')$ as is the case for $\mathtt{F}$.

2.4.1 Camera extraction

The two cameras can be extracted from the fundamental matrix. Without loss of generality we can assume that one camera is at its canonical position with the camera center at the origo looking down the Z-axis. The other camera can be formed as

$$\begin{aligned} \mathtt{P} &= [\mathtt{I} \mid \mathbf{0}] \\ \mathtt{P}' &= \left[[\mathbf{e}]_\times \mathtt{F} + \mathbf{e}\mathbf{a}^T \mid k\mathbf{e}\right] \end{aligned}$$

where $\mathbf{a}$ and k can by any vector/scalar. The scale is set by k. This is described in detail in [11].

This only produces cameras determined up to a unknown but common homography. If the camera calibration matrix is known, a similarity reconstruction of the cameras can be done, see below.

2.4.2 The essential matrix

The essential matrix is a special case of the fundamental matrix. The difference is where $\mathtt{F}$ relates image points, the essential matrix (often denoted $\mathtt{E}$) relates *normalized image points*. These are points $\hat{\mathbf{x}}$ that relates to the image coordinated like $\mathtt{K}\hat{\mathbf{x}} = \mathbf{x}$. So in a similar manner as $\mathtt{F}$ the essential matrix relates normalized image coordinates as

$$\hat{\mathbf{x}}'^T \mathtt{E} \hat{\mathbf{x}} = 0. \tag{39}$$

Substituting $\mathbf{x}$ by $\mathtt{K}\hat{\mathbf{x}}$ we get

$$\begin{aligned} (\mathtt{K}^{-1}\mathbf{x}')^T \mathtt{E} \mathtt{K}^{-1}\mathbf{x} &= \\ \mathbf{x}'^T \mathtt{K}^{-T} \mathtt{E} \mathtt{K}^{-1}\mathbf{x} &= 0. \end{aligned} \tag{40}$$

This can be viewed as the points related by the essential matrix are images projected by a camera with identity calibration matrix, $\mathtt{K} = \mathtt{I}$. This is because a the normalized image coordinate projects like $\hat{\mathbf{x}} = [\mathtt{R} \mid \mathbf{t}]\mathbf{X}$

Comparing equation 40 with the relation for $\mathtt{F}$ we induce an equation that relates $\mathtt{F}$ and $\mathtt{E}$

$$\mathtt{E} = \mathtt{K}'^T \mathtt{F} \mathtt{K}. \tag{41}$$

2.4.3 Camera extraction

Just as in the case with the fundamental matrix the two cameras can be extracted. The cameras obtained from the fundamental matrix however where only determined up to a common homography. The cameras from $\mathtt{E}$ have only a similarity ambiguity. Remember that the essential matrix relates image coordinates that has been projected by a camera with identity calibration matrix. So the first camera is set to $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$. The second camera can not be uniquely determined from the essential matrix alone. For a given matrix $\mathtt{E}$ there may be up to four different choices for the second camera. Which one to choose can be found by the fact that the back projected points should be in front of both cameras. Using the SVD to decompose $\mathtt{E}$ as

$$\mathtt{E} = \mathtt{U}\mathtt{D}\mathtt{V}^T \tag{42}$$

then, skipping the mathematics behind [11], the four possible cameras are

$$\begin{aligned}
\mathtt{P}'_a &= [\mathtt{U}\mathtt{W}\mathtt{V}^T \mid \mathbf{u}_3] \\
\mathtt{P}'_b &= [\mathtt{U}\mathtt{W}\mathtt{V}^T \mid -\mathbf{u}_3] \\
\mathtt{P}'_c &= [\mathtt{U}\mathtt{W}^T\mathtt{V}^T \mid \mathbf{u}_3] \\
\mathtt{P}'_d &= [\mathtt{U}\mathtt{W}^T\mathtt{V}^T \mid -\mathbf{u}_3]
\end{aligned}$$

Where $\mathtt{W}$ is the orthonogal 3-by-3 matrix

$$\mathtt{W} = \begin{bmatrix} & -1 & \\ 1 & & \\ & & 1 \end{bmatrix} \tag{43}$$

and $\mathbf{u}_3$ is the third column of $\mathtt{U}$.

2.5 Computation of the Fundamental matrix

A neat property of the fundamental matrix is that it can be computed from corresponding points in two images alone. Furthermore it fully describes the relation between the two images so that for example both cameras can be extracted.

Assume we have a point in one image $\mathbf{x}_i$ and another point in the other image $\mathbf{x}'_i$ that corresponds to the same 3D point. Then

$$\mathbf{x}'^T\mathtt{F}\mathbf{x} = 0, \tag{44}$$

since the point $\mathbf{x}'$ must lie on the epipolar line of $\mathbf{x}$. This is the basic equation that all methods for computing $\mathtt{F}$ is based on. Three different ones are explained below and can be found described in greater detail in [11].

2.5.1 The 8-point algorithm

We can assume that the image point vectors have been rescaled so that they all have $w = 1$. Writing $\mathbf{x} = (x, y, 1)'$ and $\mathbf{x}' = (x', y', 1)$ we can rewrite equation 44 as

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0 \tag{45}$$

where f_{ij} are the entries of $\mathtt{F}$. By putting the entries of $\mathtt{F}$ in a single row vector, $\mathbf{f}$ we can write it like this.

$$(x'x, x'y, x', y'x, y'y,' y, x, y)\mathbf{f} = 0. \tag{46}$$

As we can see each point correspondence give rise to one equation. Each such equation can be stacked on top of each other.

$$\mathtt{A}\mathbf{f} = \begin{pmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{pmatrix} \mathbf{f} = \mathbf{0} \tag{47}$$

If we have eight or more points $\mathtt{A}$ is at least an 8-by-9 matrix and will have rank 8 in the general case. The $\mathbf{f}$ vector if the null space of $\mathtt{A}$. The fundamental matrix can then be assembled from the entries in $\mathbf{f}$. However the matrix may not have rank 2. In order to make the matrix rank 2 we can use the SVD. The last singular value is $\sigma_3 = 0$ if the matrix has rank 2. So let $\mathtt{F} = \mathtt{UDV}^T$. Now replace the $\mathtt{D}$ matrix with another matrix $\mathtt{D}'$ that has its last diagonal element set to zero. Then multiply together the matrices to get $\mathtt{F}' = \mathtt{UD}'\mathtt{V}^T$. This matrix will have rank 2. It has been show that $\mathtt{F}'$ minimizes the frobenius norm $||\mathtt{F} - \mathtt{F}'||_F$ with the constraint $\det \mathtt{F}' = 0$. This algorithm extends naturally to any number of points.

2.5.2 The 7-point algorithm

If the rank 2 constraint is taken into account from the beginning only seven points are needed to compute the fundamental matrix. The last step to ensure that the matrix is rank 2 required in the 8-point algorithm is not needed.

The $\mathtt{A}$ matrix in equation 47 will have rank 7 and a two dimensional nullspace. The two column vectors of this nullspace spans the possible values for our $\mathtt{F}$. It can be parameterized as $\mathtt{F}(\alpha) = \alpha\mathtt{F}_1 + (1 - \alpha)\mathtt{F}_2$ where $\mathtt{F}_1$ and $\mathtt{F}_2$ are built from the two null space columns. The $\mathtt{F}(\alpha)$ matrix is 3-by-3 so expanding $\det \mathtt{F} = 0$ yields a cubic polynomial equation in α. This equation has up two three solutions. Using the solutions for α we can compute $\mathtt{F}$. Note that there can be up to three solutions, one for each value on α.

2.5.3 Minimizing a geometric error

The cost function minimized be the two previously described algorithms does not have meaningful geometric interpretation. An object function [11] that finds the maximum likelihood assuming that the error in the image points are gaussian is

$$\min_{\hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i, \mathtt{F}} \sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 \tag{48}$$

under the constraints that $\hat{\mathbf{x}}'^T \mathtt{F} \hat{\mathbf{x}} = 0$ and $\det \mathtt{F} = 0$. The free variables in this minimization is the $\mathtt{F}$ matrix and the refined image coordinates, $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$. This objective function is non-linear, as are the constraints. The minimization can be performed by for example the sequential quadratic programming method [13, Sec 15.5].

2.6 Triangulation

When two camera projection matrices are known and the image coordinates of a 3D point are known in both images the position of the 3D point $\mathbf{X}_i$ can be computed. This is called triangulation.

We have some point matches in the two images, $\mathbf{x}_i$ and $\mathbf{x}'_i$ and also the cameras $\mathtt{P}$ and $\mathtt{P}'$, and we want the 3D coordinates of the matched image points, $\mathbf{X}_i$. Since the projection of $\mathbf{X}$ is known in the both images we can write

$$\mathbf{x} = \mathtt{P}\mathbf{X}, \tag{49}$$
$$\mathbf{x}' = \mathtt{P}'\mathbf{X}. \tag{50}$$

Then using the same reasoning as when dealing with the DLT method described in section 2.3.1 we get

$$\mathbf{x}_i \times (\mathtt{P}\mathbf{X}_i) = \begin{pmatrix} y\mathbf{p}^{3T} - w\mathbf{p}^{2T} \\ w\mathbf{p}^{1T} - x\mathbf{p}^{3T} \\ x\mathbf{p}^{2T} - y\mathbf{p}^{1T} \end{pmatrix} \mathbf{X}_i = \mathbf{0} \tag{51}$$

where $\mathbf{p}^{jT}$ are the rows of $\mathtt{P}$ and $\mathbf{x} = (x, y, w)$. The same holds for $\mathtt{P}'$ and $\mathbf{x}'$. Only two of the rows are linearly independent so one can be removed. Assuming that $w = 1$ and reordering the terms and stacking the two equations from each image on top of each other gets us to

$$\mathtt{A}\mathbf{X} = \begin{pmatrix} x\mathbf{p}^{3T} - \mathbf{p}^{1T} \\ y\mathbf{p}^{3T} - \mathbf{p}^{2T} \\ x'\mathbf{p}'^{3T} - \mathbf{p}'^{1T} \\ y'\mathbf{p}'^{3T} - \mathbf{p}'^{2T} \end{pmatrix} \mathbf{X} = \mathbf{0}. \tag{52}$$

This are four equations in four unknowns, so in the general case there is a unique solution. But the $\mathbf{X}$ vector is a homogeneous vector so the equation system is redundant, three equations would have been enough.

The solution can be found using the singular value factorization. Let $\mathtt{A} = \mathtt{U}\mathtt{D}\mathtt{V}^T$, then the solution $\mathbf{X}$ is the last column of $\mathtt{V}$.

2.7 The Förstner interest operator

The Förstner interest operator extracts points of interest from an image. This operator is described in [5, 6]. The aim of the interest operator is to find points in an image suitable for matching with points in another image. There is also a similar algorithm that is widely used described by Harris [10].

The ideas behind the interest operator is that points with certain properties are wanted. Förstner summarizes those properties in his paper.

- Distinctness
- Invariance
- Stability

- Seldomness

The first property, distinctness, captures a very basic property of an interest point. If the exact position can not be determined the point is of no use. Like in figure 7 where we see a section of a line to the right.

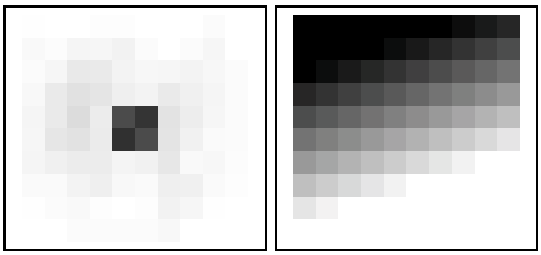


Figure 7: Left: A section of an image with a distinct point. Right: The surrounding of an ill-suited interest point

The discussion below uses the concept of gray level, but in fact the process could be done for each color channel, i.e. red, green and blue. The derivate of the image gray level is used to find interest points. A peak in the intensity of the image point is characterized by a high difference in all directions. An approximated value for the differences in x- and y-directions can be found using convolution with a differentiating kernel.

Both Harris and Förstner uses the covariance matrix of the difference in each pixel to decide how well suited it is for being an interest point. The covariance matrix

$$\Sigma = \begin{pmatrix} g_x^2 & g_x g_y \\ g_y g_x & g_y^2 \end{pmatrix} \tag{53}$$

is computed for each pixel where g_x and g_y are the differences at (x, y). A naive approach is to look at the values of g_x and g_y. We want pixels that has large g_x and g_y which indicated a rapid change in gray level in both x and y directions. A more sophisticated way is to look at the eigenvalues of the covariance matrix. The largest eigenvalue represents how much the gray value changes in the direction that changes the most, while the other eigenvalue is the change in the orthogonal direction. In a good interest point both eigenvalues are high. But to compute the eigenvalues of this matrix in each pixel can be costly even on todays computers since images can be very large. Instead a measurement based on the properties of the trace and determinant are used. For any 2-by-2 matrix `A` holds

$$\lambda_1 \lambda_2 = \det \mathtt{A} \tag{54}$$

$$\lambda_1 + \lambda_2 = \text{trace } \mathtt{A} \tag{55}$$

Where λ_1 and λ_2 are the two eigenvalues of the matrix. Förstner calculates the following isotropy measure

$$q = \frac{4 \det \Sigma}{(\text{trace } \Sigma)^2}. \tag{56}$$

This measure captures the fact that the trace and determinant have different units and is rotationally invariant because the measure is based on eigenvalues. It takes values in the interval $0 \leq q \leq 1$.

For all pixels with a high enough isotropy value enough an interest value is computed

$$w = \begin{cases} 1/\text{trace } \Sigma & q > q_{\min} \\ 0 & \text{otherwise} \end{cases} \tag{57}$$

where $q_{\min}$ may e.g. be set to 0.5. Finally the non-local w-maxima are suppressed.

In figure 8 two images are shown with the 500 best interest points marked with circles.

2.8 Matching points

In order to the compute the fundamental matrix, one need point matches between the two images. This is generally a hard problem. Various algorithms have been developed to allow point matches to be acquired automatically from an image pair.

The first step uses the fact that the surrounding pixels of a point in one image will look about the same as in the other image. To decide which points pairs to select as putative matches the cross-correlation is used as a measure. The higher S value the better the two points match. It is defined as

$$S = \frac{\sigma_J^2 \sigma_I^2}{\sigma_J \sigma_I} \tag{58}$$

Where σ_J^2 and σ_I^2 in the discrete case are defined as

$$\sigma_J^2 = \sum_{(x,y)\in A_J} J(x,y) - \bar{J} \tag{59}$$

$$\sigma_I^2 = \sum_{(x,y)\in A_I} I(x,y) - \bar{I} \tag{60}$$

$$\tag{61}$$

Over the area A_J in one image and the area A_I in the other. The values $\bar{I}$ and $\bar{J}$ are the mean pixel intensity over the respective areas. This measure is invariant to both overall intensity and contrast of the selected areas.

The area around each interest point is correlated with all interest points in the other image. The pairs with highest correlation are kept.

Figure 8: Two images of the same object with interest points marked as white circles.

Figure 8 shows two images with interest points marked. Note how some points do not

have a corresponding interest point in the other image.

Assume that the object is view from about the same angle and distance in the two images. Then for most images encountered in practice a point in one image will have its corresponding point in roughly the same area of the image. This is however not true in the general case since points may travel a lot in the image even if the camera motion is very small, e.g. the camera is close to a wall. This fact can be used in the matching process by suppressing too separated point matches, e.g. if the euclidean image distance between them is greater than 10%–20% [14].

In figure 9 the putative matches are visualized with a line from the point in that image to the point where it would be in the second image. The lines are overlaid on the first image.

Figure 9: The matches visualized with lines.

2.9 Robust estimators

When we want to fit a function to some data, the classical approach is to assume that the errors in each data point is independent of each other and follows some distribution, usually a normal distribution. In that case the optimal solution can be found using a maximum likelihood method, e.g. least squares, where the probability that the found parameters are the correct ones are as high as possible.

However sometimes the error in acquired data follows some distribution that is easy to work with, with the exception that a few data points are completely wrong. Such points are called *outliers* or *blunders* and can cause least squares solutions to perform badly. The error in these points are said to be *gross*. Even a single gross error can cause the computation to fail or diverge, so it is very important to sort those out before

a smoothing algorithm is used.

In [3] a robust method for detecting outliers is described called RANSAC, RAndom SAmple Consensus. It randomly selects a *minimum* set of data points needed to compute a model function. All other data points are measured how close they are to this model function, if it is close enough it is said that this data points supports the particular model function. This step is repeated several times and the model function with most supporting data points is selected. All data points supporting this final model are marked as *inliers* while all other points are marked as outliers.

Taking a random sample from all data points is repeated until there is a certain chance z, e.g. $z = 0.95$, that we have selected at least one "good" sample, i.e. a sample with only inliers. If we know the *á priori* probability p that a data point is an inlier we can say that

$$(1 - p^n)^k = 1 - z, \tag{62}$$

$$k = \frac{\log(1 - z)}{\log(1 - p^n)}, \tag{63}$$

where n is the minimum number of data point needed to compute the model. If the probability p is not known an upper bound can be used. In equation 62 note that the lower value for n, the fewer iterations are needed.

The algorithm is stated in a more formal form below [11, Alg 3.4].

1. Select a subset S of n points from the dataset.
2. Use the selected points to calculate the free model parameters.
3. Determine the consensus set, i.e. the set of all data points that are within some threshold from the instantiated model.
4. Repeat the above until the probability that one of the subsets S contains only inliers is larger than z.

Consider the problem of fitting a line to some data points. If there are no outliers a least squares solution will work. However, if a single data point is an outlier we will get a completely different (and wrong) line as seen in figure 10.

The RANSAC algorithm applied to the line fitting problem works as follows: Assume we know the inlier ratio $w = \frac{7}{8}$. Two points uniquely defines a line so $n = 2$. To get a 95% probability to pick at least one sample with only inliers, from equation 62 at least $k = 2$ samples are needed. To measure how close a data point is to the model function the perpendicular distance between the line and the point is used.

This method has been refined into what is called the *adaptive RANSAC*. The difference is that adaptive RANSAC tries to estimate the probability w after each iteration. This is done be estimating w with the ratio between the number of inliers from the largest subset picked and the total number of data points.

There is always a risk that some of the points that really are inliers will be marked

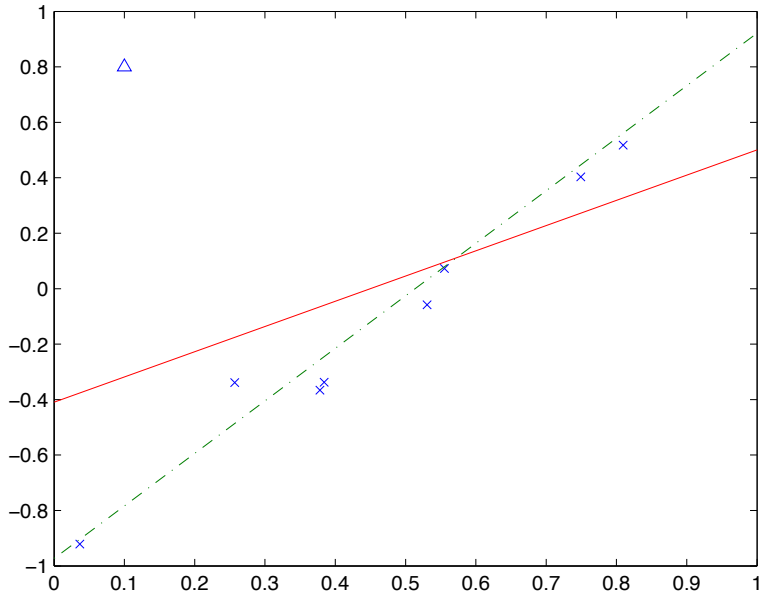


Figure 10: A line has been fitted to the data points. Another line is fitted to the data points including the inlier (marked with a slashed line)

as outliers. This happens when the threshold is too tight which causes points actually being close to the model to be classified as outliers.

After the outliers has been classified the best model function fitted only depends on the minimum number of points needed to compute it. In general the solution should be refined using *all* inliers to achieve a better result. Sometimes this is done by a iterative optimization [13] and in that case the solution from RANSAC can be used as an initial guess.

3 Reconstruction from image sequences

This section describes how to automatically reconstruct 3D coordinates and exterior orientation of the cameras. The process is sometimes called Structure and Motion Recovery (SMR) [14, 15]. The structure is the coordinates of the 3D points and the motion is the exterior orientation of the cameras.

Reconstruction methods with no prior knowledge about the internal orientation of the camera have been developed. The most common way is to initially compute a projective reconstruction then upgrade it by applying constraints on the calibration, in a *self calibration* process. [14, 15]

Another approach is to calibrate the camera separately. Using the calibration information a reconstruction with a similarity ambiguity can immediately be computed, i.e. without a subsequent self-calibration step.

3.1 The first camera pair

The first two images are used to initialize the structure and motion. It is important to select images that are well suited to start up the whole process. They must have features that can easily be matched between the views. The cameras must not be too close each other since that can cause the computation of the fundamental matrix to fail or be inaccurate [14].

Feature points found in the two images are matched together using the cross-correlation measure described in section 2.8.

The RANSAC algorithm is used to detect and remove outliers. Eight point matches are randomly selected and a resulting $\mathtt{F}$ matrix is computed using the 8-point algorithm from section 2.5.1. The measure used to threshold outliers is the distance from the epipolar line to the putative point match.

After RANSAC the fundamental matrix is recalculated with the normalized 8-point algorithm (see section 2.5.1) using all inliers and refined by minimizing the geometric error (section 2.5.3).

Knowing the fundamental matrix allows the two first cameras to be extracted as described in section 2.4.1 or 2.4.3 depending on whether the calibration matrix is known. Then the object points are triangulated from the cameras and image points. The product of this step is the first camera pair $\mathtt{P}_1, \mathtt{P}_2$, some points matches between the two images, and the triangulated object points $\mathbf{X}_i$.

3.2 Subsequent images

Subsequent images are incorporated one by one into the reconstruction. In this section the *left* image is the previous image added and the *right* image is the next to add, see figure 11. The camera matrix $\mathtt{P}_{n-1}$ and the interest point in the left image is known. The interest points in the left image are split into two sets. The set $\mathcal{A}_n$ containing points with a corresponding object point, the set $\mathcal{B}_n$ with no corresponding object points.

Interest points are located in the right image and correlated with points in the left image. The interest points are matched together, like in section 3.1 except this time all points in $\mathcal{A}_n$ are ensured to be matched with a point in the right image. For each point in $\mathcal{A}_n$ the point in the right image with highest cross correlation are matched together as a putative match.

Two sets of point matches can be identified. The first set, call it $\mathcal{A}$, contains point matches where the point in the left image is in $\mathcal{A}_n$. The other set $\mathcal{B}$ are the rest of the point matches, i.e. the ones where none of the image points in the left image has a corresponding object point. The point matches in $\mathcal{A}$ actually connects three entities, a image point in the left image, an image point in the right image, and an object point.

In figure 11 the two sets of point correspondences are shown. The point correspondence from $\mathcal{A}$, marked in the figure with $\mathbf{x}_2 - \mathbf{x}_3$, has a corresponding object point $\mathbf{X}$ since it was triangulated from an earlier point match. Those types of matches are used to compute the next camera P_n. The other point match in figure 11 $\mathbf{y}_1 - \mathbf{y}_2$ does not have

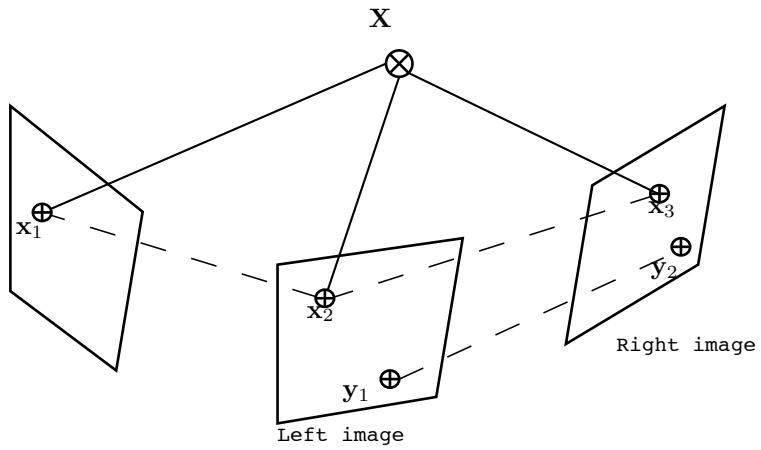


Figure 11: The two types of point correspondences between the left and right image.

an object point yet and thus belong to the set $\mathcal{B}$. Those types of point matches are used to instantiate new object points.

Next RANSAC is used to reject outliers, but instead of using the fundamental matrix, the camera is estimated directly. So this time we have correspondences between object points $\mathbf{X}$ and image points $\mathbf{x}_3$ in the right image. The task is to compute the camera for this new image in a robust way. [2]

RANSAC needs three point matches for the calibrated case and six point matches for the uncalibrated case to compute a camera $\mathtt{P}_n$.

The membership function used to determine if a point match is an inlier given a camera is the distance from the re-projected image point, $d(\mathtt{P}\mathbf{X}, \mathbf{x}_3)$.

In the uncalibrated case the camera matrix for the right image $\mathtt{P}_n$ is refined using all

[2] This was actually the example problem used in the original RANSAC paper [3].

inliers with the DLT algorithm described in section 2.3.1. In the calibrated case however, the camera is not refined since the 3-point algorithm does not generalize to more points.

Point matches in set $\mathcal{A}$ are now classified into inliers and outliers. Using the inliers in $\mathcal{A}$ a fundamental matrix relating the left and right image is computed with the 8-point algorithm (see section 2.5.1). The fundamental matrix is used to remove outliers from the set $\mathcal{B}$ by thresholding out point matches where the right image point is too far from the epipolar line line.

As a last step the correct point matches from $\mathcal{B}$ are triangulated to produce a object point. Those object points are added to the set of already known object points.

3.3 Overall algorithm

The overall algorithm for the structure and motion recovery is summarized in an psedo-code-like manner for better overview. For details refer to the more in-dept description in the prevoius sections. The second is repeated until there are no more images left to process.

1. Initialize the reconstruction using the two first images in the sequence.
   (a) Find intrest points in both images and match them together using cross-correlation.
   (b) Compute `F` from the putative point matches in a robust way using RANSAC.
   (c) Extract the first two cameras.
   (d) Trianglulate points into object points.
2. Add an image.
   (a) Find intrest points in the new image.
   (b) Match intrest points with intrest points in the prevoius image added.
   (c) Compute the camera for the new image using points matches from $\mathcal{B}$ in a robust way.
   (d) Triangulate new object points from point matches in $\mathcal{A}$, i.e. image points only visible in the new image and the previous one.

3.4 The problem of degeneracy

There are two major types of degenerate configurations that arise when doing SMR. Theese configurations makes the epipolar geometry calculations collapse into a problem with no unique solution. One is when the change in pose from the first to the second view is only a rotation about the camera center, and the other type of degeneracy is when all corresponding points in the images are coplanar (lie on a common plane) in object space. Such an image is said to contain a *dominating plane*, i.e. a plane that covers all or most of the image. Both problems occurs very often in practice. A more in-depth introduction to those problems as well as a measure to detect and distinguish between the two cases can be found in [17]. There is however a way to survive those degenerate configurations. In both cases the images will still be related via a homography [17].

4 Implementation

Two reconstruction algorithms have been implemented. One that has information about the camera calibration that includes the camera calibration matrix and radial lens distortion. The other part works without knowledge about the camera. The reconstruction obtained by the later approach is only determined up to a unknown homography.

The problems of degeneracy (see section 3.4) are not dealt with in any way so care must be taken when gathering the image sequences used as input. The camera centers should not be too close each other and dominant planes should be avoided.

Both parts start out with the two first images and find Förstner interest points in them. The differentiating kernels are sobel kernels.

$$H_{dx} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{64}$$

For smoothing the covariance matrix elements over the image (see section 2.7) a gaussian 7-by-7 kernel with $\sigma = 0.7$ pixels is used.

Image points are transformed so that they have mean about zero and variance about 1. For the calibrated case the image coordinates are transformed from pixel units to mm units, while for the uncalibrated case the pixel units are just scaled down somewhat based on the image size.

The best 500 interest points are kept. Then the seldomness operator is applied (see [5]) and this time the 250 most seldom points are kept. Then points from the two images are correlated using normalized cross correlation. The 100 best point matches are kept.

An 11-by-11 pixel wide square is use when doing the cross correlation for seldomness operator and when matching points.

The minimum distance from the epipolar line passed to RANSAC is set to 3 pixels.

When moving on to the next image the interest points in the left image is already known so they need not to be computed. When matching points together, all the points in the left image that have a corresponding object point are matched with some point in the right image. This is because we need point matches where the point in the left image that has a corresponding object point. Then an additional 100 of the best point matches are kept.

As described in section 2.3.2 the computed camera $\mathtt{P}_n$ has a four fold ambiguity. All four possible cameras are passed to RANSAC.

The threshold for rejecting outliers when computing the camera from 2D to 3D correspondences is set to 3 pixels. The minimum distance to the epipolar line when rejecting outliers in set $\mathcal{B}$ is also set to 3 pixels.

After rejecting outliers in the 2D to correspondences the camera is recomputed using all available point matches for the uncalibrated case. This is easily done since the DLT

algorithm generalizes to any number of points greater than 6.

Some outliers may survives RANSAC, i.e. incorrect point matches that coincidently lie on the epipolar line. Obvious outliers are removed in a last step by removing object points lying behind all cameras. This step cannot be done in the uncalibrated case since the overall scale may be negative causing cameras to point in the wrong direction.

5 Test data

5.1 Synthetic data

A simple stick house is used as test data, see figure 12. Seven cameras are placed in an arc aiming at the house. Image coordinates are computed by projecting the known

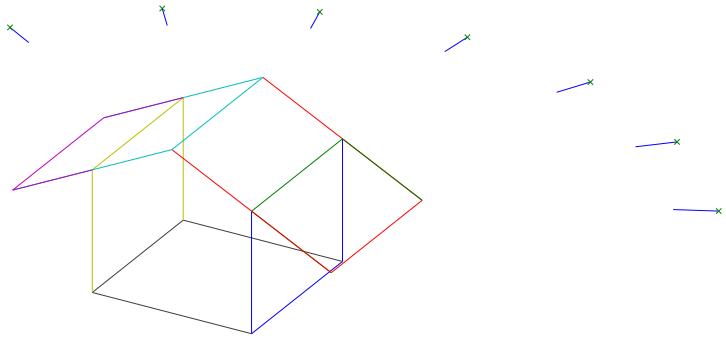


Figure 12: The house and cameras.

object points for each camera as

$$\mathbf{x}_{ij} = \mathtt{P}_i \mathbf{X}_j \tag{65}$$

with $i = 1 \ldots 7$ and j range over all 14 object points. The calibration matrices of the cameras are for simplicity set to the same as the camera used in the real images, see 5.2. Image points then undergo a slight distortion according to the lens distortion model described in 2.2.3 and a gaussian noise with $\sigma = 4$ pixels is applied. The only noise in the image coordinates for the synthetic test data is the synthetic lens distortion and gaussian noise. The resulting image points linked with lines can be seen in figure 13

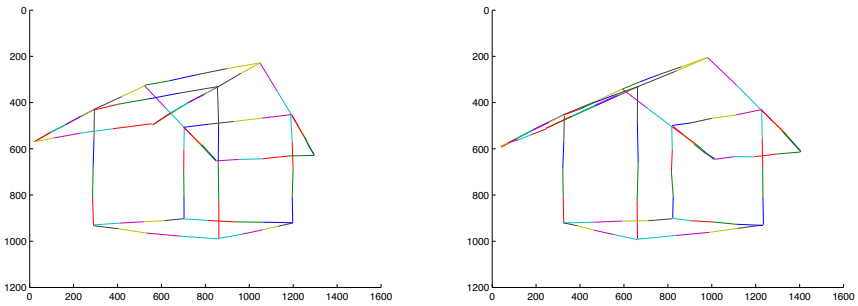


Figure 13: The house from the the first two cameras point of view.

5.2 Real images

A set of test images were taken. The object was chosen such that it did not contain a dominating plane and the camera translation was not too small. All images from the test sequence as well as some photos taken to get an overview can be found in appendix A.

The camera used was a Olympus C220 digital camera with the lens zoomed to max. This camera was first calibrated using a calibration rig, consisting of a sheet seen in figure 14.

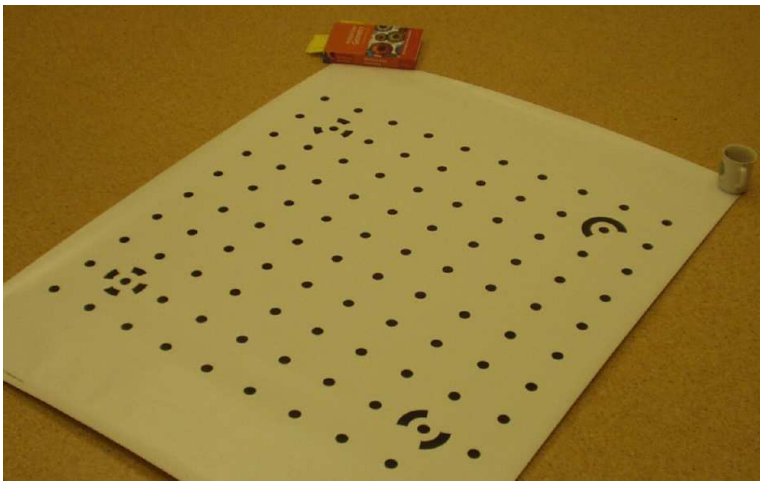


Figure 14: One of the calibration images.

6 Results

Both algorithms where run on the test data described in section 5 resulting in a two reconstructions. Information about how the reconstruction proceeded, e.g. number of point matches, execution speed, etc., was recorded.

6.1 Synthetic test run

The reconstruction algorithm was run on the synthetic test data described in section 5.1.

6.1.1 Calibrated

The calibration of the seven cameras are known beforehand, so is the lens distortion coefficients.

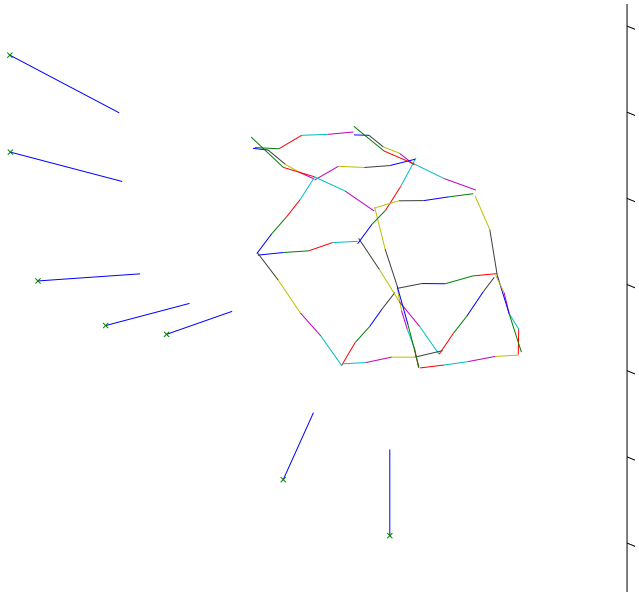


Figure 15: The reconstruction of the synthetic data, calibrated version.

Because of the errors in the image coordinates the reconstruction is not exact, and it can be seen in figure 15 how the straight line of the object are not straight in the reconstruction. The cameras are also somewhat off their exact position.

6.1.2 Uncalibrated

The reconstruction was performed using all seven images to result in a perspective reconstruction. The reconstruction plotted from a new angle can be seen in figure 16. Again, the reconstruction slightly distorted because of image errors.

A correcting homography `H` was computed from the scene knowledge to transform the reconstruction into a similarity one. How such a homography can be computed can be found in appendix C.

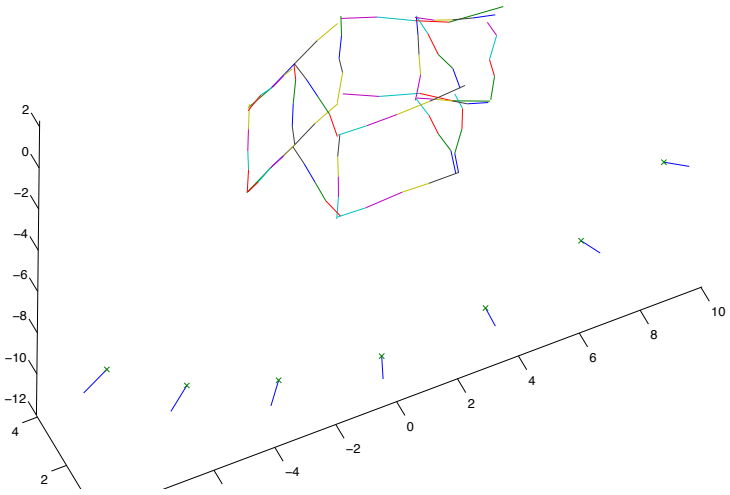


Figure 16: The reconstruction of the synthetic data, uncalibrated version. Note: A homography transforming the reconstruction into a similar one has been applied to ease viewing. The cameras still point outward because all aspects of the projective deformation cannot be undone.

6.2 Real images test run

6.2.1 Calibrated

Here are some results with the algorithm with known calibration run on the example sequence.

In order to understand the reconstruction plots displayed later in this section better, the point matches from the first image pair are shown in figure 17.

In figure 18 a plot of the reconstruction after three images has been processed. The object points are viewed from *above* in the plot leaving points located deep in the image to the left. Some artifacts can be identified from the image sequences. The rightmost point group in figure 18 corresponds to the point matches found on the leftmost book in figure 17. The tree to the right in the image has a number of point matches, i.e. the triangulated object points can be seen in the top-left corner of figure 18.

The process proceeds and in figure 19 five images has been added to the reconstruction. The five cameras can be seen in the bottom part of the image. Note the occurrence of a few outliers, e.g. just in front of the cameras. The reconstruction is not so accurate, the line corresponding to the leftmost book in figure 28 is almost parallel with the cameras rather than tilted about 45° as in the ground truth.

6.2.2 Uncalibrated

Here points and cameras from the test sequence has been reconstructed. The result after four images can be seen in figure 20. Not much can be gathered from this plot since it differs from ground truth by an unknown homography. The four cameras lie somewhat

Figure 17: Point matches between the first image pair.

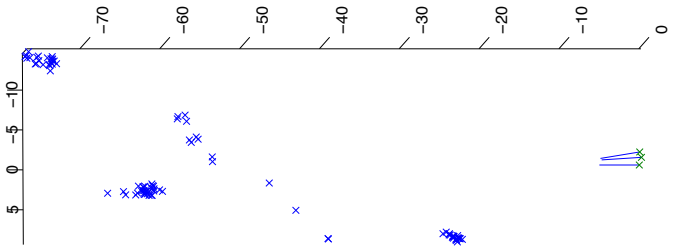


Figure 18: Object point calculated from the first three images viewed from above. The cameras can be seen to the right.

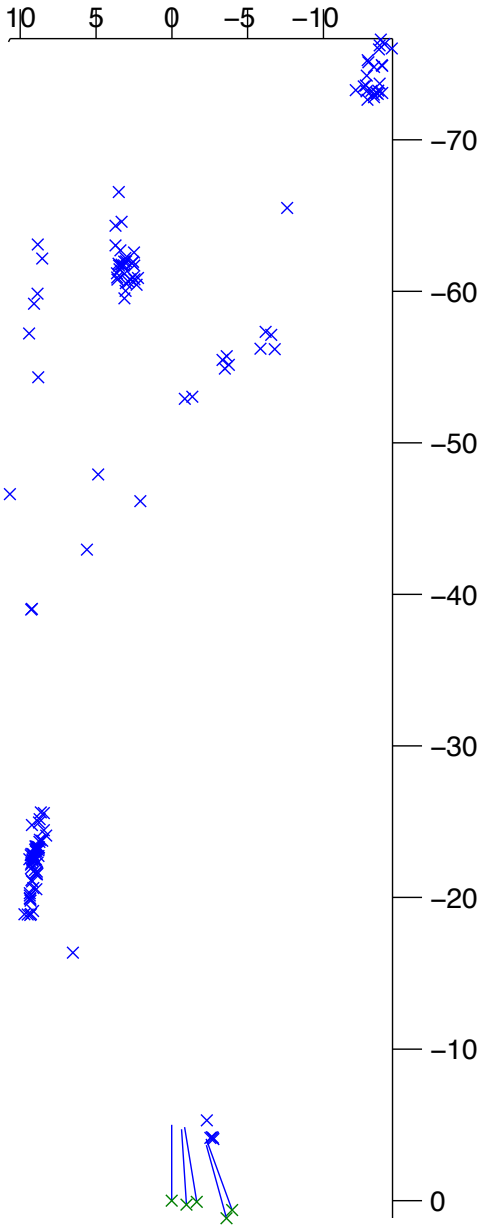


Figure 19: The reconstruction after five images has been added.

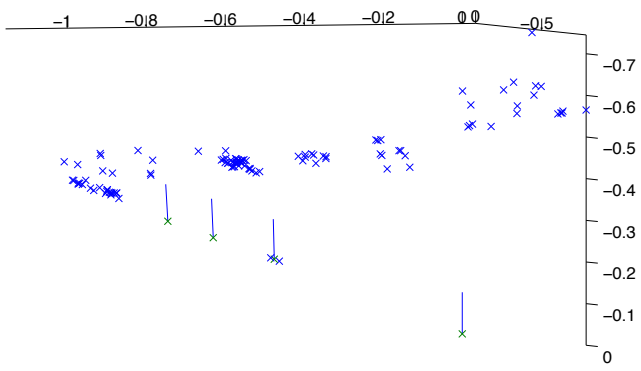


Figure 20: The reconstruction after four images has been added.

in an arc-shape, or at least on a line.

Unlike the synthetic test data, true position of object points are not known and thus a correcting homography can not be computed.

6.3 RANSAC performance

The hypothesis for this test was that since the calibrated case only requires three point correspondences to compute a camera matrix the RANSAC part would execute faster.

In figure 21 the number of samples needed for $n = 3$ and $n = 6$ respectively are plotted versus the ratio of inliers.

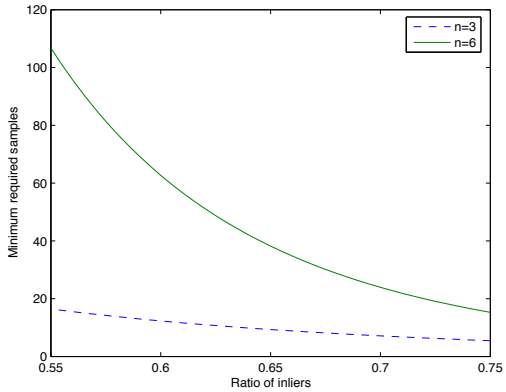


Figure 21: Number of samples needed for the RANSAC algorithm versus inlier ratio. $z = 0.95$

The implementation uses adaptive RANSAC which means that the exact number of iteration will depend on which subsets are chosen. The uncalibrated case requires more iterations as can be seen in figure 22, except for the first image pair. This image pair is somewhat special since the fundamental matrix is computed in both cases, which means the number of iterations should be about the same or possibly a bit smaller in the calibrated case because of lens distortion removal.

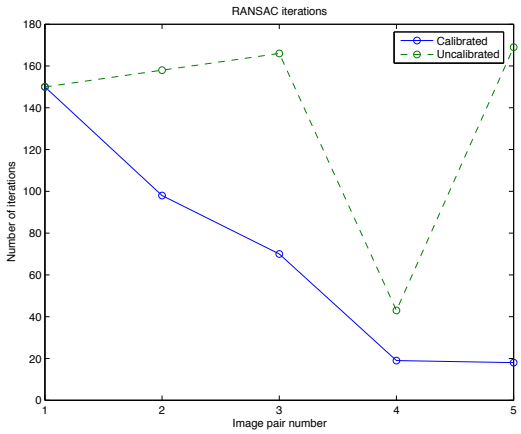


Figure 22: Number of iterations in RANSAC step .

However, test conducted shows that time spent on the RANSAC algorithm is very short compared to the total time spent on each image pair. The advantage of having lower n-value is small compared to the the amount of time spent on finding interest points and matching them together. In figure 23 the time spent in seconds on each image pair in total and just RANSAC is shown. Note how the time spent on RANSAC is disappearing low. This test was done on the calibrated case.

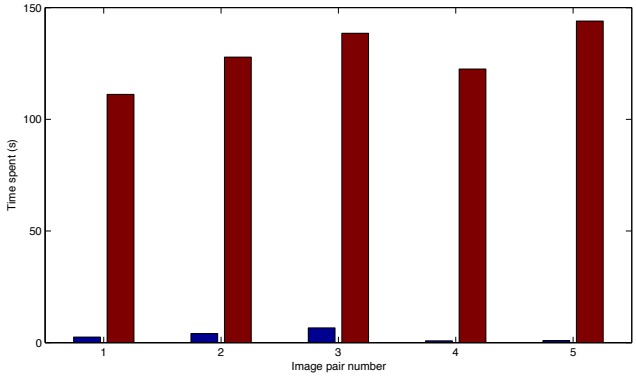


Figure 23: Total time spent in seconds per image pair. To the left of each bar is the time spent on RANSAC.

While the uncalibrated version has to pick more samples, each sample is processed faster.

Computing the camera in the uncalibrated case is a straight-forward linear algorithm while the calibrated case involves solving for a fourth degree polynomial. The effect of the slower computation of $\mathtt{P}_n$ makes the uncalibrated and the calibrated case execute at similar speeds for this implementation as can be seen in figure 24.

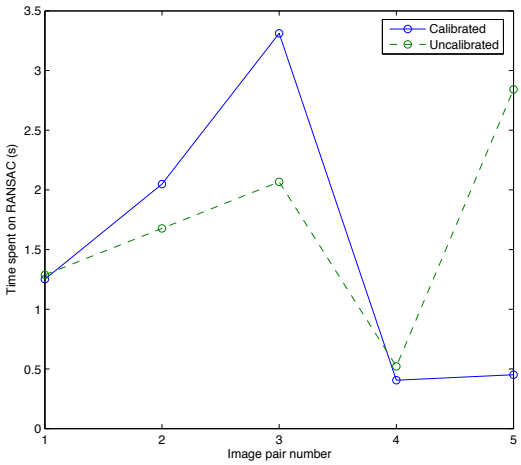


Figure 24: Time spent on RANSAC for both calibrated and uncalibrated case.

One could argue that if the number of outliers are high enough and the number of cross correlation computed are low enough, the calibrated case executes faster. Considering the vast time differences this will only happen in extreme cases.

6.4 Point matching and lens distortion

When the cameras are calibrated not only the camera calibration matrix is known, but also the size of the lens distortion.

The number of matches points between image pairs drops as more images are added as can be seen in figure 25. This is because all types of errors makes the cameras drift somewhat for each image added and thus making the matching harder. The number of matches found are higher for the calibrated case.

As can be seen in figure 26, about 63% of the interest points found in all six images are displaced less than the outlier rejection threshold of 3.5 pixels.

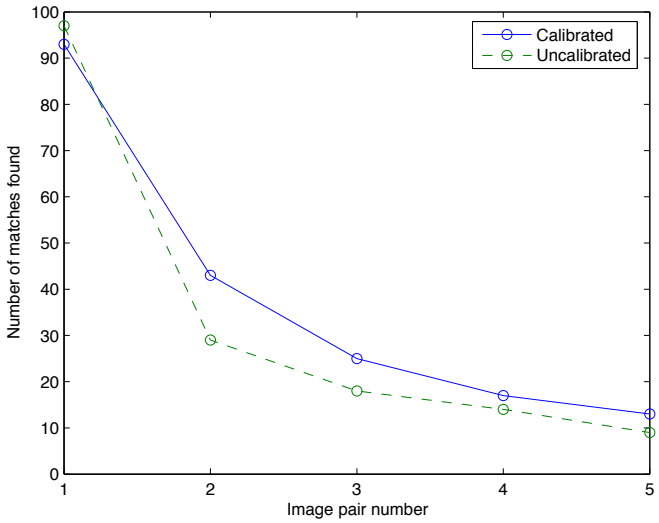


Figure 25: Number of point matches found between image pairs

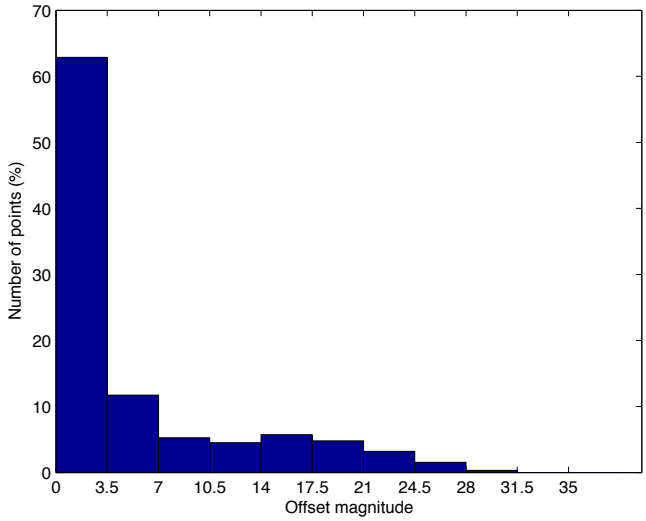


Figure 26: Distribution of points by lens distortion displacement. Each bin is 3.5 pixels wide.

7 Conclusions

Knowing the calibration matrix of the camera used allows for faster computation of the next camera in terms of number of iterations in the RANSAC step. However, test shows the computationally more expensive algorithm used when the calibration is known reduces the advantage. And moreover the time spent on RANSAC was negligible.

Lately self-calibration algorithms allowing for different focal length has been developed [15]. This increases the flexibility when gathering the image sequence since the cameras zoom can be used. The calibrated case however the focal-length is already fixed and using different focal lengths would mean taking calibration images and calibrating the camera for each focal length used.

Knowing the size of the lens distortion improves the reconstruction in several ways. More point matches are found in consecutive images since interest points lie closer to their corresponding epipolar line. Also the lens distortion is largest at the edges of the image, an area where it is desirable to have point matches.

The calibrated implementation immediately results in an similarity reconstruction, in which angles, distance ratios, etc. can be measured directly. This is of course an advantage, assuming the effort to calibrate the camera is small.

The reconstruction from the uncalibrated implementation on the other hand is of less use as it is. So a self-calibration algorithm has to be implemented as well.

In the uncalibrated case the mathematics behind are entirely projective, resulting in straight forward algorithms based on matrix decompositions. The algorithms used in the calibrated case are more cumbersome.

7.1 Future work

It would be interesting to compare the precision of the two techniques, this is however not possible unless a self-calibration algorithm is used on the projective reconstruction.

It would also be interesting to see how using the Förstner sub-pixel operator, that selects the optimal point in each window rather than the window center, affects the reconstruction

In this implementation no so called guided matching is performed where new point matches are searched for along epipolar lines after two images has been related.

In the current implementation the camera is not refined using all known 2d–3d point matches in the calibrated case, which causes the camera to drift as more images are added. A non-linear optimization should be used to refine the camera also for the calibrated case.

Points visible in more than two images can be re-triangulated using the extra image coordinates to increase the accuracy in the reconstructed object point position.

8 Acknowledgments

I would like to thank my supervisor Niclas Börlin for helping me through all phases of writing this thesis. He has been a great help in this and really put his heart in to help me get the work and the report done.

I would also like to thank Anders Hagnelius for giving feedback on the report and my former roommates Daniel, Arsalan, and Sandra for their great company.

A Test sequence

In figure 27 all six images from the test sequence are shown.

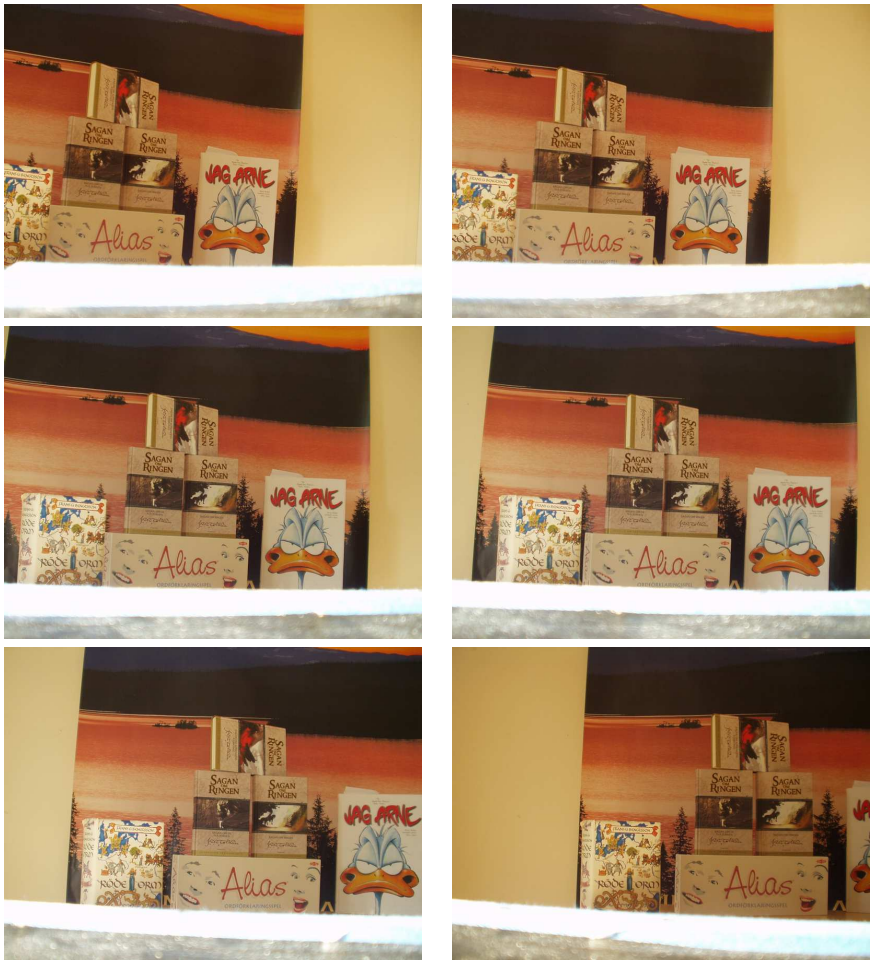


Figure 27: The test sequence.

To get a better view of the setup of the test sequence an additional image of the object is provided in figure 28. In the bottom part of the image some markings show the approximate position of the camera when the respective images where taken. The first test image corresponds to the leftmost marking.

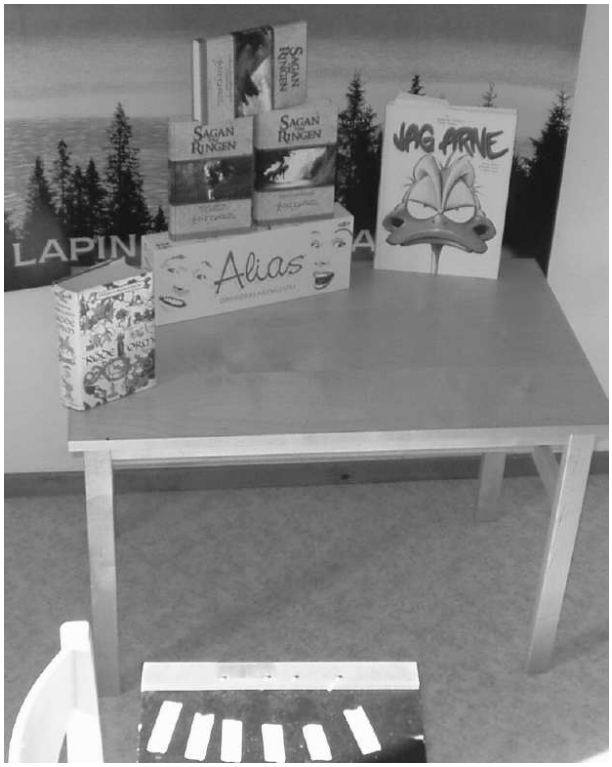


Figure 28: Overview of the object used in test sequence together with mark ups of the approximate orientation and position of the different cameras.

B Notations

Here the different notations used are described.

$\mathtt{A}$	A matrix is typeset this way.
$\mathbf{v}$	A vector is typeset like this.
$\mathtt{A}^T$	Transpose.
$\mathtt{A}^{-1}$	Inverse.
$[\mathbf{v}]_\times$	The skew matrix produced by 3-vector $\mathbf{v}$.
$\mathtt{P}$	A camera matrix (3x4).
$\mathtt{H}$	Homography.
$\mathtt{K}$	Camera calibration matrix (3x3)
$\mathtt{F}$	Fundamental matrix.
$\mathbf{X}_i$	Homogeneous point in object space (4-vector).
$\mathbf{x}_i$	Homogeneous image point (3-vector).
$\mathbf{l}$	Homogeneous representation of a 2D line (3-vector).
$\mathtt{A}^+$	Pseudo-inverse.
$d(\mathbf{x}, \mathbf{y})$	Euclidean distance between points.
$N(\mathbf{x})$	Normalization operator. $\mathbf{x}/\|\|\mathbf{x}\|\|$

C Estimating a homography in $\mathbb{P}^3$

Suppose a set of point matches $\mathbf{X}_i$ and $\mathbf{Y}_i$ in $\mathbb{P}^3$ are known and that they are related by a homography $\mathtt{H}$ transforming points as

$$\mathbf{Y}_i = \mathtt{H}\mathbf{X}_i. \tag{66}$$

A homography in $\mathbb{P}^3$ has 15 DOF and thus at least 5 point correspondences are needed to compute $\mathtt{H}$ since each point fixes three DOF.

Clever rearranging of equation 66 yields

$$\begin{bmatrix} \mathbf{X}_i^T w_i & \mathbf{0}^T & \mathbf{0}^T & -\mathbf{X}_i^T x_i \\ \mathbf{0}^T & \mathbf{X}_i^T w_i & \mathbf{0}^T & -\mathbf{X}_i^T y_i \\ \mathbf{0}^T & \mathbf{0}^T & \mathbf{X}_i^T w_i & -\mathbf{X}_i^T z_i \end{bmatrix} \mathbf{h} = \mathbf{0} \tag{67}$$

where $\mathbf{h}$ is a 16-vector holding the elements of $\mathtt{H}$ and $\mathbf{Y}_i = [x_i, y_i, z_i, w_i]^T$. The fourth row is omitted since there are only three linearly independent equations. This holds for all points correspondences i, and the equations above can be stacked on-top of each other into a $3n$-by-16 matrix $\mathtt{A}$.

$$\mathtt{A}\mathbf{h} = \mathbf{0}. \tag{68}$$

A solution can be found using the singular value decomposition.

D Singular value decomposition

The *singular value decomposition* of matrices are used through out this paper. Here is a short introduction to this.

Given a square matrix $\mathtt{A}$ the SVD is a factorization

$$\mathtt{A} = \mathtt{UDV}^T. \tag{69}$$

Both $\mathtt{U}$ and $\mathtt{V}$ are orthonogal matrices while $\mathtt{D}$ is a diagonal matrix. In order for the decomposition to be unique, the elements of $\mathtt{D}$ are sorted in descending order. The intricate inner workings of the SVD decomposition can be found in [8].

The SVD factorization has numerous application but in this paper it is mostly used to solve the minimization problem

$$\min_{\mathbf{x}} \frac{||\mathtt{A}\mathbf{x}||}{||\mathbf{x}||} \tag{70}$$

The solution to this problem is last column of the $\mathtt{V}$ matrix [11].

References

[1] D. C. Brown. Decentering distortion of lenses. *Photogrammetric Record*, 32(3):1272–1292, May 1966.

[2] A. Conrady. Decentered lens systems. *Monthly Notices of the Royal Astronomical Society*, 99:384–390, 1919.

[3] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

[4] Andrew W. Fitzgibbon and Andrew Zisserman. Automatic camera recovery for closed or open image sequences. In H. Burkhardt and B. Neuman, editors, *Computer Vision-ECCV'98*, number 1406 in Lecture Notes in Computer Science, pages 311–326. Springer Verlag, 1998.

[5] W. Förstner. A feature based correspondence algorithm for image matching. *International Archives of Photogrammetry and Remote Sensing*, 26(3/3):150–166, 1986. Rovaniemi.

[6] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and circular features. In *Intercommission Conference on Fast Processing of Photogrammetric Data*, pages 281–305, Interlaken, 1987.

[7] J. G. Fryer and D. C. Brown. Lens distortion for close-range photogrammetry. *Photogrammetric Engineering and Remote Sensing*, 52(1):51–58, 1986.

[8] Gene H. Golub and Charles F. Van Loan. *Matrix Computation*. John Hopkins Press, 3rd edition, 1996.

[9] R. M. Haralick, C. Lee, K. Ottenberg, and M. Nölle. Analysis and solutions of the tree point perspective pose estimation problem. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 592–598. IEEE, 1991.

[10] C. J. Harris and M. Stephens. A combined corner and edge detector. In *4th Alvey Vision Conference*, pages 147–151, Manchester, 1988.

[11] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[12] J. C. McGlone, editor. *Manual of Photogrammetry*. American Society of Photogrammetry, 5th edition, 2004. Index.

[13] Stephen G. Nash and Ariela Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, 1996.

[14] Marc Pollefeys. Visual 3d modeling from images. Tutorial given at the ISPRS Congress in Istanbul in July 2004.

[15] Marc Pollefeys, Luc J. Van Gool, and Marc Proesmans. Euclidean 3d reconstruction from image sequences with variable focal lenghts. In Bernard F. Buxton and Roberto Cipolla, editors, *Computer Vision-ECCV'96*, number 1064 in Lecture Notes in Computer Science, pages 31–42. Springer Verlag, 1996.

[16] C. C. Slama, editor. *Manual of Photogrammetry*. American Society of Photogrammetry, 4th edition, 1980. Index.

[17] Philip H. S. Torr, Andrew W. Fitzgibbon, and Andrew Zisserman. The problem of degeneracy in structure and motion recovery from uncalibrated image sequences. *International Journal of Computer Vision*, 32(1):27–44, 1999.

[18] Andrew Zisserman. Vanguard project. http://www.robots.ox.ac.uk/~vanguard/.

Bibtex

```
@mastersthesis{SamuelCarlsson2005,
  author = {Samuel Carlsson},
  title = {Point cloud reconstruction from image sequences ---
    Calibrated versus uncalibrated}
  year = 2005,
  school = {Umeå Universitet, Department of Computing Science}
}
```