Homography-Based Positioning and Planar Motion Recovery

Wadenbäck, Mårten

2017

Document Version:
Publisher's PDF, also known as Version of record

– CENTRUM SCIENTIARUM MATHEMATICARUM –

Homography-Based Positioning and Planar Motion Recovery

MÅRTEN WADENBÄCK

Lund University
Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Homography-Based Positioning and Planar Motion Recovery

Mårten Wadenbäck

LUND UNIVERSITY

ACADEMIC THESIS

which, by due permission of the Faculty of Engineering at Lund University, will be publicly defended on Friday the 7th of April 2017, at 13:15 in lecture hall MH:Hörmander, Matematikhuset, Sölvegatan 18, Lund, for the degree of Doctor of Philosophy in Engineering.

Faculty opponent
Dr. Juho Kannala, Aalto University, Finland

Organisation
LUND UNIVERSITY
Centre for Mathematical Sciences

Box 118
SE-221 00 Lund

Sweden

Author(s)
Mårten Wadenbäck

Document name
DOCTORAL THESIS IN MATHEMATICAL SCIENCES

Date of defence
2017-04-07

Sponsoring organisation

Title and subtitle
Homography-Based Positioning and Planar Motion Recovery

Abstract
Planar motion is an important and frequently occurring situation in mobile robotics applications. This thesis concerns estimation of ego-motion and pose of a single downwards oriented camera under the assumptions of planar motion and known internal camera parameters. The so called *essential matrix* (or its uncalibrated counterpart, the *fundamental matrix*) is frequently used in computer vision applications to compute a reconstruction in 3D of the camera locations and the observed scene. However, if the observed points are expected to lie on a plane – e.g. the ground plane – this makes the determination of these matrices an ill-posed problem. Instead, methods based on homographies are better suited to this situation.

One section of this thesis is concerned with the extraction of the camera pose and ego-motion from such homographies. We present both a direct SVD-based method and an iterative method, which both solve this problem. The iterative method is extended to allow simultaneous determination of the camera tilt from several homographies obeying the same planar motion model. This extension improves the robustness of the original method, and it provides consistent tilt estimates for the frames that are used for the estimation. The methods are evaluated using experiments on both real and synthetic data.

Another part of the thesis deals with the problem of computing the homographies from point correspondences. By using conventional homography estimation methods for this, the resulting homography is of a too general class and is not guaranteed to be compatible with the planar motion assumption. For this reason, we enforce the planar motion model at the homography estimation stage with the help of a new homography solver using a number of polynomial constraints on the entries of the homography matrix. In addition to giving a homography of the right type, this method uses only 2.5 point correspondences instead of the conventional four, which is good e.g. when used in a RANSAC framework for outlier removal.

Keywords
Homography, Planar Motion, SLAM, Motion Estimation

Classification system and/or index terms (if any)

Supplementary bibliographical information

Language
English

ISSN and key title
1404-0034

ISBN
978-91-7753-153-1 (printed)
978-91-7753-154-8 (electronic)

Recipient's notes

Number of pages
xvi+115

Price
Peppercorn

Security classification

HOMOGRAPHY-BASED POSITIONING AND PLANAR MOTION RECOVERY

MÅRTEN WADENBÄCK

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Cover image: Küchensee seen from Ratzeburg Kurpark on the 30th of April 2015

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden
http://www.maths.lu.se/

Doctoral Theses in Mathematical Sciences 2017:3
ISSN 1404-0034

ISBN 978-91-7753-153-1 (printed)
ISBN 978-91-7753-154-8 (electronic)
LUTFMA-1064-2017

Printed in Sweden by MediaTryck, Lund 2017

Abstract

Planar motion is an important and frequently occurring situation in mobile robotics applications. This thesis concerns estimation of ego-motion and pose of a single downwards oriented camera under the assumptions of planar motion and known internal camera parameters. The so called *essential matrix* (or its uncalibrated counterpart, the *fundamental matrix*) is frequently used in computer vision applications to compute a reconstruction in 3D of the camera locations and the observed scene. However, if the observed points are expected to lie on a plane – e.g. the ground plane – this makes the determination of these matrices an ill-posed problem. Instead, methods based on homographies are better suited to this situation.

One section of this thesis is concerned with the extraction of the camera pose and ego-motion from such homographies. We present both a direct SVD-based method and an iterative method, which both solve this problem. The iterative method is extended to allow simultaneous determination of the camera tilt from several homographies obeying the same planar motion model. This extension improves the robustness of the original method, and it provides consistent tilt estimates for the frames that are used for the estimation. The methods are evaluated using experiments on both real and synthetic data.

Another part of the thesis deals with the problem of computing the homographies from point correspondences. By using conventional homography estimation methods for this, the resulting homography is of a too general class and is not guaranteed to be compatible with the planar motion assumption. For this reason, we enforce the planar motion model at the homography estimation stage with the help of a new homography solver using a number of polynomial constraints on the entries of the homography matrix. In addition to giving a homography of the right type, this method uses only 2.5 point correspondences instead of the conventional four, which is good e.g. when used in a RANSAC framework for outlier removal.

Popular Science Summary

The introduction of the robotic arm in the early 1960s was an important step towards automation of industry. Much of the noisy, dangerous, and repetitious labour at assembly lines and blast furnaces could now be performed by a robotic operator instead of a human. The robots were very often individually calibrated and programmed each to perform their specific task – tasks which were now performed faster than ever, and without risking fatigue or injuries.

However, despite the increased productivity, the improved safety, and the standardised output, the robots lacked crucial skills which the earlier human operators possessed in great measure – flexibility and adaptability. In contrast to their human counterparts, the robots were highly stationary and fixated to their workspace, often bolted to the floor or at best moving along short rails, and even minor changes to their task specification required a complete reprogramming and thus time offline.

Over the past half century, efforts to endow robots with flexibility and adaptability have been major themes in robotics research. The work in this thesis is a part of those efforts, and deals with an important sub-problem called *Simultaneous Localisation and Mapping* (SLAM). Algorithms for the SLAM problem use data which the robot acquire from its sensors (sonar, laser range finders, cameras, wheel encoders, …) to determine and keep track of the surrounding environment. In other words, the robot has to create a model of its surroundings (the mapping part), and use it to determine its own position (the localisation part) relative to the model. This type of algorithms is necessary in order to enable mobile robots to move autonomously – that is, without a human operator actively controlling the robot.

Only in the last two or three decades have cameras become a realistic choice of sensor to use for robotic navigation. There are three major reasons for this. First, digital cameras have become available, and they have gone through a revolution in terms of both reduced price and improved quality. Secondly, computing power has continued to double approximately every second year, as predicted by the celebrated *Moore's law*. Thirdly, during

(a) Original image. (b) Transformed image.

Figure 1: Two images of the same planar object are related through a projective transformation called a *homography*. Here, image (b) is a synthetic view generated from image (a) by applying a suitable homography transformation.

this time, there were many milestone advances in *computer vision*, which provided practical methods for inferring geometry from images.

In this thesis I have considered the case of a camera mounted rigidly onto a mobile robot platform that moves across a planar floor, and directed in such a way that the images mainly contain the floor. By keeping track of how certain observed points move in the images as the robot moves over the floor, it is possible to deduce how the robot has moved.

If the observed points on the floor plane are visible in two different images, their coordinates in the the two images are related through a transformation called a *homography* (see Figure 1). A homography between two images can be determined in a standard way from at least four pairs of corresponding points in the two images. One part of my work is concerned with extracting motion information from such homographies, and this is done by deriving a mathematical expression for the homography in terms of a number of motion parameters, and then solving for the parameters. My conclusion for this part is that the parameters can be determined reliably and efficiently, and that therefore the robot motion can be inferred from the floor images.

Homographies constitute a very general class of transformations, and by using the conventional method of finding the homography, one is not guaranteed to find a homography that actually is compatible with the assumptions one has made – planar camera motion and constant tilt (rigidly mounted camera). One part of this thesis considers the problem of finding a homography with those additional assumptions met. The way I solved this problem was to formulate those assumptions as so called *polynomial constraints*, and then using those polynomial constraints when devising a new method for finding a homography. My findings for this part of the work are that it is possible to find a homography which is compatible with the assumptions of planar motion and constant tilt, and that this can be done using only three pairs of corresponding points (compared to four, in the standard method).

Author's Contributions

The present thesis is based on material from the works listed below. Each item is followed by a brief description of the motivation behind the work, along with a declaration of the roles played in its genesis by myself, my co-authors, and others.

A *Planar Motion and Hand-Eye Calibration Using Inter-Image Homographies from a Planar Scene* (Wadenbäck and Heyden 2013). The motivation behind this work was that we wanted to recover the planar motion parameters from a previously estimated homography matrix. Anders Heyden proposed the problem and suggested the alternating optimisation approach, and I worked out the details, produced the implementation, conducted the experiments, and wrote the paper.

B *Ego-Motion Recovery and Robust Tilt Estimation for Planar Motion Using Several Homographies* (Wadenbäck and Heyden 2014a). In this paper we wanted to address stability and robustness issues experienced with the previous method, particularly for short pure translational motions. Anders Heyden and I discussed a temporal filtering approach, but we found that we could instead make use of temporal data internally using a modification of the previous method. I implemented the method, ran the experiments, and wrote the paper.

C *Visual Odometry from Two Point Correspondences and Initial Automatic Camera Tilt Calibration* (Wadenbäck et al. 2017). In this paper we wanted to exploit the fact that the camera tilt is constant over time by performing a short initial calibration and then estimating the motion directly from point correspondences instead of from a homography matrix. I implemented the algorithms, and Martin Karlsson and I did all the experimental work and the writing.

D *Trajectory Estimation Using Relative Distances Extracted from Inter-Image Homographies* (Wadenbäck and Heyden 2014b). The purpose of this paper was to investigate an alternative method to extract information about the camera motion without having to compute the

rotations. I derived the method, Anders Heyden helped me sort out a few edge cases, I produced the implementation, and I wrote the paper.

E *A Result for Orthogonal Plus Rank-1 Matrices* (Wadenbäck 2015). At the time I was running the experiments for the previous paper, I discovered numerically that one of the formulae I was using seemed to work even without some of the assumptions I had made. I had initial discussions with Keith Hannabuss and Anders Heyden about my observations, but in the end I formulated and proved the result alone, and wrote up the note.

F *Recovering Planar Motion from Homographies Obtained using a 2.5-Point Solver for a Polynomial System* (Wadenbäck, Åström and Heyden 2016). For this paper we wanted to find a method to estimate a homography constrained by our planar motion model, instead of using a general unconstrained homography. I found a number of polynomial constraints, which Kalle Åström helped me investigate and use in the construction of a solver. I performed the numerical experiments and wrote most of the paper, while Kalle Åström made theoretical investigations of the constraints and wrote that section of the paper.

G *Manuscript in Preparation*. This manuscript contains further experiments and investigations which were intended for the paper in F but which were omitted due to lack of space. For example, the homography bundle adjustment method described in Chapter 8 is included here. While I have had useful discussions with Kalle Åström and Anders Heyden about the content, so far I have done the actual work alone.

Acknowledgements

First and foremost, I would like to thank my supervisors *Anders Heyden* and *Kalle Åström* for all their guidance, patience, helpful discussion, and encouragement over the years. It has been an utter privilege to work with you both and be given the opportunity to absorb some of the knowledge and enthusiasm that you are always so generous to share.

To my *colleagues at the Centre for Mathematical Sciences*, thank you all! I have had interesting, helpful, and/or entertaining discussions with almost all of you at some point or other during my time here. There is an old saying, 'a truly wise man surrounds himself with even wiser peers'. Regardless of whether this is a sufficient condition for being truly wise, or merely a necessary one, the department is indeed a good place to be in for anyone who harbours aspirations to be truly wise.

Martin Karlsson at the Department of Automatic Control, thank you for being a great friend, collaborator, and co-author, and thank you for all your help with the experimental work in the robotics lab. Our work has at times been hazardous (electric shock) and frustrating (cable issues, network problems, …), but it has never been boring.

Professor *Ian Reid*, thank you for inviting me to spend some time with your research group at the University of Adelaide and for giving me new perspectives on my own work. You and the other members of the group made me feel very welcome and appreciated, and my visit provided me with valuable input, inspiration, and interesting ideas for future developments and investigations.

Tobias Palmér, with whom I have shared office these five years, thank you for being an excellent friend and desk neighbour. Our countless discussions about mathematics, music, research, travels, courses, life in general, and so on in between all the focused work have been much appreciated, and our many whiteboard sessions have made some concepts in mathematics become somewhat less obscure to me. Perhaps most important of all, you and Sofia took good care of Sinus while I was in Australia. Thank you!

Finally, I gratefully acknowledge the financial support which has made my research possible. This funding has come from the *Swedish Foundation*

for Strategic Research through the project ENGROSS and from the *Swedish Research Council* through the grants SRC 2011-6150 and SRC 2015-5639. Additional funding for some of my travels has been provided by *Thorild Dahlgrens stipendiefond* and *Landshövding Per Westlings minnesfond*.

Contents

Chapter 1

Matrices and Estimation

This chapter includes background material from linear algebra and estimation theory that is needed in the later chapters. In Section 1.1 we discuss various matrix related results, notably the *singular value decomposition*. Section 1.2 contains a brief introduction to non-linear least-squares estimation, and discusses the RANSAC framework for outlier removal.

1.1 Matrix Results and Notation

A great many mathematical results, models, and methods are best expressed in terms of matrices. A matrix formulation often invites the application of results and methods from some of the most well-developed areas of mathematics, notably *linear algebra* and *operator theory*, and this can be incredibly helpful in the analysis at hand. Furthermore, when it comes to implementation in software, the numerical aspects of matrix computations are generally well understood (Golub and Van Loan, 1996; Trefethen and Bau, 1997), and high quality implementations of many fundamental 'building block' algorithms are available through LAPACK (Anderson et al., 1999).

In this thesis, matrices are typically written as boldface Roman or Greek letters, where capital letters denote matrices which (may or do) have more than one column, and where miniscule letters denote vectors (i.e. column matrices). Exceptions to these guiding principles do occur, but the reader should be able to deduce the correct meaning from the context without too much effort. When the correct size can be inferred from the context, identity matrices are written as $\boldsymbol{I}$, and zero matrices are written as $\mathbf{0}$. In the

most ambiguous cases, there might be a subscript indicating the intended size. In block matrices, zero blocks are sometimes left blank, as this better highlights the structure of the non-zero parts. Unless stated otherwise, $\|\cdot\|$ refers to the Euclidean norm (for vectors) or the operator norm induced by the Euclidean norm (for matrices).

1.1.1 The Singular Value Decomposition

In this section we shall introduce and discuss one of the most useful matrix decompositions, namely the *singular value decomposition* (SVD). The theoretical and practical importance of this decomposition cannot be sufficiently emphasised, and we cannot possibly give a full discussion of its properties and applications here. The SVD is defined through the following existence theorem.

Theorem 1.1 (Singular Value Decomposition). *Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, and denote $p = \min(m, n)$. Then there exist orthogonal $\boldsymbol{U} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times n}$, and a uniquely determined diagonal matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}$ whose diagonal entries are the **singular values** $\sigma_1 \geq \ldots \geq \sigma_p \geq 0$, such that*

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T. \tag{1.1}$$

*Any such decomposition of $\boldsymbol{A}$ is a **singular value decomposition** (SVD). If the singular values are distinct, the first p columns in $\boldsymbol{U}$ and $\boldsymbol{V}$ are uniquely determined up to a sign ambiguity.*

Quite often this theorem is stated for complex matrices, in which case $\boldsymbol{U}$ and $\boldsymbol{V}$ should be *unitary* rather than orthogonal. The matrices in this thesis are almost exclusively real-valued, however, and for this reason, the above formulation is better suited to our purposes than the more general complex version.

It can also be good to know that there is a so called *reduced SVD*, which is often used in practice due to it being cheaper to compute in many cases. In the reduced SVD, $\boldsymbol{\Sigma}$ is a square matrix of order p, and $\boldsymbol{U}$ and $\boldsymbol{V}$ are matrices of sizes $m \times p$ and $n \times p$, respectively, with orthonormal columns. Most of the theory concerning the 'full' SVD can easily be translated to the reduced case.

While Theorem 1.1 guarantees the existence of an SVD, it is not immediately clear how one practically can compute an SVD of a matrix. We will not cover any such method here, but we take great comfort in the knowledge that there exist efficient and reliable methods for this purpose (Golub and Van Loan, 1996; Trefethen and Bau, 1997).

Proofs of Theorem 1.1 can be found throughout the literature, e.g. in Horn and Johnson (1991), Golub and Van Loan (1996), and Trefethen and Bau (1997). Before discussing some useful properties of the SVD, we sententiously remark that while 'the SVD' can be used to refer to the decomposition in general, using the definite article and referring to 'the' SVD of a particular matrix should generally be avoided unless it is clear which one of the many possible SVDs is intended.

One very important observation is the following.

Theorem 1.2. *Let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be an SVD of $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and let $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$ denote column k in $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. If σ_r is the smallest non-zero singular value, then $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r\}$ is an orthonormal basis for the range (column space) of $\boldsymbol{A}$ and $\{\boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_n\}$ is an orthonormal basis for the null space of $\boldsymbol{A}$. Consequently, $r = \operatorname{rank}\boldsymbol{A}$ is precisely the number of non-zero singular values.*

Proof. First, note that the columns in $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal by definition. Now, by inspection,

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{x} = \begin{bmatrix}\sigma_1\boldsymbol{u}_1 & \ldots & \sigma_r\boldsymbol{u}_r & \boldsymbol{0} & \ldots & \boldsymbol{0}\end{bmatrix}\boldsymbol{V}^T\boldsymbol{x}, \tag{1.2}$$

which means that $\boldsymbol{A}\boldsymbol{x}$ is a linear combination of the r first (i.e. leftmost) columns in $\boldsymbol{U}$. This gives $\operatorname{rank}\boldsymbol{A} = r$, and subsequently the null space must have dimension $n - r$. But the $n - r$ vectors $\boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_n$ must lie in the null space, since

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \boldsymbol{U}\begin{bmatrix}\sigma_1\boldsymbol{v}_1 & \ldots & \sigma_r\boldsymbol{v}_r & \boldsymbol{0} & \ldots & \boldsymbol{0}\end{bmatrix}^T, \tag{1.3}$$

and since they are orthonormal, this concludes the proof. □

Low-Rank and Null Space Approximation

As we saw in Theorem 1.2, the singular value decomposition is an example of a rank-revealing matrix factorisation. The rank of a given matrix is not a continuous function of the entries, and this makes the numerical computation of the rank a very ill-conditioned problem. However, it can be shown that the singular values depend continuously on the entries, and one can then define the *effective rank* to be the number of singular values which are significantly different from zero. The effective rank is in general much less sensitive, and it can be determined reliably by computing an SVD. With all this said, the terminological distinction between the rank and the effective rank is not always strictly maintained outside the courtroom.

The SVD is not only useful for *computing* the (effective) rank of a matrix; it is also a potent device for approximating a given matrix by another matrix of lower rank, i.e. *assigning* a particular (lower) rank to a matrix. More precisely, we have the following theorem.

Theorem 1.3 (Low-Rank Approximation). *Suppose the rank of $A \in \mathbb{R}^{m\times n}$ is equal to r and let $A = U\Sigma V^T$ be an SVD of A. Then, for any $0 \leq \nu < r$ it holds that the (unique) solution to the optimisation problem*

$$\begin{aligned} \underset{X}{\text{minimise}} \quad & \|X - A\| \\ \text{subject to} \quad & \operatorname{rank} X \leq \nu \end{aligned} \tag{1.4}$$

is given by $A_\nu = U\Sigma_\nu V^T$, where Σ_ν is obtained from Σ by replacing the $r-\nu$ smallest non-zero entries with zero.

Proof. Suppose there exists some X with $\operatorname{rank} X \leq \nu$ such that

$$\|X - A\| < \|A_\nu - A\| = \|\Sigma_\nu - \Sigma\| = \sigma_{\nu+1}. \tag{1.5}$$

The null space of X must have dimension $n-\nu$, and for any x in this null space we have

$$\|Ax\| = \|(A - X)x\| \leq \|A - X\|\,\|x\| < \sigma_{\nu+1}\|x\|. \tag{1.6}$$

But for any x in the $(\nu+1)$-dimensional subspace spanned by the first $\nu+1$ columns of V we have $\|Ax\| \geq \sigma_{\nu+1}\|x\|$. This is clearly a contradiction, and we are impelled to the conclusion that there can be no such X. □

A low-rank approximation is particularly useful when the matrix entries are formed from some kind of measurements, and the underlying mathematical model dictates a particular matrix rank ν. Measurement inaccuracies, or *noise*, will often increase the rank of such a matrix. Loosely speaking, a low-rank approximation reveals the principal directions $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_\nu$ of the underlying data and projects the data on the subspace spanned by these directions, thereby achieving in effect a noise suppression.

In the same fashion, and equally important, Theorem 1.2 and Theorem 1.3 can be combined to give the best (in the Euclidean norm) approximation of the null space of a matrix. This can be used for finding non-trivial solutions to homogeneous systems of equations, sometimes referred to as *homogeneous least-squares problems* (Inkilä, 2005). This method of approximating the null space will be used frequently in the chapters ahead.

Linear Least-Squares and Pseudo-Inverse

Suppose $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. A fundamental problem in linear algebra is the *(linear) least-squares* (LS) problem

$$\underset{\boldsymbol{x}}{\text{minimise}} \quad \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|^2. \tag{1.7}$$

In many cases least-squares problems arise directly out of linear model equations, e.g. Kirchhoff's circuit laws in electrical engineering and Newton's laws of motion in classical mechanics. In other cases they are used as approximations to non-linear problems, which we will see an example of in our discussion of the Levenberg-Marquardt method in Section 1.2.1.

Before trying to solve the problem (1.7), we note that the existence of a minimiser is guaranteed since $\mathbb{R}^n$ is complete and the norm is continuous and bounded from below. There may, however, be more than one minimiser, and we are interested in finding them all.

Now suppose $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is an SVD of $\boldsymbol{A}$, and suppose $\operatorname{rank}\boldsymbol{A} = r$. Since $\boldsymbol{U}$ is orthogonal, it does not affect the norm, and we have

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|^2 = \big\|\underbrace{\boldsymbol{U}^T\boldsymbol{b}}_{=\boldsymbol{c}} - \boldsymbol{\Sigma}\underbrace{\boldsymbol{V}^T\boldsymbol{x}}_{=\boldsymbol{y}}\big\|^2 = \sum_{j=1}^{r}(c_j - \sigma_j y_j)^2 + \sum_{j=r+1}^{m} c_j^2. \tag{1.8}$$

This expression is clearly minimised when $\boldsymbol{y} = \left(\frac{c_1}{\sigma_1}, \ldots, \frac{c_r}{\sigma_r}, s_{r+1}, \ldots, s_m\right)$ for some arbitrary $s_{r+1}, \ldots, s_m$ which parametrise the minimiser. Now let $\boldsymbol{S}$ be the diagonal $n \times m$ matrix with $\sigma_1^{-1}, \ldots, \sigma_r^{-1}$ as its non-zero diagonal entries, and let $\boldsymbol{v}_k$ denote column k in $\boldsymbol{V}$. The solutions to the least-squares problem (1.7) may then be written as

$$\boldsymbol{x} = \boldsymbol{V}\boldsymbol{y} = \boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^T\boldsymbol{b} + s_{r+1}\boldsymbol{v}_{r+1} + \ldots + s_m\boldsymbol{v}_m. \tag{1.9}$$

As we saw earlier in Theorem 1.2, the vectors $\boldsymbol{v}_{r+1}, \ldots, \boldsymbol{v}_m$ span the null space of $\boldsymbol{A}$, and it is of course not surprising that we can add any linear combination of them without influencing $\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|^2$. The matrix $\boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^T$ that showed up in the solution is important.

Definition 1.1. *Let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be an SVD of $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. Let r be the number of non-zero singular values, and let $\boldsymbol{S}$ be the diagonal $n \times m$ matrix with $\sigma_1^{-1}, \ldots, \sigma_r^{-1}$ as its first r diagonal entries and with all other entries equal to zero. The matrix $\boldsymbol{A}^+ = \boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^T$ is called the (Moore-Penrose)* ***pseudo-inverse*** *of $\boldsymbol{A}$.*

A few remarks concerning $\boldsymbol{A}^+$ are in order. First, any SVD of $\boldsymbol{A}$ will give the same $\boldsymbol{A}^+$, i.e. $\boldsymbol{A}^+$ is uniquely determined by $\boldsymbol{A}$. Secondly, if $\boldsymbol{A}$ is square and invertible in the usual sense, then $\boldsymbol{A}^+$ coincides with the usual inverse. Thirdly, as we just saw, $\boldsymbol{x} = \boldsymbol{A}^+\boldsymbol{b}$ solves the least-squares problem (1.7), and it is the smallest such solution (in the Euclidean norm). Finally, it turns out that $\boldsymbol{\Sigma}^+ = \boldsymbol{S}$, and thus it is common to write the pseudo-inverse as $\boldsymbol{A}^+ = \boldsymbol{V}\boldsymbol{\Sigma}^+\boldsymbol{U}^T$ without any reference to $\boldsymbol{S}$.

We conclude this section by providing two alternative expressions for the pseudo-inverse, which hold on the condition that $\boldsymbol{A}$ has full rank. In this case we have

$$\boldsymbol{A}^+ = \begin{cases} (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T & \text{when } m \geq n, \\ \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1} & \text{when } m \leq n. \end{cases} \tag{1.10}$$

The verification of these is left as an easy exercise for the reader.

1.1.2 Block Matrices and Special Structure

When working with problems formulated in terms of matrices, there is sometimes some *special structure* present that can be used – often to great effect – e.g. in the analysis of a problem, to reduce the computational burden of an algorithm, or to simplify its implementation. While there are no sharp and clear criteria for what falls in the category of special structure, the following (sometimes overlapping) situations are typical examples:

- *Sparsity*. A sparse matrix is a matrix where most entries are zero. If a matrix is known to be sparse, and the *sparsity pattern* (i.e. where the possible non-zero entries are) is known, then many entries can safely be ignored. For instance, inverting a non-singular diagonal matrix of size $n \times n$ requires $O(n)$ operations, compared to $O(n^3)$ for a full matrix of the same size.

- *Block structure*. Some blocks in a matrix may be known to span or be orthogonal to certain subspaces (cf. the proof of Theorem 1.2), be known to be identical, be known to be factorisable in a particular way, etc. Knowledge of such block structure often enables a higher abstraction level, and can e.g. make an implementation in code easier to write and read.

- *Parametrisation*. A parametrisation of an object often provides useful insights. For example, there are various results, e.g. Theorem 1.4, that can be applied to matrices which can be written as sums where the terms have certain properties.

The aim of this section is to serve as a kind of rogues' gallery, containing some notation and useful facts for working with the situations above. The first such useful fact, which is readily verified by simply multiplying the relevant matrices together, is the *Sherman-Morrison-Woodbury formula* (Golub and Van Loan, 1996).

Theorem 1.4 (Sherman-Morrison-Woodbury formula). *If the matrices* $\boldsymbol{A}$ *and* $\boldsymbol{I} + \boldsymbol{C}^T\boldsymbol{A}^{-1}\boldsymbol{B}$ *are invertible, then* $\boldsymbol{A} + \boldsymbol{B}\boldsymbol{C}^T$ *is also invertible and its*

inverse is given by

$$(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{C}^T)^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{I}+\boldsymbol{C}^T\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{C}^T\boldsymbol{A}^{-1}. \tag{1.11}$$

A much-appreciated consequence of this is that if $\boldsymbol{A}$ is of order n and easy to invert (or the inverse has already been computed), and $\boldsymbol{B}$ and $\boldsymbol{C}$ are $n \times k$ with $k \ll n$, then applying the Sherman-Morrison-Woodbury formula allows $(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{C}^T)^{-1}$ to be computed in $O(n^2)$ operations instead of $O(n^3)$. In this thesis, however, where most of the square matrices are very small, considerations of time complexity are not the motivating factor for this theorem; instead, it is the invertibility condition and the explicit formula for the inverse that we are after.

Another useful result gives an expression for the inverse of a symmetric block matrix. Assuming all the occurring inverses exist, we have

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^T & \boldsymbol{C} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{S}^{-1} & -\boldsymbol{S}^{-1}\boldsymbol{B}\boldsymbol{C}^{-1} \\ -\boldsymbol{C}^{-1}\boldsymbol{B}^T\boldsymbol{S}^{-1} & \boldsymbol{C}^{-1}+\boldsymbol{C}^{-1}\boldsymbol{B}^T\boldsymbol{S}^{-1}\boldsymbol{B}\boldsymbol{C}^{-1} \end{bmatrix}, \tag{1.12}$$

where $\boldsymbol{S} = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{-1}\boldsymbol{B}^T$ is the so called *Schur complement* (Boyd and Vandenberghe, 2004). Block inversion is a powerful tool when solving e.g. *normal equations* in least-squares problems (see Section 1.2.1), in particular when the variables can be partitioned into a group which occurs in all equations and a group where each variable only occurs in a small number of equations. This is typically the case in *bundle adjustment* in computer vision (Triggs et al., 1999).

A familiar concept in three-dimensional vector geometry is the cross product, which is a neat supplement to the inner product when it comes to expressing relations of orthogonality or parallelism. For any vector $\boldsymbol{a} \in \mathbb{R}^3$ the mapping $\boldsymbol{z} \mapsto \boldsymbol{a} \times \boldsymbol{z}$ is linear, and can thus be expressed as $\boldsymbol{z} \mapsto \left[\boldsymbol{a}\right]_\times \boldsymbol{z}$, with the *cross product matrix* defined as

$$\left[\boldsymbol{a}\right]_\times = \frac{\partial}{\partial \boldsymbol{z}}(\boldsymbol{a} \times \boldsymbol{z}) = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \tag{1.13}$$

The cross product matrix is *skew-symmetric*, i.e. $\left[\boldsymbol{a}\right]_\times{}^T = -\left[\boldsymbol{a}\right]_\times$, and has rank two except when $\boldsymbol{a} = \boldsymbol{0}$. Additionally, any skew-symmetric matrix of order three is clearly a cross product matrix.

If $\boldsymbol{R}(\varphi)$ is a 3×3 rotation matrix, which performs a rotation an angle φ around a fixed rotation axis, then

$$\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I} \Rightarrow \frac{d\boldsymbol{R}}{d\varphi}\boldsymbol{R}^T + \boldsymbol{R}\frac{d\boldsymbol{R}^T}{d\varphi} = \boldsymbol{0}. \tag{1.14}$$

This shows that $\frac{d\boldsymbol{R}}{d\varphi}\boldsymbol{R}^T$ is skew-symmetric, i.e. there is some $\boldsymbol{v}$ for which

$$\frac{d\boldsymbol{R}}{d\varphi}\boldsymbol{R}^T = \big[\boldsymbol{v}\big]_\times \Leftrightarrow \frac{d\boldsymbol{R}}{d\varphi} = \big[\boldsymbol{v}\big]_\times \boldsymbol{R} \Leftrightarrow \boldsymbol{R}(\varphi) = \exp\big(\varphi\big[\boldsymbol{v}\big]_\times\big). \tag{1.15}$$

Furthermore, $\boldsymbol{v}$ must be the rotation axis, since it is an eigenvector to

$$\exp\big(\varphi\big[\boldsymbol{v}\big]_\times\big) = \boldsymbol{I} + \varphi\big[\boldsymbol{v}\big]_\times + \frac{\varphi^2}{2}\big[\boldsymbol{v}\big]_\times^2 + \ldots \tag{1.16}$$

with corresponding eigenvalue one. This characterisation of 3D rotations is known as the *exponential map*, and is closely related to *Rodrigues' formula* (which is discussed in Section 2.2).

The *vectorisation* operation, or *column stacking*, is defined as

$$\operatorname{vec}\boldsymbol{A} = \begin{bmatrix} a_{11} & \cdots & a_{m1} & a_{12} & \cdots & a_{m2} & \cdots & a_{1n} & \cdots & a_{mn} \end{bmatrix}^T, \tag{1.17}$$

and the *Kronecker product* is defined as

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \cdots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \cdots & a_{mn}\boldsymbol{B} \end{bmatrix}. \tag{1.18}$$

The interplay between these two operations is discussed in considerable detail in Horn and Johnson (1991), but one particular result which we shall find useful is the following.

Lemma 1.1. *For matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ of compatible sizes, it holds that*

$$\operatorname{vec}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{C}) = (\boldsymbol{C}^T \otimes \boldsymbol{A}) \operatorname{vec}\boldsymbol{B}. \tag{1.19}$$

If we know the size of $\boldsymbol{A}$, e.g. $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, then the vectorisation operation is invertible, and we can define

$$\operatorname{vec}^{-1}_{m\times n}\boldsymbol{a} = ((\operatorname{vec}\boldsymbol{I}_n)^T \otimes \boldsymbol{I}_m)(\boldsymbol{I}_n \otimes \boldsymbol{a}). \tag{1.20}$$

For any positive integers m and n it then holds that

$$\boldsymbol{a} = \operatorname{vec}(\operatorname{vec}^{-1}_{m\times n}\boldsymbol{a}), \qquad \forall \boldsymbol{a} \in \mathbb{R}^{mn}, \tag{1.21}$$

and

$$\boldsymbol{A} = \operatorname{vec}^{-1}_{m\times n}(\operatorname{vec}\boldsymbol{A}), \qquad \forall \boldsymbol{A} \in \mathbb{R}^{m\times n}. \tag{1.22}$$

1.1.3 A Note on Differentiation

Differential calculus with vector-valued functions which depend on a vector $\boldsymbol{x} \in \mathbb{R}^n$ is conceptually straightforward, but requires attention to details such as what to transpose and in which order various objects should appear. Most differentiation in this thesis can be reduced to the application of two familiar results, namely the *product rule*

$$\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{u}^T\boldsymbol{v}) = \boldsymbol{u}^T\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}} + \boldsymbol{v}^T\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}, \tag{1.23}$$

which holds whenever both $\boldsymbol{u}$ and $\boldsymbol{v}$ are differentiable functions from $\mathbb{R}^n$ to $\mathbb{R}^m$, and the *chain rule*

$$\frac{\partial \boldsymbol{w}}{\partial \boldsymbol{x}} = \frac{\partial \boldsymbol{w}}{\partial \boldsymbol{z}}\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}}, \tag{1.24}$$

where $\boldsymbol{z} : \mathbb{R}^n \to \mathbb{R}^p$ and $\boldsymbol{w} : \mathbb{R}^p \to \mathbb{R}^m$ are differentiable.

1.2 Parameter Estimation

One problem which arises time and again in mathematics and its applications is that of tuning a set of parameters detailing some mathematical model in such a way that they 'best' explain a set of measurements. Here

we shall assume that the perhaps most important part – namely, the selection of a suitable model class – has already been done, and that we thus have a model which states that $\boldsymbol{y} \in \mathbb{R}^m$ depends on $\boldsymbol{x} \in \mathbb{R}^n$ as

$$\boldsymbol{y} = \phi(\boldsymbol{x}; \boldsymbol{\beta}). \tag{1.25}$$

The function ϕ is determined by the model class, and is parametrised by the parameter vector $\boldsymbol{\beta} \in \mathbb{R}^p$. Given some data $\{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=1}^N$, the goal is to find a parameter vector $\boldsymbol{\beta}$ which makes the model (1.25) agree as well as possible with the provided data.

Occasionally it is possible to find one or more parameter vectors $\boldsymbol{\beta}$ for which $\boldsymbol{y}_j = \phi(\boldsymbol{x}_j; \boldsymbol{\beta})$ *exactly* for all $j = 1, \ldots, N$, but in many practical cases one has to tolerate some discrepancies, or *residuals*,

$$\boldsymbol{r}_j(\boldsymbol{\beta}) = \boldsymbol{y}_j - \phi(\boldsymbol{x}_j; \boldsymbol{\beta}), \quad j = 1, \ldots, N. \tag{1.26}$$

The presence of such discrepancies does not in itself indicate an inadequacy of the selected model, but may often rather be attributed to inaccurate measurements, so called *outliers* (i.e. measurements which are simply incompatible with the model or which are mislabelled – the samples which are not outliers are called *inliers*), or to the influence of phenomena which are difficult or even impossible to account for in any kind of model. In fact, if one manages to find a $\boldsymbol{\beta}$ which makes all the residuals zero, this could potentially be a symptom either of so called *overfitting* (i.e. a too general model with too many degrees of freedom) or of a too small data sample (Bishop, 2006), and one should then carefully review the modelling step before proceeding to draw any conclusion.

A natural desire, however, is to have as small discrepancies as one can possibly manage without suffering from overfitting. To this end, it is often useful to try to minimise (with regards to $\boldsymbol{\beta}$) some kind of *cost function*, which produces a scalar measure of the size of the residuals $\boldsymbol{r}_j$. All the residuals are often collected in a *residual vector*

$$\boldsymbol{\rho}(\boldsymbol{\beta}) = \begin{bmatrix} \boldsymbol{r}_1(\boldsymbol{\beta}) \\ \vdots \\ \boldsymbol{r}_N(\boldsymbol{\beta}) \end{bmatrix}. \tag{1.27}$$

The prototypical, and most influential cost function by far, is the sum of squared errors,

$$E_{\mathrm{LS}}(\boldsymbol{\beta}) = \boldsymbol{\rho}(\boldsymbol{\beta})^T \boldsymbol{\rho}(\boldsymbol{\beta}) = \sum_{j=1}^{N} \left\| \boldsymbol{r}_j(\boldsymbol{\beta}) \right\|^2, \tag{1.28}$$

famously pioneered in the early 19th century by Legendre (1805) and Gauß, and which gives rise to a so called *(non-linear) least-squares* (LS) problem. While this cost function is blessed with surprisingly many good properties, its Achilles' heel is no doubt its extreme sensitivity to outliers. Indeed, even a single outlier may in some cases render the optimal $\boldsymbol{\beta}$ useless, and it is therefore common that the use of this cost function is preceded by an outlier removal scheme such as RANSAC (see Section 1.2.2).

In addition to the sum of squared errors, there are many other cost functions with different properties to choose from. Several of the most important ones are given an in-depth treatment in well-known reference works on optimisation, e.g. Boyd and Vandenberghe (2004), and discussing them here would be an unwarranted digression. Instead, let us briefly look at a useful method for minimising E_{LS}.

1.2.1 Non-Linear Least-Squares

Let us assume that ϕ is (continuously) differentiable with respect to $\boldsymbol{\beta}$ at each data point $\boldsymbol{x}_j$. The cost function E_{LS} is non-convex for most choices of ϕ, and because of this one cannot reasonably expect to find a global minimum even when it is guaranteed to exist. However, one can make successive convex approximations of E_{LS}, and by minimising these one hopefully comes close to a good local minimum of E_{LS}. Convex quadratic approximations readily suggest themselves for this purpose, as they are both easy to compute and easy to minimise (there is even a closed-form expression for the minimum).

A linearisation of the residuals around a point $\boldsymbol{\beta}_k$ gives, for small $\boldsymbol{\Delta}$,

$$\boldsymbol{\rho}(\boldsymbol{\beta}_k + \boldsymbol{\Delta}) \approx \boldsymbol{\rho}(\boldsymbol{\beta}_k) + \boldsymbol{J}(\boldsymbol{\beta}_k)\boldsymbol{\Delta}, \tag{1.29}$$

where $\boldsymbol{J}$ is the *Jacobian*

$$\boldsymbol{J}(\boldsymbol{\beta}_k) = \frac{\partial \boldsymbol{\rho}}{\partial \boldsymbol{\beta}} = \begin{bmatrix} \frac{\partial r_1}{\partial \boldsymbol{\beta}} \\ \vdots \\ \frac{\partial r_N}{\partial \boldsymbol{\beta}} \end{bmatrix}, \tag{1.30}$$

with all the derivatives evaluated at $\boldsymbol{\beta}_k$. By simply setting $\boldsymbol{\rho}(\boldsymbol{\beta}_k + \boldsymbol{\Delta}) = \mathbf{0}$ in (1.29), the step $\boldsymbol{\Delta}_k$ given by

$$\boldsymbol{\Delta}_k = -\boldsymbol{J}(\boldsymbol{\beta}_k)^+ \boldsymbol{\rho}(\boldsymbol{\beta}_k) \tag{1.31}$$

should hopefully make the residuals smaller, and thus give a lower value of the cost function E_{LS}. This choice of $\boldsymbol{\Delta}_k$ is called a *Gauß-Newton step*, and the method of starting at some initial point $\boldsymbol{\beta}_0$ and computing successive points as $\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \boldsymbol{\Delta}_k$ using (1.31) is known as the *Gauß-Newton method*. To ensure that the method terminates in finite time, and thus becomes an algorithm (Knuth, 1997), it is in practice always used with one or more *stopping criteria* (e.g. too small step size, too many iterations, etc.). We do not cover those here.

Another instructive way of deriving the Gauß-Newton step is to replace the residuals in E_{LS} with the linearisation in (1.29) and consider the resulting quadratic function $Q_k(\boldsymbol{\Delta}) \approx E_{\text{LS}}(\boldsymbol{\beta}_k + \boldsymbol{\Delta})$. This gives us the following convex quadratic

$$Q_k(\boldsymbol{\Delta}) = \boldsymbol{\rho}(\boldsymbol{\beta}_k)^T \boldsymbol{\rho}(\boldsymbol{\beta}_k) + 2\boldsymbol{\rho}(\boldsymbol{\beta}_k)^T \boldsymbol{J}(\boldsymbol{\beta}_k)\boldsymbol{\Delta} + \boldsymbol{\Delta}^T \boldsymbol{J}(\boldsymbol{\beta}_k)^T \boldsymbol{J}(\boldsymbol{\beta}_k)\boldsymbol{\Delta}. \tag{1.32}$$

A minimum of Q_k is clearly attained when $\frac{\partial Q_k}{\partial \boldsymbol{\Delta}} = \mathbf{0}$, i.e. when $\boldsymbol{\Delta}$ satisfies the *normal equations*

$$\boldsymbol{J}(\boldsymbol{\beta}_k)^T \boldsymbol{\rho}(\boldsymbol{\beta}_k) + \boldsymbol{J}(\boldsymbol{\beta}_k)^T \boldsymbol{J}(\boldsymbol{\beta}_k)\boldsymbol{\Delta} = \mathbf{0}. \tag{1.33}$$

A minimiser of Q_k can thus be obtained as

$$\boldsymbol{\Delta}_k = \arg\min_{\boldsymbol{\Delta}} Q_k(\boldsymbol{\Delta}) = -\big(\boldsymbol{J}(\boldsymbol{\beta}_k)^T \boldsymbol{J}(\boldsymbol{\beta}_k)\big)^+ \boldsymbol{J}(\boldsymbol{\beta}_k)^T \boldsymbol{\rho}(\boldsymbol{\beta}_k), \tag{1.34}$$

which can be verified to be equal to the step in (1.31).

One problem with the Gauß-Newton method is that the $\boldsymbol{\Delta}_k$ computed from (1.31) or (1.34) might be so large that it lies outside the region where Q_k is a reasonable approximation of E_{LS}. In order to still go in the right direction, but not go too far, Levenberg proposed augmenting the normal equations by replacing $\boldsymbol{J}(\boldsymbol{\beta}_k)^T\boldsymbol{J}(\boldsymbol{\beta}_k)$ with $\boldsymbol{J}(\boldsymbol{\beta}_k)^T\boldsymbol{J}(\boldsymbol{\beta}_k) + \lambda_k\boldsymbol{I}$ for some damping parameter $\lambda_k \geq 0$ (Levenberg, 1944), and this was subsequently refined by Marquardt (1963) to instead use the augmented normal equations

$$\boldsymbol{J}(\boldsymbol{\beta}_k)^T\boldsymbol{\rho}(\boldsymbol{\beta}_k) + \big(\boldsymbol{J}(\boldsymbol{\beta}_k)^T\boldsymbol{J}(\boldsymbol{\beta}_k) + \lambda_k\boldsymbol{\Lambda}_k\big)\boldsymbol{\Delta} = \mathbf{0}, \tag{1.35}$$

where $\boldsymbol{\Lambda}_k$ is diagonal and has the same diagonal entries as $\boldsymbol{J}(\boldsymbol{\beta}_k)^T\boldsymbol{J}(\boldsymbol{\beta}_k)$.

The solution to the augmented normal equations now depends on the damping parameter λ_k as

$$\boldsymbol{\Delta}_k(\lambda_k) = -\big(\boldsymbol{J}(\boldsymbol{\beta}_k)^T\boldsymbol{J}(\boldsymbol{\beta}_k) + \lambda_k\boldsymbol{\Lambda}_k\big)^+\boldsymbol{J}(\boldsymbol{\beta}_k)^T\boldsymbol{\rho}(\boldsymbol{\beta}_k), \tag{1.36}$$

and one wants to find a λ_k such that $E_{\mathrm{LS}}\big(\boldsymbol{\beta}_k + \boldsymbol{\Delta}_k(\lambda_k)\big) < E_{\mathrm{LS}}(\boldsymbol{\beta}_k)$. It can be shown that this can always be done (as long as $\boldsymbol{\beta}_k$ is not a local minimum) by choosing λ_k very large, but this means taking a very small step towards the minimum, and a small λ_k is thus preferable. Usually, in practice, some kind of heuristic approach is used to choose the value of the damping parameter at each iteration.

This 'damped' version of the Gauß-Newton method is known as the *Levenberg-Marquardt method* for non-linear least-squares problems, and is outlined in Algorithm 1.1. The method can be thought of as an interpolation between the Gauß-Newton method and the *steepest descent method* (Boyd and Vandenberghe, 2004). Much more thorough treatments of the Levenberg-Marquardt method, particularly addressing computer vision applications and with considerations of various implementation aspects, can be found in Triggs et al. (1999) and to some extent in Hartley and Zisserman (2004).

1.2.2 Robust Estimation using RANSAC

As already mentioned, the cost function (1.28) is very sensitive to outliers. Unless the outliers are removed, the minimisation of E_{LS} will try to decrease

Input: E_{LS} of the form (1.28) and an initial point $\boldsymbol{\beta}_0$
Output: $\boldsymbol{\beta}^*$ (hopefully) close to a minimum of E_{LS}
1: $k \leftarrow 0$
2: **while** stopping criteria not reached **do**
3: Compute $\boldsymbol{J}(\boldsymbol{\beta}_k)^T\boldsymbol{J}(\boldsymbol{\beta}_k)$, $\boldsymbol{\Lambda}_k$, and $\boldsymbol{J}(\boldsymbol{\beta}_k)^T\boldsymbol{\rho}(\boldsymbol{\beta}_k)$
4: Define $\boldsymbol{\Delta}_k(\lambda_k)$ according to (1.36)
5: Choose λ_k such that $E_{\text{LS}}\big(\boldsymbol{\beta}_k + \boldsymbol{\Delta}_k(\lambda_k)\big) < E_{\text{LS}}(\boldsymbol{\beta}_k)$
6: $\boldsymbol{\beta}_{k+1} \leftarrow \boldsymbol{\beta}_k + \boldsymbol{\Delta}_k(\lambda_k)$
7: $k \leftarrow k + 1$
8: $\boldsymbol{\beta}^* \leftarrow \boldsymbol{\beta}_k$

Algorithm 1.1: The Levenberg-Marquardt method for non-linear least-squares problems works by using linearised residuals and solving augmented normal equations of the form (1.35). If $\lambda_k = 0$ for all k, the method is identical to the Gauß-Newton method, and if λ_k is very large, the method becomes similar to the *steepest descent method* (Boyd and Vandenberghe, 2004).

a large residual corresponding to an outlier at the expense of increasing (maybe even by a large measure) many of the smaller ones, which means that such bad samples have an unduly large influence on the estimated parameters $\boldsymbol{\beta}$. A much better estimate of $\boldsymbol{\beta}$ can often be obtained if one manages to identify such bad samples and exclude them from E_{LS}.

Fischler and Bolles proposed a framework called *RANdom SAmple Consensus* (RANSAC) for identifying and eliminating the outliers (Fischler and Bolles, 1981). Their idea was to fit the model to a small random subset of the data, and save the samples which do not disagree too much in a so called *consensus set*. By repeating this many times, and then fitting the model to the largest consensus set found so far, one can hopefully eliminate the outliers. The general procedure of RANSAC is shown in Algorithm 1.2.

Intuitively, if a subset of the data is chosen which only contains true inliers, most of the other true inliers should not deviate much from the estimated model, and they should therefore end up in the consensus set. If, on the other hand, some of the selected samples are not true inliers, then this will cause a poor model to be fitted and only a small number of samples

Input: Model $\boldsymbol{y} = \phi(\boldsymbol{x}; \boldsymbol{\beta})$, data $\{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=1}^{N}$, threshold τ, iterations K
Output: $\boldsymbol{\beta}$ fitted to the largest set of inliers

```
C_best ← ∅
for k = 1, ..., K do
    Select a small random subset of the data
    Fit the model to the selected subset (estimate β)
    C_k ← {(x, y) ∈ {(x_j, y_j)}_{j=1}^N | ||y − φ(x; β)|| < τ}
    if |C_k| > |C_best| then
        C_best ← C_k
Estimate β from C_best
```

Algorithm 1.2: The RANSAC framework is useful when estimating model parameters from noisy or corrupted data. The key idea is to repeatedly fit the model to a small subset and count how many samples belong to the *consensus set*, i.e. the set of samples which support the fitted model. This procedure hopefully finds a model which a large part of the data supports, and then as a final step the model is fitted to the largest consensus set found.

will – more or less 'by accident' – belong to the consensus set.

It is necessary to somehow determine the threshold τ deciding if a sample should belong to the consensus set or not, as well as the number of iterations K to run. In some situations it is possible to adaptively determine suitable values for τ and K (Hartley and Zisserman, 2004). Good automatic determination of τ is not straightforward, however, and instead one often uses some prior knowledge of how large errors one should expect. A commonly used method for determining K is to choose a desired success probability p and guess an inlier ratio w. In case the actual inlier ratio is at least w, then the probability of selecting a subset of size s containing only inliers will be at least p if one runs

$$K = \left\lceil \frac{\ln(1-p)}{\ln(1-w^s)} \right\rceil \tag{1.37}$$

iterations (Fischler and Bolles, 1981), where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x.

If one uses the smallest possible value of s, which of course depends on the model one tries to estimate, the probability of selecting a 'clean' subset is clearly maximised, and this s also gives the smallest K according to (1.37). Note that while the smallest possible s is a popular choice which works well in many instances, there are cases (especially when the data are very noisy) when larger-than-minimal subsets yield larger consensus sets, as explained in Pham et al. (2014) and Pham (2014).

Chapter 2

Camera Modelling

The main goal of this chapter is to introduce the *pinhole perspective camera model*, which we will discuss in Section 2.3. The pinhole perspective camera is best expressed using *homogeneous coordinates*, and we introduce these together with some additional projective geometry background in Section 2.1. Here we also define the important concept of a *homography* (projective transformation), and we find in Section 2.2.2 that rigid motions constitute a special class among the homographies. Section 2.4 briefly discusses the problem of finding corresponding points in two images, and acts as a bridge to the two-view material in Chapter 3.

2.1 Projective Geometry Background

The problems considered within this thesis, like many other problems in computer vision, are concerned with describing geometry. The most convenient and natural geometric framework to use when working with computer vision problems is the framework of so called *projective geometry*. While projective geometry is an interesting and fascinating field in and of itself, the purpose of this very brief introduction is mainly to provide sufficient background for working with homogeneous coordinates and to give a basic understanding of the concept of a *homography*.

2.1.1 Projective Spaces and Homogeneous Coordinates

We start by defining the fundamental structure where projective geometry takes place.

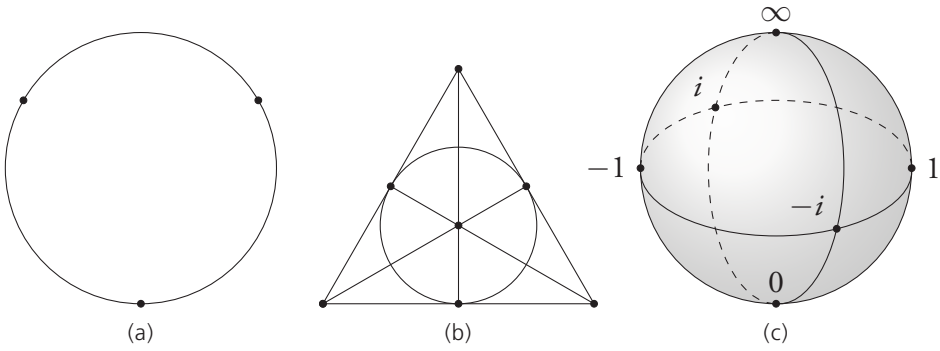


Figure 2.1: Some examples of projective spaces. Figure (a) shows a finite projective space generated by $\mathbb{Z}_2^2$ consisting of three points and one line. Figure (b) shows the much celebrated *Fano plane* which is generated by $\mathbb{Z}_2^3$ and consists of seven points and seven lines. Figure (c) tries to illustrate the *Riemann sphere*, which is generated by $\mathbb{C}^2$ and consists of infinitely many points and one line (the *extended complex line* $\mathbb{C} \cup \{\infty\}$).

Definition 2.1. *Let $V \ncong \{0\}$ be a vector space. The set of one-dimensional subspaces of V (i.e. lines through the origin) is called the* ***projective space*** *of dimension* $\dim V - 1$ *generated by V (Shafarevich, 2013; Tevelev, 2005).*

The elements of a projective space are usually referred to as *points*, regardless of how well they can be interpreted as actual geometrical points. One family of projective spaces where the geometrical interpretation is especially good, however, is the family of *real projective spaces* $\mathbb{RP}^n$ generated by $\mathbb{R}^{n+1}$. This is a consequence of $\mathbb{RP}^n$ being able to capture much of the structure present in the n-dimensional *extended Euclidean space*, that is, a Euclidean space extended with an *ideal point* (a point at infinity) in every direction. In computer vision (and in computer graphics), the 3D world, or *scene*, is usually modelled as $\mathbb{RP}^3$, and the image is modelled as $\mathbb{RP}^2$. There are countless additional examples of projective spaces, for instance the ones shown in Figure 2.1. While the discussion in this section is kept quite general, we are in the end essentially interested in $\mathbb{RP}^2$ and $\mathbb{RP}^3$.

A line through the origin of V can be represented by any of its direction vectors, and the coordinates of such direction vectors may thus be used to

identify the points in a projective space. This coordinate representation, called *homogeneous coordinates*, was pioneered by Möbius (1827) and has established itself as a natural framework for performing computations in projective spaces. In the following we will assume that all projective points are represented using homogeneous coordinates. In the specific case of the extended Euclidean spaces, an 'ordinary' point $\boldsymbol{x}$ will have homogeneous coordinates $(\lambda\boldsymbol{x}, \lambda)$ for any $\lambda \neq 0$, whereas the ideal point in the direction $\boldsymbol{v}$ will have homogeneous coordinates $(\lambda\boldsymbol{v}, 0)$ for any $\lambda \neq 0$.

In ordinary Euclidean geometry, and in the extended Euclidean spaces, it is well known that two distinct points uniquely determine a line. This is an important property, which we want to have in projective spaces as well. Two distinct points in a projective space are identified with two non-parallel lines through the origin of V, and since these will always span a unique two-dimensional subspace in V, a suitable definition of a line in a projective space would be the following:

Definition 2.2. *Let P be the projective space generated by a vector space V with* $\dim V \geq 2$. *A two-dimensional subspace of V is called a* ***line*** *in P.*

A useful consequence which follows immediately from this definition is that a line through two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in a projective space consists precisely of all points which may be written as non-zero linear combinations of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. The reader is encouraged to verify this for the examples in Figure 2.1 by introducing homogeneous coordinates there.

In the extended Euclidean plane (i.e. the two-dimensional extended Euclidean space) there exists a line containing all the ideal points, and no other point. This line is called the *ideal line* or the *line at infinity*.

In the same way as we defined points and lines, one could of course imagine defining additional geometrical objects corresponding to subspaces of higher dimensions. For some reason, however, it appears that such constructions are rarely considered in general. The typical exception, where such an object *is* considered, is the following.

Definition 2.3. *Let P be the projective space generated by a vector space V with finite dimension $n \geq 3$. A subspace of V with dimension $n-1$ is called*

*a **hyperplane** in P. (When $n = 3$, a hyperplane is the same as a line, and when $n = 4$ it is simply called a **plane**.)*

Similarly to the ideal line in the extended Euclidean plane, there is an *ideal plane* or *plane at infinity* in the three-dimensional extended Euclidean space. The same construction can of course be used to define an *ideal hyperplane* in any extended Euclidean space of higher dimension.

Any particular subspace of dimension $n-1$ is orthogonal to a unique one-dimensional subspace, and this gives a canonical bijection between points and hyperplanes in projective spaces. Using this bijection allows us to represent a hyperplane as a point $\boldsymbol{\pi}$, and the fact that a point $\boldsymbol{x}$ belongs to the hyperplane can then be expressed as

$$\boldsymbol{\pi}^T \boldsymbol{x} = 0. \tag{2.1}$$

These remarks form the essence of the so called *duality principle*, which is a very important concept in projective geometry (Busemann and Kelly, 1953). The duality principle states that for any projective theorem concerning points and hyperplanes, there is a dual theorem which can be obtained via the substitution scheme

$$\begin{array}{rcl} \text{point} & \leftrightarrow & \text{hyperplane} \\ \text{lies on} & \leftrightarrow & \text{contains} \end{array}$$

and correction of any grammatical awkwardness this may result in.

In the remainder of this thesis we will not go much beyond the above remarks when it comes to the duality principle. We will, however, benefit greatly from the algebraic description of a hyperplane (2.1).

2.1.2 Projective Cameras and Homographies

The developments in the previous section were made entirely in terms of the subspaces of a vector space V. Suppose W and V are vector spaces, and consider a mapping T from W to V which preserves the subspace structure (i.e. W is mapped onto V and every subspace in W is mapped onto some subspace in V). Then the mapping T gives rise to a corresponding mapping from the projective space generated by W to the one generated by V.

Any surjective linear transformation from W to V will preserve the subspace structure. This fact serves as inspiration for the following definition.

Definition 2.4. *Let T be a surjective linear transformation which maps a vector space $W \ncong \{0\}$ onto another vector space V. Then T is a **projection** from the projective space generated by W to the one generated by V. If T is also injective, T is called a **homography** between the two projective spaces. If $\dim W = 1 + \dim V$, then T is called a **projective camera**.*

In particular, homographies from $\mathbb{RP}^n$ to itself are non-singular square matrices of order $n+1$, and projective cameras from $\mathbb{RP}^n$ to $\mathbb{RP}^{n-1}$ are precisely the $(n+1) \times n$ matrices of rank n. Note also that there may be points in a projective space for which a particular projection is undefined. This situation cannot occur for homographies, and for projective cameras it only occurs for a single point.

Theorem 2.1. *For any projective camera, there is precisely one point for which the mapping is undefined.*

Proof. Since T is surjective and $\dim W = 1 + \dim V$, T must have a unique one-dimensional null space. The point represented by this null space would be mapped to $\mathbf{0}$, which does not represent a projective point. □

Definition 2.5. *The point in Theorem 2.1 for which the projective camera mapping is undefined is called the **camera centre**.*

From the definition of a projective camera, we know that it projects the whole space onto a hyperplane. This hyperplane is called the *image plane*.

2.1.3 The Direct Linear Transformation

In this section we will look at the important question of how to find a planar homography which maps certain given points in $\mathbb{RP}^2$ to certain other ones, in the cases when this is indeed possible. Suppose, therefore, that we want to find a homography which maps $\boldsymbol{x}_j = (x_j, y_j, z_j)$ to $\hat{\boldsymbol{x}}_j = (\hat{x}_j, \hat{y}_j, \hat{z}_j)$ for $j = 1, \ldots, N$. This means that we are looking for a non-singular 3×3 matrix $\boldsymbol{H}$ for which

$$\hat{\boldsymbol{x}}_j \sim \boldsymbol{H}\boldsymbol{x}_j, \quad j = 1, \ldots, N. \tag{2.2}$$

Now, the proportionality constraints (2.2) are equivalent to saying that all the cross products $\hat{\boldsymbol{x}}_j \times \boldsymbol{H}\boldsymbol{x}_j$ should be zero, which can be expressed for each one of them as

$$\begin{aligned} \hat{\boldsymbol{x}}_j \times \boldsymbol{H}\boldsymbol{x}_j = \boldsymbol{0} \Leftrightarrow \left[\hat{\boldsymbol{x}}_j\right]_\times \boldsymbol{H}\boldsymbol{x}_j = \boldsymbol{0} \Leftrightarrow \\ \left(\left[\hat{\boldsymbol{x}}_j\right]_\times \otimes \boldsymbol{x}_j^T\right) \mathrm{vec}(\boldsymbol{H}^T) = \boldsymbol{0}, \end{aligned} \tag{2.3}$$

or more explicitly

$$\begin{bmatrix} \boldsymbol{0} & -\hat{z}_j \boldsymbol{x}_j^T & \hat{y}_j \boldsymbol{x}_j^T \\ \hat{z}_j \boldsymbol{x}_j^T & \boldsymbol{0} & -\hat{x}_j \boldsymbol{x}_j^T \\ -\hat{y}_j \boldsymbol{x}_j^T & \hat{x}_j \boldsymbol{x}_j^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \boldsymbol{h}_3 \end{bmatrix} = \boldsymbol{0}, \tag{2.4}$$

where $\boldsymbol{h}_1^T$, $\boldsymbol{h}_2^T$, and $\boldsymbol{h}_3^T$ are the rows of $\boldsymbol{H}$. This is called a (redundant) *Direct Linear Transformation constraint* (DLT constraint).

Since the DLT constraints for the correspondences $\boldsymbol{x}_j \leftrightarrow \hat{\boldsymbol{x}}_j$ should hold simultaneously, one can form a joint DLT *system*

$$\begin{bmatrix} \left[\hat{\boldsymbol{x}}_1\right]_\times \otimes \boldsymbol{x}_1^T \\ \vdots \\ \left[\hat{\boldsymbol{x}}_N\right]_\times \otimes \boldsymbol{x}_N^T \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \boldsymbol{h}_3 \end{bmatrix} = \boldsymbol{0}, \tag{2.5}$$

which contains all the DLT constraints at once.

The trivial solution to this system, $\boldsymbol{H} = \boldsymbol{0}$, is clearly singular, so we are looking for a non-zero $\boldsymbol{H}$ in the null space of the coefficient matrix. Because of the cross product matrices, the rank of the coefficient matrix is at most $\min(2N, 9)$, which means that as long as $N \leq 4$ there is certainly an $\boldsymbol{H}$ which maps all the $\boldsymbol{x}_j$ to the $\hat{\boldsymbol{x}}_j$. Four points also determine the homography uniquely unless three of them are collinear, in which case they will produce linearly dependent DLT constraints. That four is the smallest number of point correspondences needed to determine the homography has been known since at least the late 19th century (Sturm, 1869).

If $N > 4$ it is in general not possible to find a homography which maps all the points as desired, but one which approximately accomplishes this can sometimes still be obtained as a least-squares solution to the DLT system.

Since the concept of *distance* is not applicable in projective geometry, the least-squares solution obtained in this way is not 'geometrically optimal' as would be the case in Euclidean geometry. We postpone the discussion of the so called *geometric error* to Chapter 8.

2.2 Geometric Transformations

Taking a step back to the more familiar Euclidean geometry for a moment, we will in this section consider some frequently occurring geometric transformations in $\mathbb{R}^2$ and $\mathbb{R}^3$.

2.2.1 Parametrising Rotations

For planar rotations, there is almost only one parametrisation that is ever used,[1] and the discussion of this case can be kept conveniently short. A rotation in $\mathbb{R}^2$ an angle φ is achieved by multiplication with the rotation matrix

$$\boldsymbol{R}(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}. \tag{2.6}$$

It can be shown that all orthogonal matrices in $\mathbb{R}^{2\times 2}$ with determinant equal to one can be written in this way for some angle φ.

Rotations in 3D are decidedly more complicated, and there are many ways to parametrise them. One possibility is to perform a sequence of 2D rotations around the three coordinate axes. It turns out that at most three such rotations are necessary to describe any rotation, and the three angles used in the process are the so called *Tait-Bryan angles* (α, β, γ). We will write a rotation matrix parametrised by Tait-Bryan angles as

$$\boldsymbol{R}(\alpha, \beta, \gamma) = \boldsymbol{R}_x(\alpha)\boldsymbol{R}_y(\beta)\boldsymbol{R}_z(\gamma), \tag{2.7}$$

where the $\boldsymbol{R}_x$, $\boldsymbol{R}_y$, and $\boldsymbol{R}_z$ are rotations about the coordinate axis indicated

[1] *Occasionally*, planar rotations are parametrised as complex numbers of the form $e^{j\varphi}$.

by the subscript. For instance, with this notation we have

$$\boldsymbol{R}_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.8}$$

For Tait-Bryan angles we have the following result.

Lemma 2.1. *Let $\boldsymbol{R}$ be a 3×3 rotation matrix. Then there exist triplets of angles, (α, β, γ), such that*

$$\boldsymbol{R}(\alpha, \beta, \gamma) = \boldsymbol{R}_x(\alpha)\boldsymbol{R}_y(\beta)\boldsymbol{R}_z(\gamma). \tag{2.9}$$

Furthermore, if the angles are restricted to $[0, 2\pi)$, there exist exactly two such triplets.

Proof. The existence of such a triplet follows from a QR decomposition of the rotation using Given's rotations (Golub and Van Loan, 1996). Such a process will produce $\boldsymbol{R}_x(\alpha)$, $\boldsymbol{R}_y(\beta)$ and $\boldsymbol{R}_z(\gamma)$, which satisfy (2.9). Since

$$\begin{aligned} \boldsymbol{R}(\alpha, \beta, \gamma) &= \boldsymbol{R}_x(\alpha)\boldsymbol{R}_y(\beta)\boldsymbol{R}_z(\gamma) \\ &= \boldsymbol{R}_x(\alpha + \pi)\boldsymbol{R}_x(\pi)\boldsymbol{R}_y(\beta)\boldsymbol{R}_z(\pi)\boldsymbol{R}_z(\gamma + \pi) \\ &= \boldsymbol{R}_x(\alpha + \pi)\boldsymbol{R}_y(\pi - \beta)\boldsymbol{R}_z(\gamma + \pi) \\ &= \boldsymbol{R}(\alpha + \pi, \pi - \beta, \gamma + \pi), \end{aligned} \tag{2.10}$$

we see that there are at least two possible decompositions.

To show that there are no other possible decompositions, let us consider two such decompositions of $\boldsymbol{R}$,

$$\boldsymbol{R}(\alpha, \beta, \gamma) = \boldsymbol{R}_x(\alpha)\boldsymbol{R}_y(\beta)\boldsymbol{R}_z(\gamma) = \boldsymbol{R}_x(\tilde{\alpha})\boldsymbol{R}_y(\tilde{\beta})\boldsymbol{R}_z(\tilde{\gamma}). \tag{2.11}$$

Then

$$\boldsymbol{R}_x(\underbrace{\alpha - \tilde{\alpha}}_{=\hat{\alpha}})\boldsymbol{R}_y(\beta) = \boldsymbol{R}_y(\tilde{\beta})\boldsymbol{R}_z(\underbrace{\tilde{\gamma} - \gamma}_{=\hat{\gamma}}). \tag{2.12}$$

Writing both sides out explicitly, we have

$$\boldsymbol{R}_x(\hat{\alpha})\boldsymbol{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ \sin\hat{\alpha}\sin\beta & \cos\hat{\alpha} & -\sin\hat{\alpha}\cos\beta \\ -\cos\hat{\alpha}\sin\beta & \sin\hat{\alpha} & \cos\hat{\alpha}\cos\beta \end{bmatrix} \tag{2.13}$$

and

$$\boldsymbol{R}_y(\tilde{\beta})\boldsymbol{R}_z(\hat{\gamma}) = \begin{bmatrix} \cos\tilde{\beta}\cos\hat{\gamma} & -\cos\tilde{\beta}\sin\hat{\gamma} & \sin\tilde{\beta} \\ \sin\hat{\gamma} & \cos\hat{\gamma} & 0 \\ -\sin\tilde{\beta}\cos\hat{\gamma} & \sin\tilde{\beta}\sin\hat{\gamma} & \cos\tilde{\beta} \end{bmatrix}. \tag{2.14}$$

Next, we consider the two possibilities $\cos\beta \neq 0$ and $\cos\beta = 0$.

Suppose $\cos\beta \neq 0$. Then it follows that $\sin\hat{\alpha} = \sin\hat{\gamma} = 0$ and $\cos\hat{\alpha} = \cos\hat{\gamma} = \pm 1$. This gives two possible decompositions; one where $\cos\beta = \cos\tilde{\beta}$ (when $\sin\hat{\alpha} = 1$), and one where $\cos\beta = -\cos\tilde{\beta}$ (when $\sin\hat{\alpha} = -1$).

Now, suppose instead that $\cos\beta = 0$. Then it follows that $\cos\tilde{\beta} = 0$, $\sin\beta = \sin\tilde{\beta} = \pm 1$, and $\cos\hat{\alpha} = \cos\hat{\gamma}$. We again get two possible decompositions; one where $\sin\hat{\alpha} = \sin\hat{\gamma}$ (when $\sin\beta = 1$), and one where $\sin\hat{\alpha} = -\sin\hat{\gamma}$ (when $\sin\beta = -1$). □

Another popular choice for parametrising 3D rotations is the so called *Rodrigues' formula*, which gives a very intuitive axis-angle representation of a rotation matrix. This representation can be obtained geometrically, as we will see shortly, or by grouping the terms in the exponential map (1.16).

Theorem 2.2 (Rodrigues' formula). *Let $\boldsymbol{v}$ be some fixed unit vector in $\mathbb{R}^3$, and let $\boldsymbol{R}$ be the matrix whose action results in a rotation an angle φ about $\boldsymbol{v}$. Then*

$$\boldsymbol{R} = \boldsymbol{I} + \sin\varphi\,[\boldsymbol{v}]_\times + (1-\cos\varphi)\,[\boldsymbol{v}]_\times^2. \tag{2.15}$$

Proof. See Figure 2.2 for an illustration of the geometrical situation. If $\boldsymbol{x}$ and $\boldsymbol{v}$ are parallel, then $\boldsymbol{x}$ is unchanged by the rotation, which immediately agrees with the expression (2.15) for the rotation matrix. If $\boldsymbol{x}$ and $\boldsymbol{v}$ are not parallel, recall the projection theorem, which allows us to write $\boldsymbol{x}$ as

$$\boldsymbol{x} = \boldsymbol{x}_\parallel + \boldsymbol{x}_\perp, \tag{2.16}$$

where $\boldsymbol{x}_\parallel$ is parallel to $\boldsymbol{v}$, and $\boldsymbol{x}_\perp$ is perpendicular to $\boldsymbol{v}$. The parallel part, $\boldsymbol{x}_\parallel = (\boldsymbol{v}^T\boldsymbol{x})\boldsymbol{v}$, will remain unchanged by the rotation. Note also that $\boldsymbol{x}_\perp$

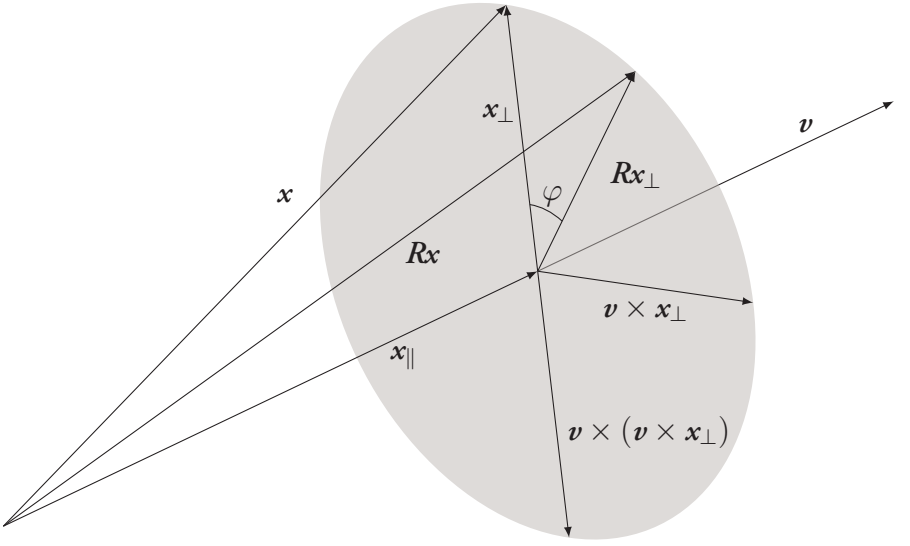


Figure 2.2: When rotating x about v, one may consider the parts $x_\parallel$ and $x_\perp$ separately. The part $x_\parallel$ is parallel to v and is unaffected by the rotation. The part $x_\perp$ is perpendicular to v, and turns in the plane orthogonal to v.

and $v \times x_\perp$ will have the same length and will make up an orthogonal basis in the plane orthogonal to v. It follows that

$$\begin{aligned} Rx_\perp &= \cos\varphi\, x_\perp + \sin\varphi\, (v \times x_\perp) \\ &= -\cos\varphi\, (v \times (v \times x_\perp)) + \sin\varphi\, (v \times x_\perp) \\ &= -\cos\varphi\, (v \times (v \times x)) + \sin\varphi\, (v \times x). \end{aligned} \tag{2.17}$$

This means that

$$\begin{aligned} Rx &= Rx_\parallel + Rx_\perp \\ &= (v^T x)v - \cos\varphi\, (v \times (v \times x)) + \sin\varphi\, (v \times x) \\ &= x + \sin\varphi\, (v \times x) + (1 - \cos\varphi)(v \times (v \times x)). \end{aligned} \tag{2.18}$$

The matrix R is readily extracted from this description. □

Finally, there is the option to parametrise rotations with the help of *quaternions*. These also provide an intuitive axis-angle representation (Szeliski,

2011; Terzakis et al., 2012), and are generally a good choice of representation for rotations. The main reason why we have not used this representation is that the planar motion model, which we will investigate in detail in Chapter 4, turned out to be easier expressed using Tait-Bryan angles.

2.2.2 Transformation Groups

Suppose $\boldsymbol{x}$ is a vector in $\mathbb{R}^2$ or $\mathbb{R}^3$. A *rigid* transformation consists of a rotation and a translation, and is a transformation of the form

$$\boldsymbol{y} = \boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}, \tag{2.19}$$

where $\boldsymbol{R}$ is a rotation matrix.

It can easily be verified that the composition of two rigid transformations is also a rigid transformation and that every rigid transformation is invertible (with the inverse also being a rigid transformation). The rigid transformation with $\boldsymbol{R} = \boldsymbol{I}$ and $\boldsymbol{t} = \boldsymbol{0}$ leaves $\boldsymbol{x}$ unmodified, and therefore acts as an identity transformation. By showing associativity, one can actually conclude that under composition the rigid transformations constitute a (non-abelian) *group* (Hungerford, 1997).

An easy method for doing this, and which is useful in other situations as well, is to make use of homogeneous coordinates. Indeed,

$$\boldsymbol{y} = \boldsymbol{R}\boldsymbol{x} + \boldsymbol{t} \quad \Leftrightarrow \quad \begin{bmatrix} \boldsymbol{y} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix}, \tag{2.20}$$

which means that the rigid transformation can be cast as a matrix multiplication, and these are known to be associative.

It also immediately follows that the rigid transformations are a special class of homographies, and constitute a subgroup among the group of homographies. In fact, there is a hierarchy of common transformations, which is shown in Table 2.1. We will not go into the details of these transformations here.

Table 2.1: Hierarchy of transformations. Each of the transformation groups contains, as subgroups, the transformations listed below it.

Transformation	Matrix
Projective	$\boldsymbol{H}, \quad \det \boldsymbol{H} \neq 0$
Affine	$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix}, \quad \det \boldsymbol{A} \neq 0$
Similarity	$\begin{bmatrix} s\boldsymbol{Q} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix}, \quad \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}, \quad s \neq 0$
Rigid	$\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix}, \quad \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}, \quad \det \boldsymbol{R} = 1$

2.3 Physical Camera Modelling

In this section we shall briefly describe the classic pinhole perspective camera model, which, as we will find, can be cast as a special case of the projective cameras introduced in Section 2.1.2. We will also discuss the Brown-Conrady model for handling lens distortion. A much more detailed discussion of these topics may be found in Hartley and Zisserman (2004) and Szeliski (2011).

2.3.1 The Pinhole Perspective Camera

Intuition about the geometrical situation may be invoked from the idealised physical model of image formation shown in Figure 2.3. Introduce an orthonormal coordinate system in which the focal point, or *camera centre*, of the camera is at the origin, and in which the image sensor lies in the plane $z = -f$. Suppose an object in front of the camera emits or reflects light, which passes through the focal point and falls onto the sensor, creating an inverted (horizontally as well as vertically flipped) image of the object. Mathematically restoring the image to a non-inverted one is equivalent to moving the image sensor to the front of the camera, at $z = f$ (which we shall call the *image plane*). The line which is perpendicular to the image plane and passes through the camera centre (here the z-axis) is

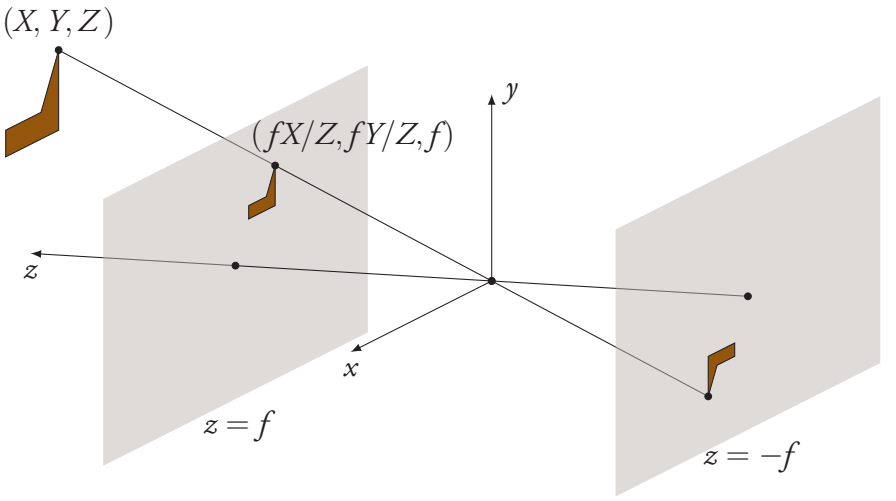


Figure 2.3: An idealised model of the image formation process. Light is emitted from the object and passes through the focal point, giving rise to an inverted image on the plane $z = -f$. Mathematically, it is more convenient to instead consider the non-inverted virtual image in front of the camera, on the *image plane* $z = f$.

termed the *optical axis*, and the image point where it intersects the image plane is called the *principal point*.

By considering similar triangles, one finds that the scene point (X, Y, Z) is projected onto the image plane at $(fX/Z, fY/Z, f)$. Clearly, the third coordinate of any point in the image plane will always be f, and we can omit this third coordinate without running the risk of information loss. Thus the camera induces a mapping from scene points (X, Y, Z) to image points, which we can write as

$$(X, Y, Z) \longmapsto (fX/Z, fY/Z). \tag{2.21}$$

By expressing both the scene points and the image points using homogeneous coordinates, introduced in Section 2.1.1, the camera mapping (2.21)

may be written neatly in matrix form as

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.22}$$

The 3×4 matrix which defines the mapping has a column rank of three, which makes it qualify as a projective camera from $\mathbb{RP}^3$ to $\mathbb{RP}^2$.

For the purpose of many algorithms which operate on images, it is more natural to work with pixel coordinates instead of the abstract image coordinates used in the mapping (2.21). Changing to pixel coordinates entails scaling the x-coordinate with a factor σ_x and the y-coordinate with a factor σ_y (if the pixels are square, these factors should be very close in magnitude), and moving the origin to one of the corners (usually the upper left corner). Introducing $f_x = \sigma_x f$ and $f_y = \sigma_y f$, the mapping to pixels is given by

$$\begin{cases} x = f_x X/Z + c_x, \\ y = f_y Y/Z + c_y, \end{cases} \tag{2.23}$$

where (c_x, c_y) are the pixel coordinates of the principal point. The camera mapping to pixel coordinates becomes[2]

$$\begin{bmatrix} xZ \\ yZ \\ Z \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \boldsymbol{K} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{2.24}$$

where the *camera calibration matrix*

$$\boldsymbol{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.25}$$

[2] Sometimes it is also necessary to introduce a *skew parameter*, accounting for non-rectangular pixels. We do not model this here.

contains the *intrinsic parameters* of the camera (the *principal point* (c_x, c_y) and the *focal lengths* f_x and f_y), can be factored out on the left.

We often want to work with a global coordinate system which is not necessarily aligned with the camera coordinate frame. Supposing the camera has coordinates $\boldsymbol{t} = (t_x, t_y, t_z)$ in this global coordinate frame and is rotated by the rotation matrix $\boldsymbol{R}$, the camera projection matrix is found to be

$$\boldsymbol{P} = \boldsymbol{KR}\begin{bmatrix}\boldsymbol{I} & -\boldsymbol{t}\end{bmatrix}. \tag{2.26}$$

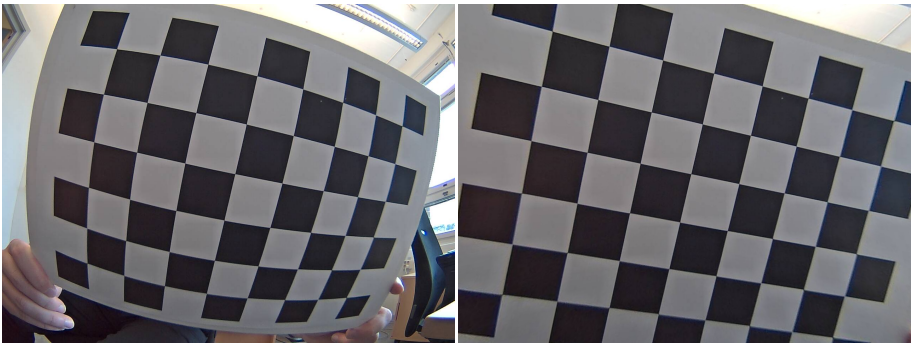
2.3.2 Lens Distortion

A very small aperture, as in the idealised pinhole camera, gives very sharp images, but a major drawback is that only a very small amount of light can reach the sensor. For this reason, most cameras are built with larger apertures to capture more light from the scene, and this light is brought closer to focus using optical lenses.

Unfortunately, optical lenses introduce a number of new challenges. For example, light takes different paths through the lenses depending on its wavelength, and this gives rise to a prism-like effect called *chromatic aberration*. Quality camera manufacturers put much effort into designing their optics in such a way as to reduce the chromatic aberration, and as end users of a good camera we are usually spared much of the suffering which chromatic aberration causes.

Optical lenses also introduce geometric distortions, in particular *radial distortion*, which cause straight lines in the scene to show up as curved lines in the image, as illustrated in Figure 2.4. These distortions are left to the end user to either accept or compensate for. One may use the *Brown-Conrady model* (Brown, 1971) to describe these non-linearities analytically and to determine a non-linear coordinate transformation which counters the distortions.

For a camera with principal point at $\mathbf{0}$, the Brown-Conrady model states that a point which under pinhole projection would have ended up at $\boldsymbol{x}$ will instead, because of lens distortions, end up at a point $\hat{\boldsymbol{x}}$ given by the



(a) Image with radial distortion. (b) Undistorted image.

Figure 2.4: Lens distortion can manifest itself in a number of ways, e.g. the *barrel distortion* shown in (a), and makes the projection disobey the pinhole camera model. If a distortion model is estimated, the image can be 'undistorted', and the resulting undistorted image can then be used together with the pinhole camera model.

expression

$$\hat{\boldsymbol{x}} = \underbrace{\gamma_{\mathrm{r}}(r)\boldsymbol{x}}_{\text{radial term}} + \underbrace{\gamma_{\mathrm{t}}(r)\left(r^2\boldsymbol{I} + 2\boldsymbol{x}\boldsymbol{x}^T\right)\begin{bmatrix}\tau_2\\ \tau_1\end{bmatrix}}_{\text{tangential term}}, \tag{2.27}$$

where $r = \|\boldsymbol{x}\|$, τ_1 and τ_2 are some constants, and where $\gamma_{\mathrm{r}}(r)$ and $\gamma_{\mathrm{t}}(r)$ are even functions with $\gamma_{\mathrm{r}}(0) = \gamma_{\mathrm{t}}(0) = 1$. For many common lenses the most severe distortion is taken care of by the radial term, and a low order polynomial as $\gamma_{\mathrm{r}}(r)$ often gives a sufficiently good model. Common choices are thus $\gamma_{\mathrm{t}}(r) = 1$ or $\gamma_{\mathrm{t}}(r) = 0$ and

$$\gamma_{\mathrm{r}}(r) = 1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6. \tag{2.28}$$

To determine the distortion parameters $(\kappa_1, \kappa_2, \kappa_3, \tau_1, \tau_2)$ one may use several images of a known calibration object such as the chequerboard shown in Figure 2.4. If one computes the image of the calibration object using pure pinhole projection, one may then apply the coordinate trans-

form (2.27) and use non-linear optimisation to tune the parameters until the virtual images agree well with the observed ones.

The BSD licensed library OpenCV includes algorithms for estimating the parameters as well as transforming images and coordinates according to the Brown-Conrady model (Itseez, Inc., 2017). Similar functionality also ships as an integrated toolbox of Matlab® since release 2013b (The MathWorks, Inc., 2013). In addition to the distortion parameters, these systems also estimate the principal point and the focal lengths f_x and f_y.

2.4 Establishing Point Correspondences

Many algorithms for multiple view geometry estimation assume that one has access to a number of point correspondences, i.e. points in different views which (are thought to) correspond to the same point in 3D. In addition to manual marking of point correspondences, which is a very time-consuming and tiresome endeavour,[3] there are algorithms which automate the process of finding such correspondences. As can be seen in Figure 2.5, which shows an example of such automatic association, one often obtains many spurious matches in addition to the valid ones. These algorithms typically address some (or all) of the three sub-problems:

1. *Interest point detection*. This problem concerns the detection of points with sufficiently distinct local appearance. Edges and corners, as well as points in highly textured areas, often provide useful interest points.

2. *Descriptor extraction*. For each of the interest points, it is necessary to extract a descriptor which describes the local appearance. In addition to being distinctive, it is often desired that the descriptor should be robust to transformations such as scaling and rotation.

3. *Matching the descriptors*. This is the problem of matching descriptors for interest points in different views. Solutions to this often rely on introducing a metric on the set of descriptors, in addition to using

[3] Having spent approximately 70 hours during the winter 2003–2004 doing precisely this, the author can back this claim with great confidence.

Figure 2.5: Automatically detected and matched feature points in two images of Wernigerode town hall. While many of the points are correctly associated, a large number of the matches are incorrect. RANSAC can be used here to remove most of the spurious matches.

some pruning technique to reduce the number of comparisons that need to be performed.

One of the, to this date, most successful methods to establish point correspondences is the *Scale-Invariant Feature Transform* (SIFT) proposed by Lowe (2004), which addresses all three sub-problems above. The interest point detection works by convolving resampled and smoothed versions of the image with a *difference of Gaußians* and using the local extrema as candidates (removing the ones with low contrast and the ones on edges). The descriptor consists of histograms over the direction of the image gradient, computed from small sub-regions around the interest point.

Another popular method is to use *Speeded-Up Robust Features* (SURF), introduced by Bay, Tuytelaars and Van Gool (2006) and which primar-

ily deal with the first two sub-problems in the list above. SURF makes heavy use of integral images and look-up tables, both for the interest point detector and the descriptor extraction, and is claimed to be several times faster than SIFT.

The last decade has seen the introduction of a myriad of different detectors, descriptors, and matching algorithms (all with catchy acronyms, of course). Recent studies, in which some of the most popular such algorithms are compared, seem to agree that it is difficult to find a clear winner among the available combinations (Gauglitz, Höllerer and Turk, 2011; Hartmann, Klüssendorf and Maehle, 2013), but there appears to be a consensus that SIFT still performs quite favourably compared to its competitors except perhaps in terms of computational cost. Implementations of many of theses algorithms are available in OpenCV (Itseez, Inc., 2017) and in the Computer Vision System Toolbox™ for Matlab® (The MathWorks, Inc., 2013).

Chapter 3

Two-View Geometry Background

This chapter covers some background theory on two-view geometry. In particular, we will consider the setup consisting of two projective cameras $\boldsymbol{P}$ and $\hat{\boldsymbol{P}}$ from $\mathbb{RP}^3$ to $\mathbb{RP}^2$. This means that $\boldsymbol{P}$ and $\hat{\boldsymbol{P}}$ both will be matrices of size 3×4 and with rank three. The main motivation is of course that we seek deeper understanding of the pinhole perspective camera from Section 2.3.1, but we will first consider the more general projective cameras introduced in Section 2.1.2.

3.1 Epipolar Geometry

We begin this section by introducing the following two important concepts:

Definition 3.1. *The image in one view of the other camera centre is called an* ***epipole****. A plane containing both camera centres is called an* ***epipolar plane****.*

The line joining two different camera centres $\boldsymbol{C}$ and $\hat{\boldsymbol{C}}$ is called the *baseline* of the pair, and the epipole in each of the views is the image point where the baseline intersects the image plane. Together with the two camera centres, a scene point $\boldsymbol{X}$ outside the baseline defines an *epipolar plane* containing these three points. This geometric situation, which is illustrated in Figure 3.1, is called *epipolar geometry* and is central to understanding two-view geometry.

Suppose $\boldsymbol{X}$ is imaged as $\boldsymbol{x}$ in the first view. The line through $\boldsymbol{C}$ and $\boldsymbol{X}$ clearly lies in the epipolar plane, and intersects the first image plane at $\boldsymbol{x}$. The image of this line in the second view, which is given by the intersection

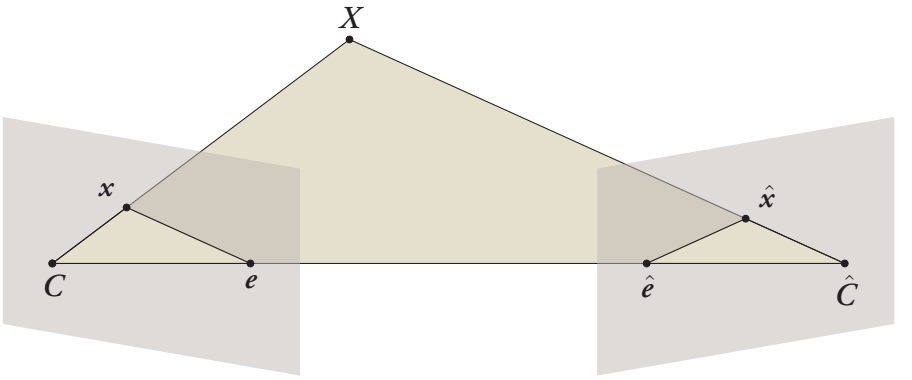


Figure 3.1: Illustration of epipolar geometry. Two cameras $\boldsymbol{P}$ and $\hat{\boldsymbol{P}}$ with camera centres $\boldsymbol{C}$ and $\hat{\boldsymbol{C}}$ observe a scene point $\boldsymbol{X}$. The epipolar plane contains $\boldsymbol{X}$ and the two camera centres.

of the epipolar plane and the second image plane, is the so called *epipolar line* corresponding to $\boldsymbol{x}$. This line consists of all $\hat{\boldsymbol{y}}$ for which $\hat{\boldsymbol{l}}^T\hat{\boldsymbol{y}} = 0$, and in particular it contains $\hat{\boldsymbol{x}} \sim \hat{\boldsymbol{P}}\boldsymbol{X}$. Correspondingly, an image point $\hat{\boldsymbol{x}}$ in the second view gives rise to an epipolar line $\boldsymbol{l}^T\boldsymbol{y} = 0$ in the first view.

3.1.1 The Fundamental Matrix

Let us now investigate the algebraic nature of the mapping from the image point $\boldsymbol{x}$ to its corresponding epipolar line $\hat{\boldsymbol{l}}$ in the other view. We know that the epipolar line $\hat{\boldsymbol{l}}$ is the image in the second view of the scene line through $\boldsymbol{C}$ and $\boldsymbol{X}$. This line clearly contains $\boldsymbol{C}$ and $\boldsymbol{P}^+\boldsymbol{x}$. Projecting these two points into the second view gives

$$\begin{cases} \hat{\boldsymbol{l}}^T\hat{\boldsymbol{P}}\boldsymbol{C} = 0 \\ \hat{\boldsymbol{l}}^T\hat{\boldsymbol{P}}\boldsymbol{P}^+\boldsymbol{x} = 0 \end{cases} \Leftrightarrow \hat{\boldsymbol{l}} \sim \hat{\boldsymbol{e}} \times \hat{\boldsymbol{P}}\boldsymbol{P}^+\boldsymbol{x} = \left[\hat{\boldsymbol{e}}\right]_\times \hat{\boldsymbol{P}}\boldsymbol{P}^+\boldsymbol{x}. \tag{3.1}$$

The (unique up to scale) matrix $\boldsymbol{F} = \left[\hat{\boldsymbol{e}}\right]_\times \hat{\boldsymbol{P}}\boldsymbol{P}^+$ is called the *fundamental matrix* associated with the ordered pair $\{\boldsymbol{P}, \hat{\boldsymbol{P}}\}$, and was introduced simultaneously and independently by Faugeras (1992) and Hartley (1992). Because of the cross product matrix involved, it follows that $\operatorname{rank}\boldsymbol{F} = 2$.

Consequently, $\boldsymbol{F}$ is not a homography, even though it does represent a linear transformation of homogeneous coordinates.

For any point $\hat{\boldsymbol{y}}$ on the epipolar line, we have $0 = \hat{\boldsymbol{l}}^T\hat{\boldsymbol{y}} = \hat{\boldsymbol{y}}^T\boldsymbol{F}\boldsymbol{x}$. In particular, this holds for $\hat{\boldsymbol{y}} = \hat{\boldsymbol{x}}$, resulting in the *epipolar constraint*

$$\hat{\boldsymbol{x}}^T\boldsymbol{F}\boldsymbol{x} = 0 \tag{3.2}$$

that corresponding points $\boldsymbol{x} \leftrightarrow \hat{\boldsymbol{x}}$ must satisfy. If we now consider the epipolar line in the first view, corresponding to $\hat{\boldsymbol{x}}$, it follows from the epipolar constraint that $\boldsymbol{l} \sim \boldsymbol{F}^T\hat{\boldsymbol{x}}$. On the other hand, a similar derivation as above results in $\boldsymbol{l} \sim \left[\boldsymbol{e}\right]_\times \boldsymbol{P}\hat{\boldsymbol{P}}^+$. Together, these facts show the following result.

Theorem 3.1. *If $\boldsymbol{F}$ is the fundamental matrix associated with the ordered pair $\{\boldsymbol{P}, \hat{\boldsymbol{P}}\}$, then $\boldsymbol{F}^T$ is the fundamental matrix associated with $\{\hat{\boldsymbol{P}}, \boldsymbol{P}\}$. Furthermore, $\boldsymbol{e}$ spans the right null space and $\hat{\boldsymbol{e}}$ spans the left null space of $\boldsymbol{F}$.*

From the discussion above we know that two camera matrices unambiguously determine a fundamental matrix. The fundamental matrix, however does not unambiguously determine the two camera matrices. In fact, the fundamental matrix is invariant to projective transformations of $\mathbb{RP}^3$, so that $\{\boldsymbol{P}, \hat{\boldsymbol{P}}\}$ and $\{\boldsymbol{P}\boldsymbol{W}, \hat{\boldsymbol{P}}\boldsymbol{W}\}$ define the same fundamental matrix for any invertible 4×4 matrix $\boldsymbol{W}$ (Hartley and Zisserman, 2004).

For a general 3×4 matrix $\boldsymbol{P}$ of rank three, an SVD will be of the form

$$\boldsymbol{P} = \boldsymbol{U}\begin{bmatrix}\sigma_1 & 0 & 0 & 0\\ 0 & \sigma_2 & 0 & 0\\ 0 & 0 & \sigma_3 & 0\end{bmatrix}\boldsymbol{V}^T = \boldsymbol{U}\begin{bmatrix}\sigma_1 & 0 & 0\\ 0 & \sigma_2 & 0\\ 0 & 0 & \sigma_3\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}\boldsymbol{V}^T. \tag{3.3}$$

Thus, we have

$$\begin{bmatrix}\boldsymbol{I} & \boldsymbol{0}\end{bmatrix} = \operatorname{diag}\left(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \frac{1}{\sigma_3}\right)\boldsymbol{U}^T\boldsymbol{P}\boldsymbol{V}, \tag{3.4}$$

which shows that it is always possible to bring one of the cameras to the form $\begin{bmatrix}\boldsymbol{I} & \boldsymbol{0}\end{bmatrix}$ by projective transformations of the scene and one of the image planes. For this reason, one may without loss of generality always assume that the two camera matrices are of the form

$$\boldsymbol{P} = \begin{bmatrix}\boldsymbol{I} & \boldsymbol{0}\end{bmatrix} \quad \text{and} \quad \hat{\boldsymbol{P}} = \begin{bmatrix}\boldsymbol{A} & \boldsymbol{b}\end{bmatrix}. \tag{3.5}$$

If we restrict ourselves to cameras of the form (3.5), then a valid choice can be computed from the fundamental matrix as $\boldsymbol{A} = \left[\hat{\boldsymbol{e}}\right]_\times \boldsymbol{F}$ and $\boldsymbol{b} = \hat{\boldsymbol{e}}$ (Luong and Viéville, 1996). The camera matrices can then be used together with the point correspondences $\boldsymbol{x} \leftrightarrow \hat{\boldsymbol{x}}$ to triangulate the location of the scene points $\boldsymbol{X}_j$. That the scene points and camera centres can be recovered uniquely up to a projective transformation is the essence of the so called *projective reconstruction theorem* (Hartley and Zisserman, 2004).

3.1.2 Computing the Fundamental Matrix

The epipolar constraint (3.2) is linear in the elements of $\boldsymbol{F}$. Lemma 1.1 allows us to express the epipolar constraint as

$$(\boldsymbol{x} \otimes \hat{\boldsymbol{x}})^T \operatorname{vec} \boldsymbol{F} = 0. \tag{3.6}$$

Analogously to the way we formed the DLT system (2.5) in Section 2.1.3, we can then stack a number of epipolar constraints into a system

$$\begin{bmatrix} (\boldsymbol{x}_1 \otimes \hat{\boldsymbol{x}}_1)^T \\ \vdots \\ (\boldsymbol{x}_N \otimes \hat{\boldsymbol{x}}_N)^T \end{bmatrix} \operatorname{vec} \boldsymbol{F} = \boldsymbol{0}. \tag{3.7}$$

If at least eight point correspondences are used, and unless the points are in certain degenerate configurations, the rank of the coefficient matrix in (3.7) will be at least eight.

The *eight-point algorithm* works by forming (3.7) for eight point correspondences, followed by computing a vector $\boldsymbol{f}$ that spans the null space. Typically, the matrix $\operatorname{vec}_{3\times3}^{-1} \boldsymbol{f}$ has rank three and therefore is not a valid fundamental matrix, but this is remedied by using Theorem 1.3 to obtain an approximation with the correct rank. The eight-point algorithm computes precisely one fundamental matrix, it is easy to implement, and if the inlier ratio is suitably large it performs exceptionally well if appropriate care is taken to numerical preconditioning (Hartley, 1997).

While the eight-point algorithm works well in many cases, it uses more point correspondences than necessary, and in the presence of a large proportion of outliers this increases the number of necessary RANSAC iterations considerably (see Section 1.2.2).

Input: Point correspondences $\boldsymbol{x}_j \leftrightarrow \hat{\boldsymbol{x}}_j$ for $j = 1, \ldots, 7$.
Output: A set $\mathcal{F}$ containing up to three possible fundamental matrices

1: Create the matrix
$$\boldsymbol{M} = \begin{bmatrix} (\boldsymbol{x}_1 \otimes \hat{\boldsymbol{x}}_1)^T \\ \vdots \\ (\boldsymbol{x}_7 \otimes \hat{\boldsymbol{x}}_7)^T \end{bmatrix}$$
2: Find non-parallel $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ in the null space of $\boldsymbol{M}$ (e.g. using SVD)
3: $\boldsymbol{F}_1 \leftarrow \mathrm{vec}_{3\times3}^{-1}\boldsymbol{f}_1$
4: $\boldsymbol{F}_2 \leftarrow \mathrm{vec}_{3\times3}^{-1}\boldsymbol{f}_2$
5: $\mathcal{F} \leftarrow \emptyset$
6: **if** $\det \boldsymbol{F}_2 = 0$ **then**
7: $\quad \mathcal{F} \leftarrow \{\boldsymbol{F}_2\}$
8: **for** each $s \in \mathbb{R}$ that solves $\det(\boldsymbol{F}_1 + s\boldsymbol{F}_2) = 0$ **do**
9: $\quad \mathcal{F} \leftarrow \mathcal{F} \cup \{\boldsymbol{F}_1 + s\boldsymbol{F}_2\}$

Algorithm 3.1: The seven-point algorithm computes a fundamental matrix compatible with the epipolar geometry from seven point correspondences.

By enforcing the rank-2 constraint at an earlier stage than is done in the eight-point algorithm, it is possible to use only seven point correspondences to determine the fundamental matrix. If only seven correspondences are used to form the system (3.7), there will in general be a two-dimensional null space spanned by $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$. In this null space there should be (up to scale) at least one element representing a true fundamental matrix. If we form $\boldsymbol{F}_1 = \mathrm{vec}_{3\times3}^{-1}\boldsymbol{f}_1$ and $\boldsymbol{F}_2 = \mathrm{vec}_{3\times3}^{-1}\boldsymbol{f}_2$, then all possible fundamental matrices are either of the form $\boldsymbol{F} = \boldsymbol{F}_2$ or the form $\boldsymbol{F} = \boldsymbol{F}_1 + s\boldsymbol{F}_2$. The former case is easily checked, and in the latter case all possibilities are found by solving $\det(\boldsymbol{F}_1 + s\boldsymbol{F}_2) = 0$. This procedure is called the *seven-point algorithm* (Hartley, 1994), and is shown in Algorithm 3.1.

Interestingly, Sturm (1869) studied and solved a problem concerning the determination of projective conic sections from seven point correspondences, which later turned out to be equivalent to computing the fundamental matrix (Sturm, 2011).

The seven-point algorithm is a good illustration of a very powerful technique, namely that of using non-linear constraints to find the 'right' element(s) in a null space computed from a number of linear constraints. This technique has successfully been used to solve numerous problems in computer vision, e.g. simultaneous estimation of a homography and radial distortion (Kúkelová et al., 2015), optimal three-view triangulation (Stewénius, Schaffalitzky and Nistér, 2005), and various methods for estimation of the essential matrix (see Section 3.1.3), to name only a few. We will use this technique again in Chapter 7 to enforce the planar motion model from Chapter 4 in a homography estimation method.

3.1.3 The Essential Matrix

Epipolar geometry in the *calibrated case*, i.e. for the pinhole perspective camera model, has been studied for a much longer time than for the *uncalibrated case* of general projective cameras. In the calibrated case, the fundamental matrix is instead called the *essential matrix*, which was introduced by Longuet-Higgins (1981). The fact that five point correspondences determine the relative pose in the calibrated case has been known for at least a hundred years (Kruppa, 1913), but despite this, it is only in recent years that useful methods to compute it have been presented. From its inception, the so called *five-point algorithm* has become a mainstream method for doing two-view reconstruction of general scenes in the calibrated case (Li and Hartley, 2006; Nistér, 2004; Stewénius, Engels and Nistér, 2006).

3.2 Planes and Homographies

The discussion thus far in this chapter has not made any specific assumptions on the scene points, except that they should not lie on the baseline. One type of structure that occurs frequently in man-made environments is that of a plane, and a frequent assumption is thus that some (or all) of the scene points should lie on a plane. For this specific situation we have the following result.

Theorem 3.2. *Assume* $\boldsymbol{P} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}$ *and* $\hat{\boldsymbol{P}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \end{bmatrix}$ *are projective cameras, and let* $\boldsymbol{\pi} = (\boldsymbol{\nu}, d)$ *be a plane which does not contain any of the camera centres. If* $\boldsymbol{X}$ *is a point on the plane and projects into the two views as* $\boldsymbol{x} \sim \boldsymbol{PX}$ *and* $\hat{\boldsymbol{x}} \sim \hat{\boldsymbol{P}}\boldsymbol{X}$*, then*

$$\hat{\boldsymbol{x}} \sim \left(\boldsymbol{A} - \boldsymbol{b}\boldsymbol{\nu}^T/d\right)\boldsymbol{x}, \tag{3.8}$$

and the matrix $\boldsymbol{H} = \boldsymbol{A} - \boldsymbol{b}\boldsymbol{\nu}^T/d$ *is a homography.*

Proof. Since $\boldsymbol{\pi}$ does not contain the camera centres, d cannot be zero, and thus we may without loss of generality assume that $d = 1$. From the image in the first view, $\boldsymbol{x} \sim \boldsymbol{PX}$, we see that $\boldsymbol{X} \sim (\boldsymbol{x}, \xi)$ for some ξ. But $\boldsymbol{X}$ lies on the plane $\boldsymbol{\pi}$, which gives $\xi = -\boldsymbol{\nu}^T\boldsymbol{x}$. The projection of $\boldsymbol{X}$ into the second view will be

$$\hat{\boldsymbol{x}} \sim \hat{\boldsymbol{P}}\boldsymbol{X} \sim \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}\xi = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\boldsymbol{\nu}^T\boldsymbol{x} = \left(\boldsymbol{A} - \boldsymbol{b}\boldsymbol{\nu}^T\right)\boldsymbol{x}. \tag{3.9}$$

The only thing that remains is to show that $\boldsymbol{A} - \boldsymbol{b}\boldsymbol{\nu}^T$ is a homography, i.e. that it is invertible. The rank of $\boldsymbol{A}$ can be either two or three. We consider these two cases separately.

If $\operatorname{rank}\boldsymbol{A} = 3$, the second camera centre will be $\hat{\boldsymbol{C}} = (-\boldsymbol{A}^{-1}\boldsymbol{b}, 1)$, and since this does not lie on $\boldsymbol{\pi}$ it follows that

$$0 \neq \boldsymbol{\pi}^T\hat{\boldsymbol{C}} = 1 - \boldsymbol{\nu}^T\boldsymbol{A}^{-1}\boldsymbol{b}, \tag{3.10}$$

and then the Sherman-Morrison-Woodbury formula (Theorem 1.4) guarantees that $\boldsymbol{A} - \boldsymbol{b}\boldsymbol{\nu}^T$ is invertible.

If $\operatorname{rank}\boldsymbol{A} = 2$, then $\boldsymbol{b}$ cannot be in the range of $\boldsymbol{A}$, which means that

$$\boldsymbol{A}\boldsymbol{x} \neq \lambda\boldsymbol{b} \tag{3.11}$$

for all $\boldsymbol{x}$ and non-zero scalars λ. We will show that the equation

$$\left(\boldsymbol{A} - \boldsymbol{b}\boldsymbol{\nu}^T\right)\boldsymbol{x} = \boldsymbol{0} \tag{3.12}$$

only has the trivial solution $\boldsymbol{x} = \boldsymbol{0}$. Clearly, there is no solution $\boldsymbol{x}$ for which $\boldsymbol{\nu}^T\boldsymbol{x} \neq 0$, as that would mean that $\boldsymbol{A}\boldsymbol{x} = (\boldsymbol{\nu}^T\boldsymbol{x})\boldsymbol{b}$. But $\boldsymbol{\nu}^T\boldsymbol{x} = 0$ is also impossible, as $\boldsymbol{x}$ then would have to span the one-dimensional null space of $\boldsymbol{A}$, and $(\boldsymbol{x}, 0)$ would in that case be the camera centre of $\hat{\boldsymbol{P}}$. □

Theorem 3.2 shows that any scene plane $\boldsymbol{\pi}$ induces a homography mapping between two views, as long as the camera centres do not lie on $\boldsymbol{\pi}$. This establishes a very important link between scene planes and homographies, and together with the next result, it gives insight into an important degeneracy for the problem of computing the fundamental matrix from point correspondences.

Theorem 3.3. *If all point correspondences $\boldsymbol{x}_j \leftrightarrow \hat{\boldsymbol{x}}_j$ are related by a homography, there are infinitely many possible fundamental matrices satisfying their epipolar constraints.*

Proof. First, we note that for any $\boldsymbol{v} \in \mathbb{R}^3$ it holds that

$$\boldsymbol{x}_j^T(\boldsymbol{v} \times \boldsymbol{x}_j) = 0 \Leftrightarrow \boldsymbol{x}_j^T \left[\boldsymbol{v}\right]_\times \boldsymbol{x}_j = 0 \Leftrightarrow (\boldsymbol{x}_j \otimes \boldsymbol{x}_j)^T \operatorname{vec}\left[\boldsymbol{v}\right]_\times = 0, \tag{3.13}$$

and therefore

$$\operatorname{rank} \begin{bmatrix} (\boldsymbol{x}_1 \otimes \boldsymbol{x}_1)^T \\ \vdots \\ (\boldsymbol{x}_N \otimes \boldsymbol{x}_N)^T \end{bmatrix} \leq 6. \tag{3.14}$$

Now, since all the correspondences are related by a homography, i.e. $\hat{\boldsymbol{x}}_j \sim \boldsymbol{H}\boldsymbol{x}_j$ for all j, the epipolar constraints yield

$$\hat{\boldsymbol{x}}_j^T \boldsymbol{F} \boldsymbol{x}_j = 0 \Leftrightarrow \boldsymbol{x}_j^T \boldsymbol{H}^T \boldsymbol{F} \boldsymbol{x}_j = 0 \Leftrightarrow (\boldsymbol{x}_j \otimes \boldsymbol{x}_j)^T \operatorname{vec}(\boldsymbol{H}^T \boldsymbol{F}) = 0. \tag{3.15}$$

It follows that there is a three-dimensional family of fundamental matrices $\boldsymbol{F}$ which are compatible with all the epipolar constraints. □

This result is crucial, because it means that we cannot estimate the fundamental matrix from point correspondences if the scene is planar. For the applications described in this thesis, where the scene is expected to be planar, it means that methods based on the fundamental matrix or the essential matrix are bound to fail, and that a homography based approach is more promising.

The case that $\boldsymbol{H}^T\boldsymbol{F}$ is skew-symmetric, which suggests itself from the proof of Theorem 3.3, is particularly important. If $\boldsymbol{H}$ is a homography and $\boldsymbol{F}$ is a fundamental matrix, and if

$$\boldsymbol{H}^T\boldsymbol{F} + \boldsymbol{F}^T\boldsymbol{H} = \boldsymbol{0}, \tag{3.16}$$

then $\boldsymbol{H}$ and $\boldsymbol{F}$ are said to be *compatible*. A homography is compatible with a fundamental matrix if and only if there exists a scene plane which induces $\boldsymbol{H}$ in the epipolar geometry defined by $\boldsymbol{F}$ (Hartley and Zisserman, 2004; Luong and Viéville, 1996).

Chapter 4

SLAM and Planar Motion

This first part of this chapter introduces the so called SLAM problem for robot navigation, and gives a brief overview of some of the existing approaches. In the sections which follow this literature review, we look at the specific case of planar robot motion, and derive a camera motion model and homography parametrisation that will be used in the subsequent chapters.

4.1 Simultaneous Localisation and Mapping

One of the long-standing efforts in robotics research has been the development of algorithms which enable robots, such as the Fraunhofer IPA rob@work platform (Fraunhofer IPA, 2012) in Figure 4.1, to navigate and move autonomously in an unknown environment. The range of applications of such algorithms is extensive and includes e.g. flexible assembly lines, robotic vacuum cleaners, logistics applications, search and rescue operations, planetary exploration, and many, many more.

A common framework that has proven successful for for this class of algorithms is *Simultaneous Localisation and Mapping* (SLAM), in which the robot makes use of various sensors (e.g. laser range finders, cameras, wheel encoders, sonar, …) to map the surrounding environment and at the same time position itself within this map (Durrant-Whyte and Bailey, 2006a,b). The type of map that is created in this process is highly dependent on what sensors are being used and what the intended application is, and can range from sparse clouds of feature points to dense and detailed textured 3D models.

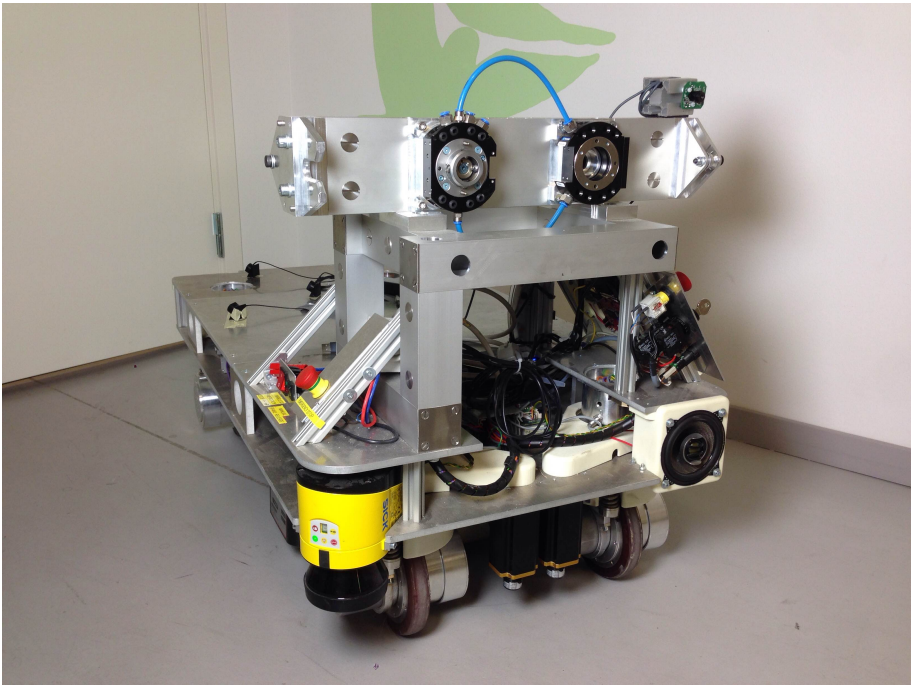


Figure 4.1: The Fraunhofer IPA rob@work (base platform) is an example of a mobile robot whose autonomy can be enhanced by SLAM algorithms. Some of the experimental data used in this thesis were obtained from a camera that was mounted onto this robot. Image courtesy of Björn Olofsson and Martin Karlsson.

4.1.1 A Brief History of Visual SLAM

Much of the early work on SLAM focused on laser range finders and wheel encoders, and how to employ statistical estimation and filtering techniques to determine ego-motion from such data. Only in the last two or three decades have cameras become a realistic choice of sensor to use for robotic navigation. There are three major reasons for this. First, digital cameras have become available, and they have gone through a revolution in terms of both reduced price and improved quality. Secondly, computing power has continued to double approximately every second year, as predicted by the famous *Moore's law*. Thirdly, during this time, there were many mile-

stone advances in computer vision, which provided practical methods for inferring geometry from images.

The probabilistic viewpoint from the early approaches has proven to be useful for camera based SLAM as well, and has stood the test of time from its early adoption by Durrant-Whyte (1987) and Harris and Pike (1988) through to more recent methods such as the MonoSLAM system by Davison et al. (2007) and the ORB-SLAM system by Mur-Artal, Montiel and Tardós (2015). Many methods use (extended) *Kalman filters* and *particle filters* to model the uncertainty, to combine data from different sensors, and in some cases to incorporate a kinematic motion model (Berntorp, 2014; Gustafsson, 2012).

Durrant-Whyte (1987) used observations of geometric features from a calibrated stereo rig, and the uncertainty of each observation was modelled as a multivariate Gaußian which was updated using a Kalman filter. A similar Kalman filter based method was presented for the monocular case by Harris and Pike (1988), and was evaluated on a sequence of 16 images. The state update to the Kalman filter was in this case obtained using non-linear minimisation of the Mahalanobis distances between the observed and the reprojected points, thus reducing the relative influence of features with large spatial uncertainty.

Davison (2003) combined Kalman filter based feature covariance modelling with a kinematic model to create a real-time camera-based SLAM system. The system requires a known calibration object to initialise, but is after this initialisation able to cope well with unconstrained 3D camera motion. The system uses features consisting of small image patches together with their spatial uncertainty as determined by the Kalman filter. MonoSLAM is an extension of this work, and adds estimated surface normals to the feature patches, along with other small improvements (Davison et al., 2007).

4.1.2 Countering the Accumulation of Error

The *accumulation of error* (or *propagation of uncertainty*, or other similar terms) is an important phenomenon which inevitably occurs when the positions of new uncertain observations are related to previous such uncertain

positions. In the context of SLAM this means that the uncertainty of the current position of the robot typically grows over time and can become arbitrarily large, if one starts from a known initial position. Equivalently, if one instead uses the current position as reference point, it means that the quality of older observations decays as time progresses.

For the purpose of e.g. collision avoidance, it is obvious that having an accurate estimate of the robot position relative to the current observations is to be preferred to having a very inaccurate position estimate relative to some old observations. Because of this, too old and uncertain landmarks are in many cases deemed worthless, and they are thus pruned from the map. This is especially true for so called *odometry methods*, which are based on 'dead reckoning', and use only fairly recent observations.

There are a number of different strategies available to mitigate (or in some cases eliminate) this unlimited error accumulation. One such idea is to try to reduce the uncertainty of the older landmarks by detecting them again, if the robot returns to that particular area of the map. This is the so called *loop closure* problem, where the goal is to join spatially close but temporally distant areas of the map. Being able to detect loops in the trajectory typically allows a drastic reduction in the accumulated positioning error, as demonstrated by e.g. Newman and Ho (2005) and Jones and Soatto (2011). However, if the loops are allowed to be of arbitrary length, the storage of, and comparison against, an increasingly large map becomes inhibiting both in terms of storage and computation time.

Another strategy that can sometimes be used to limit the uncertainty is to supplement the observations with measurements made relative to a global fixed point. Such measurements can be practically realised in a number of ways, e.g. by using a global navigation satellite system such as GPS or Galileo, but a common problem with such beacon based solutions is the requirement of supporting infrastructure around the robot. In the particular case of global navigation satellite systems, it should be noted that they do not work well indoors or on other planets, and they are not a panacea for autonomous navigation systems.

Still another scheme for countering the accumulation of error becomes available when the robot motion is known to be constrained in some way.

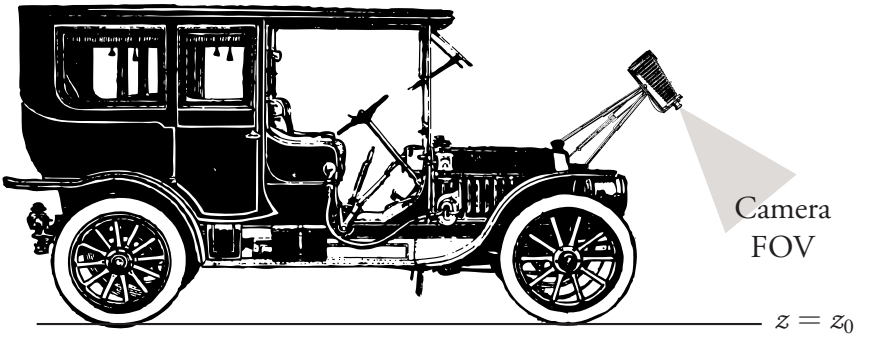


Figure 4.2: As this autonomous car drives around on reasonably flat roads, the camera (and other potential sensors not seen in the above image) will remain at a constant height above the ground plane and will thus undergo *planar motion*.

We shall discuss a frequently occurring constrained motion, namely *planar motion*, in the next section.

4.2 Planar Motion

In mobile robotics applications the cameras and other sensors are frequently mounted rigidly onto a mobile platform, which means that they will remain at a constant height above the ground. The ground can typically be assumed to be at least locally planar, and the motion of the robot is then constrained to a plane parallel to the ground (or the floor, in indoor scenarios). By considering methods which explicitly assume planar motion, the vertical positioning error of the attached sensors can automatically be bounded over arbitrarily long motion sequences.

An early method which explicitly used the planar motion assumption was presented by Ortín and Montiel (2001), who considered a forward-oriented camera mounted onto a robot. In their parametrisation, the camera was assumed to be mounted with the y-axis vertical and the motion was assumed to occur in the xz-plane. By assuming unit length translations of the form $\boldsymbol{t} = (\sin\alpha, 0, \cos\alpha)$, the essential matrix could be assumed to be

of the simple form

$$\boldsymbol{E} = \begin{bmatrix} 0 & -\cos\alpha & 0 \\ \cos(\varphi - \alpha) & 0 & \sin(\varphi - \alpha) \\ 0 & \sin\alpha & 0 \end{bmatrix}. \tag{4.1}$$

This parametrisation allowed the two motion parameters α and φ to be determined using three point correspondences and a linear method, or using two correspondences and a non-linear method. Since the translations were assumed to be of unit length, the method used a varying global scale, and was thus susceptible to scale drift.

Essentially the same motion parametrisation, together with an additional nonholonomic constraint based on the assumption that the local motion is a circular motion, was used by Scaramuzza (2011a,b). The additional nonholonomic constraint enabled local motion estimation to be performed using only one point correspondence, which allowed for an impressively efficient outlier removal scheme based on histogram voting. The proposed method demonstrated good results on relatively long motion sequences captured from a camera mounted onto a car. Though the nonholonomic constraint might be valid for the autonomous car in Figure 4.2, it is not valid for the robot in Figure 4.1 since it has omnidirectional wheels.

In contrast to the two methods mentioned above, Liang and Pears (2002) considered a camera viewing the floor, which allowed the motion to be parametrised by an inter-image homography essentially identical to the one that will be described in Section 4.4. A useful contribution of this paper is the realisation that the eigenvalues of the homography $\boldsymbol{H}$ are $\{s, se^{i\varphi}, se^{-i\varphi}\}$, where φ is the rotation angle around the vertical axis and $s \neq 0$ is an arbitrary scalar, and they additionally showed how the sign of the angle could be determined.

The approach proposed by Hajjdiab and Laganière (2004) also uses images of the floor to compute the robot motion. The idea in this method is to first transform the image to an overhead view, i.e. a synthetic view as seen straight from above (similar to Figure 4.4), and then use normalised cross-correlation to determine the translation $\boldsymbol{t}$ and rotation angle φ around the vertical axis. The transformation to the overhead view was

achieved by undoing a one-angle camera tilt, which was estimated initially and subsequently refined for the following images.

Another planar slam system using dense matching, and which is demonstrated to perform well under many different conditions, is described in Zienkiewicz and Davison (2015). The camera pose and motion are determined through a non-linear optimisation of the intensity differences, using the Huber cost function (Huber, 1964) to limit the influence of out-of-plane objects. Since the full camera pose is computed during the registration of the images, no effort has to be made in order to mount the camera in a particular way. The authors also provide a highly efficient GPU-based implementation which allows the system to run in real-time with a high frame rate.

The method described in Wadenbäck and Heyden (2013, 2014a), and which will be discussed in detail in Chapter 6, uses an iterative technique to recover both the full camera orientation and the translation vector from a homography induced by the floor plane. The idea in Hajjdiab and Laganière (2004) to consider the overhead view is applied in Wadenbäck et al. (2017), but here feature points are used instead of the normalised cross correlation, and the camera tilt is determined in the same way as in the earlier work by Wadenbäck and Heyden (2013, 2014a).

4.3 Camera Parametrisation

After this short review of planar SLAM methods, let us now turn to the problem of parametrising the camera motion under the planar motion assumptions above. Out of convenience, our choice of coordinate system will be such that the camera motion occurs in the plane $z = 0$ and the ground plane has the equation $z = 1$, as illustrated in Figure 4.3. This choice is not a restriction of the general case, since the global scale cannot be recovered from images alone. Also, we note that the transformation between the camera coordinate frame and the coordinate system of the actual robot may be found by solving the *hand-eye calibration problem* (Horaud and Dornaika, 1995; Tsai and Lenz, 1989), which, however important, is outside the scope of our considerations here.

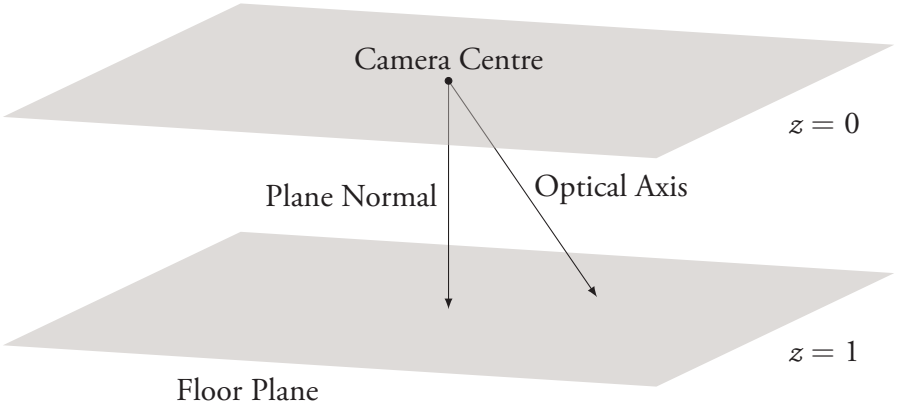


Figure 4.3: The camera moves freely in the plane $z = 0$, and can rotate about the normal of the plane, but the angle to the plane normal (tilt) is held constant.

We will assume that the camera has been calibrated in terms of lens distortion (see Section 2.3.2), so that the simple perspective projection model in Section 2.3.1 holds. Additionally, the internal camera parameters are assumed to be known, which allows us to normalise all image coordinates by applying $\boldsymbol{K}^{-1}$ to them.

We know from Section 2.3.1 that the camera position and orientation can be expressed in terms of a rotation matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{t} = (t_x, t_y, t_z)$. We choose to parametrise the rotation matrix using Tait-Bryan angles (ψ, θ, φ), as explained in Section 2.2. With the assumptions of planar motion and a rigidly mounted camera imposed, the two angles ψ and θ will be constant, and $t_z = 0$. The camera projection matrix associated with an image taken at a position $\boldsymbol{t} = (t_x, t_y, 0)$ and rotated an angle φ about the floor normal $\boldsymbol{n} = (0, 0, 1)$ will be

$$\boldsymbol{P} = \boldsymbol{R}(\psi, \theta, \varphi) \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{t} \end{bmatrix} = \boldsymbol{R}_{\psi\theta} \boldsymbol{R}_z(\varphi) \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{t} \end{bmatrix}, \tag{4.2}$$

where $\boldsymbol{R}_{\psi\theta} = \boldsymbol{R}(\psi, \theta, 0)$ encodes the camera tilt and $\boldsymbol{R}_z(\varphi)$ is the rotation about the ground plane normal (i.e. the z-axis).

In summary, a calibrated camera obeying the planar motion model can

(a) Original image. (b) Rectified image.

Figure 4.4: A typical image taken by a camera under the conditions described in this thesis is shown in (a). A rectified version, as if seen straight from above, can be seen in (b). Measurements such as distances, angles, and areas are rendered useless in (a) because of the perspective effects. In (b), these are all valid concepts.

be parametrised using a vector

$$\boldsymbol{\eta} = (\psi, \theta, \varphi, t_x, t_y). \tag{4.3}$$

The presence of a tilt $\boldsymbol{R}_{\psi\theta}$ means that the image does not correspond to a rectangular area of the floor, but an irregular quadrilateral, as illustrated in Figure 4.4.

4.4 The Inter-Image Homography

We now derive an expression for the inter-image homography between two images taken at different locations in the geometrical situation described in Section 4.3. Without loss of generality, we assume that one of the cameras has its centre at the origin, and we thus write the camera projection matrices for the two images as

$$\begin{cases} \boldsymbol{P} = \boldsymbol{R}_{\psi\theta} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}, \\ \hat{\boldsymbol{P}} = \boldsymbol{R}_{\psi\theta} \boldsymbol{R}_z(\varphi) \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{t} \end{bmatrix}. \end{cases} \tag{4.4}$$

The inter-image homography between $\boldsymbol{P}$ and $\hat{\boldsymbol{P}}$ induced by the ground plane $z = 1$ can be obtained in a number of different ways, e.g. by adapting Theorem 3.2 (or, equivalently, Corollary 5.1). Here, we shall derive it by simply projecting a point $X = (x, y, 1, 1)$ in the floor plane into the two images, and in this way obtain the homography matrix.

The projection of $X = (x, y, 1, 1)$ into the first image will be

$$\boldsymbol{x} \sim \boldsymbol{P}\boldsymbol{X} = \boldsymbol{R}_{\psi\theta} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{4.5}$$

and the projection into the second one will be

$$\hat{\boldsymbol{x}} \sim \hat{\boldsymbol{P}}\boldsymbol{X} = \boldsymbol{R}_{\psi\theta}\boldsymbol{R}_z(\varphi) \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{4.6}$$

From (4.5) and (4.6) it follows that the homography between the two images can be written as

$$\boldsymbol{H} = s\boldsymbol{R}_{\psi\theta}\boldsymbol{R}_z(\varphi)\boldsymbol{T}\boldsymbol{R}_{\psi\theta}^T, \tag{4.7}$$

for any $s \neq 0$ and with

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix} = \boldsymbol{I} - \boldsymbol{t}\boldsymbol{n}^T. \tag{4.8}$$

By ensuring $s = 1$ we have a unique representation (this may be achieved since $\det \boldsymbol{H} = s^3$). The parametrisation (4.7) gives the homography matrix in terms of physical parameters that are easily interpreted, and it is this parametrisation that will be used in the subsequent chapters.

Without surprise, we note that the homography (4.7) has a structure which allows it to be decomposed as the transformation $\boldsymbol{R}_{\psi\theta}^T$ to the overhead view followed by a 2D rigid body motion

$$\boldsymbol{R}_z(\varphi)\boldsymbol{T} = \begin{bmatrix} \cos\varphi & -\sin\varphi & \hat{t}_x \\ \sin\varphi & \cos\varphi & \hat{t}_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

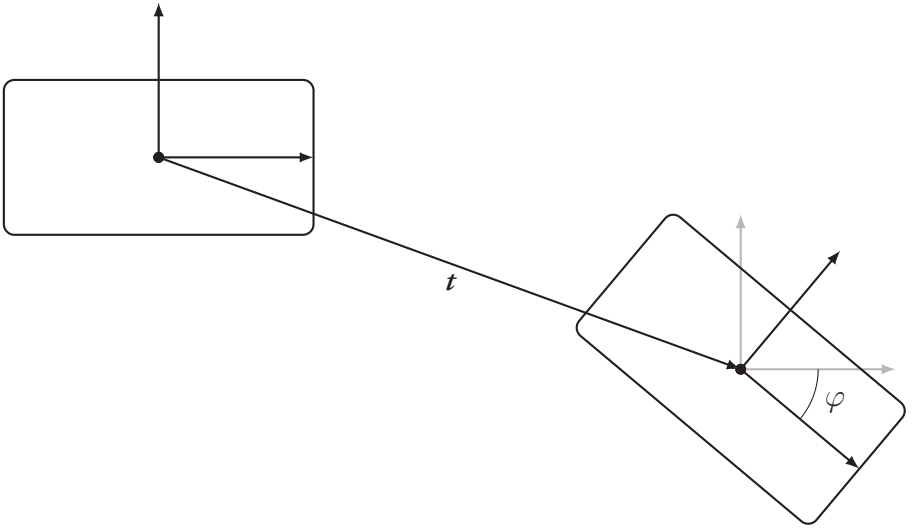


Figure 4.5: Illustration of the two-dimensional rigid body motion in the overhead view. The motion of the platform is described by a displacement $\boldsymbol{t}$ and a rotation an angle φ around the plane normal.

in the plane $z = 0$, and finally a transformation $\boldsymbol{R}_{\psi\theta}$ back from the overhead view. By using the simpler one-angle tilt $\boldsymbol{R}_x(\psi)$ instead of $\boldsymbol{R}_{\psi\theta}$ we would obtain the same decomposition into the overhead view transformation and 2D motion as Hajjdiab and Laganière (2004). Figure 4.5 illustrates the rigid motion in the overhead view.

Chapter 5

Homography Decomposition

This chapter is concerned with the recovery of the motion parameters for a homography representing planar camera motion. We start by discussing the general homography decomposition problem and reviewing the major methods for solving it. In Section 5.2 we derive in detail one such method which was introduced in Wadenbäck, Åström and Heyden (2016).

5.1 The Homography Decomposition Problem

We saw in Theorem 3.2 that a plane visible in the views of two projective cameras $\boldsymbol{P} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}$ and $\hat{\boldsymbol{P}} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \end{bmatrix}$ induces a homography between the two views, and that this homography always can be parametrised as $\boldsymbol{H} = \boldsymbol{A} - \boldsymbol{b}\boldsymbol{\nu}^T$, where $\boldsymbol{\pi} = (\boldsymbol{\nu}, 1)$ represents the plane. If we only know the homography $\boldsymbol{H}$, it would be very useful if the two camera poses and the inducing plane could somehow be extracted from the homography. If we are happy with *any* such configuration that is compatible with $\boldsymbol{H}$, then we are in for good news, as almost any choice of $\boldsymbol{b}$ and $\boldsymbol{\nu}$ will make the matrix $\boldsymbol{A} = \boldsymbol{H} + \boldsymbol{b}\boldsymbol{\nu}^T$ invertible, and thus give a valid decomposition of $\boldsymbol{H}$ (it can readily be verified that it satisfies (3.16) for $\boldsymbol{F} = \begin{bmatrix} \boldsymbol{b} \end{bmatrix}_\times \boldsymbol{A}$, and hence is compatible with the resulting epipolar geometry). If, on the other hand, we had hoped to retrieve 'the actual configuration', we must sadly accept the fact that this cannot be accomplished without additional information or requirements. This negative result is one of the concluding remarks in Zhang and Hanson (1996).

In the particular case of two *calibrated* views, i.e. $\boldsymbol{A}$ is a rotation matrix,

it turns out that there are only a finite number of possible directions for $\boldsymbol{b}$ and $\boldsymbol{\nu}$ (there is of course still a scaling ambiguity, which allows us to scale $\boldsymbol{b}$ a factor λ and $\boldsymbol{\nu}$ a factor λ^{-1}). An adaptation of Theorem 3.2 to this calibrated case gives the following.

Corollary 5.1. *(Homography Decomposition) For two calibrated views with camera matrices* $\boldsymbol{P} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}$ *and* $\hat{\boldsymbol{P}} = \boldsymbol{R}\begin{bmatrix} \boldsymbol{I} & -\boldsymbol{t} \end{bmatrix}$ *the scene plane* $\boldsymbol{\pi} = (\boldsymbol{n}, d)$ *induces a homography of the form*

$$\boldsymbol{H} = \boldsymbol{R} + \boldsymbol{R}\boldsymbol{t}\boldsymbol{n}^T/d \tag{5.1}$$

between the two views. The problem of recovering the possible configurations is known as the ***homography decomposition problem****.*

The homography decomposition problem was first solved by Faugeras and Lustman (1988), and they proved that the decomposition problem has eight different solutions except in some special cases (e.g. if $\boldsymbol{H}$ is a pure rotation then at least one of $\boldsymbol{b}$ and $\boldsymbol{\nu}$ are zero, and the other one is entirely arbitrary). Their proof of the number of solutions is constructive and exhaustively considers all cases, and thus it also provides a practical method to find the solutions (when possible).

The decomposition method proposed by Faugeras and Lustman (1988) uses a singular value decomposition $\boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ to transform the problem to an equivalent problem with a diagonal homography matrix $\boldsymbol{\Sigma}$. This transformed problem can be solved analytically, and then a transformation of the obtained solutions back to the original problem using $\boldsymbol{U}$ and $\boldsymbol{V}$ gives the sought configurations.

Out of the eight mathematical solutions, only two are in fact physically possible, and it is shown in the paper that the impossible ones can be discarded if one has access to a number of point correspondences that are mapped by the homography between the two views. This is done by requiring e.g. that all the viewed scene points are in front of both cameras.

Another similar decomposition approach, based on the consideration of a spectral decomposition $\boldsymbol{H}^T\boldsymbol{H} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$, was proposed by Zhang and Hanson (1996). Their method gives a slightly simpler handling of the special cases compared to Faugeras and Lustman (1988), and is also claimed to be computationally cheaper.

A more recent method, which gives the decomposition as explicit formulae in terms of the entries of the homography matrix, was proposed in Malis and Vargas (2007). In addition to reviewing the two earlier methods and providing a slightly clearer exposition of the method in Zhang and Hanson (1996), this report shows that the constraint

$$d + \boldsymbol{n}^T \boldsymbol{t} > 0 \tag{5.2}$$

can be used to eliminate four of the six impossible solutions. This constraint is just as valid irrespective of which of the decomposition methods is used.

Unaware, at the time, of the report by Malis and Vargas (2007) and the constraint (5.2), Wadenbäck, Åström and Heyden (2016) included as an auxiliary result, and an extension of the findings in Wadenbäck and Heyden (2014b), another SVD-based decomposition method which only results in four decompositions. While in hindsight this method does not offer any significant improvement over a combination of e.g. the method by Zhang and Hanson (1996) and the constraint (5.2), it expresses the decomposition more accessibly in terms of the generating motion parameters $\boldsymbol{\eta} = (\psi, \theta, \varphi, t_x, t_y)$ from Section 4.3. We describe this method in detail in the next section.

5.2 SVD-Based Recovery Method

We next turn to the problem of recovering the generating parameter vector $\boldsymbol{\eta}$ from a given homography of the form (4.7), i.e.

$$\boldsymbol{H} = s\boldsymbol{R}_{\psi\theta}\boldsymbol{R}_z(\varphi)\boldsymbol{T}\boldsymbol{R}_{\psi\theta}^T, \tag{5.3}$$

and normalised so that $\det \boldsymbol{H} = 1$.

Denote $\tau = \|\boldsymbol{t}\| = \sqrt{t_x^2 + t_y^2}$. With $t_x = \tau\cos\alpha$ and $t_y = \tau\sin\alpha$, the upper triangular matrix $\boldsymbol{T}$ in (5.3) may be written

$$\begin{bmatrix} 1 & 0 & -\tau\cos\alpha \\ 0 & 1 & -\tau\sin\alpha \\ 0 & 0 & 1 \end{bmatrix} = \boldsymbol{R}_z(\alpha) \underbrace{\begin{bmatrix} 1 & 0 & -\tau \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\hat{T}} \boldsymbol{R}_z^T(\alpha). \tag{5.4}$$

Since $\boldsymbol{R}_{\psi\theta}$ and $\boldsymbol{R}_z(\varphi)$ are orthogonal matrices, $\boldsymbol{H}$, $\hat{\boldsymbol{T}}$, and $\boldsymbol{T}$ must all have the same singular values, and they are found to satisfy

$$\begin{cases} \sigma_1^2 = 1 + \dfrac{\tau^2}{2} + \dfrac{\tau}{2}\sqrt{4+\tau^2}, \\ \sigma_2^2 = 1, \\ \sigma_3^2 = 1 + \dfrac{\tau^2}{2} - \dfrac{\tau}{2}\sqrt{4+\tau^2}. \end{cases} \tag{5.5}$$

Solving for τ in terms of σ_1 gives $(\sigma_1^2 - 1)^2 = [\ldots] = \tau^2\sigma_1^2$, so that

$$\tau^2 = \frac{(\sigma_1^2 - 1)^2}{\sigma_1^2} = \left(\sigma_1 - \frac{1}{\sigma_1}\right)^2, \tag{5.6}$$

and since $\sigma_1\sigma_3 = 1$ this gives

$$\tau = \sigma_1 - \sigma_3. \tag{5.7}$$

Essentially the same idea was used in Wadenbäck and Heyden (2014b) to recover τ, but expressed in a slightly more complicated way in terms of the condition number. A generalisation of the result (5.7) may be found in the note by Wadenbäck (2015).

Since eigenvalues are invariant under similarity transformations, i.e. transformations $\boldsymbol{A} = \boldsymbol{S}^{-1}\boldsymbol{B}\boldsymbol{S}$, the eigenvalues of $\boldsymbol{H}$ are the same as those of $\boldsymbol{R}_z(\varphi)\boldsymbol{T}$, and they will be 1 and $e^{\pm i\varphi}$. This result is due to Liang and Pears (2002), as mentioned earlier in Section 4.2. This allows us to compute (up to a sign ambiguity) the planar rotation as

$$\varphi = \pm \arccos\left(\frac{\operatorname{tr}\boldsymbol{H} - 1}{2}\right). \tag{5.8}$$

Now let $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \sigma_2, \sigma_3)$ and suppose $\hat{\boldsymbol{T}} = \hat{\boldsymbol{U}}\boldsymbol{\Sigma}\hat{\boldsymbol{V}}^T$ is an SVD of $\hat{\boldsymbol{T}}$ with $\det\hat{\boldsymbol{U}} = \det\hat{\boldsymbol{V}} = 1$, then

$$\boldsymbol{T} = \boldsymbol{R}_z(\alpha)\hat{\boldsymbol{U}}\boldsymbol{\Sigma}\hat{\boldsymbol{V}}^T\boldsymbol{R}_z^T(\alpha) \tag{5.9}$$

Table 5.1: The four ambiguity cases for the parameters. Out of these four cases, in fact only Case A and Case D are physically possible.

Case A	Case B	Case C	Case D
ψ	$\psi + \pi$	$\psi + \pi$	ψ
θ	$\pi - \theta$	$-\theta$	$\theta - \pi$
φ	φ	$-\varphi$	$-\varphi$
t_x	$-t_x$	$-t_x$	t_x
t_y	$-t_y$	t_y	$-t_y$

will be an SVD of $\boldsymbol{T}$. Let $\boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be an SVD of $\boldsymbol{H}$, then we must have

$$\begin{cases} \boldsymbol{U}\boldsymbol{D}_1 = \boldsymbol{R}_{\psi\theta}\boldsymbol{R}_z(\varphi)\boldsymbol{R}_z(\alpha)\hat{\boldsymbol{U}}, \\ \boldsymbol{V}\boldsymbol{D}_2 = \boldsymbol{R}_{\psi\theta}\boldsymbol{R}_z(\alpha)\hat{\boldsymbol{V}}, \end{cases} \tag{5.10}$$

where $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ are diagonal matrices with ± 1 on their diagonal entries (column-flipping matrices). Since $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{V}}$ were chosen with positive determinant, it follows that $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ have an even number of negative diagonal entries. This gives four possibilities for $\boldsymbol{D}_2$ (one with all entries equal to one, and three where two entries are minus one).

Using the second equation in (5.10) together with the fact that

$$\boldsymbol{R}_{\psi\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ \sin\psi\sin\theta & \cos\psi & -\sin\psi\cos\theta \\ -\cos\psi\sin\theta & \sin\psi & \cos\psi\cos\theta \end{bmatrix}, \tag{5.11}$$

we see that element $(1,2)$ in the matrix $\boldsymbol{V}\boldsymbol{D}_2\hat{\boldsymbol{V}}^T\boldsymbol{R}_z^T(\alpha)$ should be zero. For each of the four choices of $\boldsymbol{D}_2$, this gives two solutions for α. We use these solutions in the first condition of (5.10) to determine which of the four choices of $\boldsymbol{D}_1$ work. It is now possible to use (5.11) to recover ψ and θ from $\boldsymbol{V}\boldsymbol{D}_2\hat{\boldsymbol{V}}^T\boldsymbol{R}_z^T(\alpha)$.

The parameters $\boldsymbol{\eta} = (\psi, \theta, \varphi, t_x, t_y)$ are not uniquely determined by the homography $\boldsymbol{H}$. The four possibilities are shown in Table 5.1. Out of these, only two are in fact physically possible.

The numerical accuracy of this decomposition method was evaluated in Wadenbäck, Åström and Heyden (2016), and it was found that the generat-

ing parameters could be recovered up to full floating point precision, thus making it comparable with the other homography decomposition methods.

Chapter 6

Iterative Motion Recovery

In this chapter we shall describe the iterative scheme which was introduced in Wadenbäck and Heyden (2013) and extended in Wadenbäck and Heyden (2014a). The goal of these methods was the same as that of the SVD-based direct method in Section 5.2, i.e. to recover the generating parameter vector $\boldsymbol{\eta} = (\psi, \theta, \varphi, t_x, t_y)$ from a given homography of the form

$$\boldsymbol{H} = s\boldsymbol{R}_{\psi\theta}\boldsymbol{R}_z(\varphi)\boldsymbol{T}\boldsymbol{R}_{\psi\theta}^T, \tag{6.1}$$

with

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{6.2}$$

This is the familiar inter-image homography from Section 4.4, which obeys the planar motion assumption.

In comparison with the homography decomposition methods studied in Chapter 5, the iterative approaches described here are undeniably slower, but they have the significant advantage that they do not produce any physically impossible solutions, and they only produce *one* solution. Additionally, the extension in Wadenbäck and Heyden (2014a) allows the constant angles ψ and θ to be determined from any number of such homographies simultaneously.

The underlying idea of the iterative method is that we wish to undo the tilt and consider the rigid transformation in the overhead view, similarly to Hajjdiab and Laganière (2004). This is done by first determining the tilt $\boldsymbol{R}_{\psi\theta}$, and then extracting the rotation and translation from the rigid motion $\boldsymbol{R}_z(\varphi)\boldsymbol{T}$.

6.1 Eliminating φ and $\boldsymbol{t}$

The first step, therefore, is to eliminate the dependence on φ and $\boldsymbol{t}$, and try to obtain equations in ψ and θ only. Separating the tilt angles ψ and θ from the rigid transformations parameters $\boldsymbol{t}$ and φ in (4.7), we get

$$\boldsymbol{R}_{\psi\theta}^{T}\boldsymbol{H}\boldsymbol{R}_{\psi\theta} = s\boldsymbol{R}_{z}(\varphi)\boldsymbol{T}. \tag{6.3}$$

Here, one notes that $\boldsymbol{R}_{z}(\varphi)$ can be eliminated by multiplying with the transpose from the left on both sides. This results in the relation

$$\boldsymbol{R}_{\psi\theta}^{T}\boldsymbol{M}\boldsymbol{R}_{\psi\theta} = s^{2}\boldsymbol{T}^{T}\boldsymbol{T}, \tag{6.4}$$

with a symmetric matrix

$$\boldsymbol{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{12} & m_{22} & m_{23} \\ m_{13} & m_{23} & m_{33} \end{bmatrix} = \boldsymbol{H}^{T}\boldsymbol{H}. \tag{6.5}$$

Since both sides of (6.4) are symmetric matrices, one obtains six unique equations. Let $\mathcal{L} = \boldsymbol{R}_{\psi\theta}^{T}\boldsymbol{M}\boldsymbol{R}_{\psi\theta}$ and $\mathcal{R} = s^{2}\boldsymbol{T}^{T}\boldsymbol{T}$ be the left and right hand sides of (6.4), respectively. Evaluating $\mathcal{R}$, one obtains

$$\mathcal{R} = s^{2}\begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ -t_x & -t_y & 1 + t_x^{2} + t_y^{2} \end{bmatrix}. \tag{6.6}$$

As described in Section 2.2.1, $\boldsymbol{R}_{\psi\theta} = \boldsymbol{R}(\psi, \theta, 0)$ is a rotation of θ around the y-axis followed by a rotation of ψ around the x-axis. Direct multiplication of the rotation matrices allows us to evaluate $\mathcal{L}$, and one finds that $\mathcal{L}$ is a fourth degree polynomial expression in $\cos\psi$, $\sin\psi$, $\cos\theta$ and $\sin\theta$.

Noting that $\mathcal{R}_{11}$, $\mathcal{R}_{12}$ and $\mathcal{R}_{22}$ are independent of $\boldsymbol{t}$, the equations for ψ and θ become

$$\begin{cases} \mathcal{L}_{11} - \mathcal{L}_{22} = 0, \\ \mathcal{L}_{12} = 0. \end{cases} \tag{6.7}$$

6.2 Iterative Scheme

The system (6.7) is a system of complicated equations, but instead of trying to solve (6.7) for both ψ and θ at the same time, we will iteratively alternate between solving for one angle, with the other held fixed. This reduces the problem of solving a fourth degree trigonometric equation, so that we instead iterate and solve a second degree equation in each iteration.

Before explaining in detail how these equations are solved, we first outline in Algorithm 6.1 the iterative scheme which produces an approximation to $\boldsymbol{R}_{\psi\theta}$. Since

$$\boldsymbol{R}_{\psi\theta} = \boldsymbol{R}(\psi, \theta, 0) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ \sin\psi\sin\theta & \cos\psi & -\sin\psi\cos\theta \\ -\cos\psi\sin\theta & \sin\psi & \cos\psi\cos\theta \end{bmatrix}, \qquad (6.8)$$

it is easy to compute ψ and θ from this approximation.

Input: An inter-image homography $\boldsymbol{H}$
Output: An approximation $\boldsymbol{R}^*_{\psi\theta}$ of $\boldsymbol{R}_{\psi\theta}$
1: $\widehat{\boldsymbol{M}} \leftarrow \boldsymbol{H}^T\boldsymbol{H}$
2: $\theta_0 \leftarrow 0$
3: $\boldsymbol{R} \leftarrow \boldsymbol{R}_y(\theta_0)$
4: **for** $j = 1, \ldots, N$ **do**
5: $\quad \widehat{\boldsymbol{M}} \leftarrow \boldsymbol{R}_y(\theta_{j-1})^T\widehat{\boldsymbol{M}}\boldsymbol{R}_y(\theta_{j-1})$
6: $\quad$ Solve for ψ_j
7: $\quad \widehat{\boldsymbol{M}} \leftarrow \boldsymbol{R}_x(\psi_j)^T\widehat{\boldsymbol{M}}\boldsymbol{R}_x(\psi_j)$
8: $\quad$ Solve for θ_j
9: $\quad \boldsymbol{R} \leftarrow \boldsymbol{R}\boldsymbol{R}_x(\psi_j)\boldsymbol{R}_y(\theta_j)$
10: $\boldsymbol{R}^*_{\psi\theta} \leftarrow \boldsymbol{R}$

Algorithm 6.1: Iteratively approximate $\boldsymbol{R}_{\psi\theta}$. The steps on line 6 and line 8 are detailed in Sections 6.2.1 and 6.2.2. Since we 'embed' into $\widehat{\boldsymbol{M}}$ the current approximation, we may assume that the fixed angle is zero when solving for the free one.

6.2.1 Solving for ψ

Since $\cos\psi$ and $\sin\psi$ cannot both be zero, (6.7) is equivalent to

$$\begin{cases} \mathcal{L}_{11} - \mathcal{L}_{22} = 0, \\ \cos(\psi)\mathcal{L}_{12} = 0, \\ \sin(\psi)\mathcal{L}_{12} = 0. \end{cases} \tag{6.9}$$

By letting $\widehat{\boldsymbol{M}} = \boldsymbol{R}_y(\theta)^T \boldsymbol{M} \boldsymbol{R}_y(\theta)$ this can be written in matrix form as

$$\begin{bmatrix} \hat{m}_{11} - \hat{m}_{22} & -2\hat{m}_{23} & \hat{m}_{11} - \hat{m}_{33} \\ \hat{m}_{12} & \hat{m}_{13} & 0 \\ 0 & \hat{m}_{12} & \hat{m}_{13} \end{bmatrix} \begin{bmatrix} \cos^2\psi \\ \cos\psi\sin\psi \\ \sin^2\psi \end{bmatrix} = \mathbf{0}. \tag{6.10}$$

This means that $(\cos^2\psi, \cos\psi\sin\psi, \sin^2\psi)$ lies in the null space of the coefficient matrix in (6.10). Due to measurement noise, this matrix will have full rank, so we determine the approximate null space using the SVD.

Provided the approximate null space direction $\boldsymbol{v} = (v_1, v_2, v_3)$, one obtains ψ as

$$\psi = \frac{1}{2}\arcsin\frac{2v_2}{v_1 + v_3}. \tag{6.11}$$

6.2.2 Solving for θ

Now θ can be found in much a similar way as ψ was found. For very small angles, $\boldsymbol{R}_x(\psi)\boldsymbol{R}_y(\theta)$ should have approximately the same effect on the camera as $\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\psi)$, since they are both close to the identity matrix. Examination of the matrices confirms this for small angles.

Therefore, if $\widehat{\boldsymbol{M}} = \boldsymbol{R}_x(\psi)^T \boldsymbol{M} \boldsymbol{R}_x(\psi)$, then (6.7) is equivalent to

$$\begin{cases} \mathcal{L}_{11} - \mathcal{L}_{22} = 0, \\ \cos(\theta)\mathcal{L}_{12} = 0, \\ \sin(\theta)\mathcal{L}_{12} = 0, \end{cases} \tag{6.12}$$

which can be written in matrix form as

$$\begin{bmatrix} \hat{m}_{11} - \hat{m}_{22} & -2\hat{m}_{13} & \hat{m}_{33} - \hat{m}_{22} \\ \hat{m}_{12} & -\hat{m}_{23} & 0 \\ 0 & \hat{m}_{12} & -\hat{m}_{23} \end{bmatrix} \begin{bmatrix} \cos^2\theta \\ \cos\theta\sin\theta \\ \sin^2\theta \end{bmatrix} = \mathbf{0}. \tag{6.13}$$

In the same way as for ψ in Section 6.2.1, the null space direction vector $\boldsymbol{v}$ can be used to recover θ as

$$\theta = \frac{1}{2} \arcsin \frac{2v_2}{v_1 + v_3}. \tag{6.14}$$

For this whole tilt estimation method to succeed, it is assumed that the translation vector $\boldsymbol{t}$ is not zero, otherwise the tilt angles ψ and θ cannot be computed in this way. If indeed $\boldsymbol{t}$ is zero, the homography matrix will be a scalar multiple of an orthogonal matrix, and this case is thus easily and efficiently detected by a separate check. This situation arises in practice, since the robot may at times stop during the motion, e.g. to await new commands or to avoid collision with obstacles.

6.3 Experiments

We include in this section the experiments that were conducted and reported in Wadenbäck and Heyden (2013).

6.3.1 Random Homographies

In the first experiment, fifty homographies of the form (6.1) were generated with random values for $\boldsymbol{\eta} = (\psi, \theta, \varphi, t_x, t_y)$. The true angles and their corresponding estimates obtained by the iterative procedure is shown in Figure 6.1. The correct angles are recovered in almost all cases, but when the method fails, it fails dramatically.

6.3.2 Path Reconstruction

A simple path estimation technique was also tried in Wadenbäck and Heyden (2013) for both synthetic and real data. The QR decomposition was used to determine the translation and planar rotation from the rigid motion $\boldsymbol{R}_z(\varphi)\boldsymbol{T}$, after estimating the tilt as described in Section 6.2. In the simulation, noise of the magnitude corresponding to a few pixels was added to the points used to estimate the homographies. Results for this experiment are shown in Figure 6.2 and Figure 6.3.

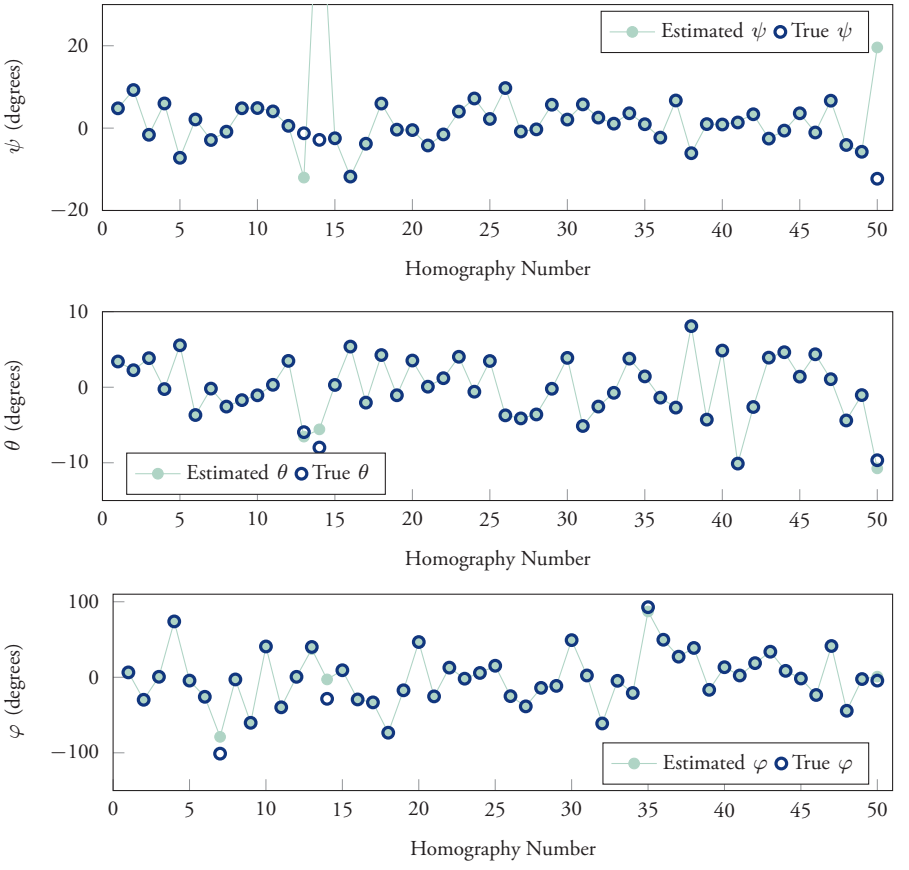


Figure 6.1: True and estimated values for ψ, θ and φ for fifty randomly generated homographies. As can be seen, the estimation works well in most instances.

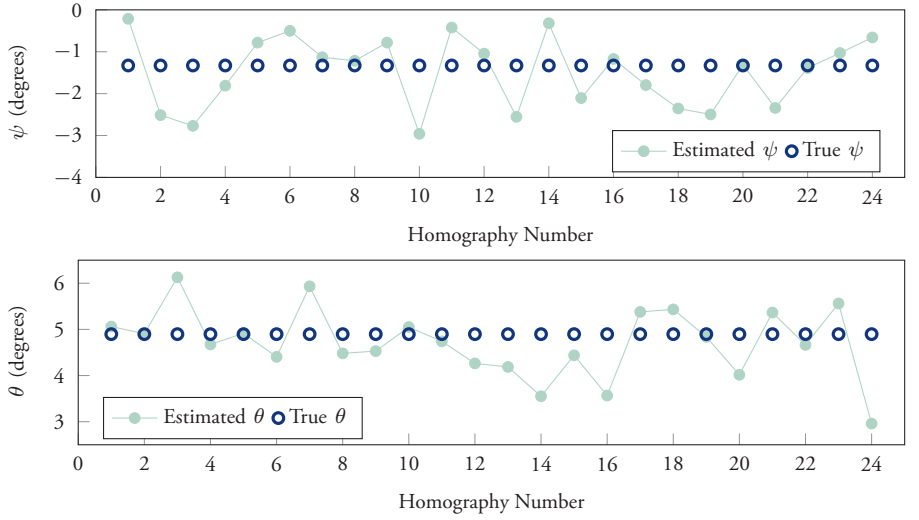


Figure 6.2: We see that the estimated values of ψ and θ are, on average, close to the true values. Since ψ and θ are constant, temporal filtering could have been used to get better estimates over time.

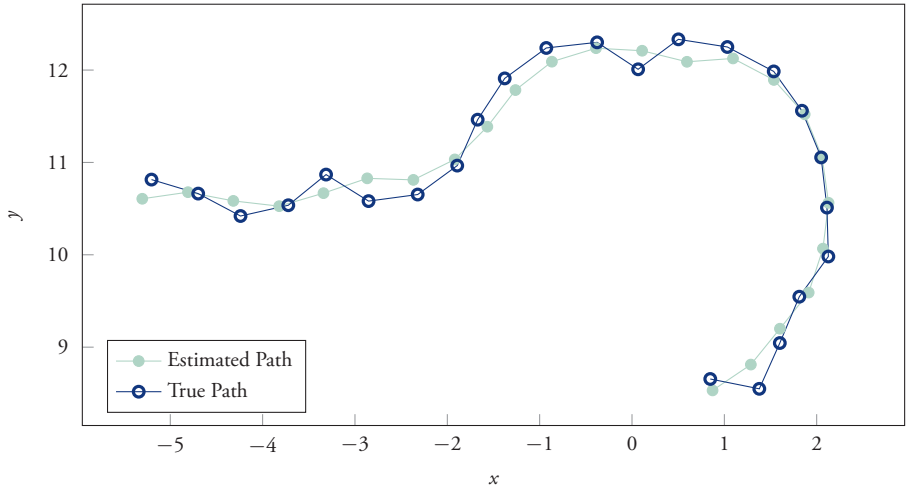


Figure 6.3: The simulated and the estimated paths. Procrustes analysis has been carried out to align the path curves for easy comparison.

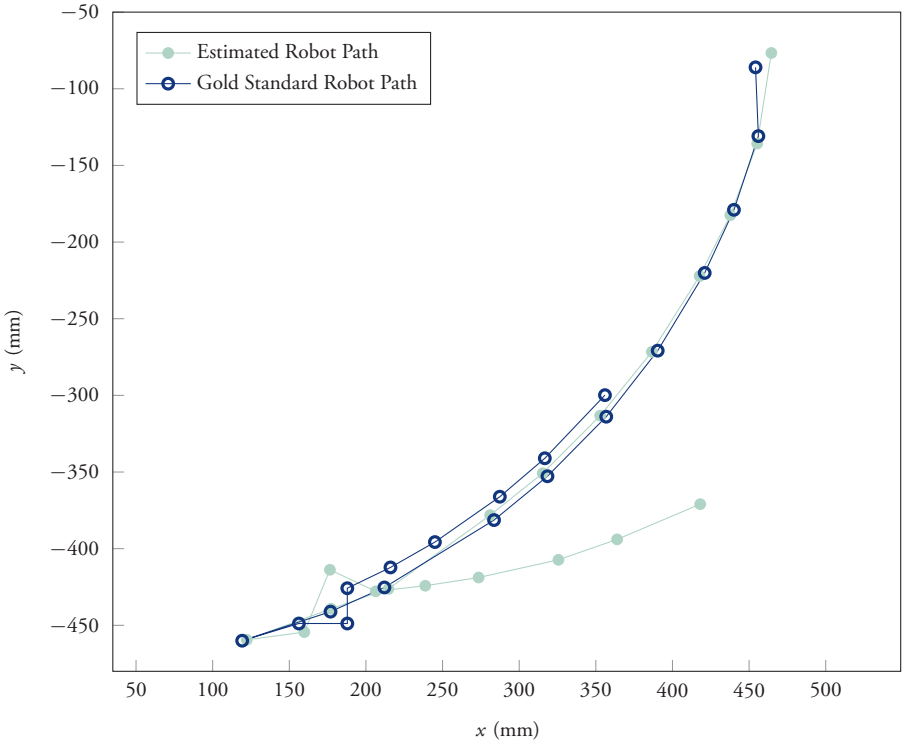


Figure 6.4: True and estimated paths for the robot experiment. At the lower part of the plot some erroneous estimates are made, which results in the estimated path being deflected away. This would be handled by a more refined path estimation method.

This simple path estimation was tried on real data as well. These data were recorded by a camera mounted onto an industrial robot arm which provided gold standard position estimates for evaluative comparison. The resulting reconstruction can be seen in Figure 6.4. For comparison, we have additionally estimated the non constant angle φ using a method based on conjugate rotations, see Liang and Pears (2002) for details. This method computes φ from the eigenvalues of the homography without estimating the tilt. Figure 6.5 shows this estimate compared to our estimate and the true value (as measured by the robot). Both methods perform well, and

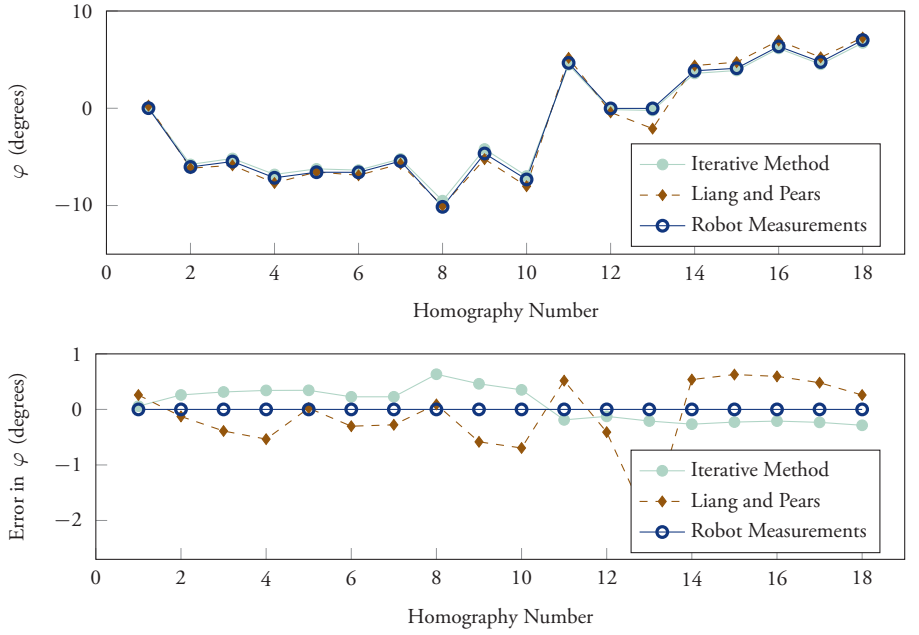


Figure 6.5: The upper plot shows the difference in orientation between consecutive images, and the lower plot shows the angular error. The plots show that our estimates of φ and the eigenvalue-based estimates of φ are both close to the truth (robot measurements).

typically give angular errors on the order of 0.5°, however tilt estimation followed by QR decomposition has a slightly favourable performance compared to the method based on the eigenvalues.

6.4 Tilt Estimation from Several Homographies

Since the camera is thought to be rigidly mounted, the tilt angles ψ and θ will be the same for all homographies arising from the camera motion. The iterative method from Section 6.2 can be extended, as was done in Wadenbäck and Heyden (2014a), to determine ψ and θ from several homographies simultaneously.

Suppose therefore that we have at our disposal a number of homographies of the form (6.1), that is,

$$\boldsymbol{H}_j = s_j \boldsymbol{R}_{\psi\theta} \boldsymbol{R}_z(\varphi_j) \boldsymbol{T}_j \boldsymbol{R}_{\psi\theta}^T, \quad j = 1, \ldots N, \tag{6.15}$$

and want to recover their generating parameters $\boldsymbol{\eta}_j = (\psi, \theta, \varphi_j, t_x^{(j)}, t_y^{(j)})$. As observed in Section 6.2, the products $\boldsymbol{R}_{\psi\theta}^T \boldsymbol{M}_j \boldsymbol{R}_{\psi\theta}$, where

$$\boldsymbol{M}_j = \begin{bmatrix} m_{11}^{(j)} & m_{12}^{(j)} & m_{13}^{(j)} \\ m_{12}^{(j)} & m_{22}^{(j)} & m_{23}^{(j)} \\ m_{13}^{(j)} & m_{23}^{(j)} & m_{33}^{(j)} \end{bmatrix} = \boldsymbol{H}_j^T \boldsymbol{H}_j, \tag{6.16}$$

are all independent of φ_j.

From each product $\boldsymbol{R}_{\psi\theta}^T \boldsymbol{M}_j \boldsymbol{R}_{\psi\theta}$ we will get equations of the form (6.10) and (6.13), where we denote the coefficient matrices

$$\boldsymbol{\Psi}_j = \begin{bmatrix} \hat{m}_{11}^{(j)} - \hat{m}_{22}^{(j)} & -2\hat{m}_{23}^{(j)} & \hat{m}_{11}^{(j)} - \hat{m}_{33}^{(j)} \\ \hat{m}_{12}^{(j)} & \hat{m}_{13}^{(j)} & 0 \\ 0 & \hat{m}_{12}^{(j)} & \hat{m}_{13}^{(j)} \end{bmatrix} \tag{6.17}$$

and

$$\boldsymbol{\Theta}_j = \begin{bmatrix} \hat{m}_{11}^{(j)} - \hat{m}_{22}^{(j)} & -2\hat{m}_{13}^{(j)} & \hat{m}_{33}^{(j)} - \hat{m}_{22}^{(j)} \\ \hat{m}_{12}^{(j)} & -\hat{m}_{23}^{(j)} & 0 \\ 0 & \hat{m}_{12}^{(j)} & -\hat{m}_{23}^{(j)} \end{bmatrix}. \tag{6.18}$$

Since all the $\boldsymbol{\Psi}_j$ should have a common null space defining ψ, we find it as the one-dimensional null space approximation of

$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\Psi}_1 \\ \vdots \\ \boldsymbol{\Psi}_N \end{bmatrix}, \tag{6.19}$$

and similarly for obtaining θ from the null space approximation of

$$\boldsymbol{\Theta} = \begin{bmatrix} \boldsymbol{\Theta}_1 \\ \vdots \\ \boldsymbol{\Theta}_N \end{bmatrix}. \tag{6.20}$$

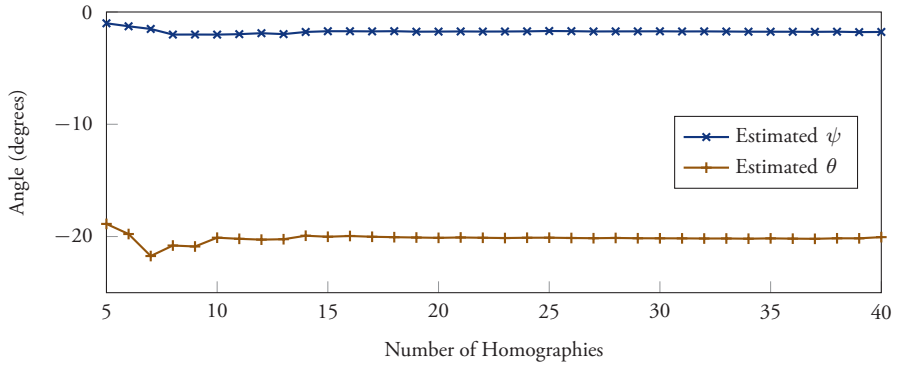


Figure 6.6: The tilt estimation reached convergence after about 15-20 frames (roughly 2 s of motion). Since the tilt estimation problem is ill-conditioned for very small translations, using fewer than five images did not give reasonable estimates with the frame rate and velocity used in the experiment.

6.5 Initial Tilt Calibration

In Wadenbäck et al. (2017), the method from Section 6.4 was used to perform an initial tilt calibration to establish the transformation $\boldsymbol{R}_{\psi\theta}^T$ to the overhead view. After this, there was no need to compute the homographies any more, as the point correspondences were used immediately to determine the rigid motion using RANSAC together with a method by Arun, Huang and Blostein (1987). The experiments were run on the Fraunhofer IPA rob@work in Figure 4.1, and the resulting tilt calibration is shown in Figure 6.6. The estimate stabilised after approximately 2 s of fairly slow motion.

6.5.1 Rigid Motion Estimation

Suppose $\hat{\boldsymbol{x}}_j \leftrightarrow \boldsymbol{x}_j$, $j = 1, \ldots, N$ are point correspondences between the first and second image. These must satisfy

$$\hat{\boldsymbol{x}}_j \sim \boldsymbol{H}\boldsymbol{x}_j \Leftrightarrow \boldsymbol{R}_{\psi\theta}^T \hat{\boldsymbol{x}}_j \sim \boldsymbol{R}_z(\varphi)\boldsymbol{T}\boldsymbol{R}_{\psi\theta}^T \boldsymbol{x}_j. \tag{6.21}$$

After the tilt has been determined using the method in Section 6.4, we

consider $\boldsymbol{R}_{\psi\theta}$ to be known. Introducing

$$\boldsymbol{y}_j \sim \boldsymbol{R}_{\psi\theta}^T \boldsymbol{x}_j \qquad \text{and} \qquad \hat{\boldsymbol{y}}_j \sim \boldsymbol{R}_{\psi\theta}^T \hat{\boldsymbol{x}}_j, \tag{6.22}$$

and using the representation with last coordinate equal to one, (6.21) becomes a planar rigid body motion in terms of $\boldsymbol{z}_j = \boldsymbol{\Pi}\boldsymbol{y}_j$ and $\hat{\boldsymbol{z}}_j = \boldsymbol{\Pi}\hat{\boldsymbol{y}}_j$. Here $\boldsymbol{\Pi}$ denotes an orthogonal projection onto the first two coordinates, i.e.

$$\boldsymbol{\Pi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{6.23}$$

If all point correspondences are correct, i.e. no outliers are present, this may be solved directly in a least-squares sense using an adaptation to 2D of the method presented in Arun, Huang and Blostein (1987). The 2D version of this problem is well-posed if at least two point correspondences are used, and the solution gives an estimate of φ and $\boldsymbol{t}$.

The least-squares solution to the rigid body motion problem presented in Arun, Huang and Blostein (1987) works by decoupling the translation and the rotation involved. It is shown that by forming

$$\boldsymbol{q}_j = \boldsymbol{z}_j - \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{z}_k \quad \text{and} \quad \hat{\boldsymbol{q}}_j = \hat{\boldsymbol{z}}_j - \frac{1}{N}\sum_{k=1}^{N} \hat{\boldsymbol{z}}_k, \tag{6.24}$$

the optimal rotation matrix is $\boldsymbol{V}\boldsymbol{U}^T$, where $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is a singular value decomposition of

$$\boldsymbol{M} = \sum_{j=1}^{N} \boldsymbol{q}_j \hat{\boldsymbol{q}}_j^T. \tag{6.25}$$

The optimal estimate $\boldsymbol{t}_{2\mathrm{D}}$ of the 2D translation vector will then be

$$\boldsymbol{t}_{2\mathrm{D}} = \Big(\frac{1}{N}\sum_{k=1}^{N} \hat{\boldsymbol{z}}_k\Big) - \boldsymbol{V}\boldsymbol{U}^T\Big(\frac{1}{N}\sum_{k=1}^{N} \boldsymbol{z}_k\Big), \tag{6.26}$$

and finally we get $\boldsymbol{t} = (\boldsymbol{t}_{2\mathrm{D}}, 0)$.

This method for recovering the rigid motion was used in a RANSAC loop for random pairs of point correspondences, and at the final step the same method was used for all the found inliers in the consensus set.

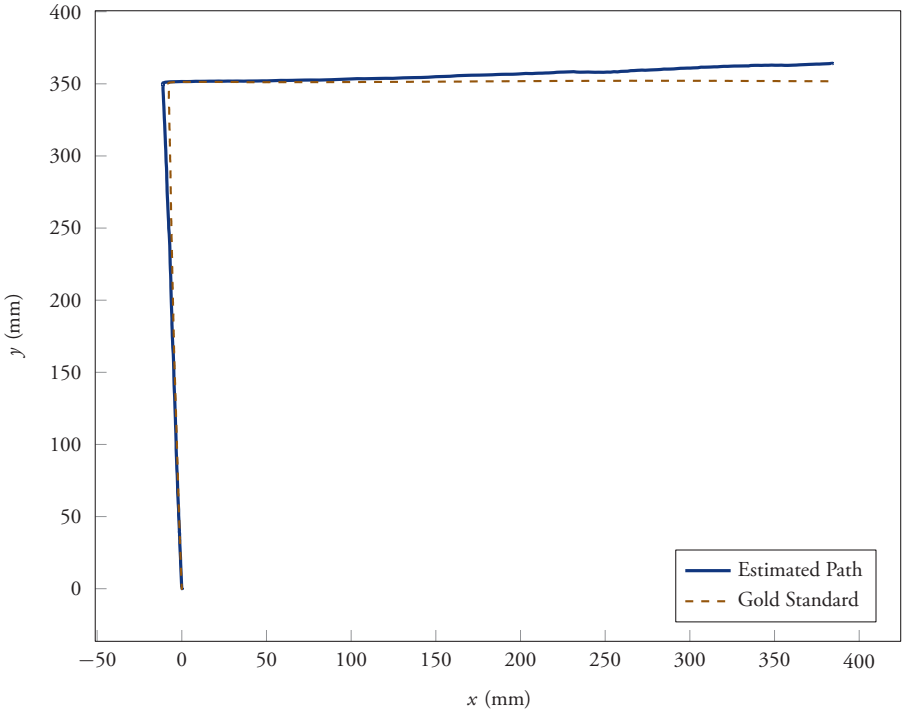


Figure 6.7: Gold standard and estimated positions for the positioning experiment. The gold standard measurements were obtained using a highly accurate optical tracking system.

6.5.2 Evaluation against Gold Standard

Among the positioning experiments presented in Wadenbäck et al. (2017) were a few evaluations against a highly accurate optical tracking system of model K600 from Nikon Metrology (Nikon Corporation 2011), which served as gold standard. This K600 system provides an absolute accuracy of less than 100 µm, and was sampled at the rate 250 Hz. We include here the positioning results for the 'parallel parking' dataset, which can be viewed in Figure 6.7. The relative positioning error, i.e. positioning error in relation to travelled distance, was found to be on the order of 2 % as shown in Figure 6.8.

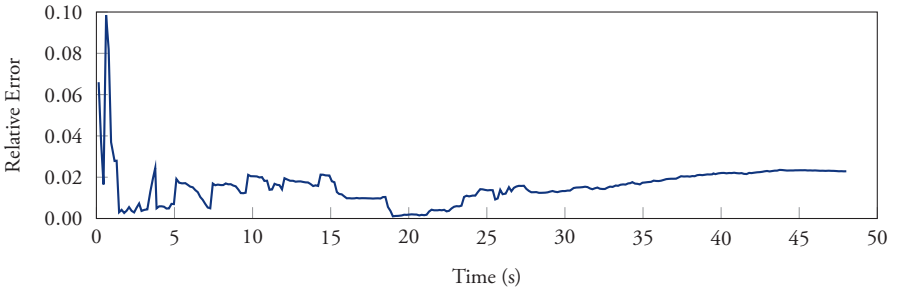


Figure 6.8: Relative estimation error, formed by dividing the absolute error with the travelled distance.

6.5.3 Evaluation on a Slightly Longer Sequence

Wadenbäck et al. (2017) also contained experiments on a slightly longer motion sequence, shaped approximately as an ellipse, with a total length of about 5.75 m and in which the robot made a full turn. The trajectory was formed such that the images at the starting position and the final position were overlapping. Due to range and workspace limitations in the Nikon system, there is no gold standard data for this experiment. Instead, the images from the starting position and the final position were used to determine the true final position, which was then compared to the final positions estimated by integrating the estimates along the trajectory. This is a meaningful comparison to make, since the accumulated error from the many homographies along the trajectory should be much larger than the error from just one homography.

The result from this experiment is shown in Figure 6.9 and Figure 6.10. The positioning error in the final position, as determined by comparing the first and final image as explained above, was found to be 0.71 % of the travelled distance.

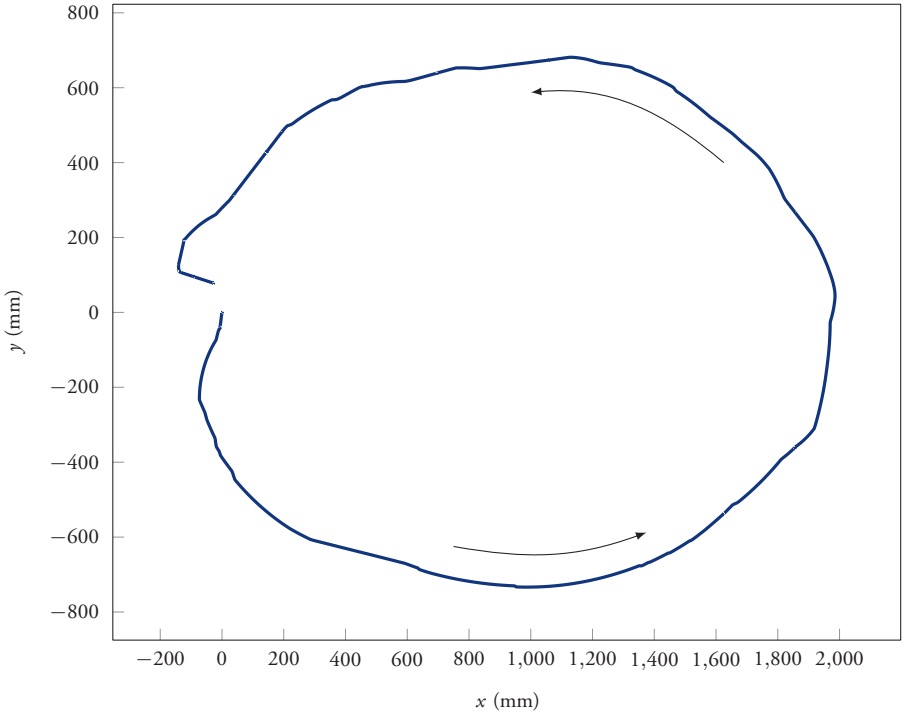


Figure 6.9: Estimated trajectory for the longer robot motion.

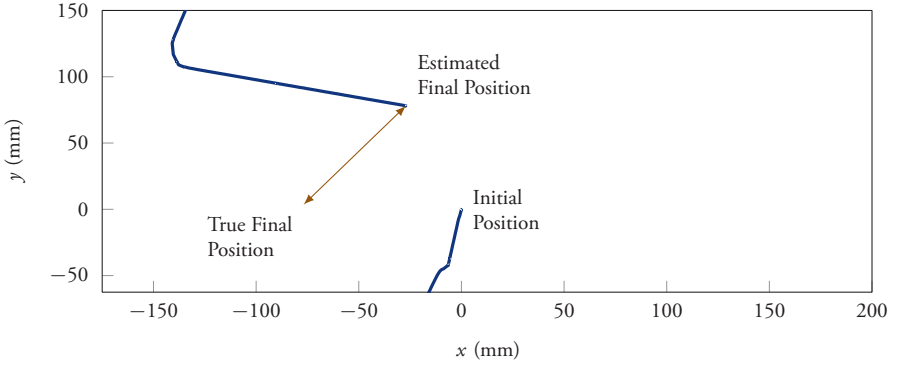


Figure 6.10: Close-up of the trajectory in Figure 6.9, showing the initial position and the final position.

Chapter 7

Compatible Homographies

In Chapter 5 and Chapter 6 we discussed a number of methods for recovering the motion parameters from an estimated homography, both with and without the assumption of the planar motion model from Chapter 4. The inter-image homography (4.7) is of a very specific form and has only five degrees of freedom, in contrast to a general planar homography which has eight degrees of freedom. If one uses a general homography estimation algorithm to estimate the inter-image homographies, one will almost certainly obtain a homography which is not of the form (4.7).

In this chapter, therefore, we investigate what can be done in order to enforce the planar motion model already at the homography estimation stage. We will thus start with a number of point correspondences $\boldsymbol{x}_j = (x_j, y_j, 1) \leftrightarrow (\hat{x}_j, \hat{y}_j, 1) = \hat{\boldsymbol{x}}_j$, and our goal is to be able to compute a homography matrix $\boldsymbol{H}$ from these correspondences, in such a way that $\boldsymbol{H}$ actually obeys the planar motion model. It is found that the elements of the inter-image homography from Section 4.4,

$$\boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = s\boldsymbol{R}_{\psi\theta}\boldsymbol{R}_z(\varphi)\boldsymbol{T}\boldsymbol{R}_{\psi\theta}^T, \tag{7.1}$$

satisfy a number of polynomial constraints, in addition to the DLT constraints (2.4). We will show how these additional constraints are enforced by a special homography solver introduced by Wadenbäck, Åström and Heyden (2016).

Before discussing this special solver, some additional theory must be covered.

7.1 Multivariate Polynomials

Multivariate polynomials, and systems of multivariate polynomials, are the focus of *algebraic geometry*. We will not develop the theory of this field, which is neatly laid out in the delightful introduction by Cox, Little and O'Shea (2007), but we will nonetheless have to introduce a fair amount of terminology and explain our notation.

Let $\boldsymbol{z} = (z_1, \ldots, z_n)$ be a vector of indeterminates and let $\boldsymbol{\alpha} \in \mathbb{N}^n$. Using *multi-index notation*, i.e. $\boldsymbol{z}^{\boldsymbol{\alpha}} = z_1^{\alpha_1} \cdots z_n^{\alpha_n}$, we define a multivariate *monomial* to be an expression of the form $c_{\boldsymbol{\alpha}} \boldsymbol{z}^{\boldsymbol{\alpha}}$ for some coefficient $c_{\boldsymbol{\alpha}}$, and by the *degree* of the monomial we mean $|\boldsymbol{\alpha}| = \alpha_1 + \ldots + \alpha_n$. A multivariate *polynomial* $f(\boldsymbol{z})$ can now be defined to be a finite sum of multivariate monomials, i.e. an expression of the form

$$f(\boldsymbol{z}) = \sum_{|\boldsymbol{\alpha}| \leq r} c_{\boldsymbol{\alpha}} \boldsymbol{z}^{\boldsymbol{\alpha}}. \tag{7.2}$$

The set of all polynomials in $\boldsymbol{z}$ with complex coefficients is the polynomial ring $\mathbb{C}[\boldsymbol{z}]$. The degree of a polynomial is defined to be the highest degree found among its monomials.

A polynomial $f(\boldsymbol{z})$ can also be conveniently expressed as

$$f(\boldsymbol{z}) = \boldsymbol{c}^T \boldsymbol{Z}(\boldsymbol{z}) \tag{7.3}$$

using a monomial vector

$$\boldsymbol{Z}(\boldsymbol{z}) = \begin{bmatrix} \boldsymbol{z}^{\boldsymbol{\alpha}_1} & \cdots & \boldsymbol{z}^{\boldsymbol{\alpha}_K} \end{bmatrix}^T \tag{7.4}$$

and a coefficient vector $\boldsymbol{c} = (c_{\boldsymbol{\alpha}_1}, \ldots, c_{\boldsymbol{\alpha}_K})$. These vectors are not uniquely determined by the polynomial, since their elements may be permuted freely with no effect. However, if one decides on a *monomial ordering* (Cox, Little and O'Shea, 2007) according to which the elements of $\boldsymbol{Z}(\boldsymbol{z})$ should be sorted, and what monomials should be included (with coefficient zero if necessary), then the vectors are in fact unique. Henceforth, we shall assume that this is done.

A polynomial equation in $\boldsymbol{z}$ can be written as

$$\boldsymbol{c}^T \boldsymbol{Z}(\boldsymbol{z}) = \mathbf{0}, \tag{7.5}$$

and similarly, a system of polynomial equations is written as

$$\boldsymbol{C}\boldsymbol{Z}(\boldsymbol{z}) = \mathbf{0}, \tag{7.6}$$

with a coefficient matrix $\boldsymbol{C}$.

The solution set of a system of polynomial equations, which can be a remarkably complicated geometric object, is called an *affine variety*. Viewing the same system algebraically, it generates a so called *ideal*, $\mathcal{I}$, in the polynomial ring $\mathbb{C}[\boldsymbol{z}]$. As already mentioned, the study of these objects and structures is carried out in the field of algebraic geometry. Specialised computer algebra systems, e.g. Macaulay2 (Grayson, Stillman and Eisenbud, 2015), have emerged to facilitate investigations in this field.

7.1.1 Solving Equations Using the Action Matrix

If a system of polynomial equations of the form (7.6) has a finite number of solutions, say k solutions, then the quotient space $\mathbb{C}[\boldsymbol{z}]/\mathcal{I}$ is isomorphic to $\mathbb{C}^k$ (Cox, Little and O'Shea, 2007). Working in $\mathbb{C}[\boldsymbol{z}]/\mathcal{I}$, this means that the linear map $f(\boldsymbol{z}) \mapsto p(\boldsymbol{z})f(\boldsymbol{z})$, for any fixed element $p(\boldsymbol{z})$, can be expressed as a square matrix of order k, which is called the *action matrix* for the *action polynomial* $p(\boldsymbol{z})$. This is done with respect to a basis of monomials in $\mathbb{C}[\boldsymbol{z}]/\mathcal{I}$, which may be collected in a monomial vector $\mathcal{B}$.

If one succeeds in computing the action matrix for a polynomial $p(\boldsymbol{z})$, then – remarkably – the eigenvectors to the action matrix will be precisely the monomial vector $\mathcal{B}$ evaluated at the k solutions of the equation system (Byröd, 2010). This can be thought of as a generalisation to the multivariate case of the *companion matrix* method (Trefethen and Bau, 1997) for solving univariate polynomial equations.

Practical methods for computing action matrices are given a thorough treatment by Byröd (2010) and Kúkelová (2013). These methods are non-trivial endeavours which require particular care to preserve numerical stability, and we can only hope to convey the key ideas and must gloss over many of the intricacies.

Let us consider a system of equations of the form (7.6) that we wish to solve. The first step to take is to choose the action polynomial $p(\boldsymbol{z})$ that will be used (this is usually taken to be just one of the variables, e.g. $p(\boldsymbol{z}) = z_1$).

Next, one determines the so called *permissible set*, $\mathcal{P}$, which is the set of all monomials in $Z(\boldsymbol{z})$ whose image after multiplication with $p(\boldsymbol{z})$ is a constant factor of some element in $Z(\boldsymbol{z})$. It is among the polynomials in the permissible set that we will later select a basis $\mathcal{B}$. The *reducible set* $\mathcal{R}$ is the set of monomials which are not in $\mathcal{P}$, but whose image after the multiplication is a polynomial multiple of some element in $\mathcal{P}$. Finally, the *excessive set* $\mathcal{E}$ contains all the monomials in $Z(\boldsymbol{z})$ which are neither in $\mathcal{P}$ nor $\mathcal{R}$.

By rearranging the system of equations, it can now be written as

$$\begin{bmatrix} C_{\mathcal{E}} & C_{\mathcal{R}} & C_{\mathcal{P}} \end{bmatrix} \begin{bmatrix} Z_{\mathcal{E}} \\ Z_{\mathcal{R}} \\ Z_{\mathcal{P}} \end{bmatrix} = \mathbf{0}, \tag{7.7}$$

where $Z_{\mathcal{E}}$ are the excessive monomials, $Z_{\mathcal{R}}$ are the reducible monomials, and $Z_{\mathcal{P}}$ are the permissible ones. By performing some form of Gaußian elimination to eliminate the excessive and reducible variables the system can be brought to the form

$$\begin{bmatrix} U_{\mathcal{E}}^{(1)} & C_{\mathcal{R}}^{(1)} & C_{\mathcal{P}}^{(1)} \\ & U_{\mathcal{R}}^{(2)} & C_{\mathcal{P}}^{(2)} \\ & & C_{\mathcal{P}}^{(3)} \end{bmatrix} \begin{bmatrix} Z_{\mathcal{E}} \\ Z_{\mathcal{R}} \\ Z_{\mathcal{P}} \end{bmatrix} = \mathbf{0}, \tag{7.8}$$

where $U_{\mathcal{E}}^{(1)}$ and $U_{\mathcal{R}}^{(2)}$ are upper triangular. Since we are not interested in the excessive variables, we continue by considering only the lower right block

$$\begin{bmatrix} U_{\mathcal{R}}^{(2)} & C_{\mathcal{P}}^{(2)} \\ & C_{\mathcal{P}}^{(3)} \end{bmatrix} \begin{bmatrix} Z_{\mathcal{R}} \\ Z_{\mathcal{P}} \end{bmatrix} = \mathbf{0}. \tag{7.9}$$

With some luck, it should now be possible to find k linearly independent columns in $C_{\mathcal{P}}^{(3)}$ which will correspond to the basis $\mathcal{B}$.

Another rearrangement of the system, followed by another Gaußian elimination, gives the equivalent system

$$\begin{bmatrix} U_{\mathcal{R}}^{(2)} & C_{\mathcal{P}\setminus\mathcal{B}}^{(2)} & C_{\mathcal{B}}^{(2)} \\ & U_{\mathcal{P}\setminus\mathcal{B}}^{(3)} & C_{\mathcal{B}}^{(3)} \end{bmatrix} \begin{bmatrix} Z_{\mathcal{R}} \\ Z_{\mathcal{P}\setminus\mathcal{B}} \\ Z_{\mathcal{B}} \end{bmatrix} = \mathbf{0}. \tag{7.10}$$

If it is possible at this point to perform a successful back-substitution, which expresses $Z_{\mathcal{R}}$ and $Z_{\mathcal{P}\setminus\mathcal{B}}$ in terms of $Z_{\mathcal{B}}$, then it is possible to form the action matrix by multiplying the elements in $\mathcal{B}$ with $p(\boldsymbol{z})$. The result of each such multiplication will by definition be among the reducible or the permissible monomials, and thanks to the back-substitution we have a way of expressing it as a linear combination of elements in $\mathcal{B}$.

One key result in Byröd (2010) is that it is sufficient that the monomial vector $\mathcal{B}$ is a *spanning set* rather than an actual basis. This means that it is not necessary to have exactly k rows in $C_{\mathcal{P}_3}$. By being able to choose $\ell > k$ monomials in $\mathcal{B}$, one trades the inconvenience of adding false solutions for some additional freedom in how to compute an $\ell \times \ell$ action matrix. This additional freedom can be used wisely to improve the numerical aspects of the method (Byröd, 2010; Byröd, Josephson and Åström, 2009).

7.2 Finding Constraints on H

After this theoretical exposé, let us return to the problem of finding polynomial constraints on the elements of the inter-image homography (7.1). We know from Section 2.1.3 that the homography $\boldsymbol{H}$ must obey (redundant) DLT constraints of the form

$$\begin{bmatrix} \mathbf{0} & -\boldsymbol{x}_j^T & \hat{y}_j \boldsymbol{x}_j^T \\ \boldsymbol{x}_j^T & \mathbf{0} & -\hat{x}_j \boldsymbol{x}_j^T \\ -\hat{y}_j \boldsymbol{x}_j^T & \hat{x}_j \boldsymbol{x}_j^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \boldsymbol{h}_3 \end{bmatrix} = \mathbf{0}, \tag{7.11}$$

where $\boldsymbol{h}_1$, $\boldsymbol{h}_2$, and $\boldsymbol{h}_3$ are the rows of $\boldsymbol{H}$, and where we have used the fact that $z_j = \hat{z}_j = 1$. Since the first two rows in (7.11) are linearly independent,

and since the whole system has rank two, we will in the following work with the equivalent (reduced) DLT constraint

$$\begin{bmatrix} \mathbf{0} & -\boldsymbol{x}_j^T & \hat{y}_j \boldsymbol{x}_j^T \\ \boldsymbol{x}_j^T & \mathbf{0} & -\hat{x}_j \boldsymbol{x}_j^T \end{bmatrix} \begin{bmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \boldsymbol{h}_3 \end{bmatrix} = \mathbf{0}. \tag{7.12}$$

We will now investigate polynomial constraints which the homography matrix entries must satisfy in addition to the linear constraints (7.12).

7.2.1 Analytical Derivation of Constraints

If we introduce intermediate parameters

$$(c_\psi, s_\psi, c_\theta, s_\theta, c_\varphi, s_\varphi) = (\cos\psi, \sin\psi, \cos\theta, \sin\theta, \cos\varphi, \sin\varphi), \tag{7.13}$$

then the affine variety defined by the polynomial equations

$$\begin{cases} c_\psi^2 + s_\psi^2 - 1 = 0, \\ c_\theta^2 + s_\theta^2 - 1 = 0, \\ c_\varphi^2 + s_\varphi^2 - 1 = 0, \end{cases} \tag{7.14}$$

is the parameter space. Multiplying the matrices on the right hand side in (7.1) together, it is clear that each h_{ij} will be a polynomial in s, t_x and t_y, and the sines and cosines of the three angles ψ, θ, and φ. This means that the set of possible homography matrices of the form (7.1) is the image under a *polynomial map* of an affine variety (Cox, Little and O'Shea, 2007). The Tarski-Seidenberg theorem guarantees that this image will be a so called *semi-algebraic set*, i.e. a set defined by a finite number of polynomial equations and polynomial inequalities (van den Dries, 1986). We will try to find the polynomial equations, and ignore the the potential polynomial inequalities.

There are a number of approaches that can be used to try to find polynomial constraints on the homography matrix. If an affine variety is given through a parametrisation, the problem of finding equations which define the same variety is a case of the *implicitisation problem*. One method to

try to solve the implicitisation problem for the set of planar motion homographies would be to use Gröbner basis computations to eliminate the generating parameters (Cox, Little and O'Shea, 2007), which could result in a number of equations containing only the h_{ij}.

While we have unsuccessfully attempted this route using Maple (Waterloo Maple Inc., 2016), it cannot be ruled out that this approach potentially could provide insights if done with enough skill, care, and patience. There is also an elimination technique based on the so called *resultant*, but this method is impractical because it inflates the degree of the polynomials excessively, and we have therefore not tried it on the present problem.

7.2.2 Numerical Derivation of Constraints

Another approach is to use the fact that we easily can generate random instances of inter-image homographies (7.1).

If there exist polynomial constraints on the elements of $\boldsymbol{H}$, then they should be of the form (7.6). Each instance $\boldsymbol{H}_j$ we generate gives a number of linear equations which the coefficient matrix must satisfy. By randomly generating sufficiently many, say N, inter-image homographies, we can find the coefficient matrix by computing the left null space of the matrix

$$\begin{bmatrix} \boldsymbol{Z}(\boldsymbol{H}_1) & \cdots & \boldsymbol{Z}(\boldsymbol{H}_N) \end{bmatrix}. \tag{7.15}$$

In order to use this idea, we need to decide what monomials we want to include in $\boldsymbol{Z}(\boldsymbol{H})$. We observe that since the homography matrix is only determined up to the scalar multiple s, any polynomial constraint on the matrix entries must be *homogeneous*, i.e. contain only monomials of the same degree. This leads us to consider $\boldsymbol{Z}(\boldsymbol{H})$ consisting of all monomial of a given degree r.

For $r \leq 3$ we did not obtain any non-trivial left null space of the matrix (7.15), whereas for $r = 4$ we obtained eleven linearly independent coefficient vectors, which were assembled to form a coefficient matrix $\boldsymbol{C}$. After bringing this coefficient matrix to *reduced row echelon form*, the coefficients were very close to being integer. Rounding them to integers, we

obtained eleven polynomial constraints

$$\begin{cases} g_1(\boldsymbol{H}) = 0, \\ \quad \vdots \\ g_{11}(\boldsymbol{H}) = 0, \end{cases} \tag{7.16}$$

with integer coefficients. These constraints were successfully confirmed symbolically using the parametrisation (7.1). To give the reader a flavour of what the discovered constraints look like, we write one of them out explicitly:

$$\begin{aligned} g_1(\boldsymbol{H}) = {} & h_{11}^2 h_{12} h_{13} + h_{11}^2 h_{22} h_{32} + h_{11}^2 h_{23} h_{33} - h_{11} h_{21} h_{12} h_{32} \\ & - h_{11} h_{21} h_{13} h_{33} - h_{11} h_{31} h_{12} h_{22} - h_{11} h_{31} h_{13} h_{23} + h_{11} h_{12}^2 h_{23} \\ & - h_{11} h_{22}^2 h_{23} - h_{11} h_{22} h_{32} h_{33} - h_{11} h_{22} h_{23} h_{33} + h_{11} h_{32} h_{13}^2 \\ & - h_{11} h_{32} h_{33}^2 + h_{21}^2 h_{12} h_{13} + h_{21} h_{31} h_{12}^2 + h_{21} h_{31} h_{13}^2 \\ & + h_{21} h_{12}^2 h_{13} + 2 h_{21} h_{12} h_{22} h_{23} + h_{21} h_{12} h_{32} h_{33} \\ & + h_{21} h_{12} h_{23} h_{33} + h_{21} h_{22}^2 h_{13} + h_{21} h_{32} h_{13} h_{23} + h_{31}^2 h_{12} h_{13} \\ & + h_{31} h_{12} h_{32} h_{23} + h_{31} h_{12} h_{13}^2 + h_{31} h_{12} h_{33}^2 + h_{31} h_{22} h_{32} h_{13} \\ & + h_{31} h_{22} h_{13} h_{23} + 2 h_{31} h_{32} h_{13} h_{33} + h_{12}^2 h_{22} h_{23} + h_{12}^2 h_{23} h_{33} \\ & - h_{12} h_{22} h_{13} h_{33} + h_{12} h_{32} h_{13} h_{23} + h_{22}^3 h_{23} + h_{22} h_{32}^2 h_{23} \\ & + h_{22} h_{32} h_{13}^2 + 2 h_{22} h_{32} h_{23}^2 + 2 h_{32}^2 h_{23} h_{33} + h_{32} h_{13}^2 h_{33} \\ & + h_{32} h_{23}^2 h_{33} + h_{32} h_{33}^3. \end{aligned} \tag{7.17}$$

7.3 The Homography Solver

We will now apply the technique of combining linear and non-linear constraints mentioned in Section 3.1.2. Suppose for that purpose that we have three point correspondences. This allows us to form a DLT system (2.5) consisting of three DLT constraints (7.12), and by removing e.g. the last row we have 2.5 DLT constraints left in the DLT system. This DLT system should have a four-dimensional null space, which allows us to parametrise

the homography as

$$\boldsymbol{H}(\boldsymbol{z}) = \boldsymbol{H}_0 + z_1\boldsymbol{H}_1 + z_2\boldsymbol{H}_2 + z_3\boldsymbol{H}_3, \tag{7.18}$$

where the $\boldsymbol{H}_k$ are basis vectors for the null space, reshaped as 3×3 matrices.

Inserting this parametrisation $\boldsymbol{H}(\boldsymbol{z})$ into the eleven polynomial constraints, i.e. forming $f_j(\boldsymbol{z}) = g_j(\boldsymbol{H}(\boldsymbol{z}))$, gives eleven new polynomial equations

$$\begin{cases} f_1(\boldsymbol{z}) = 0, \\ \quad\vdots \\ f_{11}(\boldsymbol{z}) = 0, \end{cases} \tag{7.19}$$

of degree four in the three unknowns $\boldsymbol{z} = (z_1, z_2, z_3)$. All of these involve all $35 = \binom{7}{4}$ possible monomials up to degree four.

From results obtained in the symbolic investigations of this system of equations, which we have postponed to Section 7.4.2, it turns out that before we can solve the system using the method from Section 7.1.1, we need to have more linearly independent equations in the system. This can be achieved by augmenting the system with the additional equations

$$\begin{cases} z_1 f_j(\boldsymbol{z}) = 0, \\ z_2 f_j(\boldsymbol{z}) = 0, \\ z_3 f_j(\boldsymbol{z}) = 0, \end{cases} \tag{7.20}$$

for $1 \leq j \leq 11$, resulting in 44 equations in 56 monomials. This expanded system is used for constructing the action matrix.

As we shall see in Section 7.4.2 there are fourteen solutions in general, which means that we need to select at least that many monomials as a basis (or spanning set) for the action matrix method. We obtained best results using a basis of 25 monomials, using column pivoting to improve numerical stability (Byröd, Josephson and Åström, 2008). This resulted in an efficient and numerically stable reduction of the problem to an eigenvalue problem of size 25×25. Using eigenvalue decomposition we obtain 25 putative solutions, and those that are real and fulfil the original equations are considered to be valid. The resulting homography solver is the 2.5-point solver introduced in Wadenbäck, Åström and Heyden (2016).

7.4 Investigation of the Constraints

We now turn to the investigation of the constraints (7.16) and (7.19). The investigations in Section 7.4.2 were reported in Wadenbäck, Åström and Heyden (2016), but the results in Section 7.4.1 had to be omitted due to space limitations.

7.4.1 Investigation of the Tangent Space

The constraints (7.16) on the elements of $\boldsymbol{H}$ are clearly *necessary* for a homography of the form (7.1), but we have not shown that they are *sufficient*. While the investigations in this section will not amount to proving that this is the case, we will argue that they are *for all practical purposes* sufficient.

We begin by noting that if $\boldsymbol{\eta} = (\psi, \theta, \varphi, t_x, t_y)$ and

$$\boldsymbol{H}(\boldsymbol{\eta}, s) = s\boldsymbol{R}_{\psi\theta}\boldsymbol{R}_z(\varphi) \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{R}_{\psi\theta}^T, \tag{7.21}$$

then $(\boldsymbol{\eta}, s) \mapsto \operatorname{vec}(\boldsymbol{H}(\boldsymbol{\eta}, s))$ is a parametrisation of an object in $\mathbb{R}^9$ whose dimension (at most points) cannot be greater than six.

The columns of the Jacobian of this parametrisation, evaluated at some point $(\boldsymbol{\eta}_0, s_0)$, will span the *tangent space* at $(\boldsymbol{\eta}_0, s_0)$. This means that the rank of the Jacobian is the dimension of the parametrised object at the point of evaluation. The results of evaluating this Jacobian for 100 000 random parameter values and computing the corresponding tangent space dimension is shown in Figure 7.1 (the dimension was in every single case found the be equal to 6).

Similarly, we know that the gradients ∇g_j of the constraints (7.16), evaluated at $(\boldsymbol{\eta}_0, s_0)$, should lie in the *normal space*. If they span the whole of the normal space, this means that close to $(\boldsymbol{\eta}_0, s_0)$ the constraints $g_j(\boldsymbol{H})$ are in fact sufficient to describe the parametrised object. We have computed the dimension of the linear hull of the gradients at the same 100 000 random parameter values as above, and the result can be seen in Figure 7.1. In almost all cases, the gradients spanned the entire three-dimensional normal space, and in no single instance was the dimension lower than three.

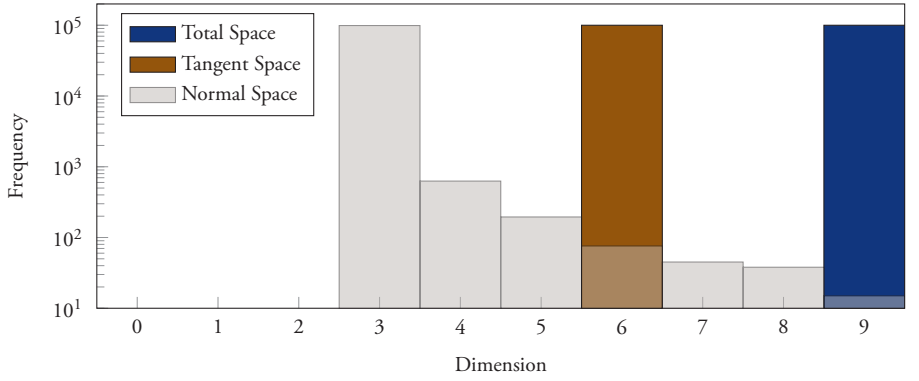


Figure 7.1: Histogram over the numerically computed dimensions of the normal space, tangent space, and the total space for 100 000 random parameter values. The dimension of the tangent space was computed as the rank of the Jacobian. The dimension of the normal space was computed as the rank of a matrix with the gradients as columns.

Since it is not theoretically possible that the sum of the dimensions of the normal space and the tangent space are more than nine in this case, the results in Figure 7.1 might be surprising, as they seem to indicate otherwise. The most likely explanation of this apparent discrepancy is that numerical effects in a few cases have resulted in a too large efficient rank (recall Section 1.1.1). This suspicion has in fact been confirmed for a small number of the cases where the dimension of the normal space was reported to be greater than three, by seeing that three of the singular values were much larger than the other ones.

7.4.2 Symbolic Investigation Using Macaulay2

As briefly mentioned at the very beginning of Section 7.1.1, the action matrix method assumes that there are finitely many solutions to the problem. There are several methods that can be used to determine if a system of polynomial equations has a finite number of solutions, and also in some cases to determine the number of solutions. One method is based on the so called mixed volume (Bernshtein, 1975). The mixed volume gives you the num-

ber of solutions for r equations in r unknowns under the assumption that all coefficients of the polynomial equations are *generic*. This is often not the case for geometric problems in computer vision, where there typically are many connections and dependencies among the coefficients.

Another method is to study the dimensionality and degree of the ideal $\mathcal{I}$. This can often be efficiently calculated assuming coefficients in $\mathbb{Z}_p$ using algebraic geometry tools such as Macaulay2 (Grayson, Stillman and Eisenbud, 2015) or Maple (Waterloo Maple Inc., 2016). This, however, requires that one generates one or several instances of the problem so that the system has integer coefficients. This makes it possible to capture dependencies between the coefficients and usually gives the relevant multiplicity of solutions. If the coefficients of the polynomials are given as inexact floating point numbers, however, cancellations might not be detected or might be erroneously detected in the elimination and an incorrect dimensionality might be reported. To investigate the constraints (7.19) symbolically, we thus need to be able to generate homographies of the form (7.1) with an exact representation, e.g. using integers.

Integer versions of the problem can be created using random rotations involving Pythagorean triples, e.g. so that $(\sin\varphi, \cos\varphi) = (3/5, 4/5)$. By multiplying the homography matrix by the integer denominators involved, one obtains an integer homography matrix of the correct type. Then the points for the first image are chosen as random integer points in homogeneous coordinates $\boldsymbol{x}$. The homogeneous coordinates of the points in the second camera are then $\hat{\boldsymbol{x}} = \boldsymbol{H}\boldsymbol{x}$, which then also become integer. The DLT-constraints (7.12) will all be integer, so it is possible to compute an integer basis for the null space, and these basis vectors can be reshaped to 3×3 integer matrices $\boldsymbol{H}_0$, $\boldsymbol{H}_1$, $\boldsymbol{H}_2$, $\boldsymbol{H}_3$. Finally since the constraints on $\boldsymbol{H}$ have integer coefficients the eleven fourth order constraints are integer for these specifically generated problems.

The constraints (7.19) were studied using Macaulay2 (Grayson, Stillman and Eisenbud, 2015) in Wadenbäck, Åström and Heyden (2016) for such integer instances of the problem. The resulting ideal has dimension zero (which means that the solution set consists of isolated points) and degree fourteen. Thus there are in general fourteen solutions to the problem.

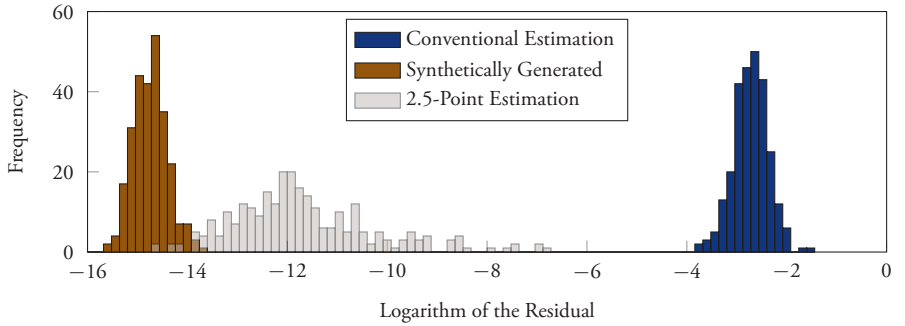


Figure 7.2: Logarithmic histogram over the residual vector $\|\boldsymbol{g}(\boldsymbol{H})\|$ for homographies obtained through conventional estimation, synthetically generated homographies, and for homographies estimated using the 2.5-point method for a real planar motion dataset. Homographies estimated using the 2.5-point solver satisfy the polynomial constraints found in Section 7.2.2 better than do the conventionally estimated homographies.

7.5 Further Numerical Experiments

From the method we used in Section 7.2.2 to derive the constraints (7.16), we already know that synthetically generated homographies should satisfy the constraints. It was claimed in the beginning of the chapter that homographies from a real dataset, if they are estimated using conventional homography estimation methods, will not be of the correct type. This claim can be substantiated by evaluating $\|\boldsymbol{g}(\boldsymbol{H})\|$, where $\boldsymbol{g} = (g_1, \ldots, g_{11})$, for conventionally estimated homographies from a real dataset and comparing it to the same norm for both homographies estimated using the 2.5-point solver in Section 7.3 and synthetically generated homographies. Figure 7.2 shows that on a real dataset the conventionally generated homographies do not satisfy the constraints, and thus do not obey the planar motion model. As seen in the same figure, the homographies obtained using the polynomial 2.5-point method have significantly smaller residuals. Although not demonstrated here, the homographies obtained using the 2.5-point method are in fact so close to satisfying the constraints that their residuals typically can be brought down to the same magnitude as the synthetically generated homo-

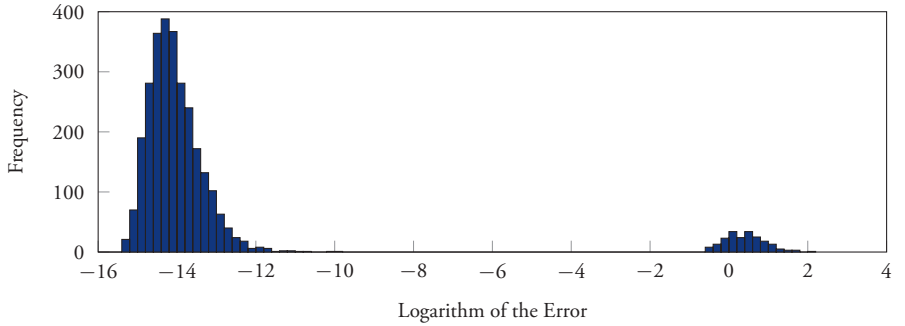


Figure 7.3: Logarithmic histogram over $\left\|\boldsymbol{H} - \tilde{\boldsymbol{H}}\right\|_F$, where $\boldsymbol{H}$ is the true homography and $\tilde{\boldsymbol{H}}$ the estimated one. The homography is recovered accurately except in a small number of failure cases.

graphies in only one or two iterations with the Newton-Raphson method (Bishop, 2006).

7.5.1 Numerical Accuracy

In order to evaluate the numerical accuracy of the 2.5-point homography solver we described in Section 7.3, we have generated 3000 random homographies of the form (7.1), along with three point correspondences for each one of these. The polynomial 2.5-point solver was then used on each triplet of correspondences for the purpose of estimating their corresponding homography. Figure 7.3 shows a histogram over the logarithm of the Frobenius norm of the errors $\left\|\boldsymbol{H} - \tilde{\boldsymbol{H}}\right\|_F$, where $\boldsymbol{H}$ is the true homography and $\tilde{\boldsymbol{H}}$ is the best estimated one. Here, all homographies are scaled to have their determinant equal to one, to eliminate the influence of the scalar factor ambiguity. From this histogram it is clear that the solver succeeds in finding the correct solution within a small numerical tolerance in most cases, but in a small percentage of the cases the correct solution is not found. This proportion of failure cases does not pose a problem for the usability of the solver, as it will generally be used in a RANSAC loop, and then hopefully a correct homography will be estimated in another iteration.

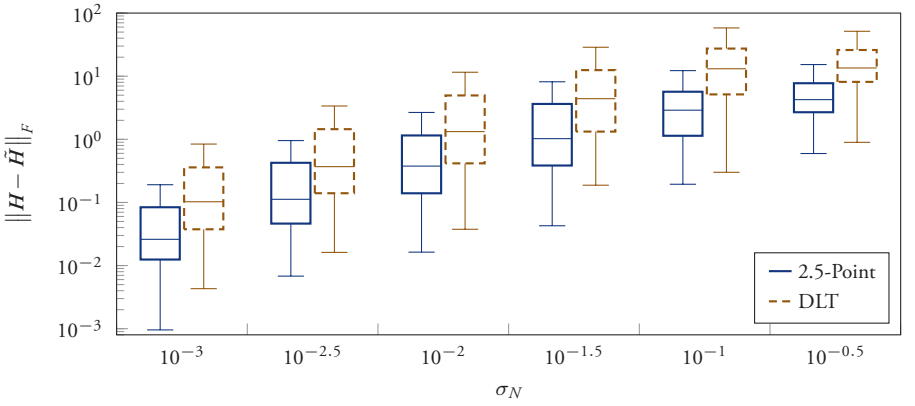


Figure 7.4: The error $\left\|\boldsymbol{H} - \tilde{\boldsymbol{H}}\right\|_F$ for different noise levels σ_N for both the polynomial 2.5-point solver and the DLT. We find that the 2.5-point method performs better than the DLT for all σ_N in the experiment.

7.5.2 Noise Sensitivity

For the purpose of investigating the noise sensitivity of the solver, random homographies $\boldsymbol{H}$ and point correspondences $\boldsymbol{x} \leftrightarrow \hat{\boldsymbol{x}}$ were generated as in Section 7.5.1. The coordinates for $\boldsymbol{x}$ were drawn from a Gaußian distribution with zero mean and unit variance, and $\hat{\boldsymbol{x}}$ was computed as $\hat{\boldsymbol{x}} = \boldsymbol{H}\boldsymbol{x}$ and normalised so that the coordinates again had close to unit variance. After this, the coordinates in $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ were contaminated by Gaußian noise with standard deviation σ_N. Next, a homography $\tilde{\boldsymbol{H}}$ was estimated from these contaminated points using both the polynomial 2.5-point solver and the DLT (using four correspondences), and normalised to have determinant equal to one. Figure 7.4 shows $\left\|\boldsymbol{H} - \tilde{\boldsymbol{H}}\right\|_F$ for different noise levels σ_N, and we observe that the homographies are more accurately recovered using the 2.5-point solver than using the DLT.

The point correspondences which were not used to estimate the homography were mapped through the estimated homography, and the reprojection errors were studied. The distribution of the mean errors is shown in Figure 7.5. In this case, the 2.5-point method and the DLT have comparable reprojection errors, and only for low noise levels does the 2.5-point

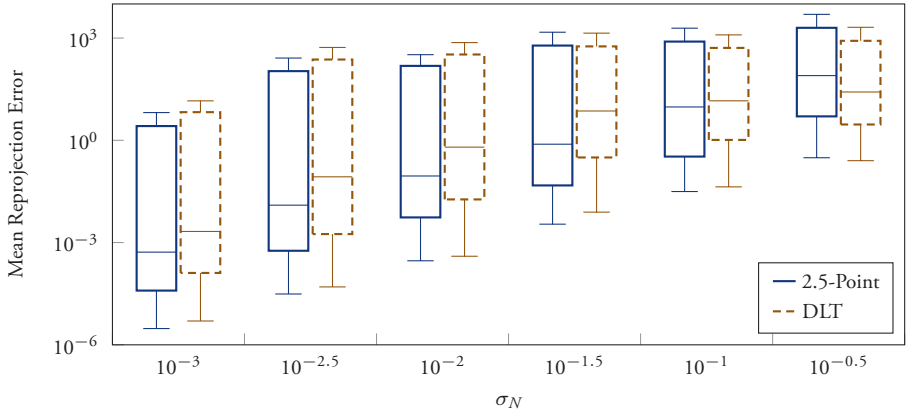


Figure 7.5: Distribution of the mean reprojection error for different noise levels σ_N for both the proposed 2.5-point solver and the DLT, where the distance is taken in the Euclidean representation of the points. We observe that the methods perform similarly, with a slight advantage for the 2.5-point method for the lower noise levels.

solver give slightly smaller ones. This is somewhat surprising, since the DLT determines more parameters than the underlying data actually have, and one might have expected a more pronounced effect of the over-fitting.

Chapter 8

Minimising the Reprojection Error

In Chapter 7 we explained in detail the polynomial 2.5-point homography solver that was introduced in Wadenbäck, Åström and Heyden (2016). This homography solver was able to use 2.5 point correspondences to compute a homography that was guaranteed to be compatible with the planar motion model from Chapter 4.

Solvers such as the 2.5-point solver or the DLT, which use as few correspondences as possible to estimate their corresponding model, are called *minimal solvers*. These are frequently used together with the RANSAC framework from Section 1.2.2, since the probability of selecting only true inliers is greater for a smaller sample size. After the largest consensus set has been found, one should fit the model to the entire consensus set, and since the 2.5-point method only works for 2.5 correspondences we need to take a different approach for this final step. Here we describe how to minimise the reprojection errors for the largest consensus set.

8.1 Geometric Error

At the end of Section 2.1.3, we mentioned that the least-squares solution obtained from an overdetermined DLT system did not correspond to minimising a geometrically sensible error function. A more reasonable cost function to minimise would be the *geometric reprojection error*,

$$E(\boldsymbol{H}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_N) = \sum_{j=1}^{N} G(\boldsymbol{x}_j^*, \boldsymbol{x}_j) + G(\hat{\boldsymbol{x}}_j^*, \hat{\boldsymbol{x}}_j), \tag{8.1}$$

subject to the constraint that $\boldsymbol{H}$ maps $\boldsymbol{x}_j$ *exactly* to $\hat{\boldsymbol{x}}_j$, and where

$$G(\boldsymbol{y}, \boldsymbol{z}) = \left\| \frac{\boldsymbol{y}}{\boldsymbol{e}_3^T \boldsymbol{y}} - \frac{\boldsymbol{z}}{\boldsymbol{e}_3^T \boldsymbol{z}} \right\|^2, \quad \boldsymbol{e}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T. \tag{8.2}$$

Here $\boldsymbol{x}_j^* \leftrightarrow \hat{\boldsymbol{x}}_j^*$ are the provided correspondences, which we may not be able to satisfy all at once, and $\boldsymbol{x}_j \leftrightarrow \hat{\boldsymbol{x}}_j$ are *geometrically corrected* versions of the correspondences. The geometric reprojection error measures the squared pixel distance between each point and its geometrically corrected counterpart, and is meaningful when the cameras are calibrated (i.e. $\boldsymbol{K}$ is known and the lens distortion model has been compensated for).

Minimising the geometric error (8.1) is a difficult constrained optimisation problem which has in total $4N + 8$ degrees of freedom, due to the eight degrees of freedom inherent to the homography matrix and the $4N$ independent parameters describing the points. We will not go into the details of how this optimisation problem is solved for a general planar homography. Further details for that problem are found e.g. in Kanatani (1998) and in Hartley and Zisserman (2004), and ready-to-run implementations are available e.g. in OpenCV (Itseez, Inc., 2017).

Algorithms in computer vision which minimise some kind of reprojection error are often referred to as *bundle adjuštment* (Triggs et al., 1999).

8.2 Planar Motion Bundle Adjustment

We are going to consider the minimisation of the geometric error (8.1) for a homography of the form (4.7). As already mentioned, this is a constrained optimisation problem, and we have not discussed how such problems are solved. Fortunately, we can exploit the camera parametrisation from Section 4.3 to reformulate the optimisation problem as an unconstrained optimisation problem, which we can solve using the Levenberg-Marquardt method from Section 1.2.1.

Suppose, then, that we have the two cameras

$$\begin{cases} \boldsymbol{P} = \boldsymbol{R}_{\psi\theta} \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}, \\ \hat{\boldsymbol{P}} = \boldsymbol{R}_{\psi\theta} \boldsymbol{R}_z(\varphi) \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{t} \end{bmatrix}, \end{cases} \tag{8.3}$$

according to the planar motion model, and that we have a number of scene points X_j which lie in the floor plane. Since only the first two coordinates in X_j can vary, it will be convenient to let $\boldsymbol{\xi}_j$ be these first two coordinates. Now let $\boldsymbol{p}_k^T$ and $\hat{\boldsymbol{p}}_k^T$ be row k in $\boldsymbol{P}$ and $\hat{\boldsymbol{P}}$, respectively. Let $\boldsymbol{\pi}_k$ be the first two coordinates of $\boldsymbol{p}_k$, and define $\hat{\boldsymbol{\pi}}_k$ correspondingly. We also introduce $\boldsymbol{\Psi} = \text{vec}(\boldsymbol{P}^T)$ and $\hat{\boldsymbol{\Psi}} = \text{vec}(\hat{\boldsymbol{P}}^T)$. Define $\boldsymbol{\Xi} = (\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_N)$ and let $\boldsymbol{\beta} = (\boldsymbol{\eta}, \boldsymbol{\Xi})$, where as usual $\boldsymbol{\eta} = (\psi, \theta, \varphi, t_x, t_y)$ is the motion parameter vector.

After defining all this notation, we will consider the reprojection error for a single point in each of the cameras. This can be written using the function G from (8.2) as $G(\boldsymbol{x}_j^*, \boldsymbol{P}X_j) = \|\boldsymbol{r}_j(\boldsymbol{\beta})\|^2$ in camera $\boldsymbol{P}$ and $G(\hat{\boldsymbol{x}}_j^*, \hat{\boldsymbol{P}}X_j) = \|\hat{\boldsymbol{r}}_j(\boldsymbol{\beta})\|^2$ in camera $\hat{\boldsymbol{P}}$, where

$$\begin{cases} \boldsymbol{r}_j(\boldsymbol{\beta}) = \begin{bmatrix} x_j^* - x_j \\ y_j^* - y_j \end{bmatrix} = \begin{bmatrix} x_j^* \\ y_j^* \end{bmatrix} - \dfrac{1}{X_j^T \boldsymbol{p}_3} \begin{bmatrix} X_j^T \boldsymbol{p}_1 \\ X_j^T \boldsymbol{p}_2 \end{bmatrix}, \\ \hat{\boldsymbol{r}}_j(\boldsymbol{\beta}) = \begin{bmatrix} \hat{x}_j^* - \hat{x}_j \\ \hat{y}_j^* - \hat{y}_j \end{bmatrix} = \begin{bmatrix} \hat{x}_j^* \\ \hat{y}_j^* \end{bmatrix} - \dfrac{1}{X_j^T \hat{\boldsymbol{p}}_3} \begin{bmatrix} X_j^T \hat{\boldsymbol{p}}_1 \\ X_j^T \hat{\boldsymbol{p}}_2 \end{bmatrix}. \end{cases} \tag{8.4}$$

The overall cost function that will be minimised is

$$E(\boldsymbol{\beta}) = \sum_{j=1}^{N} \|\boldsymbol{r}_j(\boldsymbol{\beta})\|^2 + \|\hat{\boldsymbol{r}}_j(\boldsymbol{\beta})\|^2, \tag{8.5}$$

and with the residuals defined in (8.4), the Jacobian will have the overall structure

$$\boldsymbol{J}(\boldsymbol{\beta}) = \begin{bmatrix} \frac{\partial \boldsymbol{r}_1}{\partial \boldsymbol{\eta}} & \frac{\partial \boldsymbol{r}_1}{\partial \boldsymbol{\xi}_1} & & \\ \vdots & & \ddots & \\ \frac{\partial \boldsymbol{r}_N}{\partial \boldsymbol{\eta}} & & & \frac{\partial \boldsymbol{r}_N}{\partial \boldsymbol{\xi}_N} \\ \frac{\partial \hat{\boldsymbol{r}}_1}{\partial \boldsymbol{\eta}} & \frac{\partial \hat{\boldsymbol{r}}_1}{\partial \boldsymbol{\xi}_1} & & \\ \vdots & & \ddots & \\ \frac{\partial \hat{\boldsymbol{r}}_N}{\partial \boldsymbol{\eta}} & & & \frac{\partial \hat{\boldsymbol{r}}_N}{\partial \boldsymbol{\xi}_N} \end{bmatrix}. \tag{8.6}$$

If we are able to compute all the non-zero blocks in this Jacobian, then we are ready to implement the minimisation using Algorithm 1.1.

8.2.1 Computing the Derivatives

We first compute the two derivative blocks $\frac{\partial r_j}{\partial \eta}$ and $\frac{\partial \hat{r}_j}{\partial \eta}$. These are easiest to compute via the chain rule, i.e.

$$\begin{cases} \dfrac{\partial r_j}{\partial \eta} = \dfrac{\partial r_j}{\partial \Psi} \dfrac{\partial \Psi}{\partial \eta}, \\ \dfrac{\partial \hat{r}_j}{\partial \eta} = \dfrac{\partial \hat{r}_j}{\partial \hat{\Psi}} \dfrac{\partial \hat{\Psi}}{\partial \eta}. \end{cases} \tag{8.7}$$

The derivatives $\frac{\partial \Psi}{\partial \eta}$ and $\frac{\partial \hat{\Psi}}{\partial \eta}$ are complicated, but thankfully they can be computed symbolically using a computer algebra system such as Maple (Waterloo Maple Inc., 2016).

Through careful and patient application of the differentiation results in Section 1.1.3, one finds that

$$\begin{cases} \dfrac{\partial r_j}{\partial \Psi} = \begin{bmatrix} -X_j^T p_3 & 0 & X_j^T p_1 \\ 0 & -X_j^T p_3 & X_j^T p_2 \end{bmatrix} \otimes \dfrac{X_j^T}{(X_j^T p_3)^2}, \\ \dfrac{\partial \hat{r}_j}{\partial \hat{\Psi}} = \begin{bmatrix} -X_j^T \hat{p}_3 & 0 & X_j^T \hat{p}_1 \\ 0 & -X_j^T \hat{p}_3 & X_j^T \hat{p}_2 \end{bmatrix} \otimes \dfrac{X_j^T}{(X_j^T \hat{p}_3)^2}. \end{cases} \tag{8.8}$$

Similar careful differentiation as in the previous step shows that

$$\begin{cases} \dfrac{\partial r_j}{\partial \xi_j} = \dfrac{1}{(p_3^T X_j)^2} \begin{bmatrix} p_1^T X_j \pi_3^T - p_3^T X_j \pi_1^T \\ p_2^T X_j \pi_3^T - p_3^T X_j \pi_2^T \end{bmatrix}, \\ \dfrac{\partial \hat{r}_j}{\partial \xi_j} = \dfrac{1}{(\hat{p}_3^T X_j)^2} \begin{bmatrix} \hat{p}_1^T X_j \hat{\pi}_3^T - \hat{p}_3^T X_j \hat{\pi}_1^T \\ \hat{p}_2^T X_j \hat{\pi}_3^T - \hat{p}_3^T X_j \hat{\pi}_2^T \end{bmatrix}. \end{cases} \tag{8.9}$$

We now have expressions for all the non-zero blocks in the Jacobian (8.6).

As a consequence of the structure (8.6) of the Jacobian, it can be verified that $J(\beta)^T J(\beta)$ has the 'arrow' structure

$$J(\beta)^T J(\beta) = \begin{bmatrix} \times & \times & \cdots & \times \\ \times & \times & & \\ \vdots & & \ddots & \\ \times & & & \times \end{bmatrix}, \tag{8.10}$$

and this structure is clearly preserved when adding a diagonal matrix to $\boldsymbol{J}(\boldsymbol{\beta})^T\boldsymbol{J}(\boldsymbol{\beta})$. This structure is well suited for block inversion using the Schur complement, as mentioned in Section 1.1.2, and is supposedly detected automatically if done in Matlab. Exploiting this structure speeds up the computations immensely, especially if N is large.

8.2.2 Choice of Initial Point

The Levenberg-Marquardt method requires an initial point $\boldsymbol{\beta}_0 = (\boldsymbol{\eta}_0, \boldsymbol{\Xi}_0)$ from which to start iterating. While the Levenberg-Marquardt method is less sensitive to the choice of initial point than e.g. the Gauß-Newton method, an initial point far away from the minimum will typically result in slower convergence and perhaps also a worse minimum.

In many cases, the set of point correspondences to which we will apply the bundle adjustment will be taken as a consensus set generated by using RANSAC with the 2.5-point solver from Chapter 7. The 2.5-point solver gives a homography from which the generating parameters can be determined using the SVD-based recovery method from Section 5.2, and this gives a suitable choice of $\boldsymbol{\eta}_0$. An easy method for initialising the variables in $\boldsymbol{\Xi}$ consists of using $\boldsymbol{\eta}_0$ to compute the transformation $\boldsymbol{R}_{\psi\theta}^T$ to the overhead view in Figure 4.4(b) and apply it to the points $\boldsymbol{x}_j^*$ to get $\boldsymbol{\Xi}_0$.

Chapter 9

Closing Words

As we have seen in this thesis, solutions for camera-based SLAM draw on theory and techniques from a wide range of areas, such as parameter estimation (Chapter 1), projective geometry and multiple view geometry (Chapter 2 and Chapter 3), and algebraic geometry (Chapter 7). Each of these areas has an extensive and steadily growing body of literature, and there definitely still remain many results and methods from these fields to be applied, as well as a host of interesting problems to be investigated.

The approach and attitude in this thesis have generally been close to the *odometry methods*, in that the focus has been on the local frame-to-frame motion estimation rather than on improving global consistency by focusing on the mapping part of SLAM. While it has been demonstrated in this thesis and the underlying papers that the presented methods succeed at estimating the motion, in order to actually make a fully working and useful SLAM system, the mapping part should also be considered to some extent.

The extension of the iterative method for recovering the parameter vector $\boldsymbol{\eta}$, which was discussed in Section 6.4, enabled us to use more than one homography to compute the tilt angles ψ and θ. It would be very useful to have a non-iterative method – similar to the SVD-based method in Section 5.2 – which could also be used on several homographies simultaneously.

Planar motion is a common and very useful assumption in mobile robotics, because, as we explained in Chapter 4, it can be used to bound the vertical positioning error over time and prevent scale drift. These benefits, however, come at the cost of complexity. For example, the 2.5-point solver

from Chapter 7, that was used to enforce the planar motion model at the homography estimation stage, is significantly more complex than the conventional methods for homography estimation.

In typical indoor environments, there are additional planar surfaces which could be used for robot navigation, even though the robot is not travelling parallel to all of them. To each such planar surface, there would be an associated inter-image homography, and they would all have to be compatible with the same epipolar geometry. An interesting direction of future investigations would be to find methods for determining such homographies. This problem has already been studied to some extent, e.g. in Zuliani, Kenney and Manjunath (2005), but it is only lately that the joint consistency requirement has been given much attention (Szpak, Chojnacki and van den Hengel, 2015), and there remains much to be done in this area.

Bibliography

Anderson, E. et al. (1999). *LAPACK Users' Guide*. Third Ed. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics. ISBN: 0-89871-447-8 (cited on p. 1).

Arun, K. S., Huang, T. S. and Blostein, S. D. (1987). 'Least Squares Fitting of Two 3-D Point Sets'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 9:5, pp. 698–700 (cited on pp. 77, 78).

Bay, H., Tuytelaars, T. and Van Gool, L. (2006). 'SURF: Speeded Up Robust Features'. In: *Proceedings of the 9th European Conference on Computer Vision (ECCV)*. Vol. 3951. in Series *Lecture Notes in Computer Science*. Graz, Austria: Springer-Verlag, pp. 404–417. ISBN: 978-3-540-33832-1 (cited on p. 36).

Bernshtein, D. N. (1975). 'The number of roots of a system of equations'. In: *Functional Analysis and its Applications*, Vol. 9:3, pp. 183–185 (cited on p. 93).

Berntorp, K. (2014). *Particle Filtering and Optimal Control for Vehicles and Robots*. Doctoral thesis. Lund University (cited on p. 51).

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. in Series *Information Science and Statistics*. Heidelberg, Germany: Springer-Verlag. ISBN: 978-0-387-31073-2 (cited on pp. 11, 96).

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge, England, UK: Cambridge University Press. ISBN: 978-0-521-83378-3 (cited on pp. 8, 12, 14, 15).

Brown, D. C. (1971). 'Close-Range Camera Calibration'. In: *Photogrammetric Engineering*, Vol. 37:8, pp. 855–866 (cited on p. 33).

Busemann, H. and Kelly, P. J. (1953). *Projective Geometry and Projective Metrics*. Vol. 3. in Series *Pure and Applied Mathematics*. Mineola, NY, USA: Academic Press (cited on p. 22).

Byröd, M. (2010). *Numerical Methods for Geometric Vision: From Minimal to Large Scale Problems*. Doctoral thesis. Lund University (cited on pp. 85, 87).

Byröd, M., Josephson, K. and Åström, K. (2008). 'A Column-Pivoting Based Strategy for Monomial Ordering in Numerical Gröbner Basis Calculations'. In: *Proceedings of the 10th European Conference on Computer Vision (ECCV)*. Vol. 5305. in Series *Lecture Notes in Computer Science*. Marseille, France: Springer-Verlag, pp. 130–143. ISBN: 978-3-540-88692-1 (cited on p. 91).

Byröd, M., Josephson, K. and Åström, K. (2009). 'Fast and Stable Polynomial Equation Solving and Its Application to Computer Vision'. In: *International Journal of Computer Vision*, Vol. 84:3, pp. 237–256 (cited on p. 87).

Cox, D. A., Little, J. B. and O'Shea, D. (2007). *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Third Ed. in Series *Undergraduate Texts in Mathematics*. New York, NY, USA: Springer-Verlag. ISBN: 978-0-387-35650-1 (cited on pp. 84, 85, 88, 89).

Davison, A. J. (2003). 'Real-Time Simultaneous Localisation and Mapping with a Single Camera'. In: *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV)*. Vol. 2. Nice, France: IEEE Computer Society, pp. 1403–1410. ISBN: 0-7695-1950-4 (cited on p. 51).

Davison, A. J., Reid, I. D., Molton, N. D. and Stasse, O. (2007). 'MonoSLAM: Real-Time Single Camera SLAM'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29:6, pp. 1052–1067 (cited on p. 51).

Durrant-Whyte, H. F. (1987). 'Consistent Integration and Propagation of Disparate Sensor Observations'. In: *The International Journal of Robotics Research*, Vol. 6:3, pp. 3–24 (cited on p. 51).

Durrant-Whyte, H. F. and Bailey, T. (2006a). 'Simultaneous Localization and Mapping: Part I'. In: *IEEE Robotics and Automation Magazine*, Vol. 13:2, pp. 99–108 (cited on p. 49).

Durrant-Whyte, H. F. and Bailey, T. (2006b). 'Simultaneous Localization and Mapping: Part II'. In: *IEEE Robotics and Automation Magazine*, Vol. 13:3, pp. 108–117 (cited on p. 49).

Faugeras, O. D. (1992). 'What can be seen in three dimensions with an uncalibrated stereo rig?' In: *Proceedings of the Second European Conference on Computer Vision (ECCV)*. Vol. 588. in Series *Lecture Notes in Computer Science*. Santa Margherita Ligure, Italy: Springer-Verlag, pp. 563–578. ISBN: 978-3-540-55426-4 (cited on p. 40).

Faugeras, O. D. and Lustman, F. (1988). 'Motion and Structure from Motion in a Piecewise Planar Environment'. In: *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 2:3, pp. 485–508 (cited on p. 62).

Fischler, M. A. and Bolles, R. C. (1981). 'Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography'. In: *Communications of the ACM*, Vol. 24:6, pp. 381–395 (cited on pp. 15, 16).

Fraunhofer IPA (2012). *Compact Drive Modules for Omnidirectional Robot Platforms*. URL: `http://www.care-o-bot.de/en/rob-work.html` (visited on 26/02/2016) (cited on p. 49).

Gauglitz, S., Höllerer, T. and Turk, M. (2011). 'Evaluation of Interest point Detectors and Feature Descriptors for Visual Tracking'. In: *International Journal of Computer Vision*, Vol. 94:3, pp. 335–360 (cited on p. 37).

Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations*. Third Ed. Baltimore, MD, USA: Johns Hopkins University Press. ISBN: 0-8018-5414-8 (cited on pp. 1, 3, 7, 26).

Grayson, D. R., Stillman, M. E. and Eisenbud, D. (2015). *Macaulay2: a software system for research in algebraic geometry*. URL: `http://www.math.uiuc.edu/Macaulay2/` (cited on pp. 85, 94).

Gustafsson, F. (2012). *Statistical Sensor Fusion*. Second Ed. Lund, Sweden: Studentlitteratur AB. ISBN: 978-91-44-07732-1 (cited on p. 51).

Hajjdiab, H. and Laganière, R. (2004). 'Vision-based Multi-Robot Simultaneous Localization and Mapping'. In: *Proceedings of the 1st Canadian Conference on Computer and Robot Vision (CRV)*. London, ON, Canada: IEEE Computer Society, pp. 155–162. ISBN: 0-7695-2127-4 (cited on pp. 54, 55, 59, 67).

Harris, C. G. and Pike, J. M. (1988). '3D Positional Integration from Image Sequences'. In: *Image and Vision Computing*, Vol. 6:2, pp. 87–90 (cited on p. 51).

Hartley, R. I. (1992). 'Estimation of Relative Camera Positions for Uncalibrated Cameras'. In: *Proceedings of the Second European Conference on Computer Vision (ECCV)*. Vol. 588. in Series *Lecture Notes in Computer Science*. Santa Margherita Ligure, Italy: Springer-Verlag, pp. 579–587. ISBN: 978-3-540-55426-4 (cited on p. 40).

Hartley, R. I. (1994). 'Projective Reconstruction and Invariants from Multiple Images'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16:10, pp. 1036–1041 (cited on p. 43).

Hartley, R. I. (1997). 'In Defense of the Eight-Point Algorithm'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19:6, pp. 580–593 (cited on p. 42).

Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Second Ed. Cambridge, England, UK: Cambridge University Press. ISBN: 0-521-54051-8 (cited on pp. 14, 16, 30, 41, 42, 47, 100).

Hartmann, J., Klüssendorf, J. H. and Maehle, E. (2013). 'A Comparison of Feature Descriptors for Visual SLAM'. In: *Proceedings of the European Conference on Mobile Robots (ECMR)*. Barcelona, Spain: IEEE Robotics and Automation Society, pp. 56–61. ISBN: 978-1-4799-0263-7 (cited on p. 37).

Horaud, R. and Dornaika, F. (1995). 'Hand-Eye Calibration'. In: *The International Journal of Robotics Research*, Vol. 14:3, pp. 195–210 (cited on p. 55).

Horn, R. A. and Johnson, C. R. (1991). *Topics in Matrix Analysis*. New York, NY, USA: Cambridge University Press. ISBN: 978-0-521-46713-1 (cited on pp. 3, 9).

Huber, P. J. (1964). 'Robust Estimation of a Location Parameter'. In: *The Annals of Mathematical Statistics*, Vol. 35:1, pp. 73–101 (cited on p. 55).

Hungerford, T. W. (1997). *Abstract Algebra, An Introduction*. Second Ed. Stamford, CT, USA: Brooks/Cole Thomson Learning. ISBN: 0-03-010559-5 (cited on p. 29).

Inkilä, K. (2005). 'Homogeneous Least Squares Problem'. In: *Photogrammetric Journal of Finland*, Vol. 19:2, pp. 34–42 (cited on p. 5).

Itseez, Inc. (2017). *OpenCV*. URL: `http://opencv.org/` (cited on pp. 35, 37, 100).

Jones, E. S. and Soatto, S. (2011). 'Visual-inertial navigation, mapping and localization: A scalable real-time causal approach'. In: *The International Journal of Robotics Research*, Vol. 30:4, pp. 407–430 (cited on p. 52).

Kanatani, K. (1998). 'Optimal Homography Computation with a Reliability Measure'. In: *Proceedings of the IAPR Workshop on Machine Vision Applications (MVA)*. Chiba, Japan, pp. 426–429. ISBN: 4-901122-98-3 (cited on p. 100).

Knuth, D. E. (1997). *Fundamental Algorithms*. Third Ed. Vol. 1. in Series *The Art of Computer Programming*. Boston, MA, USA: Addison-Wesley Professional. ISBN: 0-201-89683-4 (cited on p. 13).

Kruppa, E. (1913). 'Zur Ermittlung eines Objektes aus zwei Perspektiven mit innerer Orientierung'. In: *Sitzungsberichte der Kaiserlichen Akademie der Wissenschaften, Mathematisch-Naturwissenschaftlichen Classe*, Vol. 122: pp. 1939–1948 (cited on p. 44).

Kúkelová, Z. (2013). *Algebraic Methods in Computer Vision*. Doctoral thesis. Czech Techniqual University in Prague (cited on p. 85).

Kúkelová, Z., Heller, J., Bujňák, M. and Pajdla, T. (2015). 'Radial Distortion Homography'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA: IEEE, pp. 639–647. ISBN: 978-1-4673-6964-0 (cited on p. 44).

Legendre, A.-M. (1805). *Nouvelles Méthodes pour la Détermination des Orbites des Comètes*. Paris, France: Firmin Didot (cited on p. 12).

Levenberg, K. (1944). 'A Method for the Solution of Certain Non-Linear Problems in Least Squares'. In: *Quarterly of Applied Mathematics*, Vol. 2:2, pp. 164–168 (cited on p. 14).

Li, H. and Hartley, R. I. (2006). 'Five-Point Motion Estimation Made Easy'. In: *Proceedings of the 18th International Conference on Pattern Recognition (ICPR)*. Vol. 1. Hong Kong (SAR): IEEE, pp. 630–633. ISBN: 0-7695-2521-0 (cited on p. 44).

Liang, B. and Pears, N. (2002). 'Visual Navigation using Planar Homographies'. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 1. Washington, DC, USA: IEEE Robotics and Automation Society, pp. 205–210. ISBN: 0-7803-7272-7 (cited on pp. 54, 64, 74, 75).

Longuet-Higgins, H. C. (1981). 'A computer algorithm for reconstructing a scene from two projections'. In: *Nature*, Vol. 293: pp. 133–135 (cited on p. 44).

Lowe, D. G. (2004). 'Distinctive Image Features from Scale-Invariant Keypoints'. In: *International Journal of Computer Vision*, Vol. 60:2, pp. 91–110 (cited on p. 36).

Luong, Q.-T. and Viéville, T. (1996). 'Canonical Representations for the Geometries of Multiple Projective Views'. In: *Computer Vision and Image Understanding*, Vol. 64:2, pp. 193–229 (cited on pp. 42, 47).

Malis, E. and Vargas, M. (2007). *Deeper understanding of the homography decomposition for vision-based control*. Research Report 6303. Sophia Antipolis Cedex, France: INRIA (cited on p. 63).

Marquardt, D. W. (1963). 'An Algorithm for Least-Squares Estimation of Nonlinear Parameters'. In: *Journal of the Society of Industrial and Applied Mathematics*, Vol. 11:2, pp. 431–441 (cited on p. 14).

Mur-Artal, R., Montiel, J. M. M. and Tardós, J. D. (2015). 'ORB-SLAM: A Versatile and Accurate Monocular SLAM System'. In: *IEEE Transactions on Robotics*, Vol. 31:5, pp. 1147–1163 (cited on p. 51).

Möbius, A. F. (1827). *Der barycentrische Calcul - ein neues Hülfsmittel zur analytischen Behandlung der Geometrie*. Leipzig, Kingdom of Saxony: Verlag von Johann Ambrosius Barth (cited on p. 21).

Newman, P. and Ho, K. (2005). 'SLAM- Loop Closing with Visually Salient Features'. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Barcelona, Spain: IEEE Robotics and Automation Society, pp. 635–642. ISBN: 0-7803-8914-X (cited on p. 52).

Nikon Corporation (2011). *K-Series Optical CMM solutions – supporting a variety of metrology applocations*. URL: `http://www.nikonmetrology.com` (visited on 26/02/2016) (cited on p. 79).

Nistér, D. (2004). 'An Efficient Solution to the Five-Point Relative Pose Problem'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26:6, pp. 756–770 (cited on p. 44).

Ortín, D. and Montiel, J. M. M. (2001). 'Indoor robot motion based on monocular images'. In: *Robotica*, Vol. 19:3, pp. 331–342 (cited on p. 53).

Pham, T. T., Chin, T.-J., Yu, J. and Suter, D. (2014). 'The Random Cluster Model for Robust Geometric Fitting'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 36:8, pp. 1658–1671 (cited on p. 17).

Pham, T. T. (2014). *Robust Estimation in Computer Vision: Optimisation Methods and Applications*. Doctoral thesis. The University of Adelaide (cited on p. 17).

Scaramuzza, D. (2011a). '1-Point-RANSAC Structure from Motion for Vehicle-Mounted Cameras by Exploiting Non-holonomic Constraints'. In: *International Journal of Computer Vision*, Vol. 95:1, pp. 74–85 (cited on p. 54).

Scaramuzza, D. (2011b). 'Performance Evaluation of 1-Point-RANSAC Visual Odometry'. In: *Journal of Field Robotics*, Vol. 28:5, pp. 792–811 (cited on p. 54).

Shafarevich, I. R. (2013). *Basic Algebraic Geometry 1: Varieties in Projective Space*. Heidelberg, Germany: Springer-Verlag. ISBN: 978-3-642-37955-0 (cited on p. 20).

Stewénius, H., Engels, C. and Nistér, D. (2006). 'Recent developments on direct relative orientation'. In: *Journal of Photogrammetry and Remote Sensing*, Vol. 60:4, pp. 284–294 (cited on p. 44).

Stewénius, H., Schaffalitzky, F. and Nistér, D. (2005). 'How Hard is 3-view Triangulation Really?' In: *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV)*. Vol. 1. Beijing, China: IEEE, pp. 686–693. ISBN: 0-7695-2334-X (cited on p. 44).

Sturm, P. (2011). 'A Historical Survey of Geometric Computer Vision'. In: *Proceedings of the 14th International Conference on Computer Analysis of Images and Patterns (CAIP)*. Vol. 6854. in Series *Lecture Notes in Computer Science*. Seville, Spain: Springer-Verlag, pp. 1–8. ISBN: 978-3-642-23671-6 (cited on p. 43).

Sturm, R. (1869). 'Das Problem der Projectivität und seine Anwendung auf die Flächen zweiten Grades'. In: *Mathematische Annalen*, Vol. 1: pp. 533–574 (cited on pp. 24, 43).

Szeliski, R. (2011). *Computer Vision: Applications and Algorithms*. Heidelberg, Germany: Springer-Verlag. ISBN: 978-1-84882-935-0 (cited on pp. 28, 30).

Szpak, Z. L., Chojnacki, W. and van den Hengel, A. (2015). 'Robust Multiple Homography Estimation: An Ill-Solved Problem'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA: IEEE, pp. 2132–2141. ISBN: 978-1-4673-6964-0 (cited on p. 106).

Terzakis, G., Culverhouse, P., Bugmann, G., Sharma, S. and Sutton, R. (2012). *A Recipe on the Parameterization of Rotation Matrices for Non-Linear Optimization using Quaternions*. Tech. rep. Plymouth, England, UK: Plymouth University (cited on p. 29).

Tevelev, E. A. (2005). *Projective Duality and Homogeneous Spaces*. Vol. 133. in Series *Encyclopaedia of Mathematical Sciences*. Heidelberg, Germany: Springer-Verlag. ISBN: 978-3-540-22898-1 (cited on p. 20).

The MathWorks, Inc. (2013). *Computer Vision System Toolbox™ Reference*. (Note: Release 2013b). URL: `http://www.mathworks.com` (cited on pp. 35, 37).

Trefethen, L. N. and Bau III, D. (1997). *Numerical Linear Algebra*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics. ISBN: 0-89871-361-7 (cited on pp. 1, 3, 85).

Triggs, B., McLauchlan, P. F., Hartley, R. I. and Fitzgibbon, A. W. (1999). 'Bundle Adjustment – A Modern Synthesis'. In: *Vision Algorithms: Theory*

and Practice. International Workshop on Vision Algorithms (IWVA). Vol. 1883. in Series *Lecture Notes in Computer Science*. Corfu, Greece: Springer-Verlag, pp. 298–372. ISBN: 978-3-540-67973-8 (cited on pp. 8, 14, 100).

Tsai, R. Y. and Lenz, R. K. (1989). 'A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration'. In: *IEEE Transactions on Robotics and Automation*, Vol. 5:3, pp. 345–358 (cited on p. 55).

Wadenbäck, M. (2015). *A Result for Orthogonal Plus Rank-1 Matrices*. arXiv: 1512.03715 [math.GM]. URL: http://arxiv.org/abs/1512.03715 (cited on pp. x, 64).

Wadenbäck, M. and Heyden, A. (2013). 'Planar Motion and Hand-Eye Calibration Using Inter-Image Homographies from a Planar Scene'. In: *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*. Vol. 2. Barcelona, Spain: SCITEPRESS, pp. 164–168. ISBN: 978-989-8565-48-8 (cited on pp. ix, 55, 67, 71).

Wadenbäck, M. and Heyden, A. (2014a). 'Ego-Motion Recovery and Robust Tilt Estimation for Planar Motion Using Several Homographies'. In: *Proceedings of the 9th International Conference on Computer Vision Theory and Applications (VISAPP)*. Vol. 3. Lisbon, Portugal: SCITEPRESS, pp. 635–639. ISBN: 978-989-758-009-3 (cited on pp. ix, 55, 67, 75).

Wadenbäck, M. and Heyden, A. (2014b). 'Trajectory Estimation Using Relative Distances Extracted from Inter-Image Homographies'. In: *Conference on Computer and Robot Vision (CRV)*. Montréal, QC, Canada: IEEE Computer Society, pp. 232–237. ISBN: 978-1-4799-4338-8 (cited on pp. ix, 63, 64).

Wadenbäck, M., Karlsson, M., Heyden, A., Robertsson, A. and Johansson, R. (2017). 'Visual Odometry from Two Point Correspondences and Initial Automatic Camera Tilt Calibration'. In: *Proceedings of the 12th International Conference on Computer Vision Theory and Applications (VISAPP)*. Vol. 6. Porto, Portugal: SCITEPRESS, pp. 340–346. ISBN: 978-989-758-227-1 (cited on pp. ix, 55, 77, 79, 80).

Wadenbäck, M., Åström, K. and Heyden, A. (2016). 'Recovering Planar Motion from Homographies Obtained using a 2.5-Point Solver for a Polynomial System'. In: *Proceedings of the 23rd IEEE International Conference on Image Processing (ICIP)*. Phoenix, AZ, USA: IEEE, pp. 2966–2970. ISBN: 978-1-4673-9960-9 (cited on pp. x, 61, 63, 65, 83, 91, 92, 94, 99).

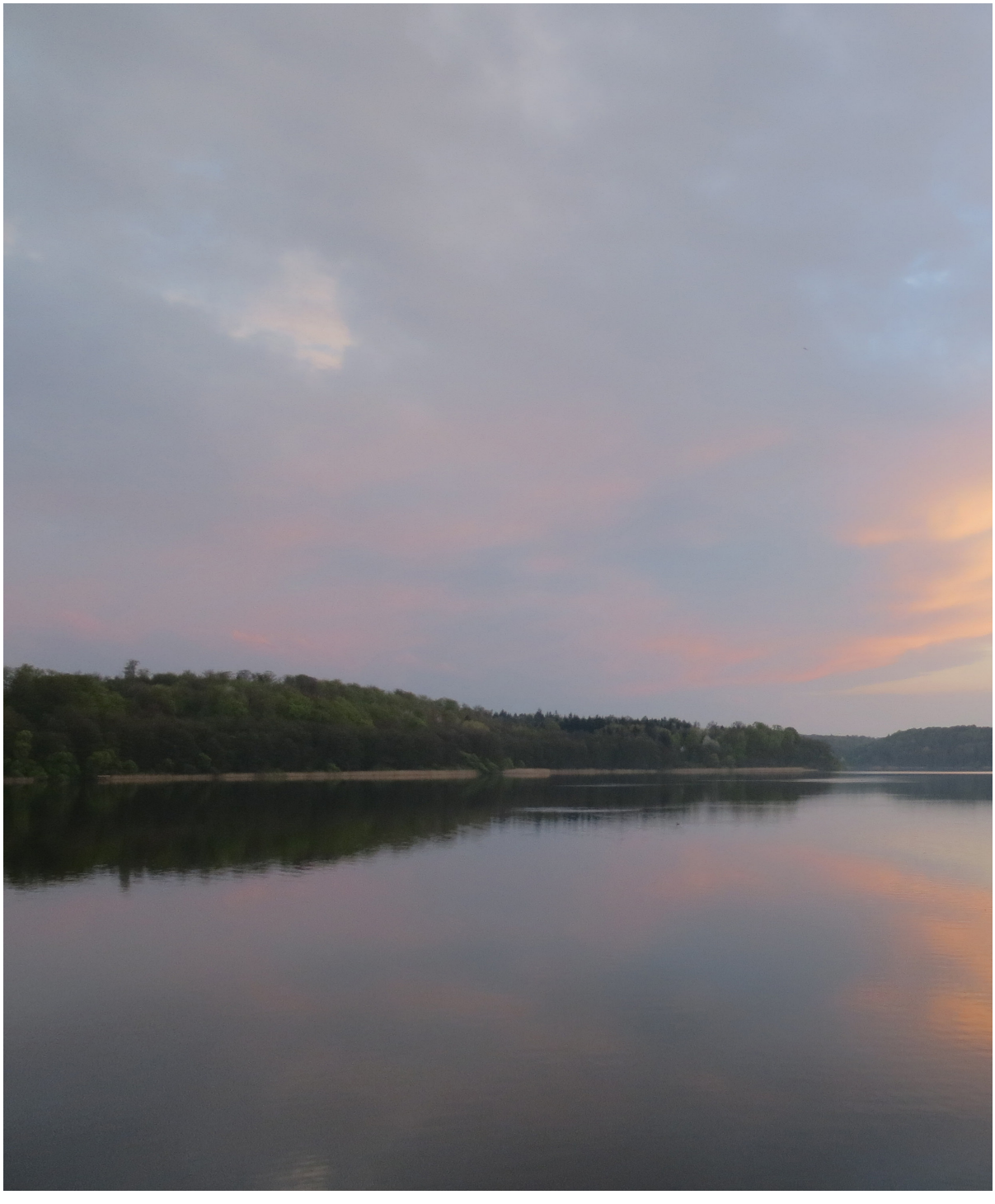
Van den Dries, L. (1986). 'A Generalization of the Tarski-Seidenberg Theorem, and Some Nondefinability Results'. In: *Bulletin of the American Mathematical Society*, Vol. 15:2, pp. 189–193 (cited on p. 88).

Waterloo Maple Inc. (2016). *Maple*. URL: `https://www.maplesoft.com/` (cited on pp. 89, 94, 102).

Zhang, Z. and Hanson, A. R. (1996). '3D Reconstruction Based on Homography Mapping'. In: *Proceedings of the 1996 ARPA Image Understanding Workshop (ARPA)*. Vol. 2. Palm Springs, CA, USA: IEEE, pp. 1007–1012. ISBN: 1-55860-401-4 (cited on pp. 61–63).

Zienkiewicz, J. and Davison, A. J. (2015). 'Extrinsics Autocalibration for Dense Planar Visual Odometry'. In: *Journal of Field Robotics*, Vol. 32:5, pp. 803–825 (cited on p. 55).

Zuliani, M., Kenney, C. S. and Manjunath, B. S. (2005). 'The MultiRANSAC Algorithm and its Applications to Detect Planar Homographies'. In: *Proceedings of the 12th IEEE International Conference on Image Processing (ICIP)*. Vol. 3. Genova, Italy: IEEE, pp. 153–156. ISBN: 0-7803-9134-9 (cited on p. 106).



LUND UNIVERSITY

Doctoral Theses in Mathematical Sciences 2017:3
ISBN 978-91-7753-153-1
ISSN 1404-0034