Computational Methods for Computer Vision

Minimal Solvers and Convex Relaxations

Viktor Larsson

LUND UNIVERSITY

ACADEMIC THESIS

which, by due permission of the Faculty of Engineering at Lund University, will be publicly defended on Friday 1st of June, 2018, at 13:15 in lecture hall MH:H, Centre for Mathematical Sciences, Sölvegatan 18, Lund, for the degree of Doctor of Philosophy in Engineering.

Faculty opponent

Dr. Hongdong Li, Australian National University, Australia.

Organization Centre for Mathematical Sciences Faculty of Engineering Lund University Box 118 SE-221 00 Lund Author(s) Viktor Larsson	Document name DOCTORATE THESIS IN MATHEMATICAL SCIENCES Date of issue June 2018 Sponsoring organization Supervisors Carl Olsson, Fredrik Kahl, Fredrik Andersson

Title and subtitle

Computational Methods in Computer Vision: Minimal Solvers and Convex Relaxations

Abstract

Robust fitting of geometric models is a core problem in computer vision. The most common approach is to use a hypothesize-and-test framework, such as RANSAC. In these frameworks the model is estimated from as few measurements as possible, which minimizes the risk of selecting corrupted measurements. These estimation problems are called minimal problems, and they can often be formulated as systems of polynomial equations. In this thesis we present new methods for building so-called minimal solvers or polynomial solvers, which are specialized code for solving such systems. On several minimal problems we improve on the state-of-the-art both with respect to numerical stability and execution time.

In many computer vision problems low rank matrices naturally occur. The rank can serve as a measure of model complexity and typically a low rank is desired. Optimization problems containing rank penalties or constraints are in general difficult. Recently convex relaxations, such as the nuclear norm, have been used to make these problems tractable. In this thesis we present new convex relaxations for rank-based optimization which avoid drawbacks of previous approaches and provide tighter relaxations. We evaluate our methods on a number of real and synthetic datasets and show state-of-the-art results.

Key words computer vision, geometric vision, minimal solvers, convex relaxations, pose estimation		
Classification system and/or index terms (if any)		
Supplementary bibliographical information		Language English
ISSN and key title 1404-0034		ISBN 978-91-7753-695-6
Recipient’s notes	Number of pages xviii+250	Price
	Security classification	

Computational Methods for Computer Vision

Minimal Solvers and Convex Relaxations

Viktor Larsson

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden

http://www.maths.lth.se/

Doctoral Theses in Mathematical Sciences 2018:4
ISSN 1404-0034

ISBN 978-91-7753-695-6 (print)
ISBN 978-91-7753-696-3 (electronic)
LUTFMA-1067-2018

Printed in Sweden by MediaTryck, Lund 2018

Abstract

Robust fitting of geometric models is a core problem in computer vision. The most common approach is to use a hypothesize-and-test framework, such as RANSAC. In these frameworks the model is estimated from as few measurements as possible, which minimizes the risk of selecting corrupted measurements. These estimation problems are called *minimal problems*, and they can often be formulated as systems of polynomial equations. In this thesis we present new methods for building so-called *minimal solvers* or *polynomial solvers*, which are specialized code for solving such systems. On several minimal problems we improve on the state-of-the-art both with respect to numerical stability and execution time.

In many computer vision problems low rank matrices naturally occur. The rank can serve as a measure of model complexity and typically a low rank is desired. Optimization problems containing rank penalties or constraints are in general difficult. Recently convex relaxations, such as the nuclear norm, have been used to make these problems tractable. In this thesis we present new convex relaxations for rank-based optimization which avoid drawbacks of previous approaches and provide tighter relaxations. We evaluate our methods on a number of real and synthetic datasets and show state-of-the-art results.

Populärvetenskaplig sammanfattning

Datorseende är idag ett väldigt aktivt forskningsområde och dyker upp i populära tillämpningar så som Virtual/Augmented Reality (VR/AR), självkörande bilar och autonoma drönare.

Gemensamt för dessa är att man vill utvinna tredimensionell information från tvådimensionella bilder. Detta kan t.ex. vara kamerans position och orientering, eller avståndet till något objekt som förekommer i bilderna. Detta kräver matematiska metoder och modeller. För att kunna vara användbara i verkligheten så måste metoderna vara både effektiva och robusta mot fel i indata. Exempelvis så är toleransen för fel väldig låg då en självkörande bil ska identifiera avstånd till medtrafikanter. För VR/AR krävs att användarens position bestäms i realtid med så lite fördröjning som möjligt, då även små fördröjningar kan leda till illamående hos användaren.

I den här avhandlingen utvecklar vi metoder för att snabbt och robust kunna lösa olika sorters geometriska estimeringsproblem, som t.ex. att bestämma kamerans position relativt en 3D-modell. Denna typ av problem kan ofta beskrivas med system av polynomekvationer. Inom avhandlingsarbetet tar vi fram metoder för att effektivt kunna lösa sådana system. Metoderna vi utvecklar är generella och kan användas på många olika problem inom datorseende.

Idag bygger de flesta metoder för 3D-rekonstruktion på antagandet att världen är statisk, d.v.s. att 3D-strukturen inte förändras mellan bilderna. Men i verkligheten så förekommer också dynamiska scener, där föremålen i scenen deformeras och rör sig både relativt kameran och varandra. Till skillnad från fallet då scenen är statisk, så blir problemet väldigt svårt att modellera matematiskt då scenen är dynamisk. Om man endast kräver att rekonstruktionen ska stämma överens med bilderna så kommer det att finnas oändligt många lösningar. För att begränsa antalet lösningar så antar man att den sökta lösningen kommer från verkligheten och är enkel i något avseende. Ett vanligt antagande är till exempel att 3D punkterna ska röra sig i en jämn bana och inte hoppa omkring mellan bilderna. I den här avhandlingen arbetar vi med antagandet att 3D punkterna ska ha låg rang. Rang är ett matematiskt begrepp som mäter hur mycket beroende det finns i strukturen. En lösning med låg rang kommer att ha många punkter som rör sig på ett liknande sätt. Inom avhandlingen undersöker vi dels olika sorters rang-modeller och dels matematiska verktyg för att kunna hantera problemformuleringar där det ingår rang-villkor.

Preface

This thesis is based on the following papers:

Main papers:

- **V. Larsson**, K. Åström, M. Oskarsson, "Efficient Solvers for Minimal Problems by Syzygy-based Reduction", *Proc. Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA, 2017.
- **V. Larsson**, K. Åström, M. Oskarsson, "Polynomial Solvers for Saturated Ideals", *Proc. International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017.
- **V. Larsson**, K. Åström, "Uncovering Symmetries in Polynomial Systems", *Proc. European Conference on Computer Vision (ECCV)*, Amsterdam, Netherlands, 2016.
- **V. Larsson**, M. Oskarsson, K. Åström, A. Wallis, Z. Kukelova, T. Pajdla, "Beyond Gröbner Bases: Basis Selection for Minimal Solvers", *Proc. Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, USA, 2018.
- **V. Larsson**, Z. Kukelova, Y. Zheng, "Making Minimal Solvers for Absolute Pose Estimation Compact and Robust ", *Proc. International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017.
- **V. Larsson**, Z. Kukelova, Y. Zheng, "Camera Pose Estimation with Unknown Principal Point", *In Proc. Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, USA, 2018.

- **V. Larsson**, C. Olsson, E. Bylow, F. Kahl, "Rank Minimization with Structured Data Patterns", *Proc. European Conference on Computer Vision (ECCV)*, Zürich, Switzerland, 2014.
- **V. Larsson**, C. Olsson, "Convex Envelopes for Low-rank Approximation", *Proc. Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, Hong Kong, China, 2015.
- **V. Larsson**, C. Olsson, "Convex Low Rank Approximation", *International Journal of Computer Vision (IJCV)*, 2016.
- **V. Larsson**, C. Olsson, "Compact Matrix Factorization with Dependent Subspaces", *Proc. Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA, 2017.

Subsidiary papers:

- **V. Larsson**, C. Olsson, F. Kahl, "A Simple Method for Subspace Estimation with Corrupted Columns", *Workshop on Robust Subspace Learning and Computer Vision (RSL-CV)*, Santiago, Chilé, 2015.
- J. Fredriksson, **V. Larsson**, C. Olsson, "Practical Robust Two-view Translation Estimation", *Proc. Computer Vision and Pattern Recognition (CVPR)*, Boston, USA, 2015.
- J. Fredriksson, **V. Larsson**, C. Olsson, F. Kahl, "Optimal Relative Pose with Unknown Correspondences", *Proc. Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, 2016.
- **V. Larsson**, J. Fredriksson, C. Toft, F. Kahl, "Outlier Rejection for Absolute Pose Estimation with Known Orientation", *Proc. British Machine Vision Conference (BMVC)*, York, England, 2016.
- J. Fredriksson, **V. Larsson**, C. Olsson, O. Enqvist, F. Kahl, "Efficient Algorithms for Robust Estimation of Relative Translation", *Image and Vision Computing (IVC)*, 2016.
- J. Alvén, F. Kahl, M. Landgren, **V. Larsson**, J. Ulén, "Shape-aware Multi-Atlas Segmentation", *Proc. International Conference on Pattern Recognition (ICPR)*, Cancun, Mexico, 2016.

- C. Olsson, M. Carlsson, F. Andersson, **V. Larsson**, “Non-Convex Rank/Sparsity Regularization and Local Minima”, *Proc. International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017.
- J. Pritts, Z. Kukelova, **V. Larsson**, O. Chum, “Radially-Distorted Conjugate Translations”, *Proc. Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, USA, 2018.

Acknowledgements

This thesis marks the end of a five year chapter of my life which I look back upon fondly. For this I would like to thank all of my colleagues at the Centre for Mathematical Sciences, especially the members of the vision group. I want to give a special thanks to my supervisors, Carl Olsson and Fredrik Kahl, for their guidance and support, in both research related matters and otherwise. I also want to thank Kalle Åström and Magnus Oskarsson for introducing me to the wonderful world of minimal solvers, and for the many papers we wrote together. I am also grateful to my other collaborators and coauthors, especially Johan Fredriksson, Yinqiang Zheng, Zuzana Kukelova, Tomas Pajdla and James Pritts.

Finally, for their encouragement and for supporting me in all aspects of life, I would like to thank my friends, my family and Rebecka Nyqvist.

This work has funded by the Swedish Research Council (grants no. 2012-4213 and 2012-4215), the Crafoord Foundation (grant no. 20150601), Strategic Research Area ELLIIT and eSSENCE, Swedish Foundation for Strategic Research project *Semantic Mapping and Visual Navigation for Smart Robots* (grant no. RIT15-0038) and Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP).

Contents

Introduction

1 Motivation

In computer vision the goal is to extract information from images. This can be information that pertains to the image at hand, e.g. if we want to count the number of cars in the image, or if we want to partition the pixels into the different visible objects. In geometric computer vision we are instead interested in extracting 3D information, that is to say something about the geometry of the actual scene where the image was taken. From only a single viewpoint the 3D geometry is ambiguous and it is therefore necessary to have multiple images from different viewpoints. The most fundamental problem, called *Structure-from-Motion*, is to determine both the 3D structure of the scene and the camera poses.

This thesis deals with two main topics from computer vision. The first topic concerns robust fitting of geometric models. Most of the problems we consider are related to camera pose estimation in different settings. Either we wish to estimate the position and orientation of a camera relative to a known 3D model (given 2D to 3D correspondences), or we wish to estimate the relative pose of two or more cameras (given 2D to 2D correspondences). These two types of camera pose estimation problems are used as building blocks for most modern Structure-from-Motion frameworks [188, 189, 195, 194, 166].

Outlier correspondences (i.e. incorrect matches) make pose estimation difficult. Usually correspondences are found using only image data without regard for the geometry, which makes outlier correspondences unavoidable. To solve this problem, robust estimation methods such as Random Sample and Consensus (RANSAC) [64] (or some of the many variations of this [212, 213, 39, 38]) are typically used. The basic idea in RANSAC is to randomly select a small subset of the data and estimate the model from this subset. The model is then validated on the entire dataset. If the sample was outlier free, the model will typically agree with a large subset of the dataset. This is then iterated and the best model (i.e. the model which has the largest consensus set) is kept. This both identifies potential outlier datapoints as well as provides a starting point for further local optimization.

To minimize the risk of selecting any outliers in the random sample, we select the minimal number of datapoints which allows us to estimate the model. These estimation problems are called *minimal problems*. For example, if we wish to fit a line, the problem is minimal with two points. If we only have a single point there will be infinitely many lines passing through it. Similarly, if we have three points, these will in general not lie on any line.

For many problems in geometric vision, the minimal problems reduce to solving systems of polynomial equations. To maximize the chance of selecting any outlier-free samples in RANSAC, we want to run as many iterations as possible, so we need to solve many instances of the same minimal problem. To make this tractable, there is a need for efficient methods for solving these systems. Designing specialized code for solving particular problems efficiently is the focus of the first part of the thesis. These specialized methods are usually called *minimal solvers* or *polynomial solvers*.

The second topic of this thesis concerns low rank models and low rank approximation. Low rank matrices occur naturally in many applications in computer vision, e.g. Structure-from-Motion (affine [211], projective [155], non-rigid [20, 72, 48] and articulated [228]), photometric stereo [17], optical flow estimation [71], motion segmentation [228, 221] and many others. There are also many applications outside of computer vision, e.g. linear system identification [63, 8], recommender systems [112] and sensor network localization [196]. These examples have in common that there is some bilinear model which explains the measurements (or at least the measurements are well-approximated with a bilinear model).

The rank of a matrix is the maximum number of linearly independent columns (or rows), and in many applications the rank serves as a measure of the complexity of the model. One such example is Non-Rigid Structure-from-Motion (NRSfM). In regular Structure-from-Motion the rigidity (i.e. the assumption that the scene is static) constrains both the structure (3D points) and the motion (camera positions and orientations). In contrast, for NRSfM the 3D points are allowed to move between each captured image, making the problem horribly under-constrained.

This is typically resolved by adding additional constraints that either the motion or the structure should be simple in some sense. In their seminal work, Bregler et al. [20] proposed to do this by adding additional rank constraints. The model was that the 3D points in each frame could be explained by a linear combination of some unknown basis shapes. In [20] they showed that the rank of the matrix containing the 2D image coordinates is related to the number of basis shapes needed.

In general, optimization problems containing rank-based penalties or constraints are very difficult. However in some special cases there are tractable algorithms for finding the optimum. The classical result by Eckart and Young [53], states that for a given matrix M the best rank r approximation (in any unitarily invariant norm), i.e.

$$X^\star = \arg\min_X \|X - M\|^2 \quad \text{s. t.} \quad \operatorname{rank}(X) \leq r, \tag{1}$$

is given by truncating the Singular Value Decomposition (SVD) of M. The problem in (1) is a non-convex optimization problem and if we add more penalties or constraints (even convex ones) the problem becomes much more difficult since the SVD-based solution is no longer applicable. One recent approach is to replace the non-convex rank function with a convex surrogate. The most common choice is the nuclear norm (or trace norm) [180, 37, 72, 175, 9], which is the sum of the singular values. The drawback of this approach is that it penalizes all singular values equally. Ideally we would like to avoid penalizing the first r singular values. One of the contributions of this thesis is a convex surrogate which avoids this problem (see Chapter 7).

2 Overview and Outline

This thesis consists of three parts. The first part (Chapters 1-4) deals with the construction of polynomial solvers for minimal problems. Different methods for building faster and more stable polynomial solvers are presented. In the second part (Chapters 5 and 6) we show two applications in pose estimation where we apply the methods from the first part. Finally, the third part (Chapters 7 and 8) concerns low rank models and their applications in computer vision.

Chapter 1. We present an automatic method for creating polynomial solvers for minimal problems. The method is similar to that of Kukelova et al. [122], in that it only requires the user to specify the equations and then generates stand-alone code for solving new instances. The main contribution is that we utilize the Gröbner bases for both the ideal and the syzygy module of the generators when creating the elimination template. The method is evaluated on a large number of problems from geometric computer vision. The chapter is based on the paper [134].

Chapter 2. We present a new method for incorporating saturation into polynomial solvers. When saturation is needed to get a zero-dimensional ideal, the typical approach in computer vision (e.g. [202, 116, 193, 32]) has been to first compute generators for the saturated ideal as a pre-processing step, and then construct an elimination template from these. In this chapter we instead propose to lift the problem into the original ideal. This allows us to use a single elimination template which can e.g. be constructed using the method presented in Chapter 1. The chapter is based on the paper [135].

Chapter 3. We present an extension to the methods from Ask et al. [13] and Kuang et al. [119] which exploits certain symmetries in polynomial systems to make smaller elimination templates. For example, consider polynomial systems where all monomials have an even degree. In this case for any solution $\boldsymbol{x} \in V$ you also have that $-\boldsymbol{x} \in V$ is a solution. This was the type of symmetry studied in [13, 119]. For these systems the action matrix can be chosen as block diagonal with the blocks corresponding to the basis elements with either even or odd degree. In [13, 119] it was then suggested to only consider a single of these blocks when constructing the elimination template. In this chapter we provide a generalization of this technique as well as show an application where this gives a large reduction in elimination template size. The chapter is based on the paper [133].

Chapter 4. In this chapter we study the problem of selecting the monomial basis in the action matrix method. The method in Chapter 1 uses the standard monomials w.r.t. a GRevLex Gröbner basis. This is an arbitrary choice and in this chapter we try to choose the basis to minimize the size of the elimination template. However, there are infinitely many monomial basis for the quotient ring, so it is not possible to perform any exhaustive search. Instead we consider two different approaches. The first is to use the so-called Gröbner fans [165], which allows us to enumerate all bases which are standard monomials to some Gröbner basis. The second approach is a heuristic random sampling method. Interestingly, we show several examples where we can find smaller templates compared to ones built using Gröbner bases. The chapter is based on the paper [142].

Chapter 5. We consider the camera pose estimation problem with unknown focal length and one parameter radial distortion (P4PFR). Given four 2D-3D correspondences, $\boldsymbol{x_i} = (u_i, v_i) \leftrightarrow \boldsymbol{X}_i = (x_i, y_i, z_i)$, we wish to estimate the camera pose $(R, \boldsymbol{t})$, the focal length f and the distortion parameter k. Using the one-parameter division model from Fitzgibbon [65] for the radial distortion, the projection equations are then[1]

$$\begin{pmatrix} \boldsymbol{x}_i \\ 1 + k\,\|\boldsymbol{x}_i\|^2 \end{pmatrix} \simeq \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} (R\boldsymbol{X}_i + \boldsymbol{t}), \tag{2}$$

This problem was originally solved by Josephsson and Byröd [102]. Later Bujnak et al. [28] proposed an improved solver by consider the planar and non-planar case separately. In this chapter we improve on the formulation of Bujnak et al. [28] and propose a solver which handles both planar and non-planar data. The chapter is based on the paper [137].

Chapter 6. In this chapter we consider the problem of pose estimation when both the focal length and principal point are unknown. The principal point is typically assumed to be centered in the image, but e.g. if the image is cropped this might not be the case. The problem is similar to the P4PFR case from the previous chapter, but instead of estimating the radial distortion parameter k, we wish to estimate the principal point $\boldsymbol{x}_0 = (u_0, v_0)$. We assume that the skew and aspect ratio are known, and by changing coordinates we can w.l.o.g. assume zero

[1]Here $\simeq$ denotes equality up to scale.

skew and unit aspect ratio. In this setting the projection equations are

$$\begin{pmatrix} \boldsymbol{x}_i - \boldsymbol{x}_0 \\ 1 \end{pmatrix} \simeq \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} (R\boldsymbol{X}_i + \boldsymbol{t}). \tag{3}$$

The camera having zero skew and unit aspect ratio poses polynomial constraints on the first 3×3 part of the camera matrix which are well known (see Faugeras [60], Heyden [88]),

$$P = \begin{bmatrix} \boldsymbol{p}_1^T & p_{14} \\ \boldsymbol{p}_2^T & p_{24} \\ \boldsymbol{p}_3^T & p_{34} \end{bmatrix}, \qquad \begin{aligned} &\det\left[\boldsymbol{p}_1,\ \boldsymbol{p}_2,\ \boldsymbol{p}_3\right] \neq 0, && (4) \\ &(\boldsymbol{p}_1 \times \boldsymbol{p}_3) \cdot (\boldsymbol{p}_2 \times \boldsymbol{p}_3) = 0, && (5) \\ &\|\boldsymbol{p}_1 \times \boldsymbol{p}_3\|^2 - \|\boldsymbol{p}_2 \times \boldsymbol{p}_3\|^2 = 0. && (6) \end{aligned}$$

Computing the saturation of (5) and (6) w.r.t. (4) we were able to find additional polynomial constraints which allowed us to drop the non-zero determinant constraint. Using the techniques presented in Chapter 1 we constructed polynomial solvers for both known (P4.5PFUV) and unknown aspect ratio (P5PFUVA). Finally, in this chapter we also considered the difficult case of both unknown principal point and radial distortion. The difficulty comes from the fact that the distortion is centered on the unknown principal point $\boldsymbol{x}_0 = (u_0, v_0)$. In this case the projection equations are

$$\begin{pmatrix} \boldsymbol{x}_i - \boldsymbol{x}_0 \\ 1 + k\,\|\boldsymbol{x}_i - \boldsymbol{x}_0\|^2 \end{pmatrix} \simeq \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} (R\boldsymbol{X}_i + \boldsymbol{t}). \tag{7}$$

The problem is minimal with 5 points, however due to the extra non-linearity we were not able to find any tractable formulation for this problem. Instead we present a relaxed two-step solver which uses 7 point correspondences. The chapter is based on the paper [138].

Chapter 7. In this chapter we look at convex relaxations for low rank approximation. One common approach (see e.g. [180, 37, 72, 175, 9]) is to replace the non-convex rank function with the convex nuclear norm

$$\|X\|_* = \sum_{k=1}^{n} \sigma_k(X), \tag{8}$$

which is simply the sum of the singular values. Similarly to how ℓ_1 regularization is used for sparse regression [210], this in some sense promotes sparsity in the singular values of X and thus also a low rank. Another motivation for using the nuclear norm is that it is the convex envelope of the rank function on the unit ball, $\{X \in \mathbb{R}^{m\times n} \mid \|X\|_2 \leq 1\}$. The restriction to the unit ball is necessary since the convex envelope on the full domain is simply zero. In this chapter we instead consider the convex envelope of the function

$$f(X) = g(\text{rank}(X)) + \|X - X_0\|_F^2\,, \tag{9}$$

where $g : \mathbb{N} \to \mathbb{R} \cup \{\infty\}$ is some penalty function. We show how to evaluate both the convex envelope of f as well as the proximal operator which is necessary for optimization. The chapter is based on the papers [136, 139, 140].

Chapter 8. In this chapter we present a variation on the traditional low rank matrix factorization, where we add additional rank constraints on sub-matrices. This results in a more compact factorization and we show in experiments that this improves inference when we only have partial observations. We also show in experiments that this model is well suited for tasks dealing with non-rigid and multi-body point trajectories. The chapter is based on the paper [141].

Author Contributions

- **V. Larsson**, K. Åström, M. Oskarsson, "Efficient Solvers for Minimal Problems by Syzygy-based Reduction", *Proc. Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA, 2017.

  MO had the original idea. Victor Ufnarovski suggested we consider syzygies. VL implemented the automatic generator. MO implemented a majority of the minimal cases for the experiments.

- **V. Larsson**, K. Åström, M. Oskarsson, "Polynomial Solvers for Saturated Ideals", *Proc. International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017.

VL had the original idea and developed the theory. MO and KÅ did most of the work on the experiments.

- **V. Larsson**, K. Åström, "Uncovering Symmetries in Polynomial Systems", *Proc. European Conference on Computer Vision (ECCV)*, Amsterdam, Netherlands, 2016.

  VL noticed the symmetry in the WPnP problem. KÅ found the necessary change of variables. VL and KÅ jointly developed the theory.

- **V. Larsson**, M. Oskarsson, K. Åström, A. Wallis, Z. Kukelova, T. Pajdla, "Beyond Gröbner Bases: Basis Selection for Minimal Solvers", *Proc. Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, USA, 2018.

  AW suggested considering different monomial orderings. TP suggested we consider Gröbner Fans. VL, MO and KÅ developed the basis sampling. VL and MO performed the experiments.

- **V. Larsson**, Z. Kukelova, Y. Zheng, "Making Minimal Solvers for Absolute Pose Estimation Compact and Robust ", *Proc. International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017.

  ZK found the new camera matrix constraints. VL, ZK and YZ jointly developed the new solvers and wrote the paper. ZK did most of the work for the experiments with real images.

- **V. Larsson**, Z. Kukelova, Y. Zheng, "Camera Pose Estimation with Unknown Principal Point", *In Proc. Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, USA, 2018.

  ZK suggested we consider the unknown principal point problems. VL, ZK and YZ jointly developed the no distortion solvers. VL developed the radial distortion solvers. ZK did most of the work for the experiments with real images.

- **V. Larsson**, C. Olsson, E. Bylow, F. Kahl, "Rank Minimization with Structured Data Patterns", *Proc. European Conference on Computer Vision (ECCV)*, Zürich, Switzerland, 2014.

CO had the original idea. VL and CO developed most of the theory. EB and FK helped perform the experiments and write the paper.

- **V. Larsson**, C. Olsson, "Convex Envelopes for Low-rank Approximation", *Proc. Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, Hong Kong, China, 2015.

  VL and CO jointly developed the theory, performed the experiments and wrote the paper.

- **V. Larsson**, C. Olsson, "Convex Low Rank Approximation", *International Journal of Computer Vision (IJCV)*, 2016.

  VL and CO jointly wrote the paper and performed the experiments.

- **V. Larsson**, C. Olsson, "Compact Matrix Factorization with Dependent Subspaces", *Proc. Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA, 2017.

  VL and CO jointly developed the model. CO proposed the optimization scheme and wrote the code. VL performed the experiments.

3 Geometric Computer Vision

In this section we give a very brief introduction to some basic concepts from geometric computer vision. For more references and a more thorough exposition of the material the reader is referred to the books by Hartley and Zisserman [81], and Forsyth and Ponce [66].

3.1 Pinhole Cameras

The most common camera model in computer vision is by far the pinhole camera [81]. In this model the projection rays intersect in a single point called the *camera center*. The 2D projections are found by intersecting these rays with a plane, which is called the *image plane*. See Figure 1.

Assume that the 3D coordinate system is chosen such that the camera center lies at the origin and the viewing direction is along the z-axis with the image plane at $z = 1$ (see Figure 1). The 2D projection $\boldsymbol{x} = (u, v)^T$ of a 3D point

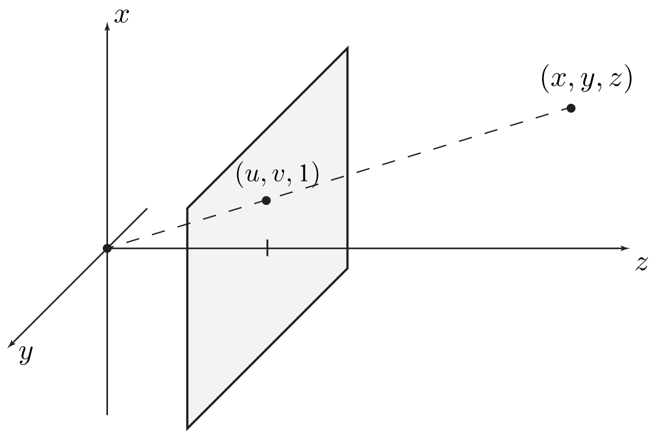


Figure 1: The pinhole camera model. The projection is formed by intersection the image plane with the line going from the camera center to the 3D point.

$\boldsymbol{X} = (x, y, z)^T$ is then found by intersecting the line from the origin to the 3D point with the plane $z = 1$, i.e

$$\boldsymbol{x} = (u, v) = (x/z, y/z)^T. \tag{10}$$

Let $\Pi : \mathbb{R}^3 \to \mathbb{R}^2$ denote the (pinhole-)projection operator which divides by the third coordinate, i.e. $\Pi(x, y, z) = (x/z, y/z)$. Now (10) was assuming that the camera was placed at the origin, viewing along the z-axis. To handle cameras in general position and orientation, we pre-compose our projection with a change of coordinate systems,

$$\boldsymbol{x} = \Pi(R\boldsymbol{X} + \boldsymbol{t}), \tag{11}$$

where R is a 3×3 rotation matrix and $\boldsymbol{t} \in \mathbb{R}^3$ is the translation. Using homogeneous coordinates the projection equations (11) can also be written as

$$\begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \simeq [R\, \boldsymbol{t}] \begin{pmatrix} \boldsymbol{X} \\ 1 \end{pmatrix}, \tag{12}$$

The matrix $P = [R\ \boldsymbol{t}]$ is called the camera matrix and encodes the orientation and position (called the *extrinsic parameters*) of the camera.

To be able to accurately represent real world cameras, we also need to model the camera's *intrinsic parameters*. These control the transformation taking us from the coordinate system of the image plane to the pixel coordinate system found in the actual image. This is encoded in the 3×3 *calibration matrix* K which has the following form,

$$K = \begin{bmatrix} \alpha f & s & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{13}$$

The full camera matrix, taking 3D point coordinates to pixel coordinates, is then given by $P = K[R\ \boldsymbol{t}]$. The intrinsic parameters in the calibration matrix are

- Focal length f.
- Skew s.
- Aspect ratio α.
- Principal point (u_0, v_0).

Typically for consumer cameras we have zero-skew ($s = 0$), unit aspect ratio ($\alpha = 1$) and the principal point centered in the image. In general any 3×4 matrix represents a projective camera and if the first 3×3 block is full rank it can be decomposed as $P = K[R\ \boldsymbol{t}]$ using QR-factorization. Since the pinhole projection $\Pi : \mathbb{R}^3 \to \mathbb{R}^2$ is scale-invariant we typically only consider camera matrices up to scale.

3.2 Affine Cameras

Affine cameras are a special case of pinhole cameras where the camera matrix has the following form

$$P = \begin{bmatrix} A & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix}, \quad A \in \mathbb{R}^{2\times 3}, \boldsymbol{t} \in \mathbb{R}^2. \tag{14}$$

Affine cameras have the nice property that the projections are especially simple. The projection $\boldsymbol{x} \in \mathbb{R}^2$ of a 3D point $\boldsymbol{X} \in \mathbb{R}^3$ becomes

$$\boldsymbol{x} = \Pi\left(P\begin{pmatrix}\boldsymbol{X}\\1\end{pmatrix}\right) = \Pi\begin{pmatrix}A\boldsymbol{X}+\boldsymbol{t}\\1\end{pmatrix} = A\boldsymbol{X} + \boldsymbol{t}. \tag{15}$$

One interpretation of affine cameras is that the camera center is infinitely far away so that the projection rays become parallel.

In [211] Tomasi and Kanade showed that if we have multiple affine cameras viewing a scene, both the cameras and 3D points can be recovered using low rank factorization (see Section 4). Let $\boldsymbol{x}_{fk} \in \mathbb{R}^2$ be the k:th 3D point seen in the f:th image, i.e. $\boldsymbol{x}_{fk} = A_f \boldsymbol{X}_k + \boldsymbol{t}_f$. The basic idea in [211] is that if we collect the image points into a matrix $M \in \mathbb{R}^{2F\times N}$ it can factorized as follows,

$$M = \begin{bmatrix} \boldsymbol{x}_{11} & \dots & \boldsymbol{x}_{1N} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x}_{F1} & \dots & \boldsymbol{x}_{FN} \end{bmatrix} = \begin{bmatrix} A_1 & \boldsymbol{t}_1 \\ \vdots & \vdots \\ A_F & \boldsymbol{t}_F \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_1 & \dots & \boldsymbol{X}_N \\ 1 & \dots & 1 \end{bmatrix}. \tag{16}$$

Thus we can recover both cameras and 3D points by finding a low rank factorization of the matrix M. In this case we have $\operatorname{rank}(M) \leq 4$.

This was later extended by Bregler et al. [20] to the case of Non-Rigid Structure-from-Motion. The assumption was then that the 3D points in each frame $X_f \in \mathbb{R}^{3\times N}$ could be written as a linear combination of some shape basis $B_1, B_2, \dots, B_K \in \mathbb{R}^{3\times N}$. The projections in each image are

$$\boldsymbol{x}_{f\cdot} = A_f \left(\sum_{k=1}^{K} c_{fk} B_k \right) + \boldsymbol{t}\mathbb{1}^T \in \mathbb{R}^{2\times N}. \tag{17}$$

The matrix M containing the image coordinates can then be factorized as

$$M = \begin{bmatrix} c_{11}A_1 & c_{12}A_1 & \dots & c_{1K}A_1 & \boldsymbol{t} \\ c_{21}A_2 & c_{22}A_2 & \dots & c_{2K}A_2 & \boldsymbol{t} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{F1}A_F & c_{F2}A_F & \dots & c_{FK}A_F & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_K \\ \mathbb{1}^T \end{bmatrix}. \tag{18}$$

In this case the matrix M has $\operatorname{rank}(M) \leq 3K + 1$.

3.3 Camera Resectioning

Resectioning (or absolute pose estimation) is the problem of estimating the camera parameters from given 2D-3D correspondences. Estimation from n points is sometimes called the Perspective-n-Points problem (or PnP for short).

Each 2D-3D correspondence, $\boldsymbol{x} = (u, v, 1) \leftrightarrow \boldsymbol{X}$, in homogeneous coordinates places two linearly independent constraints on the camera matrix P,

$$\boldsymbol{x} \simeq P\boldsymbol{X} \tag{19}$$

or equivalently,

$$uP_3\boldsymbol{X} - P_1\boldsymbol{X} = 0, \quad \text{and} \quad vP_3\boldsymbol{X} - P_2\boldsymbol{X} = 0, \tag{20}$$

where P_k denotes the k:th row of the camera matrix. If we have at least 6 point correspondences we can solve linearly for P from (20) in a least squares sense. However, this does not minimize any meaningful reprojection error in the image. Ideally we would like to instead solve the following optimization problem

$$\min_P \sum_k \left\| \begin{pmatrix} u_i \\ v_i \end{pmatrix} - \Pi(P\boldsymbol{X}_i) \right\|^2 . \tag{21}$$

This is a non-linear, non-convex optimization problem and is in general quite difficult unless a good initial guess is known.

Minimal Problems

When we have the same number of degrees of freedom as we have constraints on the camera matrix, we have a so-called *minimal problem*. In this case we are (in general) able to find a camera pose where the projection equations are satisfied exactly. There are different minimal problems depending on the how many additional constraints we add on the intrinsic parameters of the camera. Here we list some common examples which have appeared in the computer vision literature:

- **P6P/P5.5P** – For uncalibrated cameras (i.e. no constraints on the intrinsic parameters) we have 11 degrees of freedom (12 elements of P minus scale). In this case we require at least 5.5 point correspondences. The half point correspondence can be obtained by simply ignoring one of the coordinates in the projection equations for one of the point correspondences. Since there are no additional constraints on P we can linearly estimate it from the projection equations (20). This is sometimes referred to as the Direct Linear Transformation (DLT).

- **P3P** – For completely calibrated cameras (known intrinsic parameters) the remaining camera pose has 6 degrees of freedom (3 in the rotation and 3 in the translation), making it minimal with only 3 point correspondences. There have been many proposed solvers for this problem, see e.g. [110, 70].

- **P3.5PF/P4PF** – If all intrinsic parameters are known except for the focal length, the problem is instead minimal with 3.5 point correspondences. There are both methods which use the minimal number of image points, by ignoring a coordinate, see e.g. [227] (also end of Chapter 5), and methods which use all four points but ignore some other constraints [26, 234].

- **P4PFR** – If we have four points we can also use the extra half point correspondence to estimate a radial distortion parameter. This problem was first solved by Josephsson and Byröd [102], and then later improved in Bujnak et al. [28] and Larsson et al. [137] (Chapter 5).

- **P4.5PFUV/P5PFUV** – If we wish to estimate both the focal length f and the principal point $\boldsymbol{x} = (u_0, v_0)$, the problem has 9 degrees of freedom is minimal with 4.5 point correspondences. The first solver for this problem was presented by Triggs [214] which presented a non-minimal method using all 5 point correspondences. In [138] (Chapter 6) a minimal solver using 4.5 points is presented.

3.4 Epipolar Geometry

In the previous section the camera pose was relative to some coordinate system fixed by the given 3D points. Now we instead consider the case where we only have the images of two cameras viewing the same scene, and no 3D information available. In this setting it is still possible to get information on the relative poses of the cameras.

Assume that we have chosen our coordinate system such that the camera matrices are $P_1 = [I\ \mathbf{0}]$ and $P_2 = [A\ \boldsymbol{t}]$. Let $\boldsymbol{x}$ be an image point in the first image written in homogeneous coordinates, i.e. $\boldsymbol{x} = (u, v, 1)^T$. This point can be the projection of any 3D point (in homogeneous coordinates) on the line

$$\boldsymbol{X}(s) = \begin{pmatrix} \boldsymbol{x} \\ s \end{pmatrix}, \quad s \in \mathbb{R}. \tag{22}$$

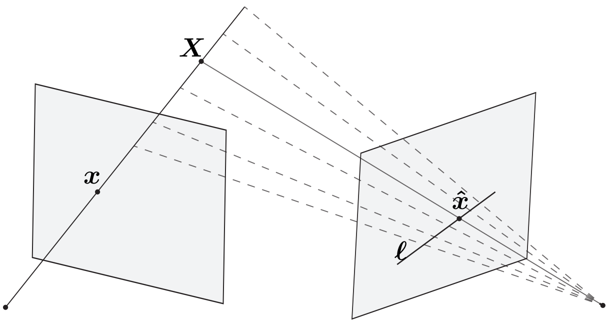


Figure 2: Epipolar geometry. Each image point in the first image backprojects onto a line in 3D, which in turn projects onto a 2D line in the second image.

In the second camera, this line projects onto the points $P_2\boldsymbol{X}(s) = A\boldsymbol{x} + s\boldsymbol{t}$. These image points all lie on the line $\boldsymbol{\ell} = \boldsymbol{t} \times A\boldsymbol{x}$, which is called the *epipolar line* corresponding to $\boldsymbol{x}$. This is illustrated in Figure 2. Now if $\hat{\boldsymbol{x}}$ is the image point in the second image corresponding to the same 3D point as $\boldsymbol{x}$, it must lie on this epipolar line, i.e. it must satisfy[2]

$$\hat{\boldsymbol{x}}^T(\boldsymbol{t} \times A\boldsymbol{x}) = \hat{\boldsymbol{x}}^T \left([\boldsymbol{t}]_\times A\right)\boldsymbol{x} = \hat{\boldsymbol{x}}^T F\boldsymbol{x} = 0. \tag{23}$$

This is called the *epipolar constraint*. The matrix $F = [\boldsymbol{t}]_\times A$ is called the *fundamental matrix* and it completely encodes the relative pose of the two cameras up to a projective transformation of the world coordinate system. The matrix F is of rank 2 and similarly to the camera matrices it is only determined up to scale. So each pair of cameras, $P_1 = [I\ \boldsymbol{0}]$ and $P_2 = [A\ \boldsymbol{t}]$, yields a fundamental matrix, and conversely for each fundamental matrix we can extract a pair of cameras (or rather a family of cameras) corresponding to this fundamental matrix.

For calibrated cameras we have that A is a rotation matrix, and the corresponding object is the *essential matrix* $E = [\boldsymbol{t}]_\times R$. The essential matrix encodes the calibrated relative pose up to an unknown similarity transform. In addition to the rank 2 constraint, the essential matrix is characterized by having two equal singular values. Algebraically this can be ensured using the *trace constraints* [197, 50, 157]

$$2EE^TE - \operatorname{tr}\left(EE^T\right)E = 0. \tag{24}$$

[2] $[\boldsymbol{t}]_\times$ denotes the 3×3 skew-symmetric matrix corresponding to the cross product with $\boldsymbol{t}$.

Together with the constraint, $\det(E) = 0$, this gives 10 equations of degree 3 in the elements of E. However it turns out that they only pose 3 independent constraints. For each essential matrix there exist four different camera pairs (up to similarity transforms) which are consistent with it.

Minimal Problems

The epipolar constraints (23) are linear in the elements of the fundamental matrix. Since the fundamental matrix has 9 elements, but is only determined up to scale, we can estimate it linearly from 8 or more point correspondences. This however does not enforce the rank 2 constraint.

- **F (7p)** – Using 7 point correspondences we have a two dimensional nullspace to the linear constraints in (23),

$$F = \alpha_1 F_1 + \alpha_2 F_2. \tag{25}$$

  Due the scale invariance we can fix $\alpha_2 = 1$. The rank constraint can be enforced by setting $\det(F) = 0$ which yields a cubic polynomial in α_1. (See e.g. [82]).

- **E (5p)** – The essential matrix has 5 degrees of freedom (3 rotation, 2 translation) so estimation becomes minimal using 5 point correspondences. In addition to the determinant constraint we must also enforce the trace constraints (24). For essential matrix estimation there have been many proposed solves, see e.g. [215, 170, 199, 147, 123].

In the computer vision literature there have been variations on these two problems, mostly dealing with partial calibration (e.g. unknown focal length), partial knowledge of extrinsics (e.g. known vertical direction) or modeling other non-linearities in the projections (e.g. radial distortion).

- **E+f, f+E+f (6p)** – If we assume unknown calibration matrices on the form $K = \text{diag}(f, f, 1)$ on either one or both of the cameras, the problem becomes minimal with 6 point correspondences. See e.g. [200, 123, 129, 145] for different approaches to this problem.

If we also model radial distortion there are a few different cases we can consider. We can have radial distortion on either one or both of the cameras, and the distortion parameter can either be shared or independent. Below are some references for the various cases which have been considered in the computer vision literature.

Problem	Points	References
E+λ	6	Kuang et al. [118]
λ+E+λ	6	Kukelova et al. [126]
E+f+λ	7	Kuang et al. [118], Kukelova et al. [129], Larsson et al. [142]
λ+f+E+f+λ	7	Jiang et al. [99]
F+λ	8	Kuang et al. [118]
λ+F+λ	8	Kukelova et al. [130]
λ_1+F+λ_2	9	Kukelova et al. [126]

4 Low Rank Approximation

Definition 1. *The **rank** of the matrix M is the maximum number of linearly independent columns.*

Proposition 2. *Each rank r matrix $X \in \mathbb{R}^{m\times n}$ can be factorized as*

$$X = AB, \tag{26}$$

where $A \in \mathbb{R}^{m\times r}$ and $B \in \mathbb{R}^{r\times n}$.

The factorization into rectangular matrices in Proposition 2 is however not unique, since for any invertible $G \in \mathbb{R}^{r\times r}$ we have $X = AB = (AG)(G^{-1}B)$ as another factorization.

Proposition 3. *For each matrix $X \in \mathbb{R}^{m\times n}$ there exist matrices*

- *$U \in \mathbb{R}^{m\times m}$ orthogonal,*
- *$V \in \mathbb{R}^{n\times n}$ orthogonal,*
- *$S \in \mathbb{R}^{m\times n}$ diagonal with non-negative elements,*

such that $X = USV^T$.

This is called the **Singular Value Decomposition** (SVD) of M. The diagonal elements of S, denoted by $\sigma_i(X)$, are usually sorted in descending order,

$$\sigma_1(X) \geq \sigma_2(X) \geq \cdots \geq \sigma_{\min(m,n)}(X) \geq 0. \tag{27}$$

These are called the **singular values** of X.

Proposition 4. *The rank is equal to the number of non-zero singular values.*

The following classical theorem by Eckart and Young [53] states that we can find the best rank r approximation by truncating the singular values.

Theorem 5. *The best rank r approximation of $X \in \mathbb{R}^{m \times n}$,*

$$X^\star = \arg\min_{Z} \|X - Z\|_F^2 \quad \textit{s.t.} \;\; \operatorname{rank}(Z) \leq r, \tag{28}$$

can be found by replacing all except for the first r singular values by zero, i.e.

$$X^\star = U \operatorname{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0) V^T, \tag{29}$$

where $X = USV^T$ is the SVD of X.

The following very useful inequality is due von Neumann [222, 161].

Theorem 6 (von Neuman's trace inequality). *Let $A, B \in \mathbb{R}^{m \times n}$ then*

$$|\operatorname{tr}(A^T B)| \leq \sum_{i=1}^{\min(m,n)} \sigma_i(A)\sigma_i(B), \tag{30}$$

with equality when A and B have SVD with the same U and V.

5 Convex Analysis

In this section we will list some basic facts and definitions from convex analysis which will be used in the thesis. For proofs and more details we refer the reader to Rockafellar [182] or Hiriart-Urrut and Lemaréchal [89].

Definition 7. *The set $\mathcal{C} \subset \mathbb{R}^n$ is called **convex** if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$ and $\lambda \in [0,1]$ we have*

$$\lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{y} \in \mathcal{C}. \tag{31}$$

So for any two points in the set, the line segment between them also belongs to the set. Figure 3 shows some examples. The analogous definition for functions follows.

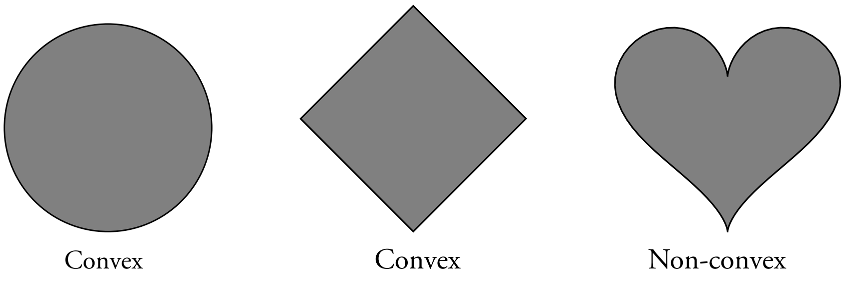


Figure 3: Examples of convex and non-convex sets.

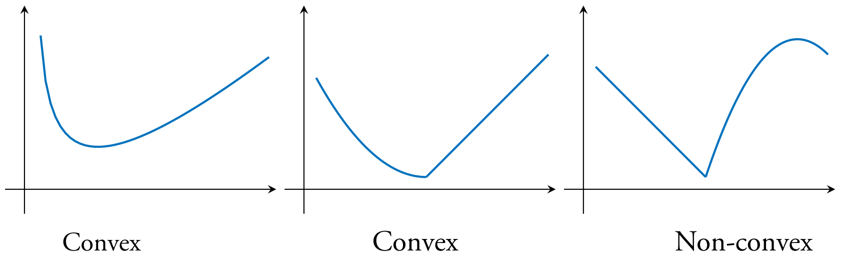


Figure 4: Examples of convex and non-convex functions.

Definition 8. *The function $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is called* ***convex*** *if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and $\lambda \in [0, 1]$ we have*

$$f(\lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1 - \lambda) f(\boldsymbol{y}) \tag{32}$$

In Figure 4 some examples of convex and non-convex functions are shown. The relationship between convex functions and sets is given by the epigraph.

Definition 9. *The* ***epigraph*** *of a function f is the set*

$$\operatorname{epi} f = \{(\boldsymbol{x}, u) \in \mathbb{R}^n \times \mathbb{R} \mid f(\boldsymbol{x}) \leq u\} \tag{33}$$

An alternative definition of convexity is that the epigraph of the function is a convex set.

Definition 10. *The function f is* ***closed*** *if the epigraph is a closed set.*

One of main attractions of convex function is the following property.

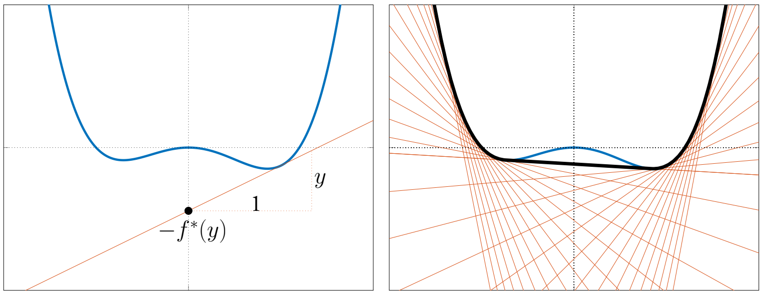


Figure 5: Geometrical interpretation of the conjugate function and the biconjugate (the convex envelope).

Proposition 11. *If f is a convex function, then any local minimizer of f is also a global minimizer.*

This makes optimization of convex functions much easier since we can infer global optimality from local information.

5.1 Fenchel Conjugacy

For each function in this section we assume that there is an affine function minorizing it.

Definition 12. *The **Fenchel conjugate** of a function f is the function f^* defined by*

$$f^*(\boldsymbol{y}) = \sup_{\boldsymbol{x}} \ \langle \boldsymbol{x}, \boldsymbol{y} \rangle - f(\boldsymbol{x}). \tag{34}$$

Even if the function f is not convex, the function f^* is always convex (as it is the pointwise supremum of affine functions). To interpret the conjugate f^* consider the following. If we fix $\boldsymbol{y}$ then

$$\forall \boldsymbol{x} \quad f^*(\boldsymbol{y}) \geq \langle \boldsymbol{x}, \boldsymbol{y} \rangle - f(\boldsymbol{x}) \quad \Leftrightarrow \quad \langle \boldsymbol{x}, \boldsymbol{y} \rangle - f^*(\boldsymbol{y}) \leq f(\boldsymbol{x}). \tag{35}$$

That is, the affine function given by $h(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle - f^*(\boldsymbol{y})$ is majorized by the function f. From the definition of f^* we can also see that this gives us the largest such affine function with the slope $\boldsymbol{y}$. This is illustrated to the left in Figure 5.

Definition 13. *The closed **convex envelope** of the function f, denoted by $\overline{\text{co}} f$, is the largest closed convex function which is majorized by f.*

Since convexity is preserved by pointwise supremum, we can write this as

$$\overline{\mathrm{co}} f(\boldsymbol{x}) = \sup \left\{ g(\boldsymbol{x}) \mid g \leq f,\ g \text{ closed convex} \right\}. \tag{36}$$

The following result allows us to express the convex envelope in terms of Fenchel conjugates.

Proposition 14. *The closed convex envelope is given by the biconjugate, i.e.*

$$\overline{co} f = (f^*)^* = f^{**}. \tag{37}$$

To gain some intuition why this is true, consider again the definition

$$f^{**}(\boldsymbol{x}) = \sup_{\boldsymbol{y}} \ \langle \boldsymbol{x}, \boldsymbol{y} \rangle - f^*(\boldsymbol{y}). \tag{38}$$

This is the pointwise supremum of all the supporting hyperplanes given by the first conjugate f^*. This is illustrated to the right in Figure 5, which shows an example of a one dimensional function (blue curve) and its envelope (black curve). Notice that due to convexity the envelope does not have any local minima (apart from the global ones).

6 Polynomial Equation Systems and Algebraic Geometry

We here give a brief overview of the topics from algebraic geometry which are pertinent to the thesis. We only state the basic definitions and results, for more complete statements and proofs, we recommend Cox et al. [44] and Cox et al. [43].

Definition 15. *A* ***monomial*** *is a finite product of variables and a* ***polynomial*** *is a finite linear combination of monomials.*

We denote the set of all polynomials with coefficients in the field $\mathbb{K}$ by $\mathbb{K}[X]$, where X is short for $X = (x_1, x_2, \ldots, x_n)$. Typically we just write X and let the variables in X be decided from context. In the thesis we will only consider $\mathbb{K} = \mathbb{C}$ and $\mathbb{K} = \mathbb{Z}_p$. However, many results hold in other fields as well.

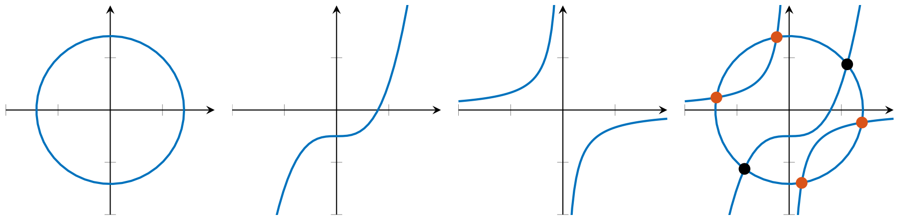
Definition 16. *The* ***ideal*** *I generated by the polynomials $f_1, \ldots, f_m \in \mathbb{K}[X]$ is*

$$I = \langle f_1, \ldots, f_m \rangle = \left\{ \sum_{k=1}^{m} h_k f_k \;\middle|\; h_k \in \mathbb{K}[X] \right\} \tag{39}$$

Definition 17. *The **affine variety** V associated with the ideal I is*

$$V(I) = \{\boldsymbol{x} \in \mathbb{K}^n \mid f(\boldsymbol{x}) = 0, \forall f \in I\} \tag{40}$$

Figure 6 shows some example of affine varieties in $\mathbb{C}[x, y]$.



$f_1 = x^2 + y^2 - 2,\quad f_2 = y^3 - x - 1/2,\quad f_3 = xy + 1/3$

Figure 6: Three polynomials f_1, f_2 and f_3 and the corresponding varieties. The last figure shows their (real) intersections. The affine variety $V(\langle f_1, f_2\rangle)$ contains two real points and is shown in black. The variety $V(\langle f_1, f_3\rangle)$ contains four points shown in red. The variety $V(\langle f_2, f_3\rangle)$ contains only complex points. Finally, $V(\langle f_1, f_2, f_3\rangle)$ is empty since there are no intersections.

6.1 Monomial Orderings

To perform division with multivariate polynomials we need to order the monomials. In the univariate case the degree gives us a natural ordering on the monomials,

$$1 < x < x^2 < x^3 < \cdots < x^n < \ldots. \tag{41}$$

However, for multivariate monomials things are not so easy. For example it is not clear if xyz should come before xy^2 or not. The following definition states the minimum requirements we should place on any ordering we choose.

Definition 18. *The relation $<$ is a **monomial ordering** if*

1. *$<$ is a total ordering on the monomials.*
2. *If f, g, α are monomials then $f < g \implies \alpha f < \alpha g$.*
3. *Every non-empty set of monomials has a smallest element under $<$.*

Below we show some examples of common monomial orderings. These orderings depend on having some underlying ordering of the variables. Typically we take this in alphabetical order, so $x > y > z$ and so on, and similarly $x_1 > x_2 > x_3 > \cdots > x_n$. Unless otherwise stated this will be the assumption for the rest of the thesis.

- Lexicographical ordering (Lex) – In Lex ordering we first compare the degree of the largest variable in the monomials. If this is equal we then consider the degrees of the second largest variable and so on.
- Graded Lex ordering (GLex) – In GLex we start by comparing the total degree of the monomials. Ties are then broken using lexicographical ordering.
- Graded Reverse Lex (GRevLex) – In GRevLex we similarly to GLex first compare the total degree of the monomials. However, ties are broken in a slightly different and more confusing manner. To break ties in GRevLex, we take the Lexicographical ordering with the variable order reversed, and then take the opposite result.

Below are some monomials ordered in the different orderings.

$$\begin{array}{rccccccc}
\text{Lex} & xz & > & x & > & y^2 & > & z^3 \\
\text{GLex} & z^3 & > & xz & > & y^2 & > & x \\
\text{GRevLex} & z^3 & > & y^2 & > & xz & > & x
\end{array}$$

Definition 19. *The* ***leading term*** $\mathrm{LT}(p)$ *of a polynomials p is the term containing the largest monomial.*

For example if we order monomials using Lexicographical order we have

$$\mathrm{LT}(3x^2 + 2y^3 + 1) = 3x^2, \tag{42}$$

but in GRevLex we instead get

$$\mathrm{LT}(3x^2 + 2y^3 + 1) = 2y^3. \tag{43}$$

6.2 The Division Algorithm and Gröbner Bases

Similarly to the univariate case there exist a division algorithm for multivariate polynomials.

Proposition 20. *Let $f \in \mathbb{K}[X]$ and $F = (f_1, \dots, f_m) \in \mathbb{K}[X]^m$ be an ordered m-tuple of polynomials. Then there exist $r, q_i \in \mathbb{K}[X]$ such that*

$$f = q_1 f_1 + \cdots + q_m f_m + r, \tag{44}$$

where either $r = 0$ or no monomials in r are divisible by any $\mathrm{LT}(f_i)$.

The division algorithm for multiple multivariate polynomials is shown below.

Algorithm 1: Division Algorithm

```
Input: $f$, $F = (f_1, \dots, f_m)$
Output: $r, q_1, \dots, q_m$
$r := q_1 := \cdots := q_m := 0$;
while $f \neq 0$ do
    for $i = 1, 2, \dots, m$ do
        if $\mathrm{LT}(f_i)$ divides $\mathrm{LT}(f)$ then
            $q_i := q_i + \frac{\mathrm{LT}(f)}{\mathrm{LT}(f_i)}$;
            $f := f - \frac{\mathrm{LT}(f)}{\mathrm{LT}(f_i)} f_i$;
            goto start of while;
        end
    end
    $r := r + \mathrm{LT}(f)$;
    $f := f - \mathrm{LT}(f)$;
end
```

Unlike in the univariate case, the remainder is not unique and depends on the order in which the polynomials are listed in the m-tuple F (and of course the monomial ordering used). This is illustrated in the following example.

Example 21. *Let $f = x^2 + x$ and consider division with $f_1 = x^2 + y$ and $f_2 = x^2 + y^2 - 1$. Dividing with (f_1, f_2) yields the remainder $x - y$ while dividing with (f_2, f_1) gives the remainder $x - y^2 + 1$.*

Definition 22. *For an ideal $I \subset \mathbb{K}[X]$, the set of polynomials $G = \{g_1, \dots, g_m\} \subset I$ is a* **Gröbner basis** *if for any $f \in I$ we have that $\mathrm{LT}(f)$ is divisible by some $\mathrm{LT}(g_i)$.*

It can be easily shown that any Gröbner basis also generates the ideal, i.e. $I = \langle g_1, \ldots, g_m \rangle$. Note again that this definition also depends on the monomial ordering used.

Proposition 23. *If $G = \{g_1, \ldots, g_m\}$ is a Gröbner basis then the remainder is unique when performing division with G regardless of the order in which the g_i are listed.*

Example 24. *Consider again f_1 and f_2 from Example 21. These polynomials are not a Gröbner basis for the ideal $I = \langle f_1, f_2 \rangle$. However, it can be verified that*

$$\begin{cases} g_1 = f_1 = x^2 + y \\ g_2 = f_2 - f_1 = y^2 - y - 1, \end{cases} \tag{45}$$

is a Gröbner basis for I w.r.t. GRevLex. Now dividing $f = x^2 + x$ with either (g_1, g_2) or (g_2, g_1) yields the remainder $x - y$.

Definition 25. *For a Gröbner basis $G = \{g_1, \ldots, g_m\}$, the **standard monomials** are the monomials of $\mathbb{K}[X]$ which are not divisible by any $\mathrm{LT}(g_i)$.*

The uniqueness of the remainder allows us to make the following definition.

Definition 26. *For a Gröbner basis G we define the **normal form** of $f \in \mathbb{K}[X]$ w.r.t. to G, denoted by $\overline{f}^G$, as the unique remainder after division with G, i.e.*

$$f = q_1 g_1 + \ldots q_m g_m + r \implies \overline{f}^G = r. \tag{46}$$

From the division algorithm we can see that the normal form $\overline{f}^G$ is always a linear combination of the standard monomials.

Definition 27. *Gröbner basis $G = \{g_1, \ldots, g_m\}$ is **reduced** if each g_i satisfies*

- *The coefficient for $\mathrm{LT}(g_i)$ is one.*
- *No monomial in g_i is divisible by any of the leading terms in $G \setminus \{g_i\}$.*

If we fix the monomial ordering, each non-empty ideal has a unique reduced Gröbner basis. In the rest of the thesis when we speak of Gröbner basis for an ideal we usually mean the reduced Gröbner basis.

6.3 Quotient Rings

Definition 28. *The **quotient ring** $\mathbb{K}[X]/I$ associated with the ideal I is the set of equivalence classes induced by the following relation,*

$$a \sim b \iff a \equiv b \mod I \iff a - b \in I. \tag{47}$$

For $p \in \mathbb{K}[X]$ we denote the class p belongs to as $[p]$, i.e.

$$[p] = \{q \in \mathbb{K}[X] \mid p - q \in I\}. \tag{48}$$

The structure of the quotient ring $\mathbb{K}[X]/I$ is closely connected to the affine variety $V(I)$ and is a key building block in the action matrix method (Section 7.3). Some of the properties of $\mathbb{K}[X]/I$ are summarized in the following proposition.

Proposition 29. *Let $I \subset \mathbb{K}[X]$ be an ideal with Gröbner basis G, then the quotient ring $\mathbb{K}[X]/I$ has the following properties:*

1. *The quotient ring $\mathbb{K}[X]/I$ is a $\mathbb{K}$-vector space.*
2. *The standard monomials (w.r.t. G) forms a (linear) basis for $\mathbb{K}[X]/I$.*
3. *If $\dim(V(I)) = 0$ then $\dim(\mathbb{K}[X]/I)$ is an upper bound for the number of solutions (i.e. distinct points in $V(I)$).*

Example 30. *Consider the ideal $I = \langle g_1, g_2 \rangle \subset \mathbb{C}[x, y]$ where*

$$g_1 = x^3 - 1, \quad g_2 = y^2 + x + 2. \tag{49}$$

Together the polynomials g_1 and g_2 forms a (reduced) Gröbner basis for I using GRevLex. The standard monomials w.r.t. G are the monomials not divisible by any of the leading terms (which in this case are x^3 and y^2), i.e.

$$\{1, x, y, x^2, xy, x^2y\}. \tag{50}$$

From this we can conclude that the dimension of the quotient ring $\mathbb{C}[x, y]/I$ is six and that there are at most six complex solutions to the system,

$$\begin{cases} g_1(x, y) = 0, \\ g_2(x, y) = 0. \end{cases} \tag{51}$$

The standard monomials and $\mathrm{LT}(g_i)$ are illustrated in Figure 7.

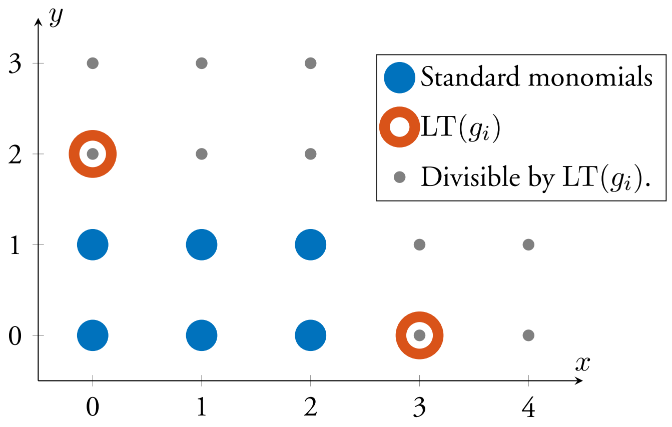


Figure 7: Monomials from Example 30. Each point (i, j) corresponds to the monomial $x^i y^j$.

6.4 Saturation

Sometimes we have systems which contain uninteresting or otherwise unwanted solutions. These can come from simplifications made during modeling, but can also be inherent to the original problem. Saturation provides a way to essentially factor these out of the ideal. One approach for integrating saturation into minimal solvers is presented in Chapter 2.

Definition 31. *The* ***saturation*** *of an ideal $I \subset \mathbb{K}[X]$ w.r.t. the polynomial $f_s \in \mathbb{K}[X]$ is defined as*[3]

$$\mathrm{Sat}(I, f_s) = \left\{p \mid \exists N \geq 0, f_s^N p \in I\right\}. \tag{52}$$

In other words, the saturated ideal consists of all polynomials such that when we multiply them with f_s sufficiently many times, we end up in the ideal I. The saturation allows us to essentially remove solutions which satisfy $f_s(\boldsymbol{x}) = 0$ from the affine variety $V(I)$. This is made formal in the following proposition.

Proposition 32. *If the field $\mathbb{K}$ is algebraically closed, the affine variety associated with the saturated ideal* $\mathrm{Sat}(I, f_s)$ *satisfies*

$$V(\mathrm{Sat}(I, f_s)) = \overline{V(I) \setminus V(\langle f_s \rangle)}, \tag{53}$$

[3]Note that while we only saturate with a single polynomial $f_s \in \mathbb{K}[X]$ here, the saturation can also be defined w.r.t. a polynomial ideal.

where the closure is taken in the Zariski topology (see [43]).

A another approach for removing unwanted solutions is the so-called *Rabinowitsch trick*, where an additional variable x_0 is added alongside the new equation

$$1 - x_0 f_s(\boldsymbol{x}) = 0. \tag{54}$$

This will remove any solution where $f_s(\boldsymbol{x}) = 0$. However we will show examples in the Chapter 2 where this approach leads to much larger polynomial solvers compared to working directly with the saturated ideal.

6.5 Syzygy Modules

Definition 33. *Let $\boldsymbol{f} \in \mathbb{K}[X]^m$, then the (**first) syzygy module** of $\boldsymbol{f}$ is*

$$\operatorname{Syz}(\boldsymbol{f}) = \operatorname{Syz}(f_1, \dots, f_m) = \left\{ \boldsymbol{s} \in \mathbb{K}[X]^m \;\middle|\; \sum_{k=1}^{m} s_k f_k = 0 \right\}. \tag{55}$$

The syzygy module encodes the ambiguity in representing polynomials in terms of the polynomials $f_1, \dots, f_m$, i.e. for any $p \in I$ we have

$$p = \sum_{k=1}^{m} h_k f_k = \sum_{k=1}^{m} (h_k + s_k) f_k, \quad \forall \boldsymbol{s} \in \operatorname{Syz}(f_1, \dots, f_m) \tag{56}$$

We can think of the syzygy module as analogous to the nullspace in linear equations, but for combinations with polynomial coefficients.

Definition 34. *The **monomials** in $\mathbb{K}[X]^m$ have the form $m\boldsymbol{e}_i$ where m is a monomial in $\mathbb{K}[X]$ and $\boldsymbol{e}_i$ is the m-tuple with 1 at the i:th position and zeros elsewhere.*

For each monomial ordering $<$ on $\mathbb{K}[X]$ we can create the following two monomial orderings in $\mathbb{K}[X]^m$, where we either first compare the monomial in $\mathbb{K}[X]$, or the position in the tuple.

- Term-Over-Position (TOP) – We have that $m_1\boldsymbol{e}_i < m_2\boldsymbol{e}_j$ if $m_1 < m_2$ or if $m_1 = m_2$ and $i < j$.
- Position-Over-Term (POT) – We have that $m_1\boldsymbol{e}_i < m_2\boldsymbol{e}_j$ if $i < j$ or if $i = j$ and $m_1 < m_2$.

Once the monomial orderings are fixed, the division algorithm extends naturally to $\mathbb{K}[X]^m$ and similarly we can also define Gröbner bases.

Example 35. *The tuple* $(2x^2 + y,\ x - 3y) \in \mathbb{C}[x, y]^2$ *contains the monomials*

$$(y, 0) < (0, y) < (0, x) < (x^2, 0). \tag{57}$$

where the ordering is from GRevLex-TOP.

7 Solving Systems of Polynomial Equations

Methods for solving systems of multivariate polynomials can in general be divided into two categories, iterative methods and algebraic methods.

The most basic iterative approaches are the root refinement methods, such as Newton's method, which aim to locally improve some solution guess. There are also variants with higher order of convergence, see e.g. [224, 90]. While these can converge very quickly they rely on good initialization. Another type of iterative approach are the homotopy continuation methods [148, 219]. In these methods the idea is to start with a simplified equation system which is then slowly perturbed into the equation system that we wish to solve. The solutions for the simplified system can then be tracked as they approach the solutions for the original system. Popular implementations are PHCpack [220] and HOM4PS [143]. One example where these have been used in computer vision is found in [184], where Salzmann used homotopy continuation methods for solving some sub-problems in an ADMM [18] optimization framework. However, for many problems in geometric computer vision these methods are too slow to be useful in practice.

For the algebraic methods there are two main approaches; resultants and methods based on Gröbner bases. In both these approaches the goal is typically to convert the problem into an eigenvalue problem (either using action matrices or reducing it to a univariate polynomial). The eigenvalue problem can then be solved using numerical methods.

7.1 Resultants

The classical resultant (sometimes called *Sylvester* resultant) gives a polynomial constraint on the coefficients of two univariate polynomials which is satisfied whenever they have a a shared root. For the two univariate polynomials,

$$\begin{cases} f(x) = a_r x^r + \cdots + a_1 x + a_0, \\ g(x) = b_s x^s + \cdots + b_1 x + b_0, \end{cases} \tag{58}$$

the resultant is the determinant of the $(r+s) \times (r+s)$ matrix as follows,

$$\text{Res}(f,g) = \det \begin{bmatrix} a_r & \cdots & \cdots & \cdots & a_0 & & \\ & \ddots & & & & \ddots & \\ & & a_r & \cdots & \cdots & \cdots & a_0 \\ b_s & \cdots & \cdots & b_0 & & & \\ & \ddots & & & \ddots & & \\ & & \ddots & & & \ddots & \\ & & & b_s & \cdots & \cdots & b_0 \end{bmatrix}. \tag{59}$$

The two polynomials f and g share a root exactly when $\text{Res}(f,g) = 0$.

If we have two polynomials in two unknowns, the resultant can be used to eliminate one of the variables by simply considering it as part of the coefficients. This is sometimes referred to as the *hidden variable trick*.

Example 36. *Let $f(x,y) = x^2 + xy + y^2 - 1$ and $g(x,y) = x + y - 1$. Hiding the variable x and constructing the resultants for $f(y)$ and $g(y)$ yields*

$$\textit{Res}_y(f,g) = \det \begin{bmatrix} 1 & x & x^2 - 1 \\ 1 & x - 1 & 0 \\ 0 & 1 & x - 1 \end{bmatrix} = x^2 - x = 0. \tag{60}$$

Thus either $x = 0$ or $x = 1$. The two solutions $(1,0)$ and $(0,1)$ can then be found by back-substitution.

Although it is possible to repeat this construction for more than two polynomials to eliminate multiple unknowns, the degree of the polynomials grows very quickly and spurious solutions are often introduced.

Macaulay [154] introduced a generalization of the resultant for multivariate polynomials and showed that it can be written as a quotient of determinants. There are other formulations of multivariate resultants such as Bezout resultants [55] and the Dixon resultants [105]. Using these multivariate resultants there are then different approaches for solving systems of polynomial equations.

In computer vision resultant-based methods have been used to solve some minimal problems, see e.g. [146, 145, 147, 214, 176]. However, as shown in

[171], these resultants sometimes suffer from poor numerical conditioning. In [124] Kukelova et al. presented a resultant-based method for converting the equation systems into polynomial eigenvalue problems [15]. This was recently extended by Heikkila [86] using techniques for constructing sparse resultants [205, 56]. For more details on resultants, resultant-based methods and how they have been used in computer vision see [177, 120].

While resultant-based methods work very well for some problem, the most common approach in geometric computer vision for solving systems of polynomial equations is based on Gröbner bases and action matrices. These are described in more detail in the remainder of this chapter.

7.2 Univariate Polynomials

We start by looking at the univariate case. Consider the polynomial

$$p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \in \mathbb{C}[x]. \tag{61}$$

For this polynomial we can create the corresponding companion matrix[4]

$$C_p = \begin{bmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_0 \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}. \tag{62}$$

This matrix has the polynomial p as its characteristic polynomial, i.e. the eigenvalues to C_p are exactly the roots of the polynomial p. So to solve the equation

$$p(x) = 0, \tag{63}$$

we can compute the eigenvalues of the matrix C_p. This offers an attractive option for root finding since there are good numerical methods for computing eigenvalues.

One interpretation of the matrix C_p comes from considering the quotient ring associated with the principal ideal $\langle p \rangle$. The quotient ring $\mathbb{C}[X]/\langle p \rangle$ is an n dimensional vector space over $\mathbb{C}$ with the monomials

$$\boldsymbol{b} = \{x^{n-1}, \dots, x^2, x, 1\} \tag{64}$$

[4]In some literature this is the transpose of the companion matrix, possibly with some permutation of the rows and columns.

as a basis. The matrix C_p then describes how the basis elements in $\boldsymbol{b}$ are mapped in $\mathbb{C}[X]/\langle p\rangle$ when we multiply with x,

$$x \begin{pmatrix} x^{n-1} \\ x^{n-2} \\ \vdots \\ x \\ 1 \end{pmatrix} = \begin{bmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_0 \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \begin{pmatrix} x^{n-1} \\ x^{n-2} \\ \vdots \\ x \\ 1 \end{pmatrix} \quad \text{mod } p(x). \tag{65}$$

The first row is simply that in $\mathbb{C}[X]/\langle p\rangle$ we have

$$[x^n] = [x^n - p(x)] = [-a_{n-1}x^{n-1} - a_{n-2}x^{n-2} \cdots - a_1 x - a_0], \tag{66}$$

while the remaining rows encode the trivial equations $[x \cdot x^k] = [x^{k+1}]$, where the basis elements are mapped to each other. In the next section the action matrix method is presented, which generalizes this idea to the multivariate case.

7.3 The Action Matrix Method

In this section we give a brief overview of the action matrix method (sometimes called eigenvalue method [44]) for finding the roots of systems of multivariate polynomials. Similarly to the companion matrix method, the main idea is to transform the system into an equivalent eigenvalue problem, for which there exist good numerical methods. For a more thorough review of the action matrix method and how it has been applied in computer vision we recommend [163], [122] and [34].

Assume that we have some set of multivariate polynomials,

$$\{f_1, f_2, \dots, f_m\} \subset \mathbb{K}[X], \tag{67}$$

and we seek to find their shared zeros. We let $I = \langle f_1, \dots, f_m\rangle$ and consider the operator $T_\alpha : \mathbb{K}[X]/I \to \mathbb{K}[X]/I$ which multiplies an element of the quotient ring with the fixed monomial[5] $\alpha \in \mathbb{K}[X]$, i.e.

$$T_\alpha\left[p(\boldsymbol{x})\right] = \left[\alpha(\boldsymbol{x})p(\boldsymbol{x})\right], \quad p \in \mathbb{K}[X]. \tag{68}$$

[5] For simplicity we take α as a monomial but the method also works for a general polynomial.

The operator T_α is a linear map and thus if we choose a (linear) basis

$$\boldsymbol{b} = \{b_1, b_2, \dots, b_K\} \tag{69}$$

for the quotient ring $\mathbb{K}[X]/I$, we can express T_α with a matrix, i.e.

$$\alpha \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{K-1} \\ b_K \end{pmatrix} = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1K} \\ m_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ m_{K1} & \dots & \dots & m_{KK} \end{bmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{K-1} \\ b_K \end{pmatrix} \mod I \tag{70}$$

or equivalently

$$[\alpha b_i] = \left[\textstyle\sum_j m_{ij} b_j\right] \quad i = 1, 2, \dots, K \quad \Leftrightarrow \quad [\alpha \boldsymbol{b}] = [M\boldsymbol{b}]\,. \tag{71}$$

The matrix $M = (m_{ij}) \in \mathbb{K}^{K \times K}$ is called the *action matrix* and is analogous to the companion matrix C_p in (65) in the univariate case. The polynomial α which we multiply with is called the *action* polynomial. Note that since $\boldsymbol{b}$ is a linear basis for $\mathbb{K}[X]/I$ the matrix M is uniquely determined.

Now similarly to the companion matrix, the eigenvalues of the matrix M is given by α evaluated at the solutions. To see this consider that for each solution $\boldsymbol{x} \in V$ we must have $M\boldsymbol{b}(\boldsymbol{x}) = \alpha(\boldsymbol{x})\boldsymbol{b}(\boldsymbol{x})$, since $M\boldsymbol{b} - \alpha\boldsymbol{b} \in I$ belongs to the ideal which vanishes on V. Thus if we evaluate α and $\boldsymbol{b}$ at the solutions we get eigenvalues and eigenvectors for the matrix M.

Note that we get one action matrix for each choice of α. However, typically it is sufficient to only consider a single action matrix, since it is often possible to recover the complete solutions either from the eigenvectors (which contains the monomials in $\boldsymbol{b}$ evaluated at the solutions) or by performing back-substitution in the original equations.

7.4 Finding Action Matrices

In the previous section we saw that once we have our action matrix it is possible to recover the solutions. In this section we focus on the problem of finding the action matrix. For the univariate case, the action matrix is easily obtained in the form of the companion matrix. However, for multivariate polynomials it requires a little bit more work.

We again let $I = \langle f_1, \ldots, f_m \rangle$ and assume that we have chosen the action polynomial $\alpha \in \mathbb{K}[X]$. Let m_{ij} denote the elements of the action matrix M, which satisfy

$$\alpha b_i - \sum_{j=1}^{K} m_{ij} b_j = 0 \mod I \quad i = 1, 2, \ldots, K, \tag{72}$$

for our choice of basis $\boldsymbol{b} = \{b_1, b_2, \ldots, b_K\}$.

We first note that if we have a Gröbner basis $G = \{g_1, \ldots, g_\ell\}$ for the ideal and choose the monomial basis $\boldsymbol{b}$ to be the standard monomials, then the action matrix can found by simply dividing αb_i with the Gröbner basis,

$$\overline{\alpha b_i}^G = \sum_{j=1}^{K} m_{ij} b_j \quad i = 1, 2, \ldots, K \tag{73}$$

since the remainders are spanned by the standard monomials.

We can use the Gröbner basis even if we choose some other basis $\hat{\boldsymbol{b}}$ for $\mathbb{K}[X]/I$. Assume that $\boldsymbol{b}$ are the standard monomials and M the action matrix w.r.t. $\boldsymbol{b}$. Then of course since $\mathbb{K}[X]/I$ is a vector space, the action matrix $\hat{M}$ w.r.t the basis $\hat{\boldsymbol{b}}$ can be obtained by a performing a simple change of basis. This basis change can be found by again dividing each $\hat{b}_i$ with the Gröbner basis, i.e. if

$$\overline{\hat{b}_i}^G = \sum_{j=1}^{K} s_{ij} b_j \quad i = 1, 2, \ldots, K \qquad \Longrightarrow \qquad [\hat{\boldsymbol{b}}] = [S\boldsymbol{b}] \tag{74}$$

then

$$[\alpha \boldsymbol{b}] = [M\boldsymbol{b}] \quad \Leftrightarrow \quad [\alpha \hat{\boldsymbol{b}}] = [\alpha S \boldsymbol{b}] = \left[(SMS^{-1})\hat{\boldsymbol{b}}\right] \tag{75}$$

where we can see that $\hat{M} = SMS^{-1}$ is the action matrix in the basis $\hat{\boldsymbol{b}}$.

One approach for computing the action matrix is to first compute a Gröbner basis for the ideal and then perform the required divisions. Given the generators of the ideal, a Gröbner basis can be computed using standard algorithms such as the Buchberger algorithm [24, 43]. However, in the polynomial solvers used in computer vision this is rarely the case for two reasons:

- Generic algorithms for computing Gröbner bases require some stopping criterion for knowing when the complete basis is computed, i.e. when all polynomials in the ideal reduce to zero. This is difficult for two reasons. Firstly, due to accumulated round-off error from using floating point arithmetic it can be difficult to determine if a polynomial is actually zero. Secondly, the coefficients in the polynomial systems are often computed from measurements in the images and there needs to be robustness to small errors in these. Even disregarding round-off error there will in some cases not be any exact solution to the system.

- For many applications runtime is very important. Often we need to solve many instances of the same problem very quickly. Thus we want to encode as much of the structure of the problem as possible into our specialized solvers, so that only things which are specific to the specific instance need to be computed at runtime.

Furthermore, it is not always necessary to find the complete Gröbner basis. Consider the example in Figure 7 on page 27. If we have the action $\alpha = x$ we will simply shift the monomials to the right in the figure when we multiply with α. In this case it turns our that we only need to divide with g_1 which has $\text{LT}(g_1) = x^3$ to form the action matrix corresponding to x.

The most common approach in computer vision literature (see Table 1.1 in Chapter 1 and the references therein) for finding the action matrices is to use a so-called *elimination template* which avoids explicit computation of the Gröbner basis and tries to directly encode the eliminations necessary for finding the action matrix.

Elimination Templates

The basic idea with elimination templates is that we don't necessarily need to divide with a Gröbner basis to find the action matrix. Any sequence of eliminations which allows to express each $[\alpha b_i]$ in terms of the basis $[b_1], [b_2], \dots, [b_k]$ will give us the action matrix. Expressing $[\alpha b_i]$ in the basis is equivalent to finding polynomials p_i in the ideal which are on the form,

$$p_i = \alpha b_i - \sum_{j=1}^{K} m_{ij} b_j \in I. \tag{76}$$

Note that since the action matrices are unique once the basis is chosen, it is sufficient to find any polynomial in the ideal which have the correct monomials (i.e. only αb_i and $\{b_1, \dots, b_K\}$). To find these polynomials we create an expanded set of equations by multiplying each f_j with some monomials. To goal is to multiply with sufficiently many monomials such that we can express the polynomials in (76) as linear combinations of the expanded set, i.e. it is possible to linearly eliminate all of the monomials except for those in (76).

To do this in practice (see e.g. [34, 120] more details) we write the expanded set of equations as $C\boldsymbol{X} = 0$, where the matrix C is called the *elimination template* and $\boldsymbol{X}$ is a vector of all the monomials occurring in the equations. By reordering the monomials we can rewrite this as

$$C\boldsymbol{X} = \begin{bmatrix} C_E & C_R & C_B \end{bmatrix} \begin{pmatrix} \boldsymbol{e} \\ \alpha\boldsymbol{b} \\ \boldsymbol{b} \end{pmatrix} = 0, \tag{77}$$

where we have grouped the monomials into *excessive* monomials $\boldsymbol{e}$, *reducible* monomials $\alpha\boldsymbol{b}$ and basis monomials $\boldsymbol{b}$. The excessive monomials are the monomials which we seek to eliminate, since they are not part of the target polynomials p_i. Note that here the monomials does not need to be ordered according to any monomial ordering.

Now we perform Gaussian elimination on (77) until we have eliminated all of the excessive monomials and get the following form on the lower part,

$$\begin{bmatrix} 0 & I & -M \end{bmatrix} \begin{pmatrix} \boldsymbol{e} \\ \alpha\boldsymbol{b} \\ \boldsymbol{b} \end{pmatrix} = 0, \tag{78}$$

from which we can extract the action matrix M. Note that in most cases some reducible monomials (i.e. αb_i) are directly available in the basis monomials, i.e. typically there will be some i and j such that $\alpha b_i = b_j$. It is not necessary to solve for these in the elimination template (these correspond to the trivial rows in M containing only zeros and ones). So with some abuse of notation, we let $\alpha\boldsymbol{b}$ in (77) and (78) denote the reducible monomials which are not in the basis and similarly for the corresponding rows in M.

One of the difficulties in constructing polynomial solvers is how to create the elimination template, i.e. choosing which monomials to multiply the equations with to generate the expanded set of equations (77). Since each extra equation corresponds to a row in the elimination template you typically want to find the smallest set possible which allow you to encode the necessary eliminations. In practice having a smaller elimination template does not only yield faster runtimes but often also better stability. Finding these small elimination templates is the focus of Chapter 1.

In this section we only considered using a single elimination template which finds all of the necessary polynomials by performing a single Gaussian elimination. However, it is also possible to use a sequence of elimination templates (see [120, 130]) and perform the eliminations in steps, similarly to how the reductions are done in the F4 algorithm [61]. It was noted in [120] that this approach sometimes yields faster solvers, however the numerical accuracy is often worse compared to using a single template.

We close this chapter with a small example.

Example 37. *Consider the following equation system,*

$$\begin{cases} f_1 = x^2 + y^2 - 1 = 0, \\ f_2 = x + ay + b = 0, \end{cases} \tag{79}$$

describing the intersection between the unit circle and a line determined by the parameters a and b.

For each choice of a and b there is (in general) two complex solutions and the monomials $\boldsymbol{b} = \{1, y\}$ forms a basis for the quotient ring $\mathbb{C}[X]/I$. This basis corresponds to the standard monomials w.r.t. a GRevLex Gröbner basis. If we choose y as our action monomial, the action matrix M will have the following form,

$$y \begin{pmatrix} y \\ 1 \end{pmatrix} = M \begin{pmatrix} y \\ 1 \end{pmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ 1 & 0 \end{bmatrix} \begin{pmatrix} y \\ 1 \end{pmatrix} \mod I. \tag{80}$$

The second row encodes the trivial equation $[y \cdot 1] = [1 \cdot y]$. From the matrix M we can find the solutions by computing the eigenvalues. This gives the action y evaluated at each of the solutions.

Now given an instance (i.e. values for a and b) we wish to recover the unknown m_{11} and m_{12}. To do this we search for polynomials in the ideal $I = \langle f_1, f_2 \rangle$ which have the same form as the first row in (80)*, i.e.*

$$p = y^2 - m_{11}y - m_{12} \in I. \tag{81}$$

To find the polynomial p we can use an elimination template. The assumption is that p can be written as a linear combination of an expanded set of our equations, in this case the set

$$\{f_1,\ xf_2,\ yf_2,\ f_2\} \tag{82}$$

is sufficient. Note that the coefficients in the expanded set only depends on the coefficients in the original equations (which in turn only depends on the parameters a and b). We then write the expanded set of equations in matrix form,

$$\begin{array}{c|cccccc|}
 & x^2 & xy & x & y^2 & y & 1 \\ \hline
f_1 & 1 & & & 1 & & -1 \\
xf_2 & 1 & a & b & & & \\
yf_2 & & 1 & & a & b & \\
f_2 & & & 1 & & a & b \\ \hline
\end{array}$$

where each row corresponds to one of the equations. In the notation from (77) we have in this example: $\mathbf{e} = \{x^2, xy, x\}$, $\alpha\mathbf{b} = \{y^2, y\}$ and $\mathbf{b} = \{y, 1\}$. Note that y is only included once even though it belongs to both $\alpha\mathbf{b}$ and $\mathbf{b}$.

Now if we perform Gaussian elimination on this template,

$$\begin{array}{|cccccc|}
x^2 & xy & x & y^2 & y & 1 \\ \hline
1 & & & & \bullet & \bullet \\
 & 1 & & & \bullet & \bullet \\
 & & 1 & & \bullet & \bullet \\
 & & & 1 & -m_{11} & -m_{12} \\ \hline
\end{array}$$

we can recover the polynomial p from the last row and construct the action matrix. In this example we only had a single monomial y^2 which we needed to express in the basis, but in general there can be multiple non-trivial rows in the action matrix.

Chapter 1

Building Polynomial Solvers

In this chapter we study the problem of automatically generating polynomial solvers for minimal problems. The main contribution is a new method for finding small elimination templates by making use of the syzygies (i.e. the polynomial relations) that exist between the original equations. These syzygies essentially parameterize the set of possible elimination templates for a given monomial basis.

We evaluate our method on a wide variety of problems from geometric computer vision and show improvement compared to both handcrafted and automatically generated solvers. Furthermore we apply our method on two previously unsolved relative orientation problems.

This chapter is based on the paper [134].

1.1 Introduction

The success of geometric computer vision is largely due to the ability to quickly and reliably solve systems of polynomial equations. This enables robust estimation schemes, such as RANSAC [64], which serve as outlier detection and initialization for non-linear optimization. To achieve robustness in RANSAC we need to solve many instances of the same minimal problem. While the different instances will have different coefficients, their structure will be essentially the same each time. This allows us to create code which is problem specific for solving the equation systems. The term *polynomial solver* (or sometimes *minimal solver*) refers to these specialized codes or procedures for solving such systems which take the instance specific coefficients as input, and output the solutions. Most polynomial solvers in computer vision recover the solutions via action matrices found with elimination templates (see Introduction chapter). The workflow for constructing these solvers was first introduced into computer vision by Stewénius [198] among others.

The idea is to consider a single instance of the problem, with the hope that the structure for this instance is representative for the whole family of problems. Fortunately it turns out that most instances are in fact representative, i.e. the set of coefficients which yields ideals with a different structure forms a nullset in the space of coefficients. To simplify computations this representative instance is usually chosen to have coefficients in some integer prime field $\mathbb{Z}_p$. Due to the exact arithmetic available in these fields, we can efficiently compute a Gröbner basis (in $\mathbb{Z}_p[X]$) for this instance. The assumption is now that the Gröbner basis for any other instance will, in general, have the same structure (i.e. same monomials and more importantly the same leading terms), but with different coefficients. This also means that the standard monomials for this Gröbner basis will, in general, be a linear basis in the quotient ring for any instance. So from this representative instance, we get the number of solutions we can expect (dimension of $\mathbb{Z}_p[X]/I$) and a candidate basis for the quotient ring. Then using these an elimination template for recovering the polynomials defining the action matrix is constructed. In the earlier works (e.g. [201]) these were very much handcrafted, however since then there been have some work proposing more systematic approaches for finding the elimination templates, see e.g. [122, 108, 168, 113].

1.1.1 Related Work

There has been a lot of work in computer vision to improve both the stability and speed of polynomial solvers. The computational cost of action matrix based polynomial solvers can be divided into three main parts.

1. **Computing the template coefficients**, i.e. finding the values which appear in the template. Since the template is usually formed by only multiplying with monomials, these coefficients are the same as the coefficients which occur in the equations. However, typically some pre-processing is required to find these coefficients from the input given to the solvers (usually some combination of image points and 3D points). For example, in the 5 point relative pose problem, we first need to compute a nullspace basis for the epipolar constraints which is used to parametrize the solutions. The coefficients which occur in the template are then polynomials in the elements

of the nullspace basis. Other than trying different parametrizations of the problem, it is difficult to speed up this part. In [129] Kukelova et al. propose a method based on elimination ideals for eliminating some of the unknowns and finding new constraints on the remaining variables, essentially finding a new set of equations for the problem. While this often results in a smaller elimination template, there is a trade-off where the new equations can become more complex and the coefficients are more expensive to compute.

2. **Linear elimination on the template** to recover the action matrix. The cost of the elimination is $\mathcal{O}(n^3)$, where n is the number of rows. Thus we aim to have as few rows as possible in our templates. There have been a few works which aim to minimize the number of rows in the template, e.g. [122, 108, 168, 113], and in this chapter we present another method for finding small elimination templates. With similar goals, Ask et al. [13] showed how to exploit certain symmetries available in some polynomial systems. This was later extended by Kuang et al. [119] and in Chapter 3 we present a further extension of these works. In Chapter 4 we present a method for selecting the quotient ring basis to minimize the size of the elimination template. In [121] Kukelova et al. presented a method which uses row and column permutations to make the elimination template into so-called Singly-Bordered Block-diagonal form. This structure can then be exploited to perform faster linear elimination when recovering the action matrix.

3. **Compute eigenvalues/vectors** of the action matrix. The size of the action matrix depends on the number of solutions to the problem. The number of solutions is typically fixed, however there are some examples where we can remove some uninteresting solutions (e.g. using saturations, see Chapter 2) or otherwise only consider a subset of the solutions (e.g. using symmetries [13, 119] and Chapter 3) to get a smaller eigenvalue problem. Sometimes the number of solutions depends on the parametrization. For example, each essential matrix corresponds to four different relative camera poses. If we were to directly parameterize the problem using the camera poses, we would get 40 solutions instead of 10 for essential matrix estimation. There has also been work on avoiding solving the full eigenvalue problem. In [29], Bujnak et al. proposed two different methods for efficiently extract-

ing univariate polynomials directly from the action matrix. The real roots of these polynomials can then be found efficiently using Sturm-sequences [91], instead of computing the full eigendecomposition of the action matrix.

There has also been some works which aim to improve the numerical stability of the polynomial solvers. In [34], Byröd et al. proposed methods to improve numerical accuracy by selecting the quotient ring basis at runtime using singular value decomposition (SVD) or by using QR factorization with column pivoting. This was later extended by Kuang et al. [113] where they optimize over the so-called *permissible* monomials (i.e. the set from which the basis is chosen from in [34]). However, the template size and numerical stability are often correlated, with smaller templates often being more stable.

For more references on how polynomial solvers are constructed and have been applied in computer vision see [34].

Related Work by Kukelova et al.

The work which is most closely related to the topic of this chapter is by Kukelova et al. [122], where the authors presented an automatic method for generating polynomial solvers. The automatic generator allows the user to specify a set of polynomial equations and then automatically generates stand-alone code for solving arbitrary problem instances. It has been widely adopted in the computer vision community and it has been used to solve several problems in geometric vision (see e.g. Table 1.1 and the references therein.) The solvers generated using [122] are built on the action matrix method (see Introduction chapter). Their generator follows the standard approach by first considering problem instances in $\mathbb{Z}_p[X]$. The method for creating polynomial solvers in [122] is summarized below.

1. Generate an instance of the equations in $\mathbb{Z}_p[X]$.

2. Using Macaulay2[75] (or some other computer algebra software) compute a Gröbner basis and the standard monomials in $\mathbb{Z}_p[X]$. The number of standard monomials gives an upper bound for the number of solutions. We also get which monomials should occur in the polynomials,

$$\alpha b_i - \sum_j m_{ij} b_j \in I \tag{1.1}$$

which define the action matrix.

3. An initial template is found by an iterative search which alternates between multiplying the equations with monomials ($x^\beta f_i$) and performing Gaussian elimination (in $\mathbb{Z}_p$) to check if all necessary polynomials are obtainable.

4. The template is then pruned by removing redundant polynomials. This is accomplished by iteratively removing a polynomial (or a subset of polynomials) from the template and performing Gaussian elimination to check if the template is still solvable.

The main drawback of the approach in [122] is that as the number of variables and equations grows the iterative search and pruning step can quickly become intractable. In this chapter we propose a new method for finding the elimination template in place of the iterative search used in [122]. We show on a large number of examples that our approach almost always produces smaller elimination templates and faster polynomial solvers.

1.2 Finding Elimination Templates

Now we will present our approach for finding elimination templates. It is similar to the method from Kukelova et al. [122] in that we also first find an initial template and then reduce it. However, we avoid the iterative search present in steps 3 and 4 above, and essentially have closed form solutions for the elimination templates.

As in [122] we start by generating an instance of the equation system in some prime field $\mathbb{Z}_p$. Using standard computer algebra software such as Macaulay2 [75], we compute a Gröbner basis for this instance,

$$G = \{g_1, g_2 \ldots, g_\ell\} \subset \mathbb{Z}_p[X], \tag{1.2}$$

and find the corresponding standard monomials,

$$\boldsymbol{b} = \{b_1, b_2 \ldots, b_K\} \subset \mathbb{Z}_p[X]. \tag{1.3}$$

Dividing αb_k with the Gröbner basis,

$$\alpha b_i = \sum_{j=1}^{\ell} q_{ij} g_j + \sum_{j=1}^{K} m_{ij} b_j, \quad q_{ij} \in \mathbb{Z}_p[X], \quad i = 1, 2, \ldots, K, \tag{1.4}$$

gives us the action matrix $M = (m_{ij}) \in \mathbb{Z}_p^{K \times K}$ for this particular integer instance, which we are not particularly interested in solving. However, by keeping track of how each of the Gröbner basis elements g_j are formed during their computation, we can express g_j in terms of the generators $f_1, \dots, f_m$,[1] i.e.

$$g_j = \sum_{k=1}^{m} c_{kj} f_k, \quad c_{kj} \in \mathbb{Z}_p[X], \quad j = 1, 2, \dots, \ell. \tag{1.5}$$

Inserting this into (1.4) we get

$$\alpha b_i - \sum_{j=1}^{K} m_{ij} b_j = \sum_{j=1}^{\ell} q_{ij} g_j = \sum_{j=1}^{\ell} q_{ij} \left(\sum_{k=1}^{m} c_{kj} f_k \right) = \sum_{j=1}^{m} h_{ij} f_j. \tag{1.6}$$

Thus we have expressed the polynomials $\alpha b_i - \sum_j m_{ij} b_j$ which define the action matrix in terms of our original equations.

The assumption is now that for different problem instances the polynomials h_{ij} in (1.6) will have the same form (i.e. same monomials) but possibly with different coefficients. These coefficients are essentially what is found by using the elimination template. To form the elimination template we multiply each f_j with all the monomials in each of the h_{ij}, so e.g. if $h_{34} = 3x^2 + 2y + 1$ we would include $\{x^2 f_4, y f_4, f_4\}$ in our elimination template.

In practice we found that elimination templates found using this method were often larger than those found by the automatic generator from Kukelova et al. [122] (see Section 1.4.1). In the next section we will present a heuristic which can simplify the polynomials h_{ij} and give a more compact elimination template.

1.2.1 Reducing the Expansion

In the previous section we showed how to obtain some polynomials $h_{ij} \in \mathbb{Z}_p[X]$ such that

$$\alpha b_i - \sum_{j=1}^{K} m_{ij} b_j = \sum_{j=1}^{m} h_{ij} f_j, \tag{1.7}$$

and that we could construct an elimination template from these. However, the polynomials h_{ij} are not unique, and in fact there are infinitely many choices of

[1]In Macaulay2 this can e.g. be accomplished using the `ChangeMatrix` option for the `gb` function.

h_{ij} which satisfy (1.7). This freedom is characterized by the syzygy module of the generators (see the Introduction chapter for more details), i.e.

$$\mathcal{S} = \mathrm{Syz}(f_1, \ldots, f_m) = \left\{ \boldsymbol{s} \in \mathbb{Z}_p[X]^m \;\middle|\; \sum_{k=1}^{m} s_k f_k = 0 \right\}. \tag{1.8}$$

For any $\boldsymbol{s} \in \mathcal{S}$ we have that

$$\alpha b_i - \sum_{j=1}^{K} m_{ij} b_j = \sum_{j=1}^{m} (h_{ij} + s_j) f_j, \tag{1.9}$$

and we can construct another valid elimination template using $\hat{h}_{ij} = h_{ij} + s_j$.

Ideally we would want to select the $\boldsymbol{s} \in \mathcal{S}$ which minimizes the size of the template, i.e. minimize the number of monomials in each h_{ij}. This is however a difficult combinatorial optimization problem. Instead we propose a simple heuristic which we experimentally show works very well.

We start by computing a Gröbner basis $G_{\mathcal{S}}$ for the syzygy module $\mathcal{S}$ using the monomial ordering GRevLex-TOP. Note that this Gröbner basis consists of m-tuples in $\mathbb{Z}_p[X]^m$. Then for each

$$\boldsymbol{h}_i = (h_{i1}, \ldots, h_{im}) \in \mathbb{Z}_p[X]^m, \tag{1.10}$$

we compute the normal forms $\overline{\boldsymbol{h}_i}^{G_{\mathcal{S}}}$ w.r.t. $G_{\mathcal{S}}$ and the template is constructed using these.

Since the monomial order first compares the degree, this promotes coefficients with low degree. One of the reasons why this heuristic works well is that having low degree is a reasonable proxy penalty for having few monomials, especially for problems with few variables.

Note that how we find the initial polynomials h_{ij} does not matter, as long as they satisfy (1.7). In fact, the approach described in Section 1.2 is not well defined since it depends on the method used for computing the Gröbner basis. This however does not matter since the normal forms are unique. It is possible to substitute other methods for finding the initial h_{ij} polynomials (such as the iterative method from [122]).

The proposed method is summarized below.

1. Generate an instance of the equations in $\mathbb{Z}_p[X]$.

2. Using Macaulay2[75] (or some other computer algebra software) compute a Gröbner basis G and the standard monomials in $\mathbb{Z}_p[X]$.

3. Using the Gröbner basis G find some polynomials $h_{ij} \in \mathbb{Z}_p[X]$ such that

$$\alpha b_i - \sum_{j=1}^{K} m_{ij} b_j = \sum_{j=1}^{m} h_{ij} f_j. \tag{1.11}$$

4. Compute a Gröbner basis $G_{\mathcal{S}}$ for the syzygy module

$$\mathcal{S} = \operatorname{Syz}(f_1, \ldots, f_m). \tag{1.12}$$

5. Reduce each $\boldsymbol{h}_i = (h_{i1}, \ldots, h_{im})$ into normal form, i.e. compute $\overline{\boldsymbol{h}_i}^{G_{\mathcal{S}}}$.

6. Construct elimination template from monomials occurring in $\overline{\boldsymbol{h}_i}^{G_{\mathcal{S}}}$.

To make this a little bit more concrete we now will illustrate our method on a small toy example.

Example 1.1. *Consider again the system from Example 37 on page 37,*

$$\begin{cases} f_1 = x^2 + y^2 - 1 = 0, \\ f_2 = x + ay + b = 0. \end{cases} \tag{1.13}$$

To create an elimination template for this problem we first consider a specific instance of the problem with integer coefficients,

$$\begin{cases} f_1 = x^2 + y^2 - 1 = 0, \\ f_2 = x + 2y - 2 = 0. \end{cases} \tag{1.14}$$

In this example we choose to work in $\mathbb{Z}_7$ for ease of presentation, however in practice a larger prime is typically used, e.g. $p = 30097$. For this instance of the ideal $I = \langle f_1, f_2 \rangle$ we can compute a Gröbner basis using Macaulay2 [75] and recover the action matrix. For this example we have

$$y \begin{pmatrix} y \\ 1 \end{pmatrix} = \begin{bmatrix} 3 & -2 \\ 1 & 0 \end{bmatrix} \begin{pmatrix} y \\ 1 \end{pmatrix} \mod I. \tag{1.15}$$

From the Gröbner basis we can also find a way to express the polynomial from the first row of (1.15),

$$p = y^2 - 3y + 2, \tag{1.16}$$

in the original equations, f_1 *and* f_2*, i.e. find some* h_1 *and* h_2 *such that* $p = h_1 f_1 + h_2 f_2$*. In this case, one possibility is*[2]

$$p = (-3x^2 + xy - x + 3)f_1 + (3x^3 + 3xy^2 + x - y + 1)f_2 \tag{1.17}$$

The coefficients h_1 *and* h_2 *are not unique. In this case the syzygy-module for* (f_1, f_2) *only consists of*

$$\mathcal{S} = \langle(-f_2, f_1)\rangle \subset \mathbb{Z}_7[X]^2 \tag{1.18}$$

thus its Gröbner basis is simply $G_{\mathcal{S}} = \{(-f_2, f_1)\}$*. Dividing* (h_1, h_2) *with* $(-f_2, f_1)$ *yields the remainder*

$$\overline{(h_1, h_2)}^{G_{\mathcal{S}}} = (3, -3x - y + 1)\,. \tag{1.19}$$

Thus we can also express p *as*

$$p = 3f_1 + (-3x - y + 1)f_2. \tag{1.20}$$

Now this only tells us how to express p *in* f_1 *and* f_2 *for this particular instance. The assumption is that in general, the polynomials* h_1 *and* h_2 *will have the same structure (i.e. same monomials), but possibly with different coefficients, i.e.*

$$p = c_0 f_1 + (c_1 x + c_2 y + c_3)f_2. \tag{1.21}$$

So to create the elimination template we choose the equations $\{f_1,\ xf_2,\ yf_2,\ f_2\}$ *(this is the template in Example 37 on page 37). The linear elimination on the template can then be seen as a way to recover the unknown coefficients* c_0, c_1, c_2 *and* c_3.

Note that using (1.17) *instead of* (1.20) *would still give us a working template. However since there are more monomials in the coefficients, the template would have been larger. More specifically the template would consist of*

$$\{x^2 f_1, xyf_1, xf_1, f_1, x^3 f_2, xy^2 f_2, xf_2, yf_2, f_2\}. \tag{1.22}$$

[2]For illustrative purposes we have chosen a slightly more complicated polynomial here than was originally returned by Macaulay2 [75]. For small problems the results are often already in normal form w.r.t. the syzygy module (see Table 1.1).

1.3 Implementation Details

We have written an automatic generator in MATLAB [94], which uses the technique described in Section 1.2 to find and reduce the elimination templates. The generator is similar to that of Kukelova et al. [122] in that it only requires the user to specify the problem equations and then generates stand-alone MATLAB or C++ code that can be used to solve arbitrary problem instances. The elimination template generation and reduction are performed in just a few lines of Macaulay2 [75]. The generator automatically identifies and exploits any variable aligned symmetries as described in [133] (see Chapter 3). In the implementation we do not perform any refinement on the elimination templates (except for the reduction step described in Section 1.2.1 and Section 1.3.2). It is possible that by using template optimization techniques such as those described in [113, 168] the results could be further improved. We have made the code for generating solvers publicly available. For most of the problems that we have tried, the solver generation time is quite small. The median running time for all the problems described in Table 1.1 of our automatic generator (executed on a standard desktop computer) is 5.7s.

1.3.1 Choosing Action Monomials

In the proposed approach the main computational cost is the computation of the Gröbner bases for the both the ideal and the syzygy module. These are both independent of the chosen action monomials. In the implementation we generate templates for all different variables as actions. We only compute the Gröbner bases once, and then perform the required divisions for each of the actions to create the templates. The action which gives the smallest elimination template is then returned.

1.3.2 Removing Redundant Columns and Rows

After finding the elimination template we optionally perform the following post-processing step to remove redundant columns and rows. This step is similar to a single iteration in the pruning step from the automatic generator from Kukelova et al. [122].

We start by reducing the matrix to row echelon form and find the pivot elements. For the excessive monomials we only keep those which correspond to the pivot elements. We also keep track of which rows were used in the elimination process. This allows us to also drop some redundant rows. However for almost all of the problems in Table 1.1, the templates after performing the syzygy-based reduction from Section 1.2.1 were already minimal in the sense that we were not able to remove any more rows during post-processing.

1.3.3 Improving Numerics with Generalized Eigenvalue Problems

In this section we present a simple trick which for some problems improves the numerical stability significantly. In the regular action matrix method we perform elimination on the template until we find the action matrix, i.e.

$$\begin{bmatrix} U & * & * \\ 0 & I & -M \end{bmatrix} \begin{pmatrix} \boldsymbol{e} \\ \alpha\boldsymbol{b} \\ \boldsymbol{b} \end{pmatrix} = 0 \implies \alpha\boldsymbol{b} - M\boldsymbol{b} = 0, \tag{1.23}$$

where U is some upper-triangular matrix. The idea now is to instead stop elimination once we have equations which only depend on $\alpha\boldsymbol{b}$ and $\boldsymbol{b}$, i.e.

$$\begin{bmatrix} U & * & * \\ 0 & A & B \end{bmatrix} \begin{pmatrix} \boldsymbol{e} \\ \alpha\boldsymbol{b} \\ \boldsymbol{b} \end{pmatrix} = 0 \implies \alpha A\boldsymbol{b} + B\boldsymbol{b} = 0. \tag{1.24}$$

This instead gives us a generalized eigenvalue problem to solve. Of course, it is now possible to get the action matrix as $M = -A^{-1}B$. However, for problems where the A matrix is poorly conditioned, we have found that it is sometimes better to solve the generalized eigenvalue problem directly instead of explicitly inverting the matrix A.

To use this method we construct our template in the same manner (since there are no problems with poor conditioning in $\mathbb{Z}_p$), but in the solver we only do eliminations until we have found the matrices A and B. Some special case has to be taken for basis elements where $\alpha b_i = b_j$ for some i and j. For these we need to add the corresponding rows to A and B. We have implemented this method in our automatic generator. In Section 2.4.2 in Chapter 2 (page 78) we show an example where we apply this method to a problem which is poorly conditioned.

1.4 Experimental Evaluation

1.4.1 Evaluation of the Reduction Step

In this section we evaluate the reduction step proposed in Section 1.2.1. Note that while the reduction does not guarantee that the template will be smaller we will show that this is often the case in practice.

To perform the evaluation we applied the automatic generator to a wide variety of minimal problems from the computer vision literature. Table 1.1 shows both the original template sizes as reported by the authors and the resulting templates from our proposed automatic generator. We have marked the templates with the fewest elements in bold. If the original solver used multiple templates they are marked with (§) and the largest template size is reported. Problems with variable-aligned symmetries (as described in [14, 119, 133] and Chapter 3) are marked with (†), these were automatically detected and removed by our generator.

It can be seen that in general the reduction step produces a smaller template. More interestingly is perhaps that the reduced template is often smaller than the template in the original paper. Note that many of the papers used the automatic generator from Kukelova et al. [122], indicated with (*) in the table. For many of the tested problems we get a large decrease in template size. For instance for the problem of estimating relative pose with a known rotation direction we go from a template of size 411×489 [187] to a template of size 39×43. This drastic reduction in size is due to a symmetry in the problem that was not used by the original authors, but automatically detected by the generator (see Chapter 3). If we assume that the time complexity of the solver is quadratic in the number of rows and linear in the number of columns this corresponds to a speed-up factor of 900.

1.4.2 Numerical Accuracy of the Solvers

The focus of the work in this chapter has been on generating fast solvers in an automatic manner, and not on numerical accuracy. However, for the proposed solvers to be usable we need them to behave in an acceptable way in terms of accuracy as well. In [168] it was reported that on a number of examples, that smaller templates yielded better numerical accuracy as well. We have verified that all of the generated solvers in Table 1.1 have reasonable numerical stability on randomly generated instances.

We will in this section give comparisons between our solvers and the original ones, for some specific problems. In terms of the underlying computer vision problem, there is often some meaningful statistical error that one can evaluate, e.g. the reprojection errors, but since our main contribution in this chapter is an automatic way of constructing solvers to systems of polynomials equations, we have opted to evaluate the actual equation residuals instead.

We compare on four different problems, where the original solvers were publicly available, namely *image stitching with unknown focal length and radial distortion* [31, 168] , the *optimal* PNP-method of Hesch et al. [87], the *optimal* PNP-method of Zheng et al. [232] and the *refractive* P5P solver from Haner et al. [77]. In Figure 1.1 the resulting error residual histograms are shown, for 5,000 runs of the solvers, with random input. The figure shows that for these problems we get similar accuracy as the original solvers while having smaller elimination templates. For the image stitching we have used the original solver presented in [31]. The smaller original template presented in Table 1.1 is from the paper of Naroditsky et al., but they reported almost identical numerics as the original solver [168].

Table 1.1: Comparison of elimination template sizes. Smallest template is highlighted in bold.

Problem	Original		Proposed generator	
	Author	template size	no reduction step	with reduction step
Rel. pose E+f 6pt (eliminated)	Kukelova et al. [129] (*)	$\mathbf{6 \times 15}$	$\mathbf{6 \times 15}$	$\mathbf{6 \times 15}$
Rel. pose E 5pt	Stewénius et al. [199]	$\mathbf{10 \times 20}$	$\mathbf{10 \times 20}$	$\mathbf{10 \times 20}$
Rel. pose F+k 8pt	Kuang et al. [118]	12×24	$\mathbf{11 \times 19}$	$\mathbf{11 \times 19}$
TDOA offset rank 2, 7,4 pts	Kuang et al. [115]	20×15	$\mathbf{14 \times 15}$	$\mathbf{14 \times 15}$
Rel. pose E+f 6pt	Bujnak et al. [27] (*)	$\mathbf{21 \times 30}$	$\mathbf{21 \times 30}$	$\mathbf{21 \times 30}$
Rel. pose f+E+f 6pt (eliminated)	Kukelova et al. [129] (*)	$\mathbf{21 \times 36}$	$\mathbf{21 \times 36}$	$\mathbf{21 \times 36}$
Abs. pose P3.5PF	Wu [227]	20×43	24×31	$\mathbf{20 \times 30}$
Rel. pose f+E+f 6pt	Kukelova et al. [122] (*)	$\mathbf{31 \times 46}$	$\mathbf{31 \times 46}$	$\mathbf{31 \times 46}$
Rel. pose k+F+k 8pt	Kukelova et al. [122] (*)	$\mathbf{32 \times 48}$	33×48	$\mathbf{32 \times 48}$
Rel. pose E+k 6pt	Kuang et al. [118]	48×70	$\mathbf{34 \times 60}$	$\mathbf{34 \times 60}$
TDOA offset rank 2, 5,6 pts	Kuang et al. [115]	105×83	74×68	$\mathbf{37 \times 42}$
Abs. pose Rolling shutter	Saurer et al. [186] (*)	48×56	50×55	$\mathbf{47 \times 55}$
Gen. abs. pose P4P+scale	Ventura et al. [218] (*)	48×56	50×55	$\mathbf{47 \times 55}$
Stitching fk+R+fk 3pt	Naroditsky et al. [168]	54×77	58×74	$\mathbf{48 \times 66}$
TDOA offset rank 3, 9,5 pts	Kuang et al. [115]	70×31	$\mathbf{30 \times 31}$	$\mathbf{30 \times 31}$
Rel. pose E+fk 7pt (elim.)	Kukelova et al. [129](*)	$\mathbf{51 \times 70}$	62×70	$\mathbf{51 \times 70}$
TDOA offset rank 3, 7,6 pts	Kuang et al. [115]	255×157	90×81	$\mathbf{52 \times 57}$
Gen. rel. pose 6pt	Stewénius et al. [201]	$\mathbf{60 \times 120}$	135×164	99×163
Abs. pose opt. PNP	Hesch et al. [87]	120×120	93×116	$\mathbf{88 \times 115}$
Triangulation satellite images	Zheng et al. [231] (*)	93×120	93×116	$\mathbf{88 \times 115}$
Abs. pose opt. PNP (Cayley)	Nakano [167] (*)	124×164	186×161	$\mathbf{118 \times 158}$
Abs. pose P4PFR	Bujnak et al. [28] (*)	136×152	$\mathbf{140 \times 144}$	140×156
Rel. pose unsynchronized	Albl et al. [6] (*)	194×210	279×175	$\mathbf{159 \times 175}$

Rel. pose k+E+k 6pt	Kukelova et al. [122] (*)	238×290	223×290	$\mathbf{149 \times 201}$
Rel. pose k_1+F+k_2 9pt	Kukelova et al. [122] (*)	179×203	355×292	$\mathbf{165 \times 189}$
Rel. pose E+fk 7pt	Kuang et al. [118]	200×231	249×214	$\mathbf{185 \times 204}$
Abs. pose weak PNP	Larsson et al. [133]	234×276	$322 \times 304^{\dagger}$	$\mathbf{138 \times 154}^{\dagger}$
Abs. pose weak PNP (2x2 sym)	Larsson et al. [133]	104×90	$42 \times 48^{\dagger}$	$\mathbf{26 \times 34}^{\dagger}$
Abs. pose rolling shutter R6P	Albl et al. [7] (*)	$\mathbf{196 \times 216}$	222×230	200×220
Abs. pose inlier opt. P4P w dir.	Svärm et al. [208]	280×252	230×229	$\mathbf{203 \times 231}$
Rel. pose E w dir. 3pt	Saurer et al. [187] (*)	411×489	287×324	$\mathbf{209 \times 217}$
Rel. pose E w dir. 3pt (using sym.)	-	-	$64 \times 68^{\dagger}$	$\mathbf{39 \times 43}^{\dagger}$
Abs. pose quivers	Kuang et al. [114]	372×386	240×244	$\mathbf{169 \times 189}$
Opt. three-view triang.	Byröd et al. [34]	$\mathbf{225 \times 209}^{\S}$		133×182
L_2 three-view triang. (Relaxed)	Kukelova et al. [131] (*)	274×305	321×328	$\mathbf{239 \times 270}$
Rel. pose E w angle 4pt	Li et al. [144] (*)	270×290	280×276	$\mathbf{266 \times 286}$
Abs. pose refractive P5P	Haner et al. [77]	280×399	410×480	$\mathbf{240 \times 256}$
Opt. vanishing points	Mirzaei et al. [162]	$2,860 \times 3,060$	263×296	$\mathbf{246 \times 296}$
TDOA offset rank 3, 6,8 pts	Kuang et al. [115]	$1,359 \times 754$	$1,359 \times 754$	$\mathbf{356 \times 345}$
Abs. pose opt. PNP	Zheng et al. [232] (*)	575×656	812×704	$\mathbf{521 \times 601}$
Abs. pose opt. PNP (using sym.)	Zheng et al. [232] (*)	348×376	$484 \times 408^{\dagger}$	$\mathbf{302 \times 342}^{\dagger}$
Abs. pose opt. inlier pose w dir 3pt	Svärm et al. [208]	$1,260 \times 1,278$	918×726	$\mathbf{544 \times 592}$
TOA (4,6)	Kuang et al. [116]	$966 \times 925^{\S}$	776×809	$\mathbf{493 \times 531}$
Abs. pose opt. PNP (quaternion)	Nakano [167] (*)	630×710	958×693	$\mathbf{604 \times 684}$
Abs. pose refractive P6PF	Haner et al. [77]	648×917	$2,196 \times 1,622^{\dagger}$	$\mathbf{636 \times 654}^{\dagger}$
Rel. pose fk+E+fk 7pt	Jiang et al. [99]	$886 \times 1,011$	$1,393 \times 1,237$	$\mathbf{581 \times 658}$
Dual-Receiver TDOA 5pt	Burgess et al. [30]	$2,625 \times 2,352$	$850 \times 1,167$	$\mathbf{455 \times 494}$
TOA (5,5)	Kuang et al. [116]	$1,386 \times 1,539^{\S}$	$1,217 \times 1,173$	$\mathbf{817 \times 859}$
Abs. pose opt. PNP (rot. matrix)	Nakano [167] (*)	$1,936 \times 1,976$	$1,698 \times 1,153$	$\mathbf{1,095 \times 1,135}$
L_2 three-view triang.	Kukelova et al. [131] (*)	$1,866 \times 1,975$	$2,541 \times 2,288$	$\mathbf{1,450 \times 1,538}$

1.4.3 Three Views with Known Intrinsic Parameters and Rotation Axis

In order to test our framework on a novel case, we turn our attention to the problem of pose estimation in three views when the rotation axis is known. Known rotation axis can occur in practice when there is additional information from sensors such as accelerometers which are common in modern cell phones. The problem is minimal when we observe one point and two lines. The problem has previously been studied for two views, where the minimal case is three points [68, 169, 209], and for two generalized cameras, where the minimal case is four points [85, 209]. Let the cameras be chosen as

$$P_1 = \begin{bmatrix} I & \mathbf{0} \end{bmatrix}, P_2 = \begin{bmatrix} R(\theta_2, \boldsymbol{v}) & \boldsymbol{t}_2 \end{bmatrix}, P_3 = \begin{bmatrix} R(\theta_3, \boldsymbol{v}) & \boldsymbol{t}_3 \end{bmatrix} \tag{1.25}$$

where $R(\theta, \boldsymbol{v})$ denotes rotation of θ radians around the rotation axis $\boldsymbol{v}$. Since the rotation axis is assumed to be known we can w.l.o.g. assume that $\boldsymbol{v} = \begin{pmatrix} 0, 1, 0 \end{pmatrix}^T$ by rotating the image coordinate systems. The two line constraints can be formulated as

$$\operatorname{rank}\left(\begin{bmatrix} P_1^T \boldsymbol{\ell}_{k1} & P_2^T \boldsymbol{\ell}_{k2} & P_3^T \boldsymbol{\ell}_{k3} \end{bmatrix}\right) = 2, \quad k = 1, 2 \tag{1.26}$$

where $\boldsymbol{\ell}_{ij}$ denotes line i observed in image j. The rank constraint can be encoded as a polynomial constraint by requiring each of the 3×3 submatrices to have zero determinant. Finally the single point correspondence gives us three sets of equations

$$\lambda_k \boldsymbol{x}_k = P_k \boldsymbol{X}, \quad k = 1, 2, 3 \tag{1.27}$$

or equivalently by eliminating λ_k

$$\boldsymbol{x}_k \times (P_k \boldsymbol{X}) = \mathbf{0}, \quad k = 1, 2, 3 \tag{1.28}$$

where $\boldsymbol{x}_k$ denotes the image point in image k.

Building an Efficient Two-step Solver

To build an efficient solver for this problem we start by noting that the top 3×3 submatrix in (1.26) only contains the rotation matrices. The determinant constraints are

$$\det\left(\begin{bmatrix} \boldsymbol{\ell}_{1k} & R_2^T \boldsymbol{\ell}_{2k} & R_3^T \boldsymbol{\ell}_{3k} \end{bmatrix}\right) = 0, \quad k = 1, 2 \tag{1.29}$$

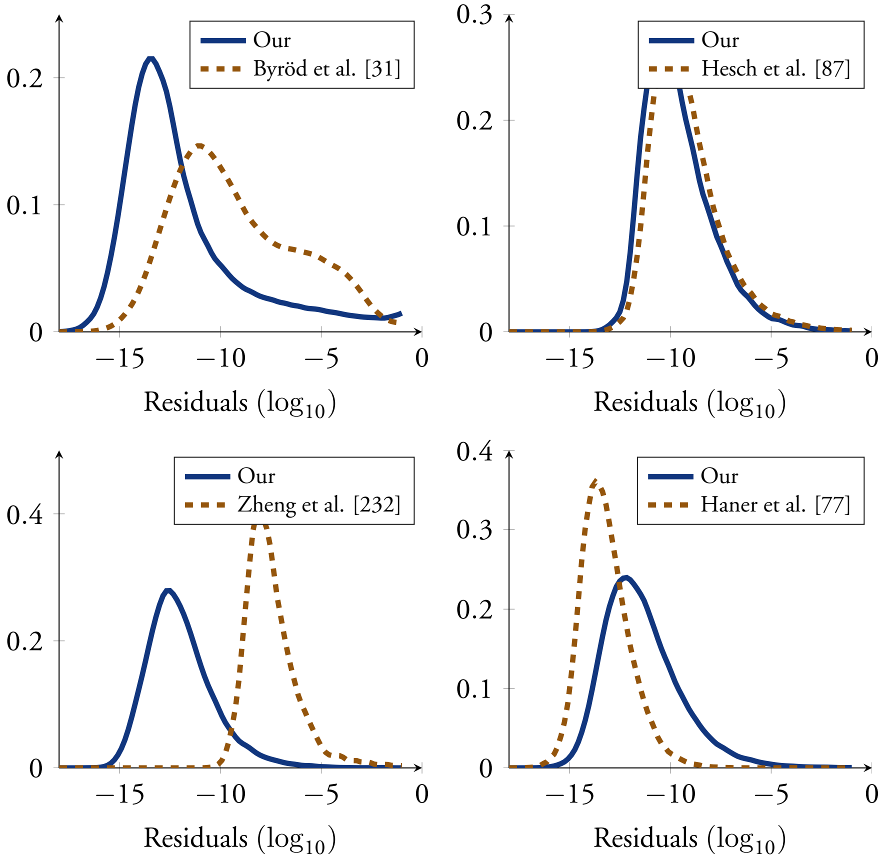


Figure 1.1: Kernel smoothed histograms of residual errors for 5,000 runs for: *image stitching with unknown focal length and radial distortion* [31, 168] , the *optimal PnP*-method of Hesch et al. [87], the *optimal PnP*-method of Zheng et al. [232] and the *refractive* P5P solver from Haner et al. [77].

which gives us two quadratic equations in the elements of R_2 and R_3. Since there are only two parameters in the rotations we can use these two equations to solve for the rotations independently from the rest of the variables.

Next we note that the constraint in (1.29) is invariant to the scale of the rotation matrices. We exploit this by parameterizing the rotations by non-unit quaternions (i.e. the Cayley parametrization), $\boldsymbol{q}_2 = \begin{pmatrix}1, & s_2\boldsymbol{v}^T\end{pmatrix}^T, \quad \boldsymbol{q}_3 = \begin{pmatrix}1, & s_3\boldsymbol{v}^T\end{pmatrix}^T$. This parameterization gives us two quartic equations in the unknowns s_2 and s_3. Fixing the first element of the quaternion introduces a degeneracy for any 180-degree rotation.

Using the proposed automatic generator we construct a solver for this system. The resulting template size is 12×20 and the system has 8 solutions. The solutions include two false solutions $s_2 = s_3 = \pm i$ that were introduced by the rotation parameterization. These can easily be discarded, and from the true solutions we can recover the correct rotations by rescaling each quaternion to unit length.

When the rotations are known we can use them to recover the translations $\boldsymbol{t}_2$ and $\boldsymbol{t}_3$. Using the point correspondence we can parameterize the two translations using the depths as

$$\boldsymbol{t}_k = \lambda_k \boldsymbol{x}_k - R_k \boldsymbol{X}, \quad k = 2, 3. \tag{1.30}$$

Since $P_1 = [I \ \mathbf{0}]$ we can select the scale such that $\boldsymbol{X} = \boldsymbol{x}_1$. Inserting (1.30) into the line constraints gives linear equations in the unknown λ_2 and λ_3, which allows us to solve for the translations. We evaluated the performance of the two-step solver on 10,000 random synthetic instances. Figure 1.2 shows the residuals for the rotation estimation and the distance from the recovered pose to the ground truth. In average solving for the rotations and finding all translations took less than one millisecond per instance. Figure 1.3 shows an example where we have used the minimal solver in a RANSAC framework to estimate the relative pose of three cameras. The 3D structure is found using DLT [81]. Note that this is the result without any bundle adjustment.

Importance of Good Parameterization

In the previous section we developed an efficient solver by choosing a clever parameterization of the problem. Finding a good parameterization is often one of the main difficulties when building polynomial solvers for minimal problems.

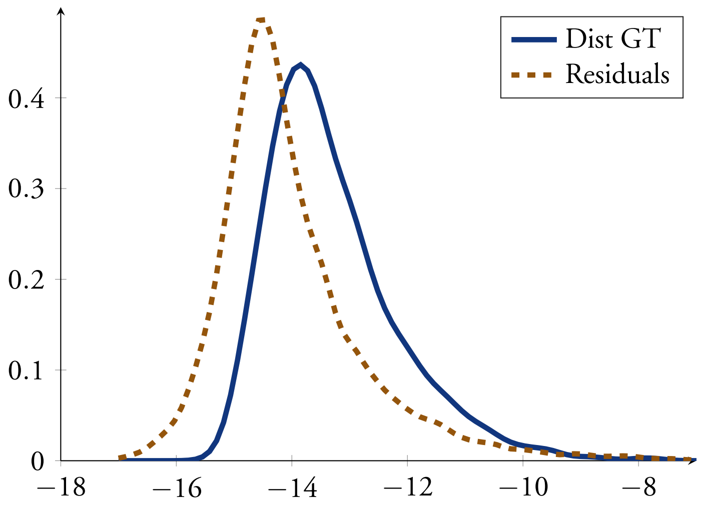


Figure 1.2: Results for 10,000 random synthetic instances for the rotation estimation in Section 1.4.3. The figure shows kernel smoothed distributions of both the residuals and the distance to the ground truth pose. The distance is defined as the maximum of $\|R_2 - R_2^{GT}\|_F$, $\|R_3 - R_3^{GT}\|_F$, $\|\boldsymbol{t}_2 - \boldsymbol{t}_2^{GT}\|_2$ and $\|\boldsymbol{t}_3 - \boldsymbol{t}_3^{GT}\|_2$.

Knowing which parameterization will yield good solvers is non-trivial and usually a trial and error approach is taken. Automatic tools such as the generator proposed in this chapter greatly speeds up this process and allows for much faster prototyping.

To illustrate the importance of good parameterization we show three alternative parameterizations for this problem.

1. First we fix the scale by setting $\boldsymbol{X} = \boldsymbol{x}_1$ and directly parameterize the two translations $\boldsymbol{t}_2 = \left(t_{21}, t_{22}, t_{23}\right)^T$ and $\boldsymbol{t}_3 = \left(t_{31}, t_{32}, t_{33}\right)^T$. The rotations are represented using unit quaternions, $\boldsymbol{q}_2 = (s_{21}, s_{22}\boldsymbol{v}^T)^T$ and $\boldsymbol{q}_3 = (s_{31}, s_{32}\boldsymbol{v}^T)^T$ with the additional constraints $\|\boldsymbol{q}_2\|^2 = \|\boldsymbol{q}_3\|^2 = 1$. This system has 24 solutions, however there exist two independent two-fold sign symmetries and by removing these we get the desired 6 solutions.

2. Next we remove the translations by parameterizing the depths λ_k in the 2nd and 3rd image, i.e. $\boldsymbol{t}_k = \lambda_k \boldsymbol{x}_k - R_k \boldsymbol{x}_1$. This reduces the number of unknowns to 6.

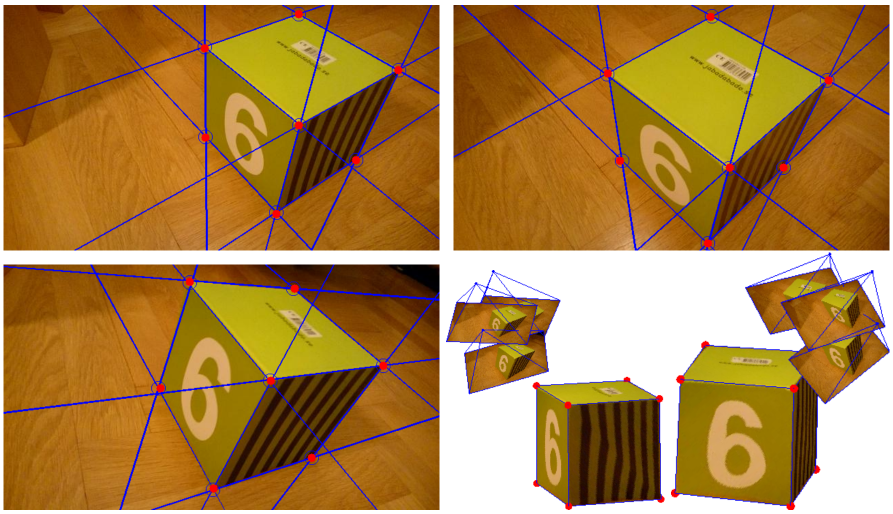


Figure 1.3: Example of three-view relative pose estimation with known rotation axis from one point and two lines. The image points and lines are shown in blue. The 3D-point reprojections are shown in red. The rightmost two images show the recovered 3D structure and poses.

3. Finally we tried using unit-quaternions when solving for the rotations using only the line constraints. The parameterization avoids the two false solutions introduced by the non-unit quaternion parameterization. However there still remains a two-fold symmetry.

The different parameterizations are summarized in Table 1.2 which also shows the template sizes and average runtimes for the solvers. Note that there is about a three orders of magnitude difference in runtime between the fastest and slowest solvers, which shows the need of finding a good parameterization.

1.4.4 Projective Reconstruction from Nine Lines in Three Views

In order to test the boundaries of our method, we have looked at a challenging unsolved minimal case. The classical problem of minimally estimating the geometry of three projective views of lines is an inherently difficult problem, [81, p. 413]. Whereas algorithms for projective reconstruction from points in three views have existed for a long time, [88, 179], there exists no practical method for the corresponding minimal problem using lines. The problem is minimal with either

Note	Unknowns	Additional equations	Template size	Runtime
Fixed scale by depth.	$s_{21}, s_{22}, s_{31}, s_{32}, \boldsymbol{t}_2, \boldsymbol{t}_3$	$\boldsymbol{X} = \boldsymbol{x}_1,\ \lVert\boldsymbol{q}_2\rVert^2 = \lVert\boldsymbol{q}_3\rVert^2 = 1$	$1,306 \times 1,261$	0.57s
Eliminated translations in the second and third image.	$s_{21}, s_{22}, s_{31}, s_{32}, \lambda_2, \lambda_3$	$\begin{cases} \boldsymbol{t}_k = \lambda_k \boldsymbol{x}_k - R_k \boldsymbol{X}, & k = 2,3 \\ \boldsymbol{X} = \boldsymbol{x}_1, \lVert\boldsymbol{q}_2\rVert^2 = \lVert\boldsymbol{q}_3\rVert^2 = 1 \end{cases}$	652×462	40ms
Unit quaternions.	$s_{21}, s_{22}, s_{31}, s_{32}$	$\lVert\boldsymbol{q}_2\rVert^2 = \lVert\boldsymbol{q}_3\rVert^2 = 1$	96×88	1.25ms
Non-unit quaternions.	s_2, s_3	$\boldsymbol{q}_k = \left(1,\ s_k \boldsymbol{v}^T\right)^T \quad k = 2,3$	8×16	0.1ms

Table 1.2: Comparison of different parameterizations for the one point two line problem from Section 1.4.3.

six points or nine lines. There are also algorithms for different minimal combinations of lines and points, cf. [176]. Linear algorithms have been developed for over-constrained solutions of at least 13 lines [78], and for combinations of lines and points [79], and there exist non-linear methods for the over-constrained cases of 10-12 lines [117]. We assume that we have three unknown uncalibrated cameras, viewing nine unknown lines in space. Each camera has eleven parameters, and each line has four degrees of freedom. In addition to this, we can only determine a solution up to a global projective coordinate system, with 15 parameters. Each viewed line in each camera gives two constraints on our parameters. Since $3 \cdot 11 + 4 \cdot 9 - 15 = 2 \cdot 3 \cdot 9$, this gives a minimal system. A major reason that the line case is much more difficult than the point case, is that the projective coordinate system can be efficiently parameterized with five points. With lines we use fewer lines, and we need to be very careful in order to avoid specialized situations, e.g. lines intersecting. We have experimented with a large number of parameterizations of our problem, and in the end we found that the following gave the most tractable solution. First of all we make coordinate changes in the images so that the first two lines in each image are represented by $\boldsymbol{\ell}_1 = (1,\, 0,\, 0)^T$ and $\boldsymbol{\ell}_2 = (0,\, 1,\, 0)^T$. This corresponds to the lines $x = 0$ and $y = 0$ respectively. We then make a projective coordinate change so that the first camera is given by $P_1 = [I\ \mathbf{0}]$. This fixes 11 of the 15 degrees of freedom, and also gives that the first 3D-line must lie on the plane $\mathbf{\Pi}_1 = P_1^T \boldsymbol{\ell}_1 = (1,\, 0,\, 0,\, 0)^T$ and the second line on the plane $\mathbf{\Pi}_2 = P_1^T \boldsymbol{\ell}_1 = (0,\, 1,\, 0,\, 0)^T$. This in turn fixes the first two lines up to two parameters each. We can now fix the final four degrees of freedom of our coordinate system by specifying two additional planes ($\mathbf{\Pi}'_1$ and $\mathbf{\Pi}'_2$) that the first two lines should lie on. (Assuming that a 3D-line is represented by two points $\boldsymbol{X}$ and $\boldsymbol{X}'$ this places two linear constraints on the coordinate change homography H, $(H\boldsymbol{X})^T \mathbf{\Pi}' = 0$ and $(H\boldsymbol{X}')^T \mathbf{\Pi}' = 0$ for each line). Choosing $\mathbf{\Pi}'_1 = (0,\, 1,\, 0,\, -1)^T$ and $\mathbf{\Pi}'_2 = (0,\, 0,\, 1,\, -1)^T$ gives that the two additional cameras can be written

$$P_2 = \begin{bmatrix} x_1 & 1 & 0 & -1 \\ 0 & x_2 & x_3 & -x_3 \\ x_4 & x_5 & x_6 & x_7 \end{bmatrix},\ P_3 = \begin{bmatrix} x_8 & x_9 & 0 & -x_9 \\ 0 & x_{10} & 1 & -1 \\ x_{11} & x_{12} & x_{13} & x_{14} \end{bmatrix}, \tag{1.31}$$

where $(x_1, \ldots, x_{14})$ are unknown parameters. We can now for each image-line triplet, $(\boldsymbol{\ell}_{i1}, \boldsymbol{\ell}_{i2}, \boldsymbol{\ell}_{i3}), i = 3, \ldots, 9$, construct the matrix

$$M_i = [P_1^T \boldsymbol{\ell}_{i1},\ P_2^T \boldsymbol{\ell}_{i2},\ P_3^T \boldsymbol{\ell}_{i3}]. \tag{1.32}$$

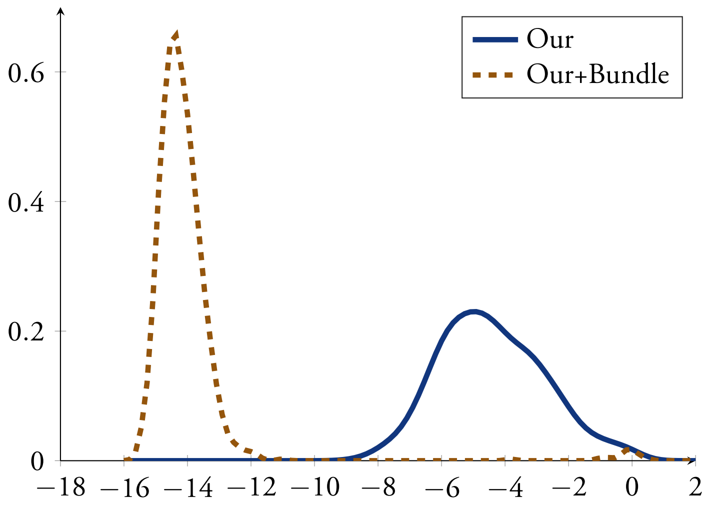


Figure 1.4: The distribution of the distance to the ground truth solution, for our initial nine-lines solver, and after non-linear refinement on the equation residuals.

If the three image lines are views of the same line, the corresponding planes should intersect, and hence rank $M_i = 2$, and all 3×3 sub-determinants of M_i should vanish. This gives three linearly independent second-degree polynomial constraints on $(x_1, \ldots, x_{14})$ for each i and in total 21 equations in the 14 parameters. In this parametrization the problem has 36 solutions. In [4] the ideal of the trifocal tensor was investigated. Here they show that the corresponding variety has degree 297. However it turns out that most of these solutions do not correspond to a valid three view camera geometry, and these degeneracies are not present in our parameterization.

We have used our automatic generator on the formulation in (1.31), resulting in a template with size $20,273 \times 14,281$ without the reduction step. The problem has a large number of variables, and the resulting equations contain a large syzygy-set, and we have not been able to calculate this in Macaulay2 entirely. We have run a partial reduction step, based on taking all possible combinations of four equations, and this leads to a template of size $16,278 \times 13,735$. To evaluate our generated solver we ran the following test. We generated 1000 random synthetic instances. We then ran our solver on the corresponding data, and compared the closest of the 36 solutions to the ground truth solution. The resulting histogram is shown in Figure 1.4. It also shows the difference to the ground truth solution after non-linear refinement of our solution. Our final elimination template is large, but very sparse. The complete solver has an average runtime of 13 s in MATLAB on a standard desktop computer (Intel I7-3930K 64 GB ram).

1.5 Conclusion

In this chapter we have presented a new method for finding elimination templates. The main contribution was to utilize the Gröbner bases for both the ideal and the syzygy module. The module encapsulates the ambiguity in representing the polynomials that are needed for constructing the action matrix. We have achieved state-of-the-art results by finding the normal form w.r.t. the syzygy module, but it is possible that a more advanced search over the syzygies would yield even better results, and this is an interesting venue for further work.

Chapter 2

Solvers for Saturated Ideals

In this chapter we present a new method for creating polynomial solvers for problems where a (possibly infinite) subset of the solutions are undesirable or uninteresting. These solutions typically arise from simplifications made during modeling, but can also come from degeneracies which are inherent to the geometry of the original problem. The proposed approach extends the standard action matrix method to saturated ideals. This allows us to add constraints that some polynomials should be non-zero on the solutions. This does not only offer the possibility of improved performance by removing superfluous solutions, but makes a larger class of problems tractable. Previously, problems with infinitely many solutions could not be solved directly using the action matrix method as it requires a zero-dimensional ideal. In contrast we only require that after removing the unwanted solutions only finitely many remain.

State-of-the-art solvers for problems in computer vision involving saturated ideals, e.g. [202, 116, 193, 32], were implemented by manually constructing equations from the saturated ideal as a preprocessing step and then applying the standard approach for constructing the action matrix. These methods are essentially hand-crafted and problem specific. In contrast our approach is entirely generic and we have extended the automatic solver generator presented in Chapter 1 to also handle saturation.

This chapter is based on preliminary results presented in Larsson et al. [135].

2.1 Action Matrices in Saturated Ideals

Let $I = \langle f_1, \ldots, f_m \rangle \subset \mathbb{K}[X]$ be an ideal such that

$$J = \mathrm{Sat}(I, f_s) = \{p \mid \exists N \geq 0,\ f_s^N p \in I\} \tag{2.1}$$

is zero-dimensional and let $\{b_k\}_{k=1}^d$ be a linear basis for the quotient ring $\mathbb{K}[X]/J$.

The goal is now to lift the properties we need from the saturated ideal J into the original ideal I. This will allow us to create an elimination template directly from our original equations which can be used to find the action matrix from the saturated ideal.

Lemma 2.1. *For each $N \geq 0$ the set $\{f_s^N b_k\}_{k=1}^d$ is linearly independent in the quotient ring $\mathbb{K}[X]/I$.*

Proof. Assume otherwise. Then there exist some $c_i \in \mathbb{K}$, not all zero, such that $\sum_i c_i f_s^N b_i \in I$. From the definition of the saturation we get

$$f_s^N \left(\textstyle\sum_i c_i b_i\right) \in I \implies \textstyle\sum_i c_i b_i \in J, \tag{2.2}$$

which is a contradiction. □

For $N \geq 0$ define[1]

$$S_N = \Big[\mathrm{span}_{\mathbb{K}}\ \{f_s^N b_k\}_{k=1}^d\Big]_I \subset \mathbb{K}[X]/I. \tag{2.3}$$

From Lemma 2.1 we know that S_N forms an d-dimensional subspace in $\mathbb{K}[X]/I$.

Lemma 2.2. *For each $\alpha \in \mathbb{K}[X]$ there exists $N \geq 0$ such that*

$$[\alpha p]_I \in S_N, \quad \forall p \in S_N, \tag{2.4}$$

i.e. S_N is stable under multiplication with α.

Proof. Let $\alpha \in \mathbb{K}[X]$ and consider $[\alpha b_k]_J$ in $\mathbb{K}[X]/J$. Since $\{b_k\}_{k=1}^d$ spans $\mathbb{K}[X]/J$ there exist $m_{ij} \in \mathbb{K}$ such that

$$[\alpha b_i]_J = \Big[\textstyle\sum_j m_{ij} b_j\Big]_J \Leftrightarrow p_i := \alpha b_i - \sum_j m_{ij} b_j \in J. \tag{2.5}$$

[1]To clarify which quotient ring the class in taken in, we add the ideal as subscript here.

By definition there exist some $N_i \geq 0$ such that $f_s^{N_i} p_i \in I$. Take some $N \geq N_i$ for all i, then in $\mathbb{K}[X]/I$ we have

$$f_s^N p_i \in I \;\Leftrightarrow\; \left[\alpha f_s^N b_i\right]_I = \left[\textstyle\sum_j m_{ij} f_s^N b_j\right]_I . \tag{2.6}$$

Now for any $p \in S_N$ we have as $[p]_I = \left[\sum c_i f_s^N b_i\right]_I$. Applying (2.6) we get

$$[\alpha p]_I = \left[\textstyle\sum_i c_i \alpha f_s^N b_i\right]_I = \left[\textstyle\sum_i \sum_j c_i m_{ij} f_s^N b_j\right]_I \in S_N, \tag{2.7}$$

which proves the lemma. □

The following theorem will show a useful relationship between the two multiplication operators[2]

$$T_\alpha^I : \mathbb{K}[X]/I \to \mathbb{K}[X]/I \quad \text{and} \quad T_\alpha^J : \mathbb{K}[X]/J \to \mathbb{K}[X]/J. \tag{2.8}$$

Theorem 2.3. *For large enough $N \geq 0$, the action matrices corresponding to $T_\alpha^I\big|_{S_N}$ and T_α^J are the same with respect to the basis $\{f_s^N b_k\}_{k=1}^d$ and $\{b_k\}_{k=1}^d$ respectively.*

Proof. While $\mathbb{K}[X]/I$ may be infinite dimensional as a vector space, we know from Lemma 2.2 that $\operatorname{Im} T_\alpha^I\big|_{S_N} \subset S_N$ for large enough N. The rest of the proof follows immediately by noting that $M_\alpha = (m_{ij})$ from (2.5) and (2.6) is indeed the action matrix for both mappings. □

To gain some intuition why this works consider the action matrix $M_\alpha = (m_{ij})$ corresponding to $T_\alpha^I\big|_{S_N}$, i.e.

$$\begin{bmatrix} & & \\ & M_\alpha & \\ & & \end{bmatrix} \begin{pmatrix} \\ f_s^N \boldsymbol{b} \\ \\ \end{pmatrix} = \alpha \begin{pmatrix} \\ f_s^N \boldsymbol{b} \\ \\ \end{pmatrix} \mod I. \tag{2.9}$$

Note that while this equation is satisfied for all $\boldsymbol{x} \in V(I)$, any solution where $f_s(\boldsymbol{x}) = 0$ will correspond to a zero vector and not an eigenvector. Thus by computing the eigenvectors of M_α it is possible to recover only the solutions in the saturated ideal.

[2]Recall that T_α is the operator which multiplies with α, i.e. $T_\alpha[p(\boldsymbol{x})] = [\alpha(\boldsymbol{x})p(\boldsymbol{x})]$.

2.2 Building Solvers with Saturation

To apply the theory presented in the previous section in practice, we extend the method for automatic solver generation from Chapter 1. The steps taken are outlined below.

1. Generate an instance of the problem with coefficients in some prime field $\mathbb{Z}_p$. This allows for efficient and accurate computations.
2. Compute the saturated ideal
$$J = \text{Sat}(I, f_s) \tag{2.10}$$
and find a linear basis $\{b_k\}_{k=1}^d$ for $\mathbb{Z}_p[X]/J$.
3. Form the polynomials used in the action matrix, i.e.
$$p_i = \alpha b_i - \textstyle\sum_j m_{ij} b_j \in J. \tag{2.11}$$
4. By iteration find the smallest $N \geq 0$ such that
$$f_s^N p_i \in I \quad \forall i. \tag{2.12}$$
5. Using the method described in Chapter 1, find $h_{ij} \in \mathbb{Z}_p[X]$ such that
$$f_s^N p_i = \textstyle\sum_j h_{ij} f_j \tag{2.13}$$
and create the elimination template from these.

Note that we essentially take the same steps as in Chapter 1, but we instead form the polynomials p_i in the saturated ideal. Then we lift both the basis $\{b_k\}_{k=1}^d$ and the polynomials p_i into the original ideal I by multiplying with f_s^N. Each of the steps listed above can be efficiently carried out in algebraic geometry software such as Macaulay2 [75].

In the implementation we restrict ourselves to saturating a single monomial. This is however not very limiting, since if we instead wish to saturate some polynomial $f(\boldsymbol{x})$ we can introduce a new variable x_0 and the equation $x_0 - f(\boldsymbol{x}) = 0$. Saturating x_0 will then be equivalent to saturating $f(\boldsymbol{x})$ in the original formulation. Note that this equation is much simpler compared to using the *Rabinowitsch trick* (see Introduction chapter),
$$x_0 f(\boldsymbol{x}) = 1, \tag{2.14}$$

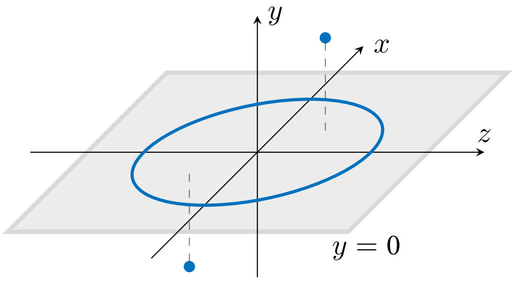


Figure 2.1: Solutions to the polynomial system in (2.16). The solution set consists of two points and a circle contained in the xz-plane. Saturating the ideal with y removes any solution where $y = 0$.

since the new variable x_0 appears linearly and is easy to eliminate.

In practice we found that for problems where we need to saturate a general polynomial $f(\boldsymbol{x})$, the elimination templates often become much smaller when the extra variable x_0 is not present in the quotient ring basis $\{b_k\}_{k=1}^{d}$. This can always be accomplished by choosing a monomial order where any monomial containing x_0 is greater than any monomial without x_0. If this is the case then clearly $\mathrm{LT}(x_0 - f(\boldsymbol{x})) = x_0$ and the standard monomials for the Gröbner basis will only contain monomials without x_0.

2.2.1 Toy Example

We will now show an overview of the steps taken to construct a polynomial solver for a toy example.

Example 2.4. *Consider the following system of equations.*

$$\begin{cases} f_1 = c_0x^2 + c_1y^2 + c_2z^2 + c_3 = 0, \\ f_2 = c_0x^2 + c_4xy + c_2z^2 + c_3 = 0, \\ f_3 = c_0x^2 + c_5yz + c_2z^2 + c_3 = 0, \end{cases} \tag{2.15}$$

where $c_0, c_1, \ldots, c_5 \in \mathbb{C}$ are constants. Note that while three quadratic equations in three variables in general have eight solutions, it is easy to see that this system becomes degenerate for $y = 0$, and the standard action matrix method breaks down.

To construct a polynomial solver for this problem we again start by considering an instance of the system where the coefficients are in $\mathbb{Z}_7$, e.g.

$$\begin{cases} f_1 = x^2 + y^2 + z^2 - 1 = 0, \\ f_2 = x^2 + 2xy + z^2 - 1 = 0, \\ f_3 = x^2 + 2yz + z^2 - 1 = 0. \end{cases} \tag{2.16}$$

Figure 2.1 shows the solution set (in $\mathbb{C}$) for these equations. For this particular instance we can use algebraic geometry software (e.g. [75]) to compute the saturation w.r.t. y.

$$J = \operatorname{Sat}(I, y) = \langle y - 2z,\ x - z,\ z^2 + 1 \rangle. \tag{2.17}$$

This also gives us a basis for the quotient ring $\mathbb{Z}_7[X]/J$,

$$\boldsymbol{b} = \{1, z\}. \tag{2.18}$$

Taking the action polynomial as x we get the action matrix

$$x \begin{pmatrix} z \\ 1 \end{pmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{pmatrix} z \\ 1 \end{pmatrix}, \qquad \text{mod } J. \tag{2.19}$$

Thus we have that the polynomials

$$p_1 = xz + 1, \quad p_2 = x - z, \tag{2.20}$$

both lie in the saturated ideal J. They are however not in $I = \langle f_1, f_2, f_3 \rangle$, so it is not possible to directly construct an elimination template using f_1, f_2 and f_3 to recover p_1 and p_2 for a general instance. However when we multiply these by the saturation variable y, we lift them into the ideal I and we can express them in terms of the original equations,

$$yp_1 = 2zf_1 + (-3x + z)f_2 + (3x - y - 3z)f_3, \tag{2.21}$$

$$yp_2 = -3f_2 + 3f_3. \tag{2.22}$$

Note that in this example we only needed to multiply by y to lift p_1 and p_2 into the ideal I (i.e. $N = 1$), but in general higher powers might be required.

Now to solve a general instance of (2.15) *we follow the approach described in the previous sections. The elimination template is created by multiplying each of the* *polynomials f_i with the monomials occurring in the coefficients in* (2.21)–(2.22). *So in this case we have*

$$\{zf_1,\ f_2,\ xf_2,\ zf_2,\ f_3,\ xf_3,\ yf_3,\ zf_3\}. \tag{2.23}$$

Using just linear combinations of these, it is then possible to recover the polynomials,

$$yp_1 = y(xz - m_{11}z - m_{12}), \tag{2.24}$$
$$yp_2 = y(x - m_{21}z - m_{22}), \tag{2.25}$$

from which we can extract the action matrix

$$M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}. \tag{2.26}$$

2.3 Saturations for Zero-dimensional Ideals

In this section we consider the special case where we have finitely many solutions before saturating the ideal, i.e. when

$$\dim \mathbb{K}[X]/J < \dim \mathbb{K}[X]/I < \infty. \tag{2.27}$$

In this case it is possible to recover the action matrix in the saturated ideal from the action matrix in the original ideal.

Let $\alpha \in \mathbb{K}[X]$ be fixed and let $\boldsymbol{b}$ be a linear basis in $\mathbb{K}[X]/I$, with the corresponding action matrix $M \in \mathbb{K}^{d\times d}$, i.e. the matrix M satisfies

$$\alpha\boldsymbol{b} - M\boldsymbol{b} = 0 \mod I. \tag{2.28}$$

Similarly let $\boldsymbol{b}_0$ be a linear basis for $\mathbb{K}[X]/J$ with action matrix $M_0 \in \mathbb{K}^{d_0\times d_0}$,

$$\alpha\boldsymbol{b}_0 - M_0\boldsymbol{b}_0 = 0 \mod J. \tag{2.29}$$

The following proposition relates the two action matrices M and M_0.

Proposition 2.5. *The action matrices M_0 and M satisfy*

$$M_0 S = SM \tag{2.30}$$

where $S \in \mathbb{K}^{d_0\times d}$ satisfies

$$f_s^N \boldsymbol{b}_0 = S\boldsymbol{b} \mod I \tag{2.31}$$

for some $N \geq 0$.

Proof. Take $N \geq 0$ large enough so that

$$f_s^N \left(\alpha \boldsymbol{b}_0 - M_0 \boldsymbol{b}_0\right) = 0 \mod I, \tag{2.32}$$

and find the matrix S such that (2.31) holds for this N. Then

$$\alpha f_s^N \boldsymbol{b}_0 - M_0 f_s^N \boldsymbol{b}_0 = 0 \mod I \implies \alpha S \boldsymbol{b} - M_0 S \boldsymbol{b} = 0 \mod I. \tag{2.33}$$

Multiplying (2.28) with S and inserting into (2.33) we get

$$\left(SM - M_0 S\right) \boldsymbol{b} = 0 \mod I. \tag{2.34}$$

Since $\boldsymbol{b}$ is a basis for $\mathbb{K}[X]/I$ we must have $SM = M_0 S$. □

For some problems it can be simpler to find the action matrix M in the original ideal I. If the matrix S is also available (e.g. if the necessary monomials are available in the template), we can recover the action matrix for J as

$$M_0 = SMS^{\dagger}. \tag{2.35}$$

This allows us to instead solve the smaller eigenvalue problem related to M_0 to compute the solutions. For problems where many solutions are removed during saturation this can be significantly cheaper.

Another way to see this is to explicitly perform a change of basis (in $\mathbb{K}[X]/I$) such that $f_s^N \boldsymbol{b}_0$ becomes part of the new basis. This is possible since we know from Lemma 2.1 that these are linearly independent in $\mathbb{K}[X]/I$ for any $N \geq 0$. Let $\hat{S} = \begin{bmatrix} S^T & S_\perp^T \end{bmatrix}^T$ be some invertible matrix formed by adding rows to some S which satisfies (2.31). Then the new action matrix $\hat{M} = \hat{S} M \hat{S}^{-1}$ satisfies

$$\alpha \begin{pmatrix} f_s \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \end{pmatrix} - \begin{bmatrix} \hat{M}_{11} & \hat{M}_{12} \\ \hat{M}_{21} & \hat{M}_{22} \end{bmatrix} \begin{pmatrix} f_s \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \end{pmatrix} = 0 \mod I, \tag{2.36}$$

where $\boldsymbol{b}_1 = S_\perp \boldsymbol{b}$. If $N \geq 0$ is chosen large enough, it follows from (2.32) that $\hat{M}_{11} = M_0$ and $\hat{M}_{12} = 0$. Then (2.30) follows directly,

$$\begin{bmatrix} M_0 & 0 \\ \hat{M}_{21} & \hat{M}_{22} \end{bmatrix} = \hat{S} M \hat{S}^{-1} \implies \begin{bmatrix} M_0 & 0 \\ \hat{M}_{21} & \hat{M}_{22} \end{bmatrix} \begin{bmatrix} S \\ S_\perp \end{bmatrix} = \begin{bmatrix} S \\ S_\perp \end{bmatrix} M, \tag{2.37}$$

where the first row is $M_0 S = SM$. This is illustrated in the following example.

Example 2.6. *Consider the ideal $I = \langle f_1, f_2\rangle \subset \mathbb{Z}_7[x,y]$ where*

$$\begin{cases} f_1 = x^2 + y^2 - 1, \\ f_2 = y^3 - xy + x - y + 1. \end{cases} \tag{2.38}$$

For this ideal we have that

$$\boldsymbol{b} = \{xy^2, xy, y^2, x, y, 1\}, \tag{2.39}$$

is a basis of $\mathbb{Z}_7[x,y]/I$ and the action matrix $M \in \mathbb{Z}_7^{6\times 6}$ for x is

$$x\begin{pmatrix} xy^2 \\ xy \\ y^2 \\ x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} -1 & 1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}\begin{pmatrix} xy^2 \\ xy \\ y^2 \\ x \\ y \\ 1 \end{pmatrix} \mod I. \tag{2.40}$$

For the saturated ideal $J = \mathrm{Sat}(I, xy-1)$ we have that $\boldsymbol{b}_0 = \{y, 1\}$ is a basis for $\mathbb{Z}_7[x,y]/J$. Now to recover the action matrix for $\boldsymbol{b}_0$ in $\mathbb{Z}_7[x,y]/J$ we simply express $f_s\boldsymbol{b}_0$ in terms of $\boldsymbol{b}$. In this case we have

$$f_s\boldsymbol{b}_0 = \begin{pmatrix} xy^2 - y \\ xy - 1 \end{pmatrix} = S\boldsymbol{b} = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}\begin{pmatrix} xy^2 \\ xy \\ y^2 \\ x \\ y \\ 1 \end{pmatrix}. \tag{2.41}$$

Now let $\hat{S}$ be some invertible matrix formed by adding rows to S. One such choice is

$$\hat{S} = \begin{bmatrix} 1 & 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.42}$$

Performing the change of basis in (2.40) *we get*

$$\alpha \hat{S}\boldsymbol{b} = (\hat{S}M\hat{S}^{-1})\hat{S}\boldsymbol{b} \mod I \quad \iff$$

$$x\begin{pmatrix} xy^2-y \\ xy-1 \\ y^2 \\ x \\ xy^2+y \\ xy+1 \end{pmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & -3 & 0 \\ 0 & 3 & -1 & 0 & 0 & -3 \\ -1 & 1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 2 & 0 & 0 \end{bmatrix} \begin{pmatrix} xy^2-y \\ xy-1 \\ y^2 \\ x \\ xy^2+y \\ xy+1 \end{pmatrix} \mod I, \tag{2.43}$$

and from the first two rows we can extract

$$(xy-1)\left(x\begin{pmatrix} y \\ 1 \end{pmatrix} - \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}\begin{pmatrix} y \\ 1 \end{pmatrix} \right) = 0 \mod I, \tag{2.44}$$

which implies that

$$M_0 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{2.45}$$

is the action matrix in $\mathbb{Z}_7[x,y]/J$.

2.4 Applications with Saturated Ideals

We will in the following sections show how our method can be applied to a number of real world problems. These examples will show the benefits of our approach in terms of ease of construction of solvers without manual saturation (Section 2.4.1 and 2.4.2), significant speed-up (Section 2.4.2 and 2.4.3) and the benefit over introduction of auxiliary variables as in (2.14). The new solvers presented in this section are summarized in Table 2.1. In Pritts et al. [178] the saturation technique presented in this chapter was used to create minimal solvers for affine rectification and distortion estimation from coplanar repeated structure.

	Template size	# Sol.	Exec. time
Three view triang.			
Stewénius et al. [202]	-	47	> 1 s
Byröd et al. [32] (relaxed ideal)	$225 \times 209^{\dagger}$	154	60 ms
Byröd et al. [34] (basis select.)	$225 \times 209^{\dagger}$	58	3 ms
Our	571×676	47	10 ms
Our (new param.)	209×265	50	3 ms
Time-of-Arrival			
Kuang et al. [116] (4,6)	$966 \times 925^{\dagger}$	38	0.6 s
Kuang et al. [116] (5,5)	$1,386 \times 1,539^{\dagger}$	42	1.4 s
Our (4,6)	569×692	38	22 ms
Our (5,5)	$938 \times 1,301$	42	95 ms
Vanishing point est.			
Mirzaei et al. [162]	$2,860 \times 3,060$	40	1.8 s
Our	246×397	40	5 ms

Table 2.1: Overview over template sizes for the investigated applications in the paper. †: Several elimination steps are used, the largest of the elimination templates is reported.

2.4.1 Triangulation

In this section we will investigate how our saturation framework can be used in multiple view triangulation. Geometrically triangulation seems easily done, by simply intersecting the back-projected image rays. However to find the global minimizer of the reprojection error for many views is an inherently difficult problem. For two views one can find the solution by solving a sixth-degree polynomial, [83], but for three views it becomes numerically and theoretically harder, and there have many papers dealing with this problem [202, 32, 34, 131]. There are also iterative methods, that do not guarantee a global optimum, [3, 104, 151, 84], methods that minimize the L_∞-error [80, 160] and Branch-and-Bound methods whose worst case convergence is exponential [103, 153].

Optimal Three View Triangulation

We will initially model the triangulation problem in three views in the same way as described in [202]. To find the optimal solution we want to minimize the reprojection error

$$\underset{\mathbf{X}}{\text{minimize}} \sum_{i=1}^{3} (\frac{P_i^1\mathbf{X}}{P_i^3\mathbf{X}} - \mathbf{x}_i^1)^2 + (\frac{P_i^2\mathbf{X}}{P_i^3\mathbf{X}} - \mathbf{x}_i^2)^2, \tag{2.46}$$

where $\mathbf{X} = [X\,Y\,Z\,1]^T$ is the unknown 3D-point, and superscript denotes row-index. This is a non-convex problem and we will solve it by evaluating all stationary points. We have the freedom to make projective world coordinate changes without changing the error function. Similar to [202] we will use this to simplify our formulation, by choosing a coordinate system so that the third rows of the cameras are given by [1 0 0 0], [0 1 0 0] and [0 0 1 0] respectively. This will in turn change the denominators in (2.46) to X, Y and Z respectively. The gradient is easily calculated, but in order to get polynomial constraints we need to cross-multiply all the fractions with the denominators. In this way we end up with three equations, each of total degree 6 – giving rise to a one-dimensional ideal. In [202] they solved this by manually saturating the ideal, and after saturation with X, Y and Z ended up with 47 solutions. But the problem is very numerically challenging and the authors needed to use 128 bit arithmetic, making the solver impractically slow to use in practice. In [32] they used the same parametrization but showed that, by using a slightly different saturation approach and then considering a relaxed ideal, a practical solver could be developed. The solver still suffered from poor conditioning. In [34] the authors developed new methodology to handle poor conditioning by numerically choosing the linear basis during runtime in the minimal solver, using either SVD or QR factorization. We can use our automatic saturation process to avoid the cumbersome manual saturation steps. After saturation with X, Y and Z we end up with 47 solutions and get an elimination template of size 571×676.

In order to compare our solver with the publicly available solver from [34] we ran a simple test on the well-known dinosaur sequence. Using 36 calibrated frames with a total of 2592 points, seen in at least three views, we extracted for each of these points the corresponding first, last and middle camera. We ran our solver and the solver from [34]. The resulting mean of the reprojection error for all points was 8.90×10^{-5} for both methods (in the calibrated images). Using the

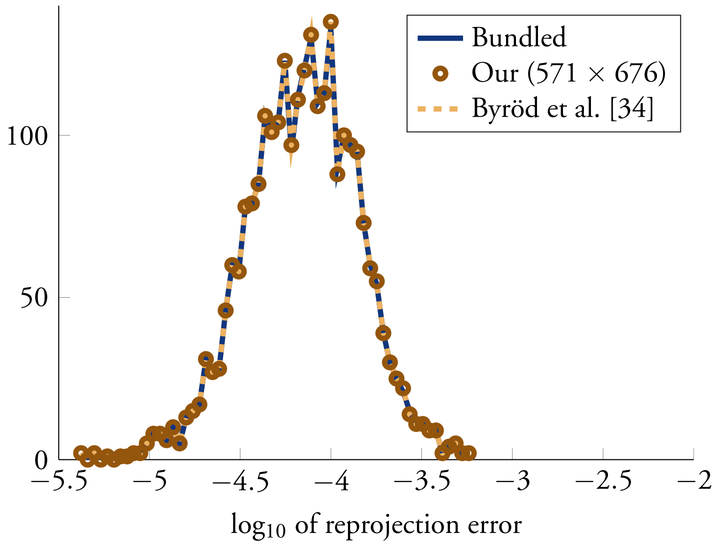


Figure 2.2: Error histograms for the dinosaur experiment for the different methods, on a log-scale.

chosen cameras we can for each point triangulate its 3D position linearly and then perform non-linear optimization on the reprojection error. This gave the same error (8.90×10^{-5}). In Figure 2.2 the error histogram is shown for all methods. One can see that they produce very similar results. Our developed solver has a runtime of around 10 ms, compared to around 3 ms for the handcrafted solver from [34].

New Parameterization

We can actually further simplify our problem formulation by setting the last row of camera three to $[0\,0\,0\,1]$, instead of as previously $[0\,0\,1\,0]$ (i.e. move the third camera's image plane to infinity). This means that we don't need to cross-multiply with factors of Z. This leads to three equations of total degree 6, 6 and 5 respectively. Furthermore, in this parameterization it turns out that it is sufficient to saturate only one of the variables. In this case the saturated ideal has 50 solutions. This approach gives a substantially smaller template of size 209×265, and in turn a much faster solver, with runtime under 3 ms. We have evaluated this new solver on both synthetic and real data. For a synthetic test, we randomly placed a 3D

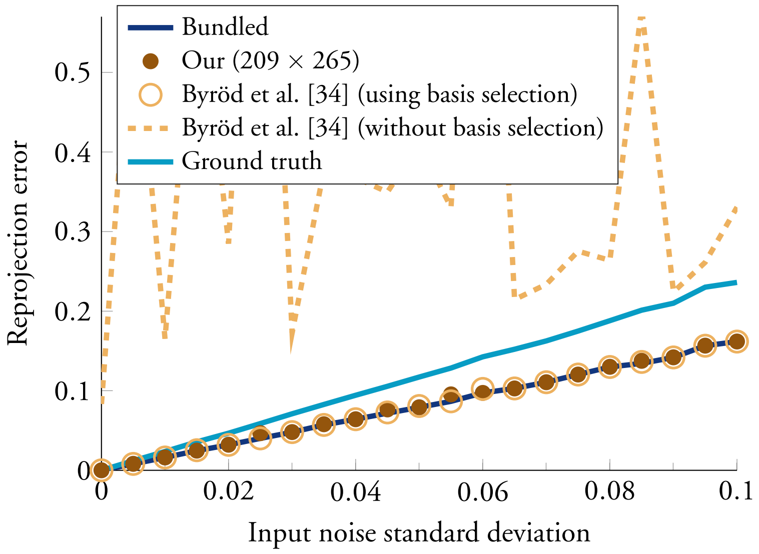


Figure 2.3: Synthetic triangulation test, comparing our method to non-linear optimization for different amounts of added image noise.

point and three cameras in a box with side-length 100. We then randomly chose an orientation for each camera with the constraint that the point would be visible, for a field of view of around 70°. We finally added normally distributed noise to the projection points with varying standard deviation. The result of running our algorithm, for varying degrees of noise, is shown in Figure 2.3. Also shown is the corresponding reprojection error for the randomly set ground truth 3D-point as well as the non-linearly optimized 3D-point position, using the ground truth position as initialization. The results for each noise level are evaluated on 1,000 random instances. We compare our results with the state-of-the-art solver from Byröd et al. [34]. Note that our solver achieves similar results without the special techniques for improving numerics from [34]. For comparison we have also included the results from the solver from [34] without basis selection. To test our triangulation method on real data, we ran it on three large scale problems where the camera matrices are available. In Figure 2.4 triangulated 3D points for the Notre Dame dataset [195] with around 120,000 3D-points, the Orebro Castle dataset [174] with around 50,000 3D-points, and the Arts Quad dataset [45] with around 1,400,000 3D-points are shown.

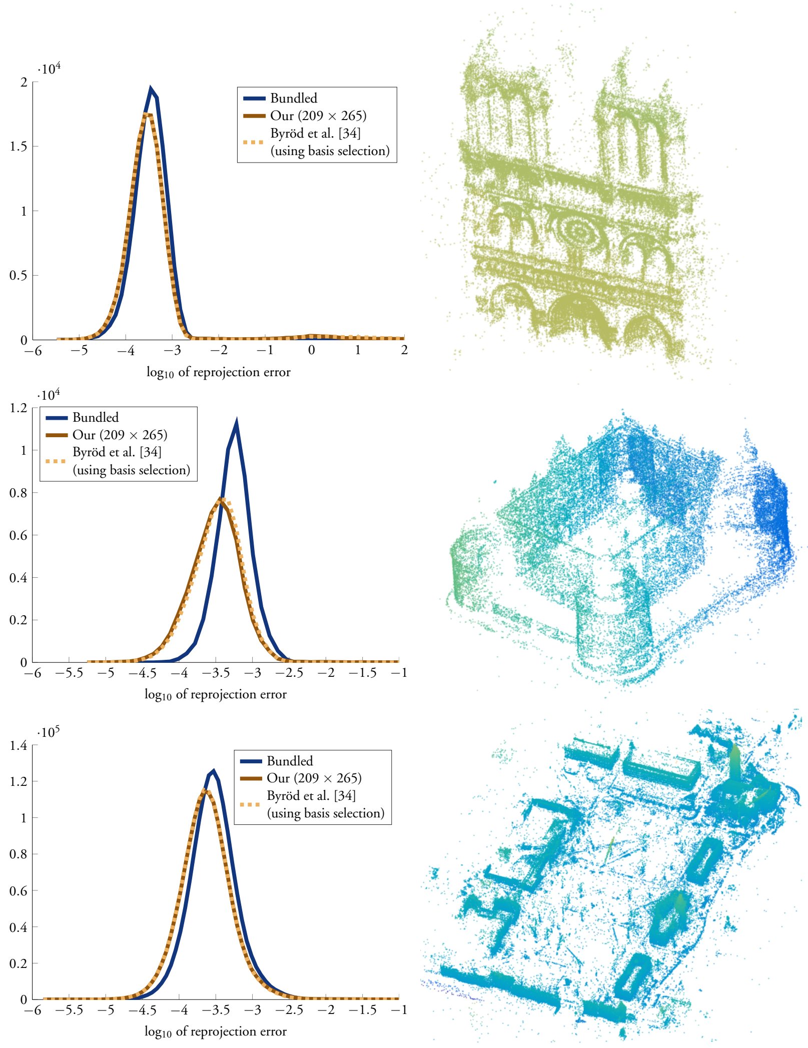


Figure 2.4: Triangulated points for three real problems using our triangulation method. Top to bottom: the Notre Dame dataset [195] with around 120,000 3D-points, the Orebro Castle dataset [174] (around 50,000 3D-points), and the Arts Quad dataset [45] (around 1,400,000 3D-points). Histograms show reprojection errors for our method compared to the solutions provided in the original datasets (which are bundled over all available cameras, hence the slighly larger median error) and the solver from [34].

2.4.2 Time-of-Arrival Self-Calibration

The term Time-of-Arrival (ToA) or alternatively Time-of-Flight (ToF), denotes the travel time of a signal from a transmitter to a receiver. If the speed of the medium is known such measurements provide distances from the transmitters to the receivers. The ToA self-calibration problem is the problem of estimating both transmitter positions $\mathbf{s}_j$ and receiver positions $\mathbf{r}_i$ given distance measurements,

$$d_{ij}^2 = \|\mathbf{r}_i - \mathbf{s}_j\|_2^2. \tag{2.47}$$

Following [116] we rearrange the equations in (2.47) into four types,

$$\begin{aligned}
d_{11}^2 &= (\mathbf{r}_1 - \mathbf{s}_1)^T(\mathbf{r}_1 - \mathbf{s}_1), && (2.48)\\
d_{1j}^2 - d_{11}^2 &= -2\mathbf{r}_1^T(\mathbf{s}_j - \mathbf{s}_1) + \mathbf{s}_j^T\mathbf{s}_j - \mathbf{s}_1^T\mathbf{s}_1, && (2.49)\\
d_{i1}^2 - d_{11}^2 &= -2(\mathbf{r}_i - \mathbf{r}_1)^T\mathbf{s}_1 + \mathbf{r}_i^T\mathbf{r}_i - \mathbf{r}_1^T\mathbf{r}_1, && (2.50)\\
d_{ij}^2 - d_{i1}^2 - d_{1j}^2 + d_{11}^2 &= -2(\mathbf{r}_i - \mathbf{r}_1)^T(\mathbf{s}_j - \mathbf{s}_1). && (2.51)
\end{aligned}$$

Introducing the two matrices

$$R = \begin{bmatrix}(\mathbf{r}_2 - \mathbf{r}_1)\dots(\mathbf{r}_m - \mathbf{r}_1)\end{bmatrix} \tag{2.52}$$

$$S = \begin{bmatrix}(\mathbf{s}_2 - \mathbf{s}_1)\dots(\mathbf{s}_n - \mathbf{s}_1)\end{bmatrix} \tag{2.53}$$

the $(m-1)(n-1)$ constraints in (2.51) can then be written $B = R^T S$, where

$$B = \begin{pmatrix} \frac{d_{22}^2 - d_{21}^2 - d_{12}^2 + d_{11}^2}{-2} & \dots & \frac{d_{2n}^2 - d_{21}^2 - d_{1n}^2 + d_{11}^2}{-2} \\ \vdots & \ddots & \vdots \\ \frac{d_{m2}^2 - d_{m1}^2 - d_{12}^2 + d_{11}^2}{-2} & \dots & \frac{d_{mn}^2 - d_{m1}^2 - d_{1n}^2 + d_{11}^2}{-2} \end{pmatrix}. \tag{2.54}$$

By factorizing $B = \tilde{R}^T\tilde{S}$, we can almost solve the self-calibration problem, however the factorization is not unique. If $B = \tilde{R}^T\tilde{S}$, then $B = \tilde{R}^T A A^{-1}\tilde{S}$ is also a valid factorization. Furthermore both the sender position $\mathbf{s}_1$ and the receiver position $\mathbf{r}_1$ are unknown.

Since the choice of coordinate system is arbitrary, one may without loss of generality set $\mathbf{r}_1$ to the origin. Also since any matrix A can be QR-factorized as a rotation matrix times a triangular matrix, one may assume that A is triangular, i.e. $A = L$. These choices fixate most of the freedom in the coordinate system.

Thus we parametrize the problem with $(L, \mathbf{b})$ so that

$$\begin{aligned} &\mathbf{r}_1 = \mathbf{0},\ \mathbf{s}_1 = L\mathbf{b},\ \mathbf{r}_i = L^{-T}\tilde{R}_i,\ i = 2\ldots m, \\ &\mathbf{s}_j = L(\tilde{S}_j + \mathbf{b}),\ j = 2\ldots n. \end{aligned} \tag{2.55}$$

Using this parametrization the equations (2.48)–(2.50) become

$$d_{11}^2 \quad = \mathbf{b}^T H^{-1}\mathbf{b}, \tag{2.56}$$
$$d_{1j}^2 - d_{11}^2 \quad = \tilde{S}_j^T H^{-1}\tilde{S}_j + 2\mathbf{b}^T H^{-1}\tilde{S}_j, \tag{2.57}$$
$$d_{i1}^2 - d_{11}^2 \quad = \tilde{R}_i^T H\tilde{R}_i - 2\mathbf{b}^T\tilde{R}_i, \tag{2.58}$$

using the symmetric matrix $H = (L^T L)^{-1}$. When both receivers and senders are in general 3D positions, there are two minimal problems to the time-of-arrival self-calibration problem – 6 receivers and 4 senders, and 5 receivers and 5 senders [116]. The solution strategy is now to first consider the constraints in (2.58), which are linear in the unknowns $(H, \mathbf{b})$. For the minimal problem with 6 receivers and 4 senders there are 5 such linear constraints on the 9 parameters in $(H, \mathbf{b})$. Thus it is possible to parametrize $(H, \mathbf{b})$ linearly in four unknowns (x_1, x_2, x_3, x_4). The remaining equations (2.56) and (2.57) involve the inverse of H. By rewriting $H^{-1} = \operatorname{adj} H / \det H$ and multiplying with $\det H$, the remaining four equations

$$d_{11}^2 \det H - \mathbf{b}^T \operatorname{adj} H\mathbf{b} \;= 0, \tag{2.59}$$
$$(d_{1j}^2 - d_{11}^2)\det H - \tilde{S}_j^T \operatorname{adj} H\tilde{S}_j - 2\mathbf{b}^T \operatorname{adj} H\tilde{S}_j \;= 0 \tag{2.60}$$

become polynomial equations in (x_1, x_2, x_3, x_4). The problem has 38 solutions for which $\det H \neq 0$. However there is a one dimensional family of solutions for which $\det H = 0$. The other minimal case with 5 receivers and 5 senders can in a similar manner be reduced to a system of five equations (2.59) and (2.60) in five unknowns. This problem has 42 solutions for which $\det H \neq 0$ and again a one dimensional family of solutions where $\det H = 0$. For other cases see [192, 193].

Using our approach we generated solvers for the two minimal cases $(4, 6)$ and $(5, 5)$. The two solvers have similar performance and we only evaluate the $(4, 6)$ case here. In Figure 2.5 we show the relative error in the distances for 1000 synthetic instance. We compare with the state-of-the-art solver from [116]. The underlying problem is numerically challenging, and in [116] the online basis selection method from [33] was used to improve the numerical performance.

The template sizes and runtimes are shown in Table 2.1. We also include the results using the generalized eigenvalue technique from Chapter 1 Section 1.3.3 (page 49) which improves the numerical stability.

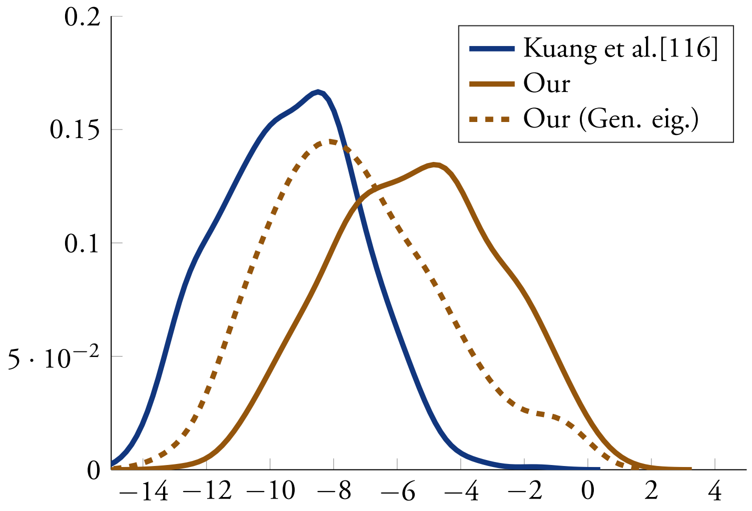


Figure 2.5: Histogram of $\log_{10}$ of residuals for solutions using the solver from Kuang et al. [116] and from proposed system.

2.4.3 Vanishing Point Estimation

In [162] the authors present a method for estimating the vanishing points in a Manhattan world. It is based on solving the following minimization problem optimally,

$$\min_R E = \sum_{i=1}^{N} (\mathbf{n}_i^T R \mathbf{m}_i)^2, \; R^T R = I, \; \det(R) = 1, \tag{2.61}$$

where $\mathbf{n}_i$ is the known vanishing point in the canonical frame for each line i given by its unit normalized representation $\mathbf{m}_i$. The sought R is the rotation matrix that takes the camera to the canonical frame. The rotation is then parameterized using the Cayley-Gibbs-Rodrigues formulation using $\mathbf{s}^T = [s_1, s_2, s_3]$, and

$$R(\mathbf{s}) = \frac{(1 - \mathbf{s}^T\mathbf{s})I + 2[\mathbf{s}]_\times + 2\mathbf{s}\mathbf{s}^T}{1 + \mathbf{s}^T\mathbf{s}}, \tag{2.62}$$

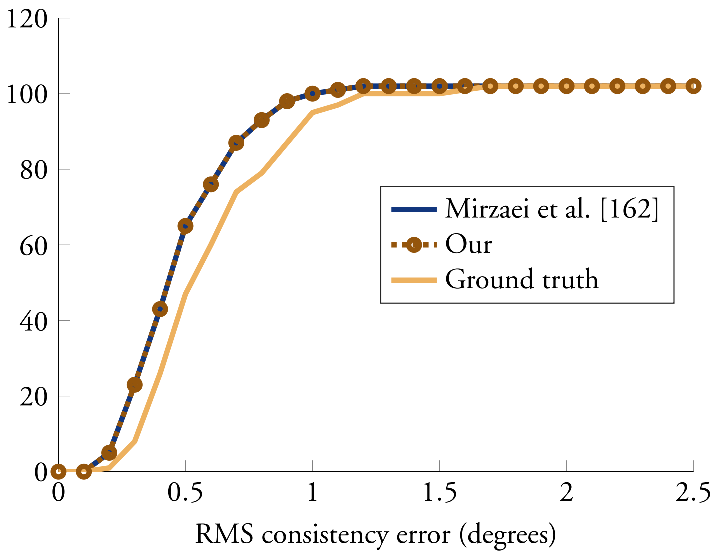


Figure 2.6: Vanishing point estimation RMS consistency on the York Urban dataset [51]. The figure shows that our method gives the same results as [162] on this dataset.

where $[\mathbf{s}]_\times$ is the cross-product matrix. The solution to (2.61) is found by enumerating all stationary points of E, by calculating the derivatives of (2.61) with respect to $\mathbf{s}$ and setting these to zero. The equations are given by

$$f_j(\mathbf{s}) = (1 + \mathbf{s}^T\mathbf{s})\frac{\partial \hat{E}}{\partial s_j} - 4s_j\hat{E} = 0, \quad j = 1, 2, 3, \tag{2.63}$$

where $\hat{E}(\mathbf{s}) = (1 + \mathbf{s}^T\mathbf{s})^2 E(\mathbf{s})$. The affine variety related to (2.63) contains a one-dimensional solution set corresponding to the imaginary hypersphere $1 + \mathbf{s}^T\mathbf{s} = 0$. This means that the standard action matrix method cannot be used directly. In [162] the authors solve this by introducing an auxiliary variable s_0 and a new equation, $f_0 = s_0(1 + \mathbf{s}^T\mathbf{s}) - 1 = 0$, which results in a zero-dimensional solution set, containing at most 40 solutions. This results in a stable, but comparably slow solver, based on an elimination template of size $2,860 \times 3,060$.

We have tested our automatic saturation methodology on this problem, by directly solving (2.63) and saturating with $1 + \mathbf{s}^T\mathbf{s}$. This results in an orders of magnitude faster solver with an elimination template of size 246×397. In order to compare the numerical properties of our solver we evaluated it on the same dataset as in [162], namely the York Urban Dataset (YUD), [51]. In Figure 2.6 the cumulative histogram of the consistency error is shown for our method compared to [162] and the ground truth estimate. The consistency error is defined as the RMS of $\sin^{-1}(\mathbf{v}_i^T \mathbf{m}_j)$ over all lines j for each image, where $\mathbf{v}_i$ is the corresponding estimated vanishing point. One can note that we get the same distribution as [162] (see their paper for the discussion on the slightly better results compared to the ground truth estimate). The average running time of our solver on YUD is 5 ms, compared to 1.8 s for [162]. This corresponds to a speed-up of more than a factor of 300, which shows that in this case it is much more beneficial to saturate directly compared to introducing an auxiliary variable as in (2.14). Our faster solver is also more suited to be used in a RANSAC framework, to eliminate errors in the classification of the lines.

2.5 Conclusions

In this chapter we have presented a new technique for building polynomial solvers for saturated ideals. In contrast to previous approaches the method avoids explicitly computing generators of the saturated ideal at runtime and instead lifts the problem into the original ideal.

Chapter 3

Exploiting Symmetries in Polynomials Systems

In this chapter we study certain symmetries in polynomial equation systems and how they can be integrated into the action matrix method. The main contribution is a generalization of the partial p-fold symmetry from Ask et al. [13] and Kuang et al. [119], and we provide new insights as to why these methods work.

3.1 Related Work

In [13] Ask et al. considers polynomial systems, where the degree of each monomial has the same remainder modulo p. This introduces a p-fold symmetry into the solution set. By taking this symmetry into account they construct smaller and more stable polynomial solvers. This work was later extended by Kuang et al. [119] to polynomial systems where this symmetry only exists in a subset of the variables. This type of symmetry has been used in [109, 232, 12, 77, 57]. In this chapter we further extend the work from [13, 119] and show how to handle multiple independent symmetries.

In [41] Corless et al. also consider symmetries in the action matrix method. Their approach is based on studying the group structure of the symmetry using tools from invariant theory. The paper present theory for symmetries of general group structure and the focus is mostly showing how to solve the equation system assuming that a Gröbner basis is known.

In contrast, in this chapter the focus is to use these symmetries to construct smaller elimination templates, and similarly to [13, 119] we instead classify the symmetry based on the monomials appearing in the equations. This allows the method to integrate more naturally into the standard pipeline for building polynomial solvers using elimination templates. While we are only able to handle a subset of the symmetries using this formulation, we will show many interesting applications where this is sufficient.

3.2 Symmetries in Minimal Problems

Now we present our generalization of the results from the Ask et al. [13] and Kuang et al. [119]. In this section we will make heavy use of the multi-index notation for monomials, i.e.

$$\boldsymbol{x}^{\boldsymbol{\alpha}} = \boldsymbol{x}^{(\alpha_1,\ldots,\alpha_n)} = \prod_{k=1}^{n} x_k^{\alpha_k}. \tag{3.1}$$

We start with a simple example.

Example 3.1. *Consider the following system of polynomial equations*

$$\begin{cases} x^2 + y - 2 = 0, \\ x^2y^2 - 1 = 0. \end{cases} \tag{3.2}$$

The system has six solutions given by

$$(\pm 1, 1),\ (\pm\varphi, -\varphi^{-1}),\ (\pm\varphi^{-1}, \varphi) \quad \textit{where} \quad \varphi = \frac{1+\sqrt{5}}{2}. \tag{3.3}$$

Since each monomial has the x-variable raised to an even power, we can for any solution flip the sign of x and get another solution.

This type of symmetry was studied in [13, 119] and is characterized in the following definition.

Definition 3.2. *The polynomial $f(\boldsymbol{x}, \boldsymbol{y})$ has a **partial** p**-fold symmetry** in $\boldsymbol{x}$ if the sum of the exponents for $\boldsymbol{x}$ of each monomial has the same remainder q modulo p, i.e.*

$$f(\boldsymbol{x}, \boldsymbol{y}) = \sum_k a_k \boldsymbol{x}^{\boldsymbol{\alpha}_k} \boldsymbol{y}^{\boldsymbol{\beta}_k} \implies q \equiv \mathbb{1}^T \boldsymbol{\alpha}_k \bmod p \quad \forall k. \tag{3.4}$$

In [119] it was shown that if we have a system of polynomials with this property the solution set will also have a p-fold symmetry. More specifically if V is the set of solutions then

$$(\boldsymbol{x}, \boldsymbol{y}) \in V \implies (e^{2\pi i \frac{k}{p}} \boldsymbol{x}, \boldsymbol{y}) \in V \quad k = 0, 1, 2, ..., p-1. \tag{3.5}$$

Example 3.3. *The polynomial system*

$$\begin{cases} x^3 - 1 = 0, \\ xy - 1 = 0 \end{cases} \tag{3.6}$$

has three solutions given by $V = \{(1,1), (\frac{-1+i\sqrt{3}}{2}, \frac{-1-i\sqrt{3}}{2}), (\frac{-1-i\sqrt{3}}{2}, \frac{-1+i\sqrt{3}}{2})\}$. *While this system does not have any partial p-fold symmetries, the solution set has the following property*

$$(x, y) \in V \implies (e^{2\pi i \frac{1}{3}} x, e^{2\pi i \frac{2}{3}} y) \in V, \tag{3.7}$$

which is similar to that in (3.5).

In this work we consider a natural generalization of the partial p-fold symmetry characterized by the following definition.

Definition 3.4. *The polynomial $f(\boldsymbol{x})$ has a **weighted p-fold symmetry** with weights $\boldsymbol{c} \in \mathbb{Z}_p^n$ if the $\boldsymbol{c}$-weighted sum of the exponents for $\boldsymbol{x}$ of each monomial has the same remainder q modulo p, i.e.*

$$f(\boldsymbol{x}) = \sum_k a_k \boldsymbol{x}^{\boldsymbol{\alpha}_k} \implies q \equiv \boldsymbol{c}^T \boldsymbol{\alpha}_k \bmod p \quad \forall k. \tag{3.8}$$

Example 3.5. *Below are three examples*

$$\begin{array}{lll} f_1(x,y) = x^3 - x^2y^2 + y^3, & p = 3, & \boldsymbol{c} = (2,1), \\ f_2(x,y) = x^5 + x^3y + x, & p = 4, & \boldsymbol{c} = (1,2), \\ f_3(x,y,z) = x + y^2 + yz - 1, & p = 2, & \boldsymbol{c} = (0,1,1). \end{array}$$

Note that the vector $\boldsymbol{c}$ is not unique, e.g. for the first polynomial $\boldsymbol{c} = (1,2)$ would also work. If p is not prime then for any factor of p we also have a symmetry, e.g. the polynomial f_2 also has a 2-fold symmetry. From the last example it becomes clear that the partial p-fold symmetry from [119] is a special case of the weighted symmetry where the weights are binary, corresponding to the symmetry variables.

Similarly to (3.5) we will see that for any polynomial equation system with $\boldsymbol{c}$-weighted p-fold symmetry, the solution set has a corresponding symmetry. For any vector $\boldsymbol{c} \in \mathbb{Z}_p^n$ we define the matrix

$$D_p^{\boldsymbol{c}} = \text{diag} \left\{ \exp(2\pi i \frac{c_k}{p}) \right\}_{k=1}^{n}. \tag{3.9}$$

Definition 3.6. *The set $V \subset \mathbb{C}^n$ has a **c-weighted** p**-fold symmetry** if it is stable under $D_p^{\mathbf{c}}$, i.e.*

$$D_p^{\boldsymbol{c}} V \subset V. \tag{3.10}$$

The following two theorems are directly adapted from Kuang et al. [119], where they are proved for regular partial p-fold symmetry.

Theorem 3.7. *Let $I = \langle f_1, \dots, f_m \rangle$. If each f_i has a* **c***-weighted p-fold symmetry then*

$$\boldsymbol{x} \in V(I) \implies D_p^{\boldsymbol{c}} \boldsymbol{x} \in V(I). \tag{3.11}$$

Proof. Take any $\boldsymbol{x} \in V(I)$. Consider some f_i and let q be the remainder from Definition 3.4. Let $\boldsymbol{x}^\beta$ be any monomial from f_i and consider the effect of $D_p^{\boldsymbol{c}}$,

$$(D_p^{\boldsymbol{c}} \boldsymbol{x})^{\boldsymbol{\beta}} = \prod_k (e^{2\pi i \frac{c_k}{p}} x_k)^{\beta_k} = \prod_k e^{2\pi i \frac{c_k \beta_k}{p}} x_k^{\beta_k} = e^{2\pi i \frac{\boldsymbol{c}^T \boldsymbol{\beta}}{p}} \boldsymbol{x}^{\boldsymbol{\beta}} = e^{2\pi i \frac{q}{p}} \boldsymbol{x}^{\boldsymbol{\beta}}. \tag{3.12}$$

Then if $f_i(\boldsymbol{x}) = \sum_k a_k \boldsymbol{x}^{\boldsymbol{\alpha}_k}$ we have

$$f_i(D_p^{\boldsymbol{c}} \boldsymbol{x}) = \sum_k a_k (D_p^{\boldsymbol{c}} \boldsymbol{x})^{\boldsymbol{\alpha}_k} = \sum_k a_k e^{2\pi i \frac{q}{p}} \boldsymbol{x}^{\boldsymbol{\alpha}_k} = e^{2\pi i \frac{q}{p}} f_i(\boldsymbol{x}) = 0, \tag{3.13}$$

and since this holds for any $i = 1, 2, \dots, m$ we must have that $D_p^{\boldsymbol{c}} \boldsymbol{x} \in V(I)$. □

Theorem 3.8. *Let $I = \langle f_1, \dots, f_m \rangle$ be an ideal where $V(I)$ satisfies* (3.11). *Then there exist a set of generators $I = \langle g_1, \dots, g_n \rangle$ where each g_i has* **c***-weighted p-fold symmetry.*

Proof. Let $\boldsymbol{x} \in V$. Then for any f_i we have

$$f_i\left((D_p^{\boldsymbol{c}})^k \boldsymbol{x}\right) = 0, \quad k = 0, 1, 2, ..., p-1. \tag{3.14}$$

Decompose f_i into $f_i(\boldsymbol{x}) = g_0(\boldsymbol{x}) + g_1(\boldsymbol{x}) + ... + g_{p-1}(\boldsymbol{x})$ such that each monomial in g_q has the $\boldsymbol{c}$-weighted remainder q modulo p, i.e.

$$g_q(\boldsymbol{x}) = \sum_k a_k \boldsymbol{x}^{\gamma_k} \implies \boldsymbol{c}^T \gamma_k \equiv q \bmod p. \tag{3.15}$$

Then if we denote $\omega = e^{2\pi i \frac{1}{p}}$ we have

$$f_i(\boldsymbol{x}) = g_0(\boldsymbol{x}) + g_1(\boldsymbol{x}) + ... + g_{p-1}(\boldsymbol{x}), \tag{3.16}$$

$$f_i(D_p^{\boldsymbol{c}}\boldsymbol{x}) = g_0(\boldsymbol{x}) + \omega g_1(\boldsymbol{x}) + ... + \omega^{p-1} g_{p-1}(\boldsymbol{x}), \tag{3.17}$$

$$\cdots$$

$$f_i((D_p^{\boldsymbol{c}})^{p-1}\boldsymbol{x}) = g_0(\boldsymbol{x}) + \omega^{p-1} g_1(\boldsymbol{x}) + ... + \omega g_{p-1}(\boldsymbol{x}). \tag{3.18}$$

Since each $f_i((D_p^{\boldsymbol{c}})^k\boldsymbol{x}) = 0$ we can rewrite this as

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{p-1} & \omega^{p-2} & \dots & \omega \end{bmatrix} \begin{bmatrix} g_0(\boldsymbol{x}) \\ g_1(\boldsymbol{x}) \\ g_2(\boldsymbol{x}) \\ \vdots \\ g_{p-1}(\boldsymbol{x}) \end{bmatrix} = \mathbf{0}. \tag{3.19}$$

Since the matrix is non-singular we must have $g_k(\boldsymbol{x}) = 0$ for $k = 0, 1, 2, \dots, p-1$. By definition each g_k will have a $\boldsymbol{c}$-weighted p-fold symmetry and by replacing each equation $f_i(\boldsymbol{x}) = 0$ by the equations $\{g_k(\boldsymbol{x}) = 0\}_{k=0}^{p-1}$ it is clear that we get new generators which has the correct symmetry.

$\square$

Corollary 3.9. *Take any polynomial $f \in I = \langle f_1, f_2, \dots, f_m \rangle$ where $V(I)$ satisfy* (3.11) *and decompose*

$$f(\boldsymbol{x}) = g_0(\boldsymbol{x}) + g_1(\boldsymbol{x}) + ... + g_{p-1}(\boldsymbol{x}), \tag{3.20}$$

such that each monomial in g_q has the $\boldsymbol{c}$-weighted remainder q modulo p, then each component g_q is a polynomial in I.

Proof. This is a consequence of the proof of the previous theorem. □

Corollary 3.10. *Let $I \subset \mathbb{C}[X]$ be an ideal where $V(I)$ has $\mathbf{c}$-weighted p-fold symmetry, i.e. satisfies* (3.11)*, and let $\mathcal{B}$ be a monomial basis in $\mathbb{C}[X]/I$. For any $h \in \mathbb{C}[X]$ with $\mathbf{c}$-weighted exponent remainder q modulo p we have that*

$$[h(\boldsymbol{x})] \in [\operatorname{span} \mathcal{B}_q], \tag{3.21}$$

where $\mathcal{B}_q$ are the basis elements with $\mathbf{c}$-weighted exponent remainder q modulo p.

Proof. Since $\mathcal{B}$ is a linear basis for $\mathbb{C}[X]/I$ there exist coefficients a_k such that

$$[h(\boldsymbol{x})] = \left[\sum_k a_k b_k(\boldsymbol{x})\right], \quad \text{where} \quad b_k \in \mathcal{B}. \tag{3.22}$$

Then we have $h(\boldsymbol{x}) - \sum_k a_k b_k(\boldsymbol{x}) \in I$. Using Corollary 3.9 we can split this into the different exponent remainder classes, which gives us,

$$h(\boldsymbol{x}) - \sum_{b_k \in \mathcal{B}_q} a_k b_k(\boldsymbol{x}) \in I \implies [h(\boldsymbol{x})] = \left[\sum_{b_k \in \mathcal{B}_q} a_k b_k(\boldsymbol{x})\right]. \tag{3.23}$$

□

While the following corollary was not stated explicitly in Ask et al. [13] and Kuang et al. [119], it is what allowed them to only consider e.g. even monomials when constructing their elimination templates. This also makes it clear how to handle multiple independent symmetries. In Corless et al. [41] a similar result is presented.

Corollary 3.11. *For any action polynomial $\alpha(\boldsymbol{x}) \in \mathbb{C}[X]$ with $\mathbf{c}$-weighted remainder zero the corresponding action matrix becomes block diagonal.*

Proof. If $\alpha(\boldsymbol{x})$ has $\mathbf{c}$-weighted remainder zero then all polynomials in $\alpha(\boldsymbol{x})\mathcal{B}_q$ will have remainder q. From Corollary 3.10 we get $[\alpha(\boldsymbol{x})\mathcal{B}_q] \subset [\operatorname{span} \mathcal{B}_q]$ and the result follows. □

3.2.1 Solving Equation Systems with Symmetries

Once the symmetries have been identified they can be used to construct more compact polynomial solvers. From Corollary 3.11 we know that if we choose our action polynomial $a(\boldsymbol{x}) \in \mathbb{C}[X]$ to be invariant with respect to the symmetry the corresponding action matrix will be block diagonal. The idea is then to only consider a single block of the matrix, i.e. we only consider the action of $\alpha(\boldsymbol{x})$ on a subset of the basis monomials $\mathcal{B}$. The elimination template then only needs to be able to construct the polynomials required for this smaller sub-block of the action matrix. Since these only contain monomials of a certain exponent remainder class, we only need to include equations which has this class in our template. This typically allows for much smaller elimination templates to be used.

Next we show two concrete examples where we construct the partial action matrix and use it to recover the solutions.

Example 3.12. *Consider again the system in Example 3.1,*

$$\begin{cases} x^2 + y - 2 = 0, \\ x^2y^2 - 1 = 0. \end{cases} \tag{3.24}$$

We saw earlier that this system has six solutions and a 2-fold partial symmetry in the x variable, or equivalently a $(1,0)$-weighted 2-fold symmetry. For this system the quotient ring $\mathbb{C}[X]/I$ is spanned by the monomials

$$\mathcal{B} = \{1,\ x,\ y,\ xy,\ y^2,\ xy^2\}. \tag{3.25}$$

We can group these into two sets, based on their $(1,0)$-weighted remainder modulo 2,

$$\mathcal{B}_0 = \{1,\ y,\ y^2\} \quad \textit{and} \quad \mathcal{B}_1 = \{x,\ xy,\ xy^2\}. \tag{3.26}$$

Now instead of working with the entire basis $\mathcal{B}$ we will only consider the subset $\mathcal{B}_0$. If we choose x^2 to be our action polynomial (note that this has $(1,0)$-remainder zero) we have the following multiplication maps[1]

$$T_{x^2}[1] = x^2 = 2 - y, \quad T_{x^2}[y] = x^2y, \quad T_{x^2}[y^2] = x^2y^2 = 1. \tag{3.27}$$

Since one of the monomials was not in the span of $\mathcal{B}_0$ we need to generate more equations. Multiplying the first equation by y we get

$$x^2y + y^2 - 2y = 0 \implies T_{x^2}[y] = x^2y = 2y - y^2 \in \operatorname{span}\mathcal{B}_0. \tag{3.28}$$

[1]$T_\alpha : \mathbb{C}[X]/I \to \mathbb{C}[X]/I$ is the linear map corresponding to multiplication by $\alpha(\boldsymbol{x})$.

Finally we can construct our action matrix,

$$\begin{bmatrix} 0 & 0 & 1 \\ -1 & 2 & 0 \\ 0 & -1 & 2 \end{bmatrix} \begin{pmatrix} y^2 \\ y \\ 1 \end{pmatrix} = x^2 \begin{pmatrix} y^2 \\ y \\ 1 \end{pmatrix}. \tag{3.29}$$

Even though the system has six solutions we only have to solve a 3×3 eigenvalue problem. From each eigenvector we can construct two solutions with different signs for x.

Next we show a similar example but where the equation system has two independent symmetries.

Example 3.13. *Consider the equation system*

$$\begin{cases} x^2 + y^2 - 2 = 0, \\ xy^2 - x = 0. \end{cases} \tag{3.30}$$

This system has a 2-fold partial symmetry in the x variable and 2-fold partial symmetry in y, or equivalently two 2-fold symmetries with weights $\mathbf{c}_1 = (1, 0)$ and $\mathbf{c}_2 = (0, 1)$. The equation system has six solutions and a basis for the quotient ring $\mathbb{C}[X]/I$ is given by

$$\mathcal{B} = \{1,\ x,\ y,\ xy,\ y^2,\ y^3\}. \tag{3.31}$$

Grouping the basis monomials based on their $\mathbf{c}_k$-weighted remainders modulo 2:

$$\mathcal{B}_{0,0} = \{1,\ y^2\},\ \mathcal{B}_{0,1} = \{y,\ y^3\},\ \mathcal{B}_{1,0} = \{x\},\ \mathcal{B}_{1,1} = \{xy\}. \tag{3.32}$$

Let us choose to work with $\mathcal{B}_{0,0}$. By multiplying the second equation with x we get that all the monomials in the system have the same remainder as our monomial basis,

$$\begin{cases} x^2 + y^2 - 2 = 0, \\ x^2y^2 - x^2 = 0. \end{cases} \tag{3.33}$$

Choosing again the action polynomial as x^2 we get the following multiplications

$$T_{x^2}[1] = x^2 = 2 - y^2 \in \operatorname{span} \mathcal{B}_{0,0}, \quad T_{x^2}[y^2] = x^2y^2 = x^2 = 2 - y^2 \in \operatorname{span} \mathcal{B}_{0,0}, \tag{3.34}$$

which allows us to construct a 2×2 *action matrix*

$$\begin{bmatrix} -1 & 2 \\ -1 & 2 \end{bmatrix} \begin{pmatrix} y^2 \\ 1 \end{pmatrix} = x^2 \begin{pmatrix} y^2 \\ 1 \end{pmatrix}. \tag{3.35}$$

This matrix has eigenvalues 0 and 1 with corresponding eigenvectors $\begin{pmatrix} 2 \\ 1 \end{pmatrix}$, $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$. *The two eigenpairs give us the following possibilities*

$$\begin{cases} y^2 = 2 \\ x^2 = 0 \end{cases} \quad \text{and} \quad \begin{cases} y^2 = 1 \\ x^2 = 1 \end{cases}. \tag{3.36}$$

The first eigenvector gives two solutions $(0, \sqrt{2})$ *and* $(0, -\sqrt{2})$ *and the second gives four solutions* $(1, \pm 1)$, $(-1, \pm 1)$. *The full action matrix has the following form*

$$\begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & -1 & 2 & & \\ & & -1 & 2 & & \\ & & & & -1 & 2 \\ & & & & -1 & 2 \end{bmatrix} \begin{pmatrix} xy \\ x \\ y^3 \\ y \\ y^2 \\ 1 \end{pmatrix} = x^2 \begin{pmatrix} xy \\ x \\ y^3 \\ y \\ y^2 \\ 1 \end{pmatrix}. \tag{3.37}$$

In this example we chose the basis $\mathcal{B}_{0,0}$ *but we could also have used* $\mathcal{B}_{0,1}$ *to recover the solutions. However the other two choices would not have allowed us to recover the complete solution set.*

3.2.2 Implementation

We have implemented support for using this type of symmetry in the automatic generator presented in Chapter 1. It automatically finds these variable aligned symmetries by simply computing the remainders for the monomial exponents for all the weight vectors $\boldsymbol{c}$ up to some predefined maximum weight. For small problems this can be done very quickly. To find the elimination template we only need to partition the quotient ring basis into the different remainder classes and make sure to choose an action monomial invariant to the symmetry. One elimination template is then created for each exponent remainder class and the smallest one is kept.

3.3 Unaligned Symmetries

In the previous section we have studied symmetries which depend on the exponents of the monomials. These properties are however not preserved under a linear change of variables and for some problems there can exist weighted symmetries which only appear after a change of variables.

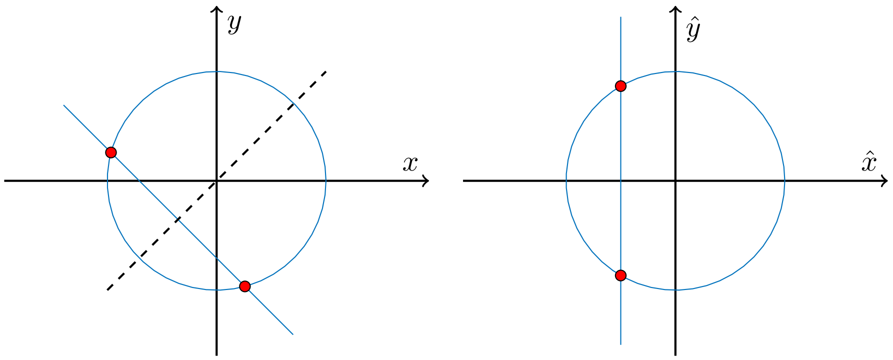
Example 3.14. *For $\theta \in \mathbb{R}$ consider the following family of polynomial systems*

$$\begin{cases} x^2 + y^2 = 1 \\ x + y = \theta \end{cases} . \tag{3.38}$$

Clearly the solution set is stable under the transform which switches x and y since the equation system is unchanged. But in this formulation the system does not have any weighted p-fold symmetries. Performing a change of variables

$$\begin{cases} \hat{x} = x + y \\ \hat{y} = x - y \end{cases} \implies \begin{cases} \frac{1}{2}\hat{x}^2 + \frac{1}{2}\hat{y}^2 = 1 \\ \hat{x} = \theta \end{cases} \tag{3.39}$$

reveals a 2-fold symmetry in the $\hat{y}$ variable. The solutions before and after the change of variables is illustrated below. The axis of symmetry is dashed in the left figure. After the change of variables, the symmetry axis is aligned with the x axis.



The previous example showed a polynomial system which was invariant to a specific linear transform. After a change of variables the symmetry was transformed into a weighted p-fold symmetry as in Section 3.2. The following theorem shows that under some weak assumptions this can be done in general.

Theorem 3.15. *Let $f_i(x) = 0$, $i = 1, 2, \dots, m$ be a polynomial system with a finite number of solutions. If there exist an invertible matrix $A \neq I$ such that the solution set $V = \{x \mid f(x) = 0\} \subset \mathbb{C}^n$ is stable under A, then the polynomial system exhibits a $\boldsymbol{c}$-weighted p-fold symmetry after a linear change of variables.*

Proof. We start by noting that we can without loss of generality assume that $\operatorname{span} V = \mathbb{C}^n$. If this does not hold it will be sufficient to consider the restriction of A to the span of V, i.e. $A_{|V} : \operatorname{span} V \to \operatorname{span} V$.

Since V is finite and A injective we have that $AV = V$. This means that A acts as a permutation on the elements of V. It follows that there must exist $p \in \mathbb{N}$ such that

$$A^p s = s \quad \forall s \in V, \tag{3.40}$$

since there are only a finite number of possible permutations. This implies

$$A^p = I, \tag{3.41}$$

since the elements of V span $\mathbb{C}^n$. This is a sufficient condition for A to be diagonalizable and that the eigenvalues of A are p:th roots of unity, i.e. $\lambda_k = e^{2\pi i \frac{c_k}{p}}$. Let S be the matrix which diagonalizes A, then

$$A = SDS^{-1} = S \operatorname{diag}\{e^{2\pi i \frac{c_k}{p}}\}_{k=1}^{n} S^{-1}. \tag{3.42}$$

Note that by definition $D = D_p^{\boldsymbol{c}}$ for $\boldsymbol{c} = (c_1, c_2, \dots, c_n) \in \mathbb{Z}_p^n$ and if we perform the change of variables $\hat{\boldsymbol{x}} = S^{-1}\boldsymbol{x}$ the solution set instead becomes stable under $D_p^{\boldsymbol{c}}$ and thus the system has a $\boldsymbol{c}$-weighted p-fold symmetry. $\square$

If the system has multiple symmetry matrices we can use all of them if we are able to diagonalize them simultaneously. It is a well-known fact from linear algebra that a set of diagonalizable matrices can be simultaneously diagonalized if and only if they commute.

In [41], Corless et al. also consider symmetries where the corresponding matrices do not commute. In this case it is not always possible to choose a monomial basis for $\mathbb{K}[X]/I$ which makes the action matrix block diagonal. However, for non-monomial quotient ring bases it is not as clear how to construct the elimination templates efficiently.

3.3.1 Finding Unaligned Symmetries in Practice

Unless there is some problem specific knowledge it can be difficult to find the change of variables which reveals the symmetry. In this section we present a simple heuristic method for determining if a given problem has any symmetries of the type presented in the previous section. The idea is to, similarly to the homotopy continuation methods, generate solution trajectories for a family of problems. From these trajectories it is then possible to identify the symmetries.

Assume that we are given a family of polynomial systems $\{f_i(\boldsymbol{x}, \boldsymbol{a}) = 0\}$, which depends on some data $\boldsymbol{a} \in \mathbb{C}^m$, i.e. for fix $\boldsymbol{a}$ each $f_i(\boldsymbol{x}, \boldsymbol{a})$ is polynomial in $\boldsymbol{x}$. To find the symmetries we do the following:

1. Take some instance $\boldsymbol{a}^0 \in \mathbb{C}^m$ and solve the problem $\{f_i(\boldsymbol{x}, \boldsymbol{a}^0) = 0\}$ using any method. This can for example be accomplished by selecting some $\boldsymbol{a}^0$ where the solutions are known, or by using some numerical solver (e.g. PHCPack [220]). Denote the solutions $\boldsymbol{s}_k^0,\ k = 1, 2, \ldots, N$.

2. Next we generate a sequence of polynomial systems by updating data in small increments, $\boldsymbol{a}^{t+1} = \boldsymbol{a}^t + \boldsymbol{\epsilon}$. For each solution $\boldsymbol{s}_k^0$ we generate a *solution trajectory* by tracking the solution using non-linear refinement methods (e.g. Newton iterations). So to find $\boldsymbol{s}_k^{t+1}$ we solve $\{f_i(\boldsymbol{x}, \boldsymbol{a}^{t+1}) = 0\}$ by starting non-linear refinement at $\boldsymbol{s}_k^t$. This is repeated until the matrices
$$S_k = \begin{bmatrix} \boldsymbol{s}_k^0 & \boldsymbol{s}_k^1 & \ldots & \boldsymbol{s}_k^t \end{bmatrix} \in \mathbb{C}^{n \times t}, \quad k = 1, 2, ..., N \tag{3.43}$$
are all of full rank and $t > n$.

3. For each pair, $i \neq j$, we try to find a matrix $A_{ij} \in \mathbb{C}^{n \times n}$ such that
$$A_{ij} S_i = S_j. \tag{3.44}$$
Since the system is overdetermined and the solution matrices S_i might have some small errors we solve (3.44) in a least square sense. If the residuals are sufficiently close to zero, we add the matrix A_{ij} to a list of possible symmetry matrices.

4. For each matrix A_{ij} we check if the solution set is stable, i.e. if for each k there exist $\ell \neq k$ such that $A_{ij} S_k = S_\ell$. The matrices which satisfy this are the symmetry matrices.

5. Finally we generate new instances and check if the symmetry matrices work.

3.4 Application with Symmetries

Now we will present one problem where we can build significantly smaller solvers by exploiting the symmetric structure of the problem. Other examples (marked with †) can be found in Table 1.1 in Chapter 1.

3.4.1 Weak Perspective-n-Points

In this section we show a practical example where we can construct a more compact polynomial solver by exploiting a symmetry in the problem. We consider the problem of estimating a weak perspective camera[2] (also known as scaled orthographic camera) from n 2D-3D point correspondences. We find the pose which minimizes the squared reprojection error.

To find the pose which minimizes the squared reprojection error we want to solve the following constrained minimization problem

$$\min_{s,R,\boldsymbol{t}} \left\|RA + \boldsymbol{t}\mathbb{1}^T - B\right\|_F^2 \quad \text{s.t.} \quad RR^T = s^2 I_2, \tag{3.45}$$

where $A \in \mathbb{R}^{3\times n}$ are the 3D points and $B \in \mathbb{R}^{2\times n}$ are the corresponding 2D projections. By differentiating the cost w.r.t. $\boldsymbol{t}$ we get

$$2(RA + \boldsymbol{t}\mathbb{1}^T - B)\mathbb{1} = 0 \implies \boldsymbol{t} = \frac{1}{n}(B - RA)\mathbb{1}. \tag{3.46}$$

Inserting into the original cost

$$\left\|(RA - B)(I - \frac{1}{N}\mathbb{1}\mathbb{1}^T)\right\|_F^2 = \left\|R\tilde{A} - \tilde{B}\right\|_F^2, \tag{3.47}$$

where $\tilde{A}$ is simply A with the row-mean subtracted. The problem can be further simplified. Let $\tilde{A} = U\begin{bmatrix}\Sigma & 0\end{bmatrix}\begin{bmatrix}V_1^T \\ V_2^T\end{bmatrix}$ be the singular value decomposition of $\tilde{A}$. Then

$$\left\|R\tilde{A} - \tilde{B}\right\|_F^2 = \left\|(RU)\Sigma - \tilde{B}V_1\right\|_F^2 \tag{3.48}$$

[2]In some literature different scales for x and y-axis are allowed (i.e. unknown aspect ratio), however here we consider a weak perspective camera to have unit aspect ratio.

and by an orthogonal change of variables the optimization problem is reduced to

$$\min_{s,R} \left\| R \ \mathrm{diag}(a_1, a_2, a_3) - \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix} \right\|_F^2 \tag{3.49}$$

$$\text{s.t.} \quad RR^T = s^2 I_2, \tag{3.50}$$

where $a_1 \geq a_2 \geq a_3 \geq 0$ are the singular values of $\tilde{A}$.

Parameterizing the Constraints

We use the unconstrained quaternion parametrization of the scaled 2×3 rotation,

$$R(q) = \begin{bmatrix} q_1^2 + q_2^2 - q_3^2 - q_4^2 & 2(q_2 q_3 - q_1 q_4) & 2(q_1 q_3 + q_2 q_4) \\ 2(q_1 q_4 + q_2 q_3) & q_1^2 - q_2^2 + q_3^2 - q_4^2 & 2(q_3 q_4 - q_1 q_2) \end{bmatrix}, \tag{3.51}$$

where $q = (q_1, q_2, q_3, q_4)$ and $\|q\|_2 = s$. In this parametrization the problem in (3.49) becomes unconstrained and the cost function

$$f(q) = \|R(q)A - B\|_F^2. \tag{3.52}$$

is a quartic polynomial in the elements of q. Since each element in $R(q)$ is of degree two we have that $f(q)$ only contains monomials of degrees $0, 2$ and 4.

We find the optimal pose by studying the first-order necessary conditions for (3.52). Since the problem is now unconstrained in the unscaled quaternion representation we simply solve for the critical points, i.e.

$$g(q) = \nabla_q f(q) = 0. \tag{3.53}$$

Since $f(q)$ only contains even terms, this equation system $g(q) = 0$ can only contain odd terms (degree 1 or 3). Thus we can directly see that the equation system will have at least a two-fold symmetry. This symmetry correspond to the sign ambiguity of the quaternion representation, i.e. $R(q) = R(-q)$.

Additional Symmetries in the Solutions

The quaternion parametrization is inherently ambiguous since the sign of the quaternion does not matter. But it turns out that there is a further ambiguity which comes from the fact that the third row of the rotation matrix is ignored. By studying (3.51) we can see that $R(q_1, q_2, q_3, q_4) = R(iq_4, iq_3, -iq_2, -iq_1)$ holds for all $q \in \mathbb{C}^4$. For the full 3×3 rotation matrix this corresponds to changing sign of the third row. Together these ambiguities introduce a four-fold symmetry into the solution set which can be described by the matrices

$$A_1 = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \quad \text{and} \quad A_2 = \begin{pmatrix} 0 & 0 & 0 & i \\ 0 & 0 & i & 0 \\ 0 & -i & 0 & 0 \\ -i & 0 & 0 & 0 \end{pmatrix}. \tag{3.54}$$

To simultaneously diagonalize the matrices we perform the following change of variables,

$$\hat{\boldsymbol{q}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & -i & -i & 0 \\ -1 & 0 & 0 & 1 \\ i & 0 & 0 & i \\ 0 & -1 & 1 & 0 \end{bmatrix} \boldsymbol{q}. \tag{3.55}$$

In these new variables the rotation matrix becomes

$$R(\hat{\boldsymbol{q}}) = \begin{bmatrix} \hat{q}_1^2 - \hat{q}_2^2 - \hat{q}_3^2 + \hat{q}_4^2 & -i\hat{q}_1^2 - i\hat{q}_2^2 + i\hat{q}_3^2 + i\hat{q}_4^2 & 2(\hat{q}_1\hat{q}_2 + 2\hat{q}_3\hat{q}_4) \\ -i\hat{q}_1^2 + i\hat{q}_2^2 - i\hat{q}_3^2 + i\hat{q}_4^2 & -\hat{q}_1^2 - \hat{q}_2^2 - \hat{q}_3^2 - \hat{q}_4^2 & 2i(\hat{q}_3\hat{q}_4 - \hat{q}_1\hat{q}_2) \end{bmatrix}. \tag{3.56}$$

Note that the only mixed terms are $\hat{q}_1\hat{q}_2$ and $\hat{q}_3\hat{q}_4$. This leads to one 2-fold symmetry in $(\hat{q}_1, \hat{q}_2)$ and one 2-fold symmetry in $(\hat{q}_3, \hat{q}_4)$.

In Chapter 6 we present another problem where this symmetry appears.

Constructing a Polynomial Solver

By studying the problem in Macaulay2 [75] and Maple [164] we find that the system has 33 solutions when parameterizing it using quaternions. Ignoring the trivial solution, the rest of the solutions can be grouped into eight groups of four solutions.

We used the automatic generator from Chapter 1 to create polynomial solvers for this problem, both using only the 2-fold symmetry (as in [14] and [119]) as well as the full 2×2-fold symmetry as described in the previous section. For the 2×2 symmetry the following eight basis monomials,

$$\mathcal{B}_{1,1} = \{\hat{q}_1\hat{q}_3,\ \hat{q}_1\hat{q}_3\hat{q}_4^2,\ \hat{q}_1\hat{q}_4,\ \hat{q}_1\hat{q}_4^3,\ \hat{q}_2\hat{q}_3,\ \hat{q}_2\hat{q}_3\hat{q}_4^2,\ \hat{q}_2\hat{q}_4,\ \hat{q}_2\hat{q}_4^3\}, \tag{3.57}$$

gave the smallest elimination template with the action monomial $\alpha(\hat{\boldsymbol{q}}) = \hat{q}_1\hat{q}_2$.

We also generated solvers without using any of the symmetries. We applied both the automatic generator from Chapter 1 as well as the automatic generator from Kukelova et al. [122]. The template sizes were 231 × 263 and 243 × 273. In the experimental evaluation we only include the former.

Another Parametrization from Wu

In [227] Wu presented a solver for the P3.5Pf problem. For this problem there exist a similar symmetry where the sign change in the first two rows of the rotation matrix can be canceled by changing the sign of the focal length, i.e.

$$\begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_1^T \\ \boldsymbol{r}_2^T \\ \boldsymbol{r}_3^T \end{bmatrix} = \begin{bmatrix} -f & & \\ & -f & \\ & & 1 \end{bmatrix} \begin{bmatrix} -\boldsymbol{r}_1^T \\ -\boldsymbol{r}_2^T \\ \boldsymbol{r}_3^T \end{bmatrix}. \tag{3.58}$$

To avoid this symmetry Wu proposed used a re-parametrization of the problem. The idea is to factor out a rotation around the z-axis,

$$\begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & \\ \sin(\theta) & \cos(\theta) & \\ & & 1 \end{bmatrix} R(\phi,\gamma) = \begin{bmatrix} x & -y & \\ y & x & \\ & & 1 \end{bmatrix} R(\phi,\gamma), \tag{3.59}$$

and then introduce new unknowns, $x = f\cos(\theta)$ and $y = f\sin(\theta)$ where the symmetry is not present, i.e.

$$x = f\cos(\theta) = (-f)(-\cos(\theta)). \tag{3.60}$$

Since only the two first rows are considered for the WPnP problem, we can use the parametrization for the scaled 2×3 rotation. In this parametrization we get in addition to the 8 correct solutions, 4 solutions where $x = y = 0$, corresponding to the zero scale solution. Using the automatic generator from Chapter 1 we created a polynomial solver for this parametrization. The elimination template was of size 208 × 220.

Experimental Evaluation

In this section we experimentally evaluate the polynomial solvers from the previous sections. The sizes of the elimination templates and action matrices for the three methods can be seen in Table 3.1.

	2×2-sym.	2-sym.	No sym.	Wu
Elimination template	26×34	138×154	231×263	208×220
Action matrix	8×8	16×16	32×32	12×12
Runtime	0.22 ms	0.79 ms	1.9 ms	0.91 ms

Table 3.1: Size of the elimination template and action matrix for the four solvers.

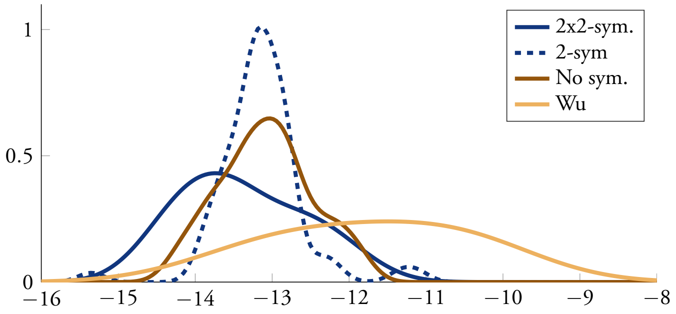


Figure 3.1: Histogram over the residuals for 1000 random instances. For the solver using the parametrization from Wu [227] we prune the zero-scale solutions before computing the residuals.

We evaluated the four solvers on synthetic instances. Figure 3.1 shows the distribution of the residuals over 1000 random instances. All solvers are sufficiently stable for practical purposes. The solver using the parametrization from Wu has slightly worse performance. For the 1000 instances, the maximum relative error in the best residual was less than 10^{-4} for all methods. The average runtimes are shown in Table 3.1.

Degeneracies

Under some conditions the problem changes nature and the polynomial solvers break down. For this problem we empirically found that this happens when either the structure is planar ($a_3 = 0$) or if two of the singular values are equal ($a_1 = a_2$ or $a_2 = a_3$). Close to these configurations the solvers become numerically unstable. We compare the performance of the polynomial solver on random instances close to these degeneracies. We generated random instances where the third singular value approached zero and instances where $|a_2 - a_3|$ approached zero. Figure 3.2 shows the percentage of successful instances (defined as all residuals less than 10^{-6}) as we approach the degenerate configurations. The solver using the parametrization from Wu [227] is the most sensitive to close to degenerate configurations. We can also see that the solver which only uses the two-fold symmetry (2-sym.) has the best performance for the close to planar case ($a_3 = 0$).

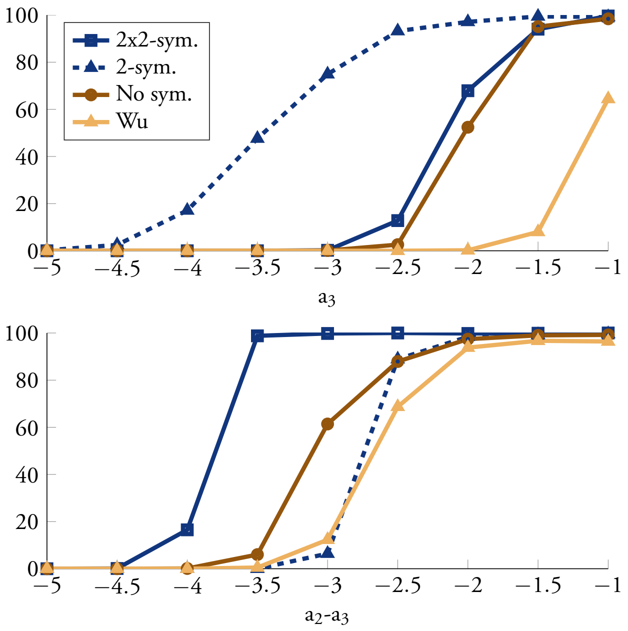


Figure 3.2: Percentage of successful instances (all residuals smaller than 10^{-6}) when the structure is close to planar, i.e. $a_3 \approx 0$ (Top) and when there are two almost equal singular values, i.e. $a_2 \approx a_3$. (Bottom)

3.5 Conclusions

In this chapter we have revisited the symmetries studied in Ask et al. [13] and Kuang et al. [119]. These symmetries can be used to construct smaller elimination templates by only considering a subset of the basis monomials in the quotient ring. Our contribution is a more clear presentation of the symmetries from [13, 119] and we provide some more theoretical motivations why these methods work.

This also revealed how to handle multiple independent symmetries, resulting in even more compact polynomial solvers. Furthermore, we have shown that these symmetries are not restricted to theoretical examples, but occur in real problems from computer vision.

Chapter 4

Basis Selection for Minimal Problems

Many state-of-the-art polynomial solvers in computer vision are based on Gröbner bases and the action-matrix method, and there are now powerful tools available for the automatic generation of such solvers (see Chapter 1 and [122, 134, 108]). In this chapter we target a specific part of this pipeline, namely the choice of monomial basis in the quotient ring. We will show how careful selection of the monomial bases can give significant speed-up in the resulting solvers. Previously, little attention has been paid to the choice of basis to gain speed in polynomial solvers, and usually a Gröbner basis is used to select the monomial basis. We will in the chapter describe how we can test *all possible* Gröbner bases. We will further show that going beyond Gröbner bases leads to faster solvers in a number of cases.

Specifically, our contributions in this chapter are:

- Minimizing elimination template size by enumerating all possible Gröbner bases for an ideal.
- A heuristic method for sampling feasible monomial bases which do not come from any Gröbner basis.
- State-of-the-art performance on a number of geometric estimation and calibration problems in terms of speed.

4.1 Related Work

When creating polynomial solvers in computer vision, the basis for the quotient ring $\{b_1, \ldots, b_K\}$ is typically chosen as the standard monomials from the Gröbner basis w.r.t. the monomial ordering GRevLex (this is e.g. done in Chapter 1 and [122, 134, 108]). However, this is an arbitrary choice and the methods work for any basis. In this chapter we focus on the problem of selecting this basis with the aim of reducing the size of the elimination template. We show that for some problems there are better choices that yield significantly faster solvers.

Given a monomial basis it is still a difficult problem to find the smallest elimination template. In this work we use the automatic generator from Chapter 1 ([134]) to construct the templates, but this method is not guaranteed to find the optimal template. The results in this chapter are w.r.t. this particular template construction method. However, any other method for constructing the template (such as the one from [122]) could be substituted.

Our approach can be used to optimize other characteristics of the solvers as well, such as accuracy or stability. However, for practical purposes these are typically secondary to runtime as long as they are sufficiently good. For example if you are estimating the focal length it usually does not matter whether the errors are 10^{-6} or 10^{-16}.

In [34], Byröd et al. presented different methods for choosing the basis during runtime. However in their setting the size of the template was fixed, and the online basis selection was done solely to improve the numerics of the solver. In [113] the authors further improved stability by carefully selecting the so-called *permissible* monomials (i.e. the set from which the basis is chosen from in [34]).

4.2 Exhaustive Search over Gröbner Bases

For an ideal I, the (reduced) Gröbner basis depends on the monomial ordering chosen in the polynomial ring $\mathbb{K}[X]$. Different orderings can yield different Gröbner bases, and thus different sets of standard monomials. For polynomial solvers in computer vision, the most popular ordering is GRevLex [43], since it has been empirically observed to typically give small elimination templates [202, 200, 122, 134]. It has also been noted in computational algebraic geometry and cryptography [190] that graded orderings [43] (i.e. archimedean [181]) often lead to faster Gröbner basis computations compared to, e.g. lexicographical order-

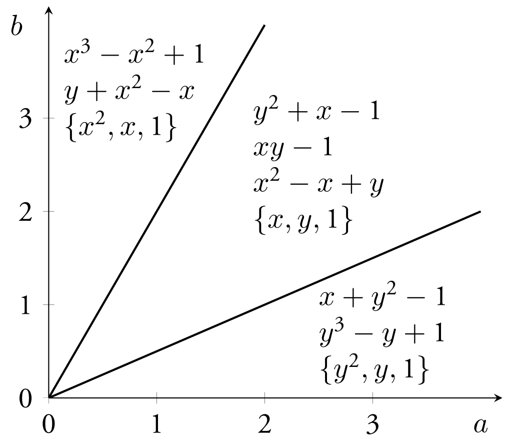


Figure 4.1: The Gröbner fan of the ideal $I = \langle x + y^2 - 1, x\, y - 1\rangle$ consists of three two-dimensional cones. For each cone, there is exactly one reduced Gröbner basis of I, and a corresponding basis for $\mathbb{K}[X]/I$ of standard monomials. All monomial orderings generated by all weight vectors from one cone give the same reduced Gröbner basis of I. Hence, there are exactly three different reduced Gröbner bases for I over all possible different monomial orderings.

ings [43]. However, there exist examples where this is not the case. Hence, this suggests to investigate the efficiency of Gröbner basis (and hence action matrix) construction w.r.t. all possible different monomial orderings.

4.2.1 Gröbner Fans

While there are infinitely many different monomial orderings, Mora and Robbiano [165] showed that for a given ideal I there are only finitely many different reduced Gröbner bases [69]. To present this theory is beyond the scope of this chapter, but we will try to describe the main ideas and relate how this can be used in our problem setting. The set of all reduced Gröbner bases of an ideal can be computed [69, 97, 96] using the *Gröbner fan* of the ideal [165, 206]. The Gröbner fan of an ideal was defined by Mora and Robbiano in 1988 [165]. It is a finite fan of polyhedral cones indexing the distinct monomial initial ideals with respect to monomial orderings or, equivalently, indexing the reduced Gröbner bases of the ideal. See [165, 206, 69] for the full account of the theory.

Here we will illustrate it on a simple example computed using the software package Gfan [96, 97]. Consider the polynomial system $I = \langle x+y^2-1, x\,y-1\rangle$. Figure 4.1 shows the Gröbner fan of I together with the corresponding reduced Gröbner bases and standard monomials. It consists of three two-dimensional cones. For each cone, there is exactly one reduced Gröbner basis of I, giving in total three different reduced Gröbner bases. To connect the different reduced Gröbner bases to the fans in Figure 4.1, consider the exponent vectors $[a, b]^\top$ that correspond to monomials $x^a y^b$, e.g. $[2, 3]^\top$ represents x^2y^3. Now, for every monomial ordering $\prec$ on $\mathbb{C}[x, y]$ one can find a (set of) real non-negative (weight) vectors $w \in \mathbb{R}^2$ such that if $x^a y^b \prec x^c y^d$, then $[a, b] \cdot w \le [c, d] \cdot w$. In this way, every ordering is connected to a set of its (compatible) real weight vectors. Finally, for a fixed I, the union of all the sets of weight vectors corresponding to all monomial orderings producing the same reduced Gröbner basis is a full (here two) dimensional cone in $\mathbb{R}^2$. There are only finitely many such cones for a fixed I. In our situation, there are three two-dimensional cones, see Figure 4.1.

4.2.2 Building Minimal Solvers using Gröbner Fans

In Chapter 1 we presented an automatic generator for polynomial solvers which was evaluated on a large test-bed of polynomial equation systems from geometric computer vision. Even though some of the problem formulations are no longer state-of-the-art for their respective problem, they still serve as a good benchmark set to test our methods. For each of these problems we tried to compute the Gröbner fan, aborting the computations if they lasted more than 12 hours. Using the automatic generator we then constructed a polynomial solver for each of the reduced Gröbner bases found. Table 4.1 (page 111) shows some problems where we were able to find a smaller elimination template compared to using the GRevLex basis. Note that the number of Gröbner bases can increase very quickly and it is not always tractable to compute the complete Gröbner fan for larger problems. For the six point relative pose with shared radial distortion problem we ran the Gröbner fan computation for a week before aborting the computation.

Figure 4.2 shows a histogram of the different template sizes for the solvers constructed from the Gröbner fan for the P4PFR formulation from Bujnak et al. [28]. Many of the found bases yield very large templates. To avoid these uninteresting bases, as well as the long runtimes for computing the Gröbner fan, we propose to use a guided random sampling approach in the next section.

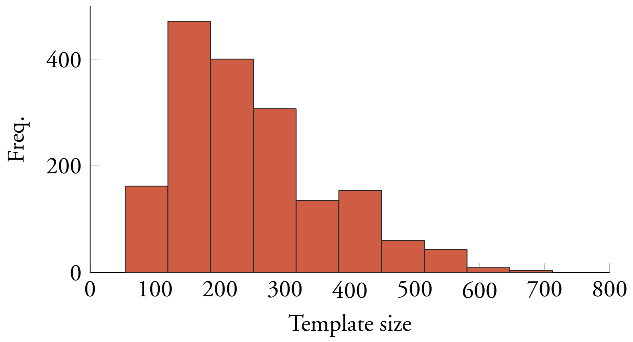


Figure 4.2: Template size (rows) for the Gröbner fan bases for the P4PFR formulation from Bujnak et al. [28].

4.3 Beyond Gröbner Bases

In the previous section we computed all reduced Gröbner bases for a problem and used these to select quotient ring bases. However, it is not necessary to select a standard monomial basis that comes from a Gröbner basis for some monomial ordering, since any spanning and linearly independent set will do. In this section we instead consider bases which do not come as standard monomial bases from any Gröbner basis, and show that for some problems this allows us to find even smaller elimination templates.

4.3.1 Random Sampling for Basis Selection

Once you drop the Gröbner basis constraint you have infinitely many choices for monomial bases, so it is no longer possible to do any exhaustive search. Even if we restrict ourselves to monomials below some fixed degree, the combinatorial explosion of choices often makes it intractable to try them all.

Instead we propose a random sampling approach. The sampling is guided by several heuristics based on empirical observations. We will try to motivate our choices later on, but we will start by describing our proposed algorithm. The heuristics we use for basis selection are:

H1. We try to have as many of the basis monomials and reducible monomials (i.e. αb_k) appearing in the original equations as possible.

H2. We try to minimize the degree in some of the unknowns. This usually helps when the variables occur in an unbalanced way in the equations. E.g. if our problem is parameterized using a quaternion (for rotation) and a focal length, we have seen that it is typically good to try to minimize the degree of the focal length.

H3. We try to select a connected block of monomials.

To generate the initial set of monomials that we sample from we use the following strategy: We start with the monomials occurring in the equations. If these do not contain any basis (see Section 4.3.2) we multiply with all first degree monomials that occur in the equations and add these. If they still do not contain any basis, we again multiply with all the second degree monomials and so on (in some special cases we need to add some extra low-degree monomials to get an independent set). We denote these monomials by $\mathcal{M}$ and the monomials that occur in the original equations by $\mathcal{E} \subset \mathcal{M}$.

Now, to sample a basis we start by randomly choosing a binary weight vector $\omega = \{0, 1\}^n$. This represents the direction we want to minimize in H2. For each monomial $m \in \mathcal{M}$ we assign a weight $w_d(m)$ penalizing the weighted degree using ω. So, e.g. if $\omega = (0, 1, 1)$, the monomial $m = xyz^2$ would have the weighted degree $0 + 1 + 2 = 3$. Next we select an action variable α. It is chosen uniformly in the direction which is minimized by ω. So, in the previous example we would have chosen either y or z. If ω is all zero we choose uniformly from all variables. Note that this α is used only for guiding the random sampling. When we construct the solvers we try every variable as action.

The basis is then sampled iteratively, with one monomial added at a time. Given a partial basis $\mathcal{B} \subset \mathcal{M}$ we select the next monomial to add as follows:

1. Find monomials $\mathcal{M}_{\mathcal{B}} \subset \mathcal{M}$ that are linearly independent from the partial basis $\mathcal{B}$ (see Section 4.3.2)

2. For each monomial $m \in \mathcal{M}_{\mathcal{B}}$ compute a weight

$$w(m) = \mathbb{I}\,(m \in \mathcal{E}) + \mathbb{I}\,(\alpha m \in \mathcal{E} \cup \mathcal{B}) + w_d(m) + \epsilon \tag{4.1}$$

   where ϵ is a small number.

3. Find the neighboring monomials of $\mathcal{B}$ in $\mathcal{M}_{\mathcal{B}}$.

4. Sample proportionally to $w(m)$ from the neighboring monomials. (If there are no feasible neighboring monomials, sample instead from all of $\mathcal{M}_{\mathcal{B}}$).

These steps are iterated until we have a complete basis.

4.3.2 Checking Linear Independence

When we sample basis elements, we need to be able to quickly determine if a set of monomials are linearly independent in the quotient ring $\mathbb{C}[X]/I$ (or typically $\mathbb{Z}_p[X]/I$ since we do most of our calculations in $\mathbb{Z}_p$ to speed up computations and avoid round-off errors).

We start by computing any (reduced) Gröbner basis for the ideal and find the standard monomials $\{b_1, b_2, \ldots, b_K\}$ for this basis. Then, since these monomials form a basis for the quotient ring, we write each $m \in \mathcal{M}$ as

$$m = \sum_k c_k b_k \mod I, \tag{4.2}$$

by simply dividing with the Gröbner basis. This associates vector

$$\boldsymbol{c} = (c_1, c_2, \ldots, c_K), \tag{4.3}$$

to each monomial in $\mathcal{M}$. To check if a set of monomials is linearly independent in the quotient ring, we can now equivalently check if the corresponding vectors are independent in $\mathbb{C}^K$ (or $\mathbb{Z}_p^K$) by performing the standard Gaussian elimination.

4.3.3 Building Minimal Solvers with Sampled Bases

We applied our random sampling strategy in an experiment similar to the one in Section 4.2.2. For each problem, we randomly sampled 100 bases and constructed the corresponding solvers. Some results are shown in Table 4.1. Using our sampling strategy we can find smaller elimination templates for some problems. Note that for some problems the best basis did not come from any Gröbner basis. We were also able to find smaller solvers for problems where the Gröbner fan computation took too long and was aborted.

In Figure 4.3 we show the monomial bases which gave the smallest templates from our sampling scheme for two problems, 8 point relative pose F+λ and 3 point image stitching fλ+R+fλ. The monomials are here represented by their corresponding exponents as vectors. We also show the standard GRevLex bases, that give significantly larger templates for these two problems. In general the GRevLex ordering will lead to a basis that has a low total degree. We have found that, for some problems, if it is possible to keep the maximum degree low in one variable, even if the total degree becomes larger, this is beneficial for the template size. Figure 4.3 left shows an example of this. Another important aspect that we

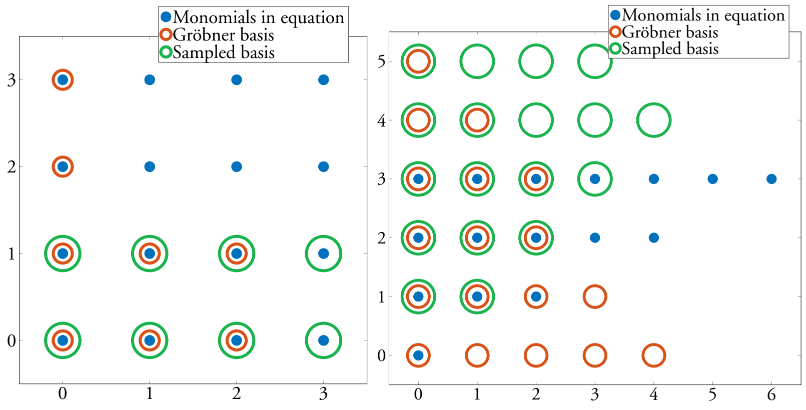


Figure 4.3: The figure shows the basis monomials for two example problems, namely 8pt rel. pose F+λ (left) and 3pt image stitching fλ+R+fλ (right). Both these problems have two variables, and for both these problems the proposed basis sampling scheme gives significantly smaller template compared to the Gröbner basis variants.

have seen, is that we should choose, if possible, monomials within the original equations, as these are available directly. In Figure 4.3 right, all the monomials occurring in the original equations are shown as blue dots. In this case the sampled basis better aligns with the structure of the monomials in the equations compared to GRevLex.

4.3.4 Experiment: Heuristic vs. Uniform Sampling

In this section we show a comparison of our heuristic with sampling basis monomials uniformly. We compare three different approaches: (i) our heuristic sampling from the monomials in $\mathcal{M}$ (as defined in Section 4.3.1), (ii) uniformly sampling from $\mathcal{M}$, and (iii) uniformly sampling from all monomials of the same degree as those in $\mathcal{M}$. Figure 4.4 shows the distribution of the template sizes (number of rows) for 1,000 random samples for the P4Pfr formulation from Bujnak et al. [28]. Our sampling heuristic and the strategy for selecting $\mathcal{M}$ both give significant improvements for this example.

Problem	Author	Original	[134]	GFan+ [134]	(#GB)	Heuristic+[134]
Rel. pose F+λ 8pt	Kuang et al. [118]	12×24	11×20	11×20	(10)	$\mathbf{7 \times 16}$
Rel. pose E+f 6pt	Bujnak et al. [27]	21×30	21×30	$\mathbf{11 \times 20}$	(66)	$\mathbf{11 \times 20}$
Rel. pose f+E+f 6pt	Kukelova et al. [122]	31×46	31×50	31×50	(218)	$\mathbf{21 \times 40}$
Rel. pose E+λ 6pt	Kuang et al. [118]	48×70	34×60	34×60	(846)	$\mathbf{14 \times 40}$
Stitching $f\lambda$+R+$f\lambda$ 3pt	Naroditsky et al. [168]	54×77	48×66	48×66	(26)	$\mathbf{18 \times 36}$
Abs. Pose P4PFR	Bujnak et al. [28]	136×152	140×156	$\mathbf{54 \times 70}$	(1745)	$\mathbf{54 \times 70}$
Rel. pose λ+E+λ 6pt	Kukelova et al. [122]	238×290	149×205	-	?	$\mathbf{53 \times 115}$
Rel. pose λ_1+F+λ_2 9pt	Kukelova et al. [122]	179×203	165×200	$\mathbf{84 \times 117}$	(6896)	$\mathbf{84 \times 117}$
Rel. pose E+$f\lambda$ 7pt	Kuang et al. [118]	200×231	181×200	$\mathbf{69 \times 90}$	(3190)	$\mathbf{69 \times 90}$
Rel. pose E+$f\lambda$ 7pt (elim. λ)	-	-	52×71	37×56	(332)	$\mathbf{24 \times 43}$
Rel. pose E+$f\lambda$ 7pt (elim. $f\lambda$)	Kukelova et al. [129]	$\mathbf{51 \times 70}$	$\mathbf{51 \times 70}$	$\mathbf{51 \times 70}$	(3416)	$\mathbf{51 \times 70}$
Abs. pose quivers	Kuang et al. [114]	372×386	216×258	-	?	$\mathbf{81 \times 119}$
Rel. pose E angle+4pt	Li et al. [144]	270×290	266×329	-	?	$\mathbf{183 \times 249}$
Abs. pose refractive P5P	Haner et al. [77]	280×399	240×324	$\mathbf{157 \times 246}$	(8659)	240×324

Table 4.1: Size of the elimination templates for some minimal problems. For the relative pose problems unknown radial distortion is denoted with λ and unknown focal length with f, and the position describes which camera it refers to. The table shows the original template size from the author, the template size found using the method from Chapter 1 [134] (GRevLex basis), the template size from doing an exhaustive search over Gröbner bases (Section 4.2.2) and the random sampling approach (Section 4.3.1). Missing entries are when the Gröbner fan computation took longer than 12 hours.

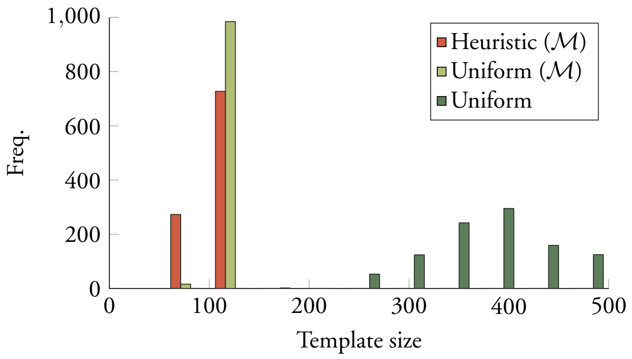


Figure 4.4: Template size (rows) for 1,000 randomly sampled bases for the P4PFR formulation from Bujnak et al. [28].

4.4 Panoramic Stitching $f\lambda + R + f\lambda$

We will now show how our method can be used to construct fast solvers for stitching images from cameras with radial distortion and where the focal length is unknown. This problem was formulated and solved using Gröbner basis techniques in [31]. In [168] a technique for numerically optimizing the size of the elimination template was presented, and a new faster solver with a template of size 54×77 was constructed. In Chapter 1 ([134]) a slightly faster solver was presented, based on a template of size 48×66. We will follow the derivations in [22] and [31] when we construct our solver for two-view stitching using three point correspondences. We will additionally show that we can use the exact same solver to solve the minimal problem of three-view stitching using two point correspondences.

4.4.1 Two View Image Stitching

We assume that we have a camera undergoing some unknown rotation R, taking two images of a number of unknown 3D points $\boldsymbol{X}_i$. We denote the points in the two images with $\boldsymbol{u}_i$ and $\boldsymbol{u}'_i$ respectively. We will describe how we handle the radial distortion later, and will assume that we only need to handle the unknown focal length f just now. The projection equations can then be formulated as

$$\gamma_i \boldsymbol{u}_i = K\boldsymbol{X}_i, \quad \gamma'_i \boldsymbol{u}'_i = KR\boldsymbol{X}_i, \tag{4.4}$$

Author	Execution time (ms)
Proposed	0.16
Larsson et al. [134]	0.38
Byröd et al. [31]	0.89

Table 4.2: Timing of three point stitching with unknown focal length and radial distortion, using MATLAB implementations running on a standard desktop computer.

where γ_i and γ_i' are the depths, and $K = \text{diag}(f, f, 1)$. We can remove the dependence of γ_i, γ_i' and R by solving for $\boldsymbol{X}_i$ and taking scalar products, giving the constraints

$$\frac{\langle K^{-1}\boldsymbol{u}_j, K^{-1}\boldsymbol{u}_k\rangle^2}{|K^{-1}\boldsymbol{u}_j|^2|K^{-1}\boldsymbol{u}_k|^2} = \frac{\langle \boldsymbol{X}_j, \boldsymbol{X}_k\rangle^2}{|\boldsymbol{X}_j|^2|\boldsymbol{X}_k|^2} = \frac{\langle K^{-1}\boldsymbol{u}_j', K^{-1}\boldsymbol{u}_k'\rangle^2}{|K^{-1}\boldsymbol{u}_j'|^2|K^{-1}\boldsymbol{u}_k'|^2}, \tag{4.5}$$

for two points j and k. Cross-multiplying with denominators will give polynomials in the unknown f. We will now add radial distortion to our problem, and model it using Fitzgibbon's division model [65] so that for the radially distorted image coordinates $\boldsymbol{x}_i$ we have $\boldsymbol{u}_i \sim \boldsymbol{x}_i + \lambda\boldsymbol{z}_i$, where $\boldsymbol{z}_i = \begin{bmatrix}0 & 0 & x_i^2 + y_i^2\end{bmatrix}^T$, and λ is the radial distortion parameter. Inserting this into (4.5) gives us our final constraints in the unknown λ and f. Using two points will only give us one equation so we need at least three point correspondences (this actually gives three constraints, so it is slightly over-determined, but we only use two of the equations).

We have run both the exhaustive Gröbner basis selection and our proposed basis sampling scheme, see Table 4.1. The Gröbner bases do not give any improvement over the state-of-the-art solver but our sampling gives a significantly smaller template of size 18×36.

4.4.2 Three View Image Stitching

The constraints (4.5) only compare pairs of images, using two point correspondences. So if we, instead of having three point correspondences in two views, have two point correspondences in three views, we get the same type of constraints, namely

$$\frac{\langle K^{-1}\boldsymbol{u}_1, K^{-1}\boldsymbol{u}_2\rangle^2}{|K^{-1}\boldsymbol{u}_1|^2|K^{-1}\boldsymbol{u}_2|^2} = \frac{\langle K^{-1}\boldsymbol{u}_1', K^{-1}\boldsymbol{u}_2'\rangle^2}{|K^{-1}\boldsymbol{u}_1'|^2|K^{-1}\boldsymbol{u}_2'|^2}, \tag{4.6}$$

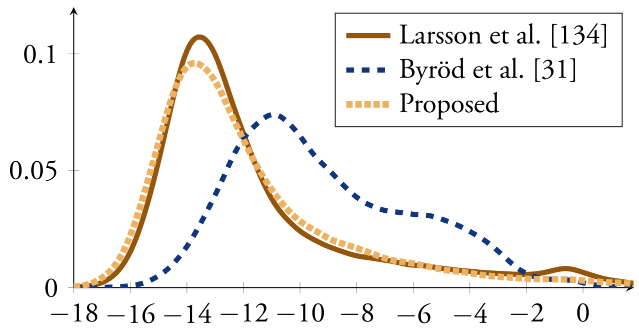


Figure 4.5: The figure shows histograms of equation residuals for 10,000 examples of the 3 pt stitching problem.

and

$$\frac{\langle K^{-1}\boldsymbol{u}_1'', K^{-1}\boldsymbol{u}_2''\rangle^2}{|K^{-1}\boldsymbol{u}_1''|^2|K^{-1}\boldsymbol{u}_2''|^2} = \frac{\langle K^{-1}\boldsymbol{u}_1', K^{-1}\boldsymbol{u}_2'\rangle^2}{|K^{-1}\boldsymbol{u}_1'|^2|K^{-1}\boldsymbol{u}_2'|^2}, \tag{4.7}$$

where double primes are used for image three. We can hence use the exact same solver to solve this case. In this case we have a true minimal case, since we only get two constraints on f and λ.

4.4.3 Evaluation

We have implemented our solver in MATLAB, where all image coordinate input and manipulation were done using mex-compiled C++ routines. In order to have a fair comparison of our method with [134] and [31], we modified their code so that the corresponding image coordinate manipulations also were done using mex-compiled code. The timing comparison is shown in Table 4.2, and one can see a clear speed-up. The solvers were run on a standard desktop computer. In order to check the numerical stability of our solver, we generated synthetic data, and evaluated the equation residuals. The results can be seen in Figure 4.5. In order to see how well our method works in practice, we did an automatic panoramic stitching of two images, with a fish-eye lens and unknown focal length, shown to the left in Figure 4.6. We then ran our solver in a standard RANSAC framework, with tentative correspondences based on SURF features and descriptors. The results can be seen to the right in Figure 4.6. Here the panoramic image was done without any blending in order to show the correctness of the stitching. The trans-

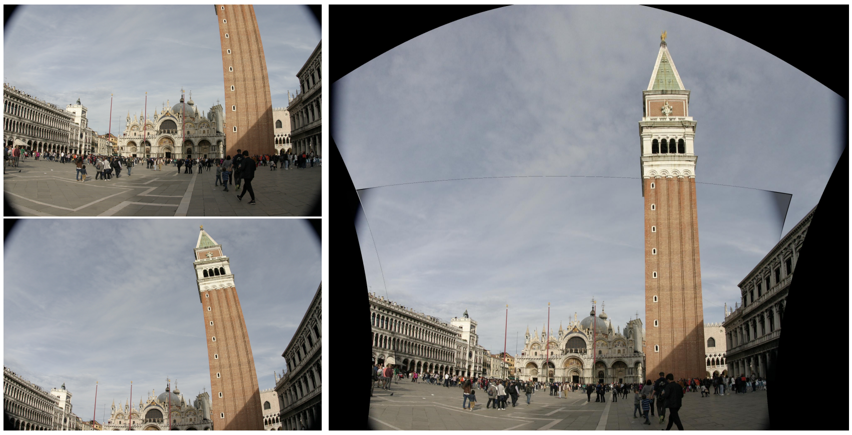


Figure 4.6: Stitching of two images with large radial distortion using on our three-point solver in a standard RANSAC framework. The resulting panorama (right) is based on the best RANSAC three-point solution without any additional non-linear refinement.

formation used was based on the best RANSAC solution from our solver based on only three point correspondences, without any further refinement.

4.5 Relative Pose $E + f\lambda$

As another example we consider the relative pose problem where the calibration and distortion parameter are known for only one of the two cameras. The goal is to find a fundamental matrix F and distortion parameter λ that satisfy the epipolar constraints

$$\left[\hat{x}_i,\ \hat{y}_i,\ 1\right] F \left[x_i,\ y_i,\ 1 + \lambda(x_i^2 + y_i^2)\right]^T = 0, \tag{4.8}$$

as well as a focal length f, that makes

$$E = F\text{diag}(f, f, 1) \tag{4.9}$$

an essential matrix. The problem is minimal with seven point correspondences and has 19 solutions. The first solver was presented by Kuang et al. [118] and was recently improved by Kukelova et al. [129].

4.5.1 Formulation of Kuang et al.

Now we give a brief overview of the formulation used in Kuang et al. [118]. The scale of the fundamental matrix is fixed by setting $f_{33} = 1$, and the epipolar constraints yield seven equations in the monomials

$$\{\lambda f_{13}, \lambda f_{23}, \lambda, f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, 1\}. \tag{4.10}$$

Using the first six equations, Kuang et al. linearly eliminate the first two columns of the fundamental matrix[1]

$$[f_{11}, f_{12}, f_{21}, f_{22}, f_{31}, f_{32}]^T = G\,[\lambda f_{13}, \lambda f_{23}, \lambda, f_{13}, f_{23}, 1]^T \tag{4.11}$$

where $G \in \mathbb{R}^{6\times 6}$. Finally, the last equation expresses the monomial λf_{13} as a quadratic function $h(\lambda, f_{13}, f_{22})$, which gives the additional equation

$$\lambda f_{13} - h(\lambda, f_{13}, f_{23}) = 0. \tag{4.12}$$

Parametrizing the inverse focal length w, the essential matrix is given by $E = F\text{diag}(1, 1, w)$, and it must satisfy the equations

$$2EE^TE - \text{tr}(EE^T)E = 0, \quad \det(F) = 0. \tag{4.13}$$

This gives 11 equations in unknowns w, λ, f_{13} and f_{23}. Using these equations, Kuang et al. [118] constructed a polynomial solver with a template of size 200×231.

Computing the Gröbner fan, we found that there are 3190 different reduced Gröbner bases for this problem. Constructing solvers for all of these bases, we found an elimination template of size 69×90. Applying the random approach in Section 4.3.1, we did not find any better solver. While this solver is significantly smaller than the original solver from Kuang et al. [118] (200×231), it is still slightly larger than the state-of-the-art solver from Kukelova et al. [129] (51×70).

4.5.2 Formulation of Kukelova et al.

In [129] the authors present another formulation for this problem based on computing elimination ideals to eliminate both the radial distortion and focal length.

[1]Note that here we have the focal length and distortion on the right side of F, while it was on the left in [118].

Since the radial distortion makes the epipolar constraints non-linear they first employ a lifting technique to remove the non-linearity. They introduce new variables y_1, y_2 and y_3 and construct an extended fundamental matrix as in [21],

$$\hat{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} & y_1 \\ f_{21} & f_{22} & f_{23} & y_2 \\ f_{31} & f_{32} & f_{33} & y_3 \end{bmatrix}, \tag{4.14}$$

together with the equations $y_i = \lambda f_{i3}$. Now the epipolar constraints are linear constraints on $\hat{F}$,

$$\left[\hat{x}_i,\ \hat{y}_i,\ 1\right] \hat{F} \left[x_i,\ y_i,\ 1,\ x_i^2 + y_i^2\right]^T = 0. \tag{4.15}$$

Using the (now) linear constraints on $\hat{F}$ they parametrize it using four unknowns. Finally using the elimination ideal trick they eliminate both the focal length and radial distortion parameter to get new polynomial constraints on the elements on $\hat{F}$. Using these new equations, they were able to construct a solver with a template size 51×70 using the automatic generator from [122].

We computed the Gröbner fan for this parametrization and found that there are 3416 reduced Gröbner basis. Among these we found no solver better than the GRevLex solver built by Kukelova et al. We also performed the random sampling approach without finding any improvement. This matches our intuition that for equation systems where the unknowns are balanced in the monomials, GRevLex performs very well.

4.5.3 Our Approach

Empirically we have seen that our basis selection approach works best when the monomials appear in some unbalanced way in the equations. In the parametrization from Kukelova et al. [129], the only unknowns are the nullspace parameters from the linear equations.

To get more imbalanced equations, we propose another formulation which is a combination of the two previous approaches. In particular, we use the elimination ideal trick to eliminate the focal length, but keep the radial distortion parameter as an unknown. This avoids the extra unknowns introduced by the lifting in (4.14). Using similar linear eliminations as Kuang et al. [118], the fundamental matrix is expressed in λ, f_{13} and f_{23}. Then, instead of directly parametrizing the focal length and adding the essential matrix constraints (4.13), we add the eliminated constraints for one-sided focal length from [129] which only depend on the

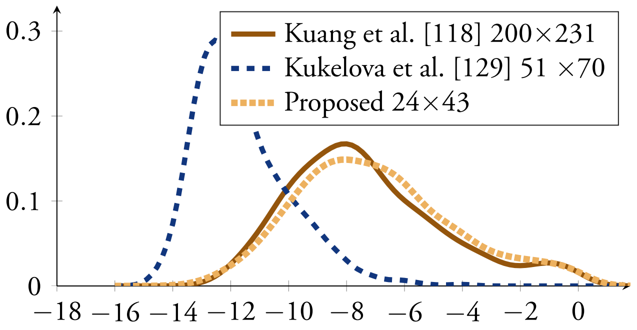


Figure 4.7: Relative error in focal length for 1,000 random instances.

elements of the fundamental matrix. Together with the constraint from (4.12), we get five equations in only three unknowns. Computing the Gröbner fan, we find 332 different reduced Gröbner bases. The best solver was of size 37×56. Finally using the random sampling approach we find a solver with an elimination template of size 24×43.

4.5.4 Evaluation

We performed a synthetic experiment to evaluate the numerical stability of the new solver. We generated 1,000 random (but feasible) synthetic instances. The calibration parameters were set to $f_{gt} = 10$ and $\lambda_{gt} = -0.1$. For each solver we recorded the solution with the smallest focal length error. Figure 4.7 shows the distribution of the $\log_{10}$ relative focal length error $\frac{|f-f_{gt}|}{f_gt}$ for all 1000 instances. The numerical stability of the new solver is similar to the solver from Kuang et al. [118]. Note that while the stability is worse than the solver from Kukelova et al. [129], it is still stable enough for practical purposes. The new solver is however significantly faster with an average runtime of 1.2 ms, compared to 10 ms for the solver from Kukelova et al. [129] (both solvers are implemented in MATLAB). Note that this increase in speed is not only due to the smaller elimination template, but the coefficients in the template are less complex and cheaper to compute.

4.6 Conclusions

We have explored how basis selection can be used to make polynomial solvers based on the action matrix method faster. The concept of Gröbner fans is an efficient representation of the possible reduced Gröbner bases that arise from (infinitely many) different monomial orderings. This gives us a tool to enumerate and test all monomial bases that arise from different Gröbner bases. We have shown that this in some cases gives significantly smaller elimination templates, and hence much faster solvers. We have also introduced a novel sampling scheme, that optimizes some heuristic criteria that we have experimentally found to often give small templates. Our initial motivation for sampling was that the calculation and testing of all Gröbner fans in some cases takes very (or even unfeasibly) long time, but we found that going beyond Gröbner bases can yield even smaller templates. Our motivation has here been to optimize the template size but the framework could easily be modified to optimize other criteria such as numerical stability (as was done in Kuang et al. [113]). We have tested our method on a large number of minimal problems, and shown that we get significant speed-ups in many cases. We have also explored in more depth how our method can be used in two applications, namely panoramic stitching with unknown focal length and radial distortion and relative pose with unknown one-sided focal length and radial distortion.

Chapter 5

Absolute Pose with Unknown Focal Length and Radial Distortion

In this chapter we revisit the problem of camera pose estimation with unknown focal length and radial distortion (P4PFR). Previous approaches suffer from artificial degeneracies which come from their formulation and not the geometry of the original problem. In this chapter we show how to avoid these false degeneracies to create a more robust solver. Combined with recently published techniques for Gröbner basis solvers we are also able to construct solvers which are significantly smaller. Finally we show that a similar approach can be directly applied to the P3.5PF problem to get a non-degenerate solver, which is competitive with the current state-of-the-art.

This chapter is based on the paper [137].

5.1 Introduction

Estimating the pose of a camera from minimal 2D-3D point correspondences is an important problem in geometric computer vision, as the minimal solvers often form the building blocks for 3D reconstruction frameworks. For estimating a calibrated camera only three points are required [70, 110]. If the intrinsic parameters are only partially known, more point correspondences are necessary.

When all the intrinsic parameters are unknown, the direct linear transform

(DLT) [81] algorithm applies, which uses at least five and a half point correspondences. In practice, considering that a part of the intrinsic parameters are usually known a priori, the DLT algorithm suffers from over-parametrization, and usually gives inaccurate estimates because of overfitting in the presence of noisy data.

The most common case is that all intrinsic parameters are known except for the focal length. For the pose estimation with unknown focal length there have been many proposed minimal solvers using four points [214, 26, 234] and recently Wu [227] presented the first truly minimal solver using 3.5 points (obtained by ignoring one image point coordinate).

In this chapter we consider the problem when the camera suffers from unknown radial distortion which needs to be estimated alongside the camera pose. The problem with both unknown radial distortion and unknown focal length becomes minimal with four 2D-3D point correspondences and is usually denoted P4PFR (see Figure 5.1). It was first solved by Josephson and Byröd [102], however the elimination template for this solver was quite large (1134×720), which limits its practical use. In [28] Bujnak et al. presented polynomial solvers with much smaller elimination templates by considering the planar and non-planar problems separately.

In [125] Kukelova et al. presented a non-minimal solver which uses five point correspondences. The solver is very fast but has the drawbacks of requiring more data and solving an overconstrained problem.

The P4PFR problem becomes degenerate when the 3D points lie in a plane parallel to the image plane. In this case there exist infinitely many solutions by translating the camera towards the plane and changing the focal length. This degeneracy is inherent to the problem itself and cannot be avoided. However the previous solvers from [102] and [28] both suffer from additional degeneracies which are artificial in the sense that they come from the specific formulation used, instead of the geometry of the original problem.

In this chapter we show how to avoid these unnecessary degeneracies and construct minimal solvers which are both more robust and have better performance. The main contributions of this chapter are

- We present new polynomial solvers for P4PFR which outperform the current state-of-the-art.
- We avoid the nullspace and rotation degeneracies which are present in competing methods.

- Using the elimination ideal technique for minimal solvers from Kukelova et al. [129] we are able to find additional constraints on the camera matrix and produce significantly smaller elimination templates.
- We create a single solver which works for both planar and non-planar data.
- We also apply our approach to the case with no radial distortion (P3.5Pf) and get results comparable to the current state-of-the-art.

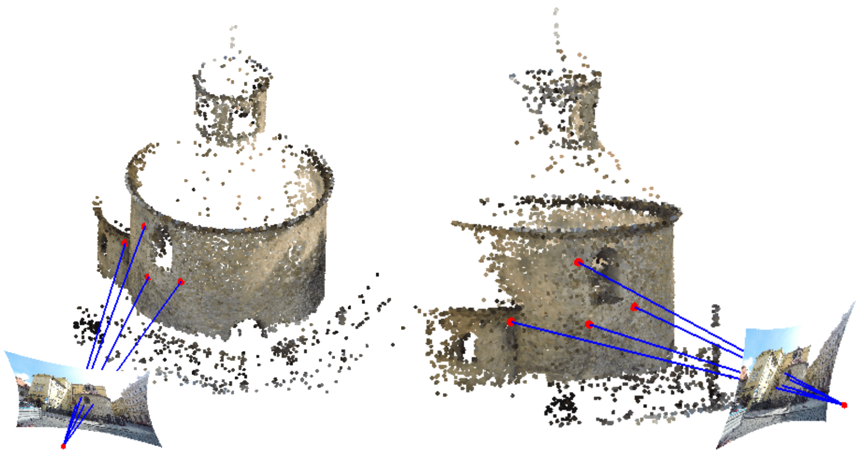


Figure 5.1: The camera pose, focal length and radial distortion are estimated from four 2D-3D correspondences. The examples above are from the *Rotunda* dataset (Section 5.4.4).

5.2 Background and Previous Work

5.2.1 Unknown Radial Distortion and Focal Length

If the image has undergone radial distortion, the standard pinhole camera model is no longer valid, and more complicated models are required to handle the extra non-linearity. For this, different models have been proposed, e.g. [52, 65, 40]. One of the more popular models (which is used in e.g. [16, 100, 102, 28, 126]) is the one parameter division model presented by Fitzgibbon [65], since it provides a good trade-off between representation accuracy and model complexity.

The model assumes the undistorted image coordinates (u_u, v_u) are given by

$$(u_u, v_u) = \frac{1}{1 + k(u_d^2 + v_d^2)}(u_d, v_d) \tag{5.1}$$

where (u_d, v_d) are the distorted image coordinates observed in the image. The strength of the distortion is controlled by the parameter k, which is typically negative in the presence of barrel distortion.

In this model the projection equations can be written as

$$\lambda_i \begin{pmatrix} u_i \\ v_i \\ 1 + kd_i \end{pmatrix} = P\boldsymbol{X}_i = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \tag{5.2}$$

where $d_i = u_i^2 + v_i^2$. We assume that the camera has zero skew, unit aspect ratio and that the principal point lies in the center of the image. The camera matrix must then satisfy

$$P \sim K[R\,\boldsymbol{t}], \quad K = \text{diag}(f, f, 1), \quad R^T R = I \tag{5.3}$$

Since the camera matrix has 7 degrees of freedom and the radial distortion parameter k is unknown, the problem becomes minimal with four 2D-3D point correspondences.

5.2.2 Minimal Solver from Josephson and Byröd

The first solution to the P4PFR problem was presented in [102] by Josephson and Byröd. In the paper they proposed to parametrize the problem directly using quaternions for the rotation matrix. The scale of the camera matrix was fixed by setting the first element of the quaternion to one. Finally, by using a clever choice of coordinate systems, they were able to eliminate the translation. This formulation leads to 5 equations in the 5 remaining unknowns (three quaternion parameters, the inverse focal length and the distortion parameter).

Using the techniques from [33] they constructed a Gröbner basis based solver for this system. The solver performs linear elimination on a matrix of size 1134×720 followed by eigendecomposition of a 24×24 matrix. The large template results in poor numerical stability and long running time, both of which make the solver unsuitable for practical applications.

The solver works for both planar and non-planar 3D points, but has a non-trivial degeneracy for any 180° rotation introduced by setting the first quaternion element to one. In addition, the number of solutions from this solver (24) is unnecessarily doubled, because of an ambiguity between the focal length and the rotation [227] (see also Section 3.4.1 in Chapter 3).

5.2.3 Minimal Solver from Bujnak et al.

In [28] Bujnak et al. presented another minimal solver for the P4PFR problem. Their solver is based on the observation that if we eliminate the scalars λ_i from the first two equations in (5.2), we get

$$v_i P_1 \boldsymbol{X}_i - u_i P_2 \boldsymbol{X}_i = 0, \quad i = 1,2,3,4 \tag{5.4}$$

which is linear in the first two rows of the camera matrix and does not contain the radial distortion parameter k. These equations constrain the projections to lie on the lines going from the image plane origin to the image points. Since there are four such equations, this can be rewritten as

$$M\boldsymbol{v} = \mathbf{0}, \quad M \in \mathbb{R}^{4\times 8}, \tag{5.5}$$

where $\boldsymbol{v} = [p_{11}, p_{12}, p_{13}, p_{14}, p_{21}, p_{22}, p_{23}, p_{24}]$. This is used to parametrize the first two rows of the camera matrix using only four parameters,

$$\boldsymbol{v} = \alpha_1 \boldsymbol{v}_1 + \alpha_2 \boldsymbol{v}_2 + \alpha_3 \boldsymbol{v}_3 + \alpha_4 \boldsymbol{v}_4. \tag{5.6}$$

where $\{\boldsymbol{v}_i\}_{i=1}^4 \subset \mathbb{R}^8$ is a basis for the nullspace of M. Since the reprojection equations are homogeneous in the camera matrix, the scale is fixed by setting $\alpha_4 = 1$. From (5.2) the remaining linearly independent equations can be written as

$$(1 + kd_i)P_1 \boldsymbol{X}_i - u_i P_3 \boldsymbol{X}_i = 0, \quad i = 1,2,3,4 \tag{5.7}$$

Collecting the terms properly will lead to

$$A[p_{31}, p_{32}, p_{33}, p_{34}]^T = B[\boldsymbol{\alpha}, k\boldsymbol{\alpha}, k, 1]^T, \tag{5.8}$$

where $A \in \mathbb{R}^{4\times 4}$, $B \in \mathbb{R}^{4\times 8}$ and $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3]^T$. Multiplying with the inverse of A, the third camera row is expressed in the four unknowns $\alpha_1, \alpha_2, \alpha_3$ and k.

The left-most 3×3 part of the camera matrix should correspond to KR (5.3). This gives constraints that the rows should be pairwise orthogonal and that the first two rows should have the same norm, as follows

$$p_{21}p_{31} + p_{22}p_{32} + p_{23}p_{33} = 0, \tag{5.9}$$

$$p_{11}p_{31} + p_{12}p_{32} + p_{13}p_{33} = 0, \tag{5.10}$$

$$p_{11}p_{21} + p_{12}p_{22} + p_{13}p_{23} = 0, \tag{5.11}$$

$$p_{11}^2 + p_{12}^2 + p_{13}^2 - p_{21}^2 - p_{22}^2 - p_{23}^2 = 0. \tag{5.12}$$

Using these equations Bujnak et al. [28] created a solver which performs Gaussian elimination on a template of size 136×152. Once the camera matrices are found, the focal length can be recovered by solving a quadratic polynomial. This formulation has 16 solutions, but only 12 of these are geometrically valid for the original problem.

This solver greatly reduces the template size compared to the solver from Josephson and Byröd [102]. However, it only works for non-planar 3D points. In the paper, the authors proposed a special solver to handle the planar case separately. The planar solver has an elimination template of size 12×18.

5.3 Our Approach for P4Pfr

Now we will present our approach for solving the P4Pfr problem. It builds on the formulation from Bujnak et al. [28], but improves it in three key aspects:

- The artificial degeneracy introduced by fixing the scale with $\alpha_4 = 1$ is removed.
- Using the recent elimination ideal technique [129] we get a significantly smaller elimination template.
- We remove the planar degeneracy and create a unified solver that works for both planar and non-planar scenes.

5.3.1 Removing Nullspace Degeneracy

In the solver by Bujnak et al. [28], the scale is fixed by setting $\alpha_4 = 1$ in (5.6). This has the benefit of reducing the number of unknowns by one. However it introduces a degeneracy for any camera matrix which correspond to $\alpha_4 = 0$. Since the nullspace is only determined up to a 4×4 change of variables, this essentially excludes a random set of camera matrices. It is unlikely in practice that the true solution has $\alpha_4 = 0$ exactly, however any solutions close to these degenerate configurations can result in bad numerics (see Section 5.4.3 for experiments on this).

To avoid this degeneracy we propose a simple method for ensuring that the camera matrices which are excluded are geometrically uninteresting. To accomplish this we instead fix the scale by setting $\lambda_1 = 1$. Then for the first point the projection equations become

$$\begin{pmatrix} u_1, & v_1, & 1 + kd_1 \end{pmatrix}^T = P\boldsymbol{X}_1 \tag{5.13}$$

Using this technique the first point now gives two linear constraints on the first two camera rows, P_1 and P_2. The homogeneous linear system in (5.5) now becomes inhomogeneous,

$$M\boldsymbol{v} = \boldsymbol{b}, \quad M \in \mathbb{R}^{5\times 8}, \quad b \in \mathbb{R}^8 \tag{5.14}$$

and has one additional row. The solutions to this system can be parametrized as

$$\boldsymbol{v} = \boldsymbol{v}_0 + \alpha_1\boldsymbol{v}_1 + \alpha_2\boldsymbol{v}_2 + \alpha_3\boldsymbol{v}_3 \tag{5.15}$$

where $M\boldsymbol{v}_0 = \boldsymbol{b}$ and $\{\boldsymbol{v}_i\}_{i=1}^3 \subset \mathbb{R}^8$ forms a basis for the nullspace of M.

Note that this is essentially the same parametrization as before, but we have made sure that the degeneracy now instead occurs when the first point has zero depth, i.e. the first 3D point coincides with the camera center.

5.3.2 New Camera Matrix Constraints

In [129] Kukelova et al. presented a new technique for using elimination ideals to construct smaller polynomial solvers. The approach is based on the observation that for many problems the equations can be divided into two groups; linear equations which depend on the data and non-linear equations which are independent of the data. By computing elimination ideals [42] for the non-linear equations, it is possible to eliminate some of the unknowns before constructing the elimination template. For a more detailed description of the process see [129].

In the P4PFR problem the non-linear equations are[1]

$$P = \text{diag}(f, f, 1)\begin{bmatrix} R & \boldsymbol{t} \end{bmatrix}, \quad R^T R = sI \tag{5.16}$$

[1]We add the unknown s since the camera matrix is only determined up to scale.

Computing the elimination ideal which eliminates all unknowns except for P yields the following set of equations

$$p_{13}^2 p_{32} - p_{21}^2 p_{32} - p_{22}^2 p_{32} - p_{12}p_{13}p_{33} - p_{22}p_{23}p_{33} = 0 \tag{5.17}$$

$$p_{12}p_{13}p_{32} + p_{22}p_{23}p_{32} - p_{12}^2 p_{33} + p_{21}^2 p_{33} + p_{23}^2 p_{33} = 0 \tag{5.18}$$

$$p_{11}p_{13}p_{32} + p_{21}p_{23}p_{32} - p_{11}p_{12}p_{33} - p_{21}p_{22}p_{33} = 0 \tag{5.19}$$

$$p_{13}^2 p_{31} - p_{22}^2 p_{31} + p_{21}p_{22}p_{32} - p_{11}p_{13}p_{33} = 0 \tag{5.20}$$

$$p_{12}p_{13}p_{31} + p_{22}p_{23}p_{31} - p_{11}p_{12}p_{33} - p_{21}p_{22}p_{33} = 0 \tag{5.21}$$

in addition to the constraints (5.9)–(5.12). These constraints ensure that the first 3×3 part of the camera matrix can be factorized as $\text{diag}(f, f, 1)R$, where R is a scaled rotation. To the best of our knowledge, the constraints (5.17)–(5.21) are new and have not been used in the computer vision literature before.

After adding these new equations, the formulation now correctly has 12 solutions, in contrast to 16 in [28] and 24 in [102]. A closer study revealed that the four solutions removed (compared to [28]) with these new constraints are complex and yield camera matrices for which

$$p_{11}^2 + p_{12}^2 + p_{13}^2 = p_{21}^2 + p_{22}^2 + p_{23}^2 = 0. \tag{5.22}$$

Using the automatic solver generator from [122], we generate a polynomial solver with an elimination template of size 28×40. While the solver is significantly smaller than the non-planar solver from [28] (136×152), it also suffers from the same planar degeneracy.

In [129] the authors also considered radial distortion, but for relative pose. To avoid the extra non-linearity introduced by the radial distortion they used a lifting approach. We tried to apply the same technique for our problem and also eliminate the radial distortion, but the resulting solvers were too large for practical use. The size of the elimination template was 296×330.

5.3.3 Removing Planar Degeneracy

In this section we will extend the formulation to handle both planar and non-planar scenes.

For planar scenes we can without loss of generality assume that all $z_i = 0$, i.e. the 3D points lie in the $xy-$plane. As was noted in Bujnak et al. [28], for such scenes the third column of the A matrix in (5.8) becomes zero, and it is impossible to eliminate p_{33} this way. To avoid this situation we instead add p_{33} as an additional unknown. Then using only three of the four points we can express p_{31}, p_{32} and p_{34} in the unknowns $\boldsymbol{\alpha}$, k and p_{33}, i.e.

$$A[p_{31}, p_{32}, p_{34}]^T = B[\boldsymbol{\alpha}, k\boldsymbol{\alpha}, k, p_{33}, 1]^T \tag{5.23}$$

where $A \in \mathbb{R}^{3\times3}$ and $B \in \mathbb{R}^{3\times9}$. Note that this works for both planar and non-planar data. Since we have only used three of the four equations (5.7), we need to add the last projection equation separately. Using the first point (which was used to fix the scale (5.13)) this equation is simply

$$1 + kd_1 = P_3\boldsymbol{X}_1 \tag{5.24}$$

Studying the equations in Macaulay2 [75] we found that this formulation also has 12 solutions. Using the automatic generator presented in Chapter 1 we generated a solver with template size 38×50. To our surprise this solver did not directly work for planar data.

Further investigations of the problem in Macaulay2 [75] revealed that for planar instances, the monomial basis,

$$\{1, \alpha_1, \alpha_2, \alpha_3, k, p_{33}, \alpha_1\alpha_3, \alpha_2^2, \alpha_2\alpha_3, \alpha_3^2, \alpha_3 k, \alpha_3 p_{33}\} \tag{5.25}$$

which was used for the quotient ring $\mathbb{C}[X]/I$ became linearly dependent. Thus making it impossible to express the action matrix using it. Furthermore we found that if we added the monomial $\alpha_1\alpha_2 k$ to the basis, it would span the quotient ring for both planar and non-planar data. Since the structure of the ideal is different for the two types of instances, different monomials are needed in the elimination template. Using the automatic generator from Chapter 1 we created elimination templates for both planar and non-planar instances and then constructed a single template by taking the union of all the necessary equations. Since the ideals are very similar for both cases, we were able to find a merged template which is only slightly bigger than the template created using non-planar data only.

The final solver performs Gaussian elimination on a single template of size 40×50 and then solves a 13×13 eigenvalue problem (due to the extra basis element). However for any instance only 12 of the eigenvectors correspond to actual solutions. This solver does not suffer from any of the artificial degeneracies present in previous solvers and it works for both planar and non-planar data. Compared to the current state-of-the-art general solver from Josephson and Byröd [102] the size is orders of magnitude smaller (40×50 vs. 1134×720).

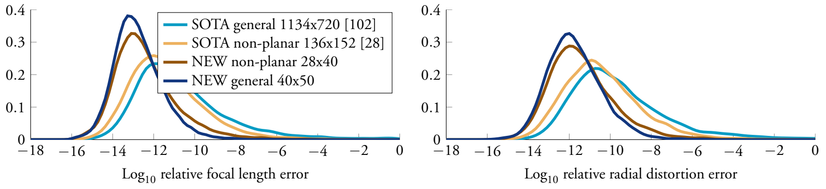


Figure 5.2: Histograms of log_{10} relative errors of the estimated focal lengths (left) and radial distortions (right) for non-planar scenes.

5.4 Experimental Evaluation

5.4.1 Numerical Stability

To evaluate the stability and accuracy of the new P4PFR solvers we use a similar experiment setup as was used in [234, 227]. We generate synthetic scenes by uniformly sampling four 3D points in the box $[-2, 2] \times [-2, 2] \times [2, 8]$ in the camera's local coordinate system. The 3D points are then transformed by a random rotation and translation. The focal length is randomly chosen in the interval $f_{gt} \in [0.5, 2.5]$. Radial distortion using the division model [65] was added to all image points to generate noiseless distorted points. The radial distortion parameter was randomly drawn from the interval $k_{gt} \in [-0.45, 0]$.

To perform the experiment we generated 10000 random scenes as described above. Figure 5.2 shows the histograms of the log_{10} relative errors in the estimated focal length and radial distortion parameter obtained by selecting the real root closest to the ground truth values f_{gt} and k_{gt}. We also ran the same experiment for planar scenes, generated by projecting the four 3D points to the closest plane using SVD. The results are shown in Figure 5.3.

The new general P4PFR (40x50) solver is stable for both the planar and non-planar setting. Moreover, the new solver is more stable than the state-of-the-art general solver from Josephson and Byröd [102] for non-planar setting. The same holds true for the new non-planar solver compared to the state-of-the-art non-planar solver from Bujnak et al. [28].

5.4.2 Noise Experiment

In the next experiment we studied the performance of the new solvers in the presence of image noise. We again compare both presented solvers (the general

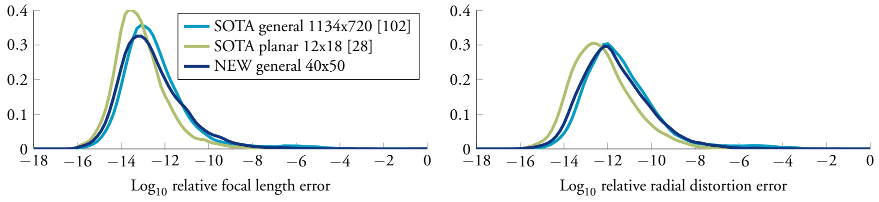


Figure 5.3: Histograms of log_{10} relative errors of the estimated focal lengths (left) and radial distortions (right) for planar scenes.

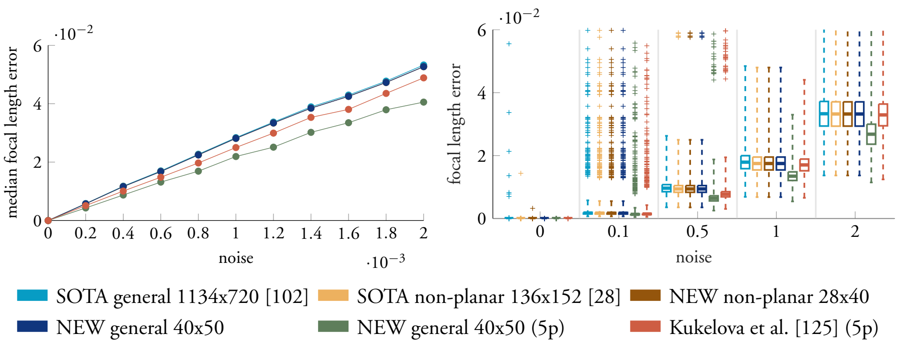


Figure 5.4: Comparison of the errors in the focal length estimated by different solvers for varying levels of noise. The ground truth values were set to $f_{gt} = 1.5$ and $k_{gt} = -0.4$. *Left:* Median focal length error. *Right:* Boxplot of focal length errors.

solver and the non-planar solver) with the the state-of-the-art general solver from Josephson and Byröd [102] and the state-of-the-art non-planar solver from Bujnak et al. [28].

In this experiment we used the same setup as was used in the previous stability experiment, however with the fixed ground truth focal length $f_{gt} = 1.5$ and the fixed radial distortion $k_{gt} = -0.4$. For each noise level 1000 estimates for random scenes and camera positions were made. Figure 5.4 (Left) shows the median focal length errors for different noise levels. Figure 5.4 (Right) shows the errors for the focal lengths using the MATLAB boxplot function which shows values 25% to 75% quantile as a box with horizontal line at median. The crosses show data beyond 1.5 times the interquartile range. Here we only show the errors in the focal length since the other errors are qualitatively similar.

All minimal solvers perform equally well. This is caused by the fact that these solvers are all algebraically equivalent. The only difference is caused by numerical instabilities. This is e.g. visible in Figure 5.4 (Right) for the state-of-the-art general solver by Josephson and Byröd [102] in the noiseless case.

Estimation from Non-Minimal Point Sets

In [125] Kukelova et al. presented a non-minimal solver which uses five point correspondences. To perform a comparison with this solver we generated a fifth point for each of the scenes. The results are included in Figure 5.4 and we can see that in the presence of noise the performance is superior compared to the four point solvers, which is reasonable since more data is used.

Since our solver is based on a nullspace parametrization it is possible to use five points in our solver as well. For estimation with non-minimal point sets the only changes we need to make is to compute an approximate nullspace in (5.14) using SVD and to solve (5.23) in a least squares sense. Figure 5.4 shows that our solver using 5 points is more accurate for noisy data compared to the solver from [125].

5.4.3 Stability Close to Degenerate Configurations

In this section we evaluate the stability of the polynomial solvers close the degenerate configurations. First we consider the quaternion based degeneracy in the solver from Josephson and Byröd [102]. We generated random scenes where the ground truth camera pose was close to the degenerate configuration (i.e. first element of the quaternion representing the rotation is close to zero). Figure 5.5 shows the median relative error in the focal length as the first element of the quaternion tends to zero.

Next we consider the nullspace based degeneracy from Section 5.3.1. To perform the experiment we randomly generate a scene and computed a basis for the nullspace (as in (5.5)). We then find the coefficients α which correspond to the ground truth camera matrix. Next we perform a random rotation in the nullspace which brings α_4 close to zero. Figure 5.6 shows the result for our solver both using the parametrization from Section 5.3.1 and using the degenerate nullspace created as above. Since this degeneracy is also present in the solver from Bujnak et al. [28] we include the results of running their solver with the degenerate nullspace basis as well.

Finally we consider close-to-planar degeneracy. To evaluate the performance

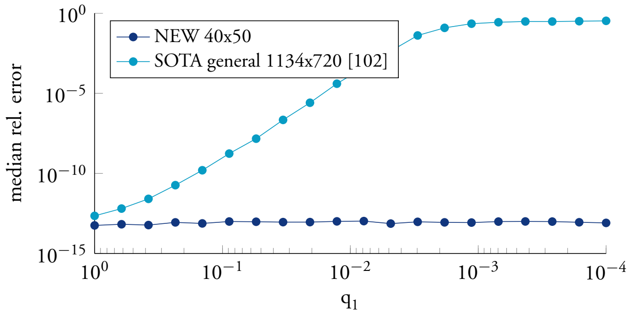


Figure 5.5: The relative focal length error as the first element of the quaternion approaches zero. Each point shows the median error over 1000 instances.

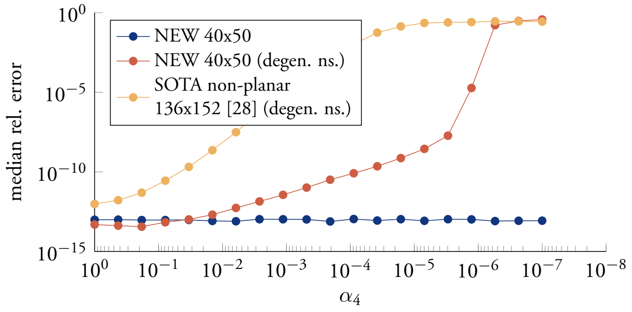


Figure 5.6: The relative focal length error as the last nullspace coefficient α_4 approaches zero. Each point shows the median error over 1000 instances. For our solver we show the results both with and without the scale fixing in Section 5.3.1.

of our new P4PFR solvers on close-to-planar scenes we use a similar experiment setup as was used in [28]. We created a synthetic scene where we were able control the scene planarity by a scalar value a. First, we generated three random non-collinear 3D points. These three points define our plane. Then we randomly generated the fourth point at the distance sa from the plane, where the scale s was the maximum distance from the first three points to their center of gravity. The fourth point was generated such that its distance from the center of gravity was not greater than s. This means that for $a = 0$ we got four points on the plane and for $a = 1$ we got a well defined non-planar four-tuple of 3D points. For each given planarity value a we created a scene consisting of these four 3D points and an additional 100 random 3D points. Each 3D point was projected by a camera

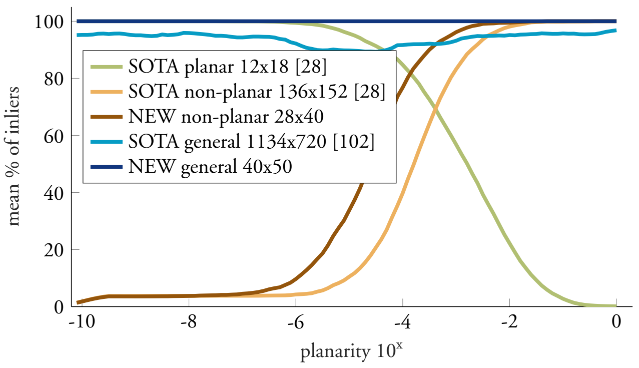


Figure 5.7: Mean of the number of inliers for a near planar scene.

with random feasible orientation and position, $f_{gt} = 1.5$ and $k_{gt} = -0.4$. The first four points were kept noise free, while we added some small noise (standard deviation of 0.5 pixels) to the remaining 100 points.

For each given planarity value a we computed the camera pose, focal length and radial distortion from the first four-tuple of correspondences. This four-tuple was not affected by a noise and hence the only deviation from the ground truth solutions comes from the numerical instability of the solvers itself. To evaluate the impact of this instability we used the estimated camera pose, focal length and radial distortion to project all remaining 3D points to the image plane. Then we measured the number of inliers, i.e. the number of points that were projected closer than one pixel to its corresponding 2D image point.

In this experiment, for each given planarity value a we created 100 random scenes. Figure 5.7 shows the mean number of inliers found by different solvers. It can be seen that the new non-planar solver performs better than the state-of-the-art non-planar solver from Bujnak et al. [28] for close-to-planar scenes and is therefore more suitable for a "joined general solver" presented in [28]. The new general P4PFR solver doesn't have problems with close-to-planar or planar scenes and this solver was able to find all inliers. The state-of-the-art general solver [102] has some problems with numerical stability due the decomposition of a huge template matrix (1134×720). Therefore, the average number of inliers returned by this solver was less than 100.

5.4.4 Evaluation on Real Images

Finally we evaluate our method on real image data. We consider the *Rotunda* dataset [127] and the *Graffiti* dataset [128]. The Rotunda dataset consists of 62 images captured using a GoPro Hero4 camera with significant radial distortion. The Graffiti dataset consists of 12 images captured using GoPro Hero3 camera and 7 images captured using a HTC Desire 500 mobile phone. Some example images are shown in Figure 5.8. Using the RealityCapture software [2] we built a 3D reconstructions of both scenes. The Rotunda reconstruction contains 170994 3D points and the average reprojection error was 1.4694 pixels over 549478 image points. The Graffiti reconstruction contains 26078 3D points and the average reprojection error was 1.0778 pixels over 91518 image points.

Then to perform the experiment we used the 3D model to estimate the pose of each image using the new minimal solver (40×50) in a RANSAC framework. Since the dataset only contains image data, we used the camera and distortion parameters obtained from RealityCapture as ground truth for the experiment. Table 5.1 shows the errors for the focal length and radial distortion, as well as the camera pose. Overall the errors are quite small, with slightly larger errors for the more difficult planar dataset (Graffiti).

Dataset	*Rotunda*			*Graffiti*		
	avg.	med.	max	avg.	med.	max
Focal (%)	0.08	0.07	0.29	0.44	0.33	1.76
Distortion (%)	0.51	0.45	1.85	2.16	1.23	8.85
Rotation (degree)	0.03	0.03	0.10	0.12	0.11	0.27
Translation (%)	0.07	0.07	0.26	1.30	0.70	5.06

Table 5.1: Errors for the real datasets. The errors are relative to the ground truth for all except rotation which is shown in degrees.

5.5 Our Approach for P3.5Pf

Now we show how we can apply the same approach to the closely related problem of pose estimation with unknown focal length. This problem has 7 degrees of freedom and is overconstrained with four points. In [227] Wu presented a min-

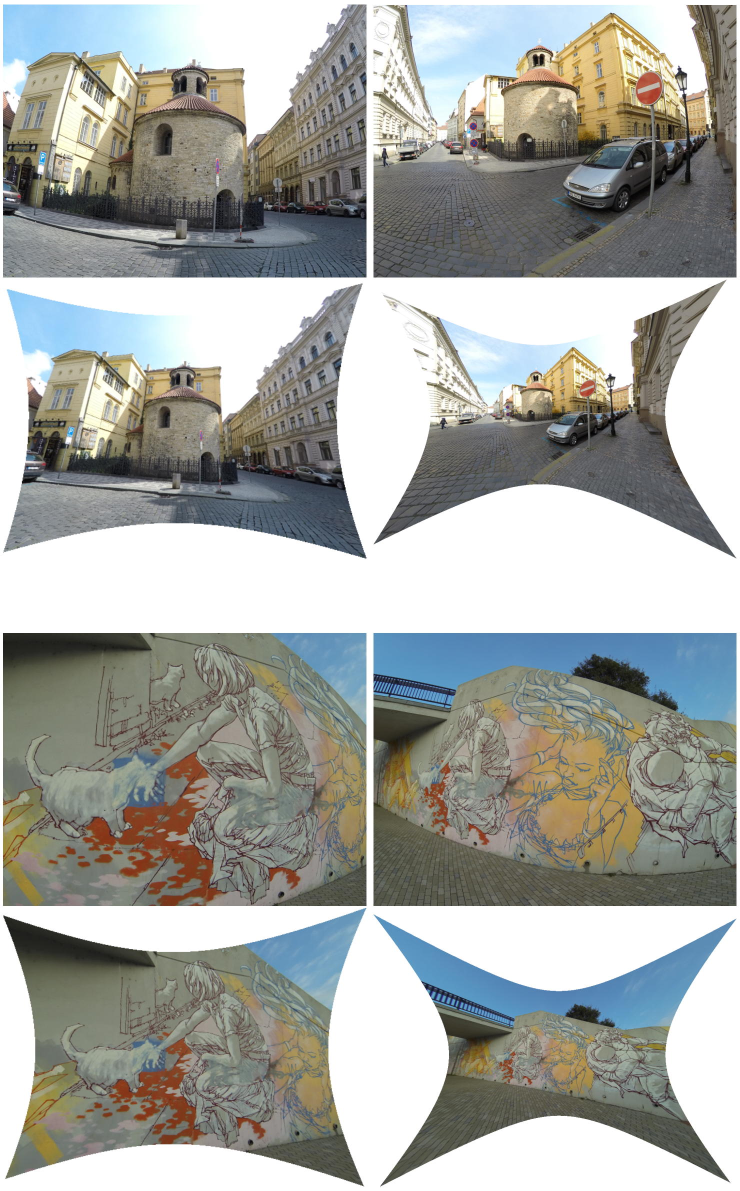


Figure 5.8: Some examples of the images in the *Rotunda* (Top) and *Graffiti* (bottom) datasets.

imal solver using 3.5 points (i.e. ignoring one of the image coordinates for one of the points). This solver works for both planar and non-planar data but has a degeneracy introduced by setting one quaternion element to one.

In this section we develop a new solver for this problem which has comparable performance to [227], but does not introduce any artificial degeneracies. The approach is essentially the same as for our P4PFR solver, but the formulation is simplified slightly since the projection equations (5.2) become completely linear when the radial distortion is removed. Each point correspondence now gives us two linearly independent equations in the camera matrix,

$$P_1\boldsymbol{X}_i - u_iP_3\boldsymbol{X}_i = 0, \quad P_2\boldsymbol{X}_i - v_iP_3\boldsymbol{X}_i = 0, \tag{5.26}$$

for $i = 1, 2, 3, 4$. Ignoring one of these eight equations gives a minimal problem. Using the same trick as in Section 5.3.1, we fix the scale by setting the first depth to one,

$$u_1 = P_1\boldsymbol{X}_1, \quad v_1 = P_2\boldsymbol{X}_1, \quad 1 = P_3\boldsymbol{X}_1. \tag{5.27}$$

Rewriting the linear constraints as

$$M\boldsymbol{v} = \boldsymbol{b}, \quad M \in \mathbb{R}^{8\times 12}, \tag{5.28}$$

we can parametrize the problem with only four unknowns using the nullspace of M. Note that for this problem the constraints on the camera matrix are exactly same as in Section 5.3.2. Using these 9 equations in the four unknowns, we generated a polynomial solver using the automatic generator from [122]. The resulting solver has a template of size 25×35, comparable to the current state-of-the-art [227] (20×30). However in contrast to [227] this formulation does not contain any additional degeneracies.

In this formulation the problem has 10 solutions for general data, and 8 solutions for planar data. For this problem the quotient basis and template we found from the non-planar instances works for planar instances as well. In the case of planar data the solver still returns 10 solutions, but only 8 will correspond to actual solutions.

5.5.1 Experiment

To evaluate the stability and accuracy of the polynomial solver we use a similar experiment setup as was used in [234, 227]. We generated random instances by uniformly sampling four 3D points in the box $[-2, 2] \times [-2, 2] \times [2, 8]$ in the camera's local coordinate system. The points were transformed by a random rotation and translation. The focal length was chosen uniformly in the interval $[200, 2000]$. Figure 5.9 shows the log_{10} relative focal length error for 1000 instances for both planar and non-planar data. For comparison we include the results for the best ratio and distance based solvers from Bujnak [25] and the GP4PF solver from Zheng et al. [234]. The new P3.5PF solver is stable for both the planar and non-planar setting. We were unable to directly compare with the solver from [227], since the code has not been made available. However in [227] they report comparable results to the solver from Zheng et al. [234].

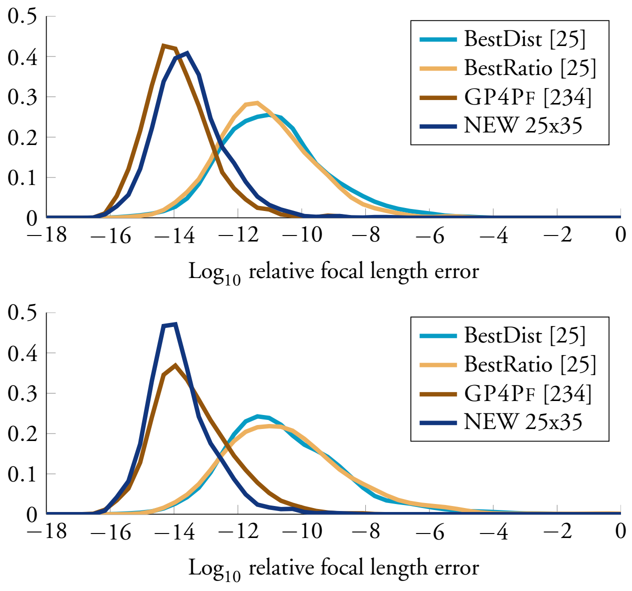


Figure 5.9: Stability of the P3.5PF solver. The figure shows the relative focal length errors over 1000 random instances. *Top:* General 3D data. *Bottom:* Planar 3D data.

5.6 Conclusions

In this chapter we revisited the absolute pose estimation problem with unknown focal length and radial distortion (P4PFR). We improved on previous approaches in several aspects.

First we removed the degeneracy introduced by fixing the scale in the nullspace parametrization by manually choosing which camera matrices to exclude. It is possible that this trick could be applied for other problems as well, e.g. in relative pose problems where nullspace parametrizations are commonly used.

Next we applied the elimination ideal techniques from [129] to get new constraints on the camera matrix. These constraints are satisfied whenever the focal length is the only unknown intrinsic parameter. It is possible that this technique could be applied to settings with other partial calibrations as well.

Finally we were able to construct a solver which worked for both planar and non-planar scenes. The key idea was to make sure that the monomial basis spanned both quotient rings and then creating a single merged elimination template which contains the equations necessary to solve for both types of instances. We believe that this template merging strategy could be applied to other problems as well. Not only for planar/non-planar degeneracies, but any time the structure of the problem depends on the input data.

Chapter 6

Absolute Pose with Unknown Focal Length and Principal Point

In terms of partially calibrated intrinsic parameters, the majority of existing work on pose estimation assume that the principal point lies in the image center (see e.g. Chapter 5 and the references therein). Unfortunately, this assumption is not always true, especially in the case of asymmetrically cropped images. When the offset is trivial, it can be partially compensated by the translation, without severely affecting the rotation. However, in the presence of significant offset such solvers with centered principal point will give poor rotation and translation estimates.

Triggs [214] was the first to consider estimation of the camera pose with unknown focal length and principal point. A non-minimal 5-point solver was presented, although the minimal case is with four and a half points only. As noted in [214], this 5-point solver is very sensitive to noise due to the non-minimal parametrization.

In this chapter we present the first exactly minimal solver for the case of unknown principal point and focal length, using four and a half point correspondences (called P4.5PFUV). In addition to the classical constraints on the camera matrix for enforcing zero skew and unit aspect ratio [60, 88], we derive novel polynomial constraints which allow us to avoid solutions corresponding to rank deficient camera matrices. Next we consider the problem where the aspect ratio is also unknown, and construct a minimal solver which uses five point correspondences (called P5PFUVA). In this case the equations reduce to a simple quartic polynomial which allows for a closed form solver that is both extremely fast and stable. In experiments we show that the new solvers are superior in terms of both stability and efficiency compared to the previous state-of-the-art five point solver from Triggs [214].

Finally, we consider the case of both unknown principal point and radial distortion. This problem is very difficult and highly non-linear due the radial distortion being centered on the unknown principal point. We develop the first practical non-minimal solver by using seven point correspondences (called P7PFRUV) instead of the minimum five.

The contributions in this chapter are:

- To derive new polynomial constraints on the camera matrix for the case of unit aspect ratio and zero skew.
- To develop the first minimal P4.5PFUV solver (unit aspect ratio and zero skew) as well as an extremely fast P5PFUVA solver (zero skew).
- To explore the extremely challenging case of unknown principal point and radial distortion, and develop the first practical non-minimal solver using seven point correspondences.

This chapter is based on the paper [138].

6.1 Unit Aspect Ratio and Zero Skew

In this section, we first focus on the absolute pose problem without radial distortion. Given point correspondences $\boldsymbol{x}_i \leftrightarrow \boldsymbol{X}_i$, we wish to estimate the camera pose $(R, \boldsymbol{t})$, the focal length f and the principal point $\boldsymbol{x}_0 = (u_0, v_0)$. For this setting the projection equations become

$$\begin{pmatrix} \boldsymbol{x}_i - \boldsymbol{x}_0 \\ 1 \end{pmatrix} \simeq \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} (R\boldsymbol{X}_i + \boldsymbol{t}). \tag{6.1}$$

Since the focal length and principal point are both unknown, the only remaining constraints on the camera matrix come from the aspect ratio and skew. We assume that the cameras have zero skew ($s = 0$) and unit aspect ratio ($\alpha = 1$). These are natural constraints which are satisfied by most consumer cameras with a modern CCD/CMOS sensor.

6.1.1 Camera Matrix Constraints

The constraints for a camera matrix P to admit the following factorization

$$P = K \begin{bmatrix} R & \boldsymbol{t} \end{bmatrix}, \quad K = \begin{bmatrix} f & & u_0 \\ & f & v_0 \\ & & 1 \end{bmatrix}, \tag{6.2}$$

are well known and summarized in the following theorem.

Theorem 6.1 (Faugeras [60], Heyden [88]). *The matrix*

$$P = \begin{bmatrix} \boldsymbol{p}_1^T & p_{14} \\ \boldsymbol{p}_2^T & p_{24} \\ \boldsymbol{p}_3^T & p_{34} \end{bmatrix} \tag{6.3}$$

corresponds to a perspective camera with zero skew and unit aspect ratio if and only if

$$\det \begin{bmatrix} \boldsymbol{p}_1, & \boldsymbol{p}_2, & \boldsymbol{p}_3 \end{bmatrix} \neq 0 \tag{6.4}$$

and

$$(\boldsymbol{p}_1 \times \boldsymbol{p}_3) \cdot (\boldsymbol{p}_2 \times \boldsymbol{p}_3) = 0 \tag{6.5}$$

$$\|\boldsymbol{p}_1 \times \boldsymbol{p}_3\|^2 - \|\boldsymbol{p}_2 \times \boldsymbol{p}_3\|^2 = 0 \tag{6.6}$$

If only (6.5) *holds the camera has non-unit aspect ratio.*

Although formulated differently, the constraints (6.5) and (6.6) are equivalent to the ones used to create the solver from Triggs [214].

6.1.2 New Camera Matrix Constraints

The non-zero determinant constraint in (6.4) is difficult to incorporate in polynomial solvers. Ignoring this constraint adds false solutions corresponding to rank deficient camera matrices. In this section we use tools from algebraic geometry (see e.g. [42]) to find additional polynomial constraints which ensure that we only recover the true camera matrices.

Let I be the ideal generated by the original constraints,

$$I = \left\langle (\boldsymbol{p}_1 \times \boldsymbol{p}_3) \cdot (\boldsymbol{p}_2 \times \boldsymbol{p}_3), \|\boldsymbol{p}_1 \times \boldsymbol{p}_3\|^2 - \|\boldsymbol{p}_2 \times \boldsymbol{p}_3\|^2 \right\rangle. \tag{6.7}$$

Using Macaulay2 [75] we find that this ideal is of dimension 7[1] and degree 16. This means that if we add 7 linear constraints we will in general have 16 solutions. Now some of these solutions might correspond to rank deficient camera matrices. To remove these solutions we compute the saturation of I w.r.t to the determinant, i.e.

$$J = \left\{ f(\boldsymbol{x}) \;\middle|\; \exists N \geq 0,\ \det(\left[\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3\right])^N f(\boldsymbol{x}) \in I \right\}. \tag{6.8}$$

The saturated ideal J contains additional polynomial constraints which should be satisfied by the correct camera matrices (i.e. non-zero determinant). We find that this ideal is also of dimension 7 but only of degree 10. This means that in general there are 6 false solutions corresponding to rank deficient camera matrices if you only use the constraints in (6.5) and (6.6).

The ideal J is generated by the two constraints from (6.5) and (6.6), as well as 5 polynomials of degree 5, listed below.

$$p_{11}p_{12}p_{32}^2p_{33}+p_{11}p_{12}p_{33}^3-p_{11}p_{13}p_{32}^3-p_{11}p_{13}p_{32}p_{33}^2-p_{12}^2p_{31}p_{32}p_{33}+p_{12}p_{13}p_{31}p_{32}^2-p_{12}p_{13}p_{31}p_{33}^2+p_{13}^2p_{31}p_{32}p_{33}+p_{21}p_{22}p_{32}^2p_{33}+p_{21}p_{22}p_{33}^3-p_{21}p_{23}p_{32}^3-p_{21}p_{23}p_{32}p_{33}^2-p_{22}^2p_{31}p_{32}p_{33}+p_{22}p_{23}p_{31}p_{32}^2-p_{22}p_{23}p_{31}p_{33}^2+p_{23}^2p_{31}p_{32}p_{33}=0$$

$$p_{11}p_{12}p_{31}p_{33}^2-p_{11}p_{13}p_{31}p_{32}p_{33}+p_{12}^2p_{32}p_{33}^2-p_{12}p_{13}p_{31}^2p_{33}-2p_{12}p_{13}p_{32}^2p_{33}+p_{13}^2p_{31}^2p_{32}+p_{13}^2p_{32}^3-p_{21}^2p_{32}^3-p_{21}^2p_{32}p_{33}^2+2p_{21}p_{22}p_{31}p_{32}^2+p_{21}p_{22}p_{31}p_{33}^2+p_{21}p_{23}p_{31}p_{32}p_{33}-p_{22}^2p_{31}^2p_{32}-p_{22}p_{23}p_{31}^2p_{33}=0$$

$$p_{11}^2p_{32}^2p_{33}+p_{11}^2p_{33}^3-p_{11}p_{12}p_{31}p_{32}p_{33}-p_{11}p_{13}p_{31}p_{32}^2-2p_{11}p_{13}p_{31}p_{33}^2+p_{12}p_{13}p_{31}^2p_{32}+p_{13}^2p_{31}^2p_{33}+p_{21}p_{22}p_{31}p_{32}p_{33}-p_{21}p_{23}p_{31}p_{32}^2-p_{22}^2p_{31}^2p_{33}-p_{22}^2p_{33}^3+p_{22}p_{23}p_{31}^2p_{32}+2p_{22}p_{23}p_{32}p_{33}^2-p_{23}^2p_{32}^2p_{33}=0$$

$$p_{11}^2p_{31}p_{33}^2+p_{11}p_{12}p_{32}p_{33}^2-2p_{11}p_{13}p_{31}^2p_{33}-p_{11}p_{13}p_{32}^2p_{33}-p_{12}p_{13}p_{31}p_{32}p_{33}+p_{13}^2p_{31}^3+p_{13}^2p_{31}p_{32}^2-p_{21}^2p_{31}p_{32}^2+2p_{21}p_{22}p_{31}^2p_{32}+p_{21}p_{22}p_{32}p_{33}^2-p_{21}p_{23}p_{32}^2p_{33}-p_{22}^2p_{31}^3-p_{22}^2p_{31}p_{33}^2+p_{22}p_{23}p_{31}p_{32}p_{33}=0$$

$$p_{11}^2p_{31}p_{32}p_{33}-p_{11}p_{12}p_{31}^2p_{33}-p_{11}p_{12}p_{33}^3-p_{11}p_{13}p_{31}^2p_{32}+p_{11}p_{13}p_{32}p_{33}^2+p_{12}p_{13}p_{31}^3+p_{12}p_{13}p_{31}p_{33}^2-p_{13}^2p_{31}p_{32}p_{33}+p_{21}^2p_{31}p_{32}p_{33}-p_{21}p_{22}p_{31}^2p_{33}-p_{21}p_{22}p_{33}^3-p_{21}p_{23}p_{31}^2p_{32}+p_{21}p_{23}p_{32}p_{33}^2+p_{22}p_{23}p_{31}^3+p_{22}p_{23}p_{31}p_{33}^2-p_{23}^2p_{31}p_{32}p_{33}=0$$

6.1.3 Building a Polynomial Solver - P4.5PFUV

Cameras with unit aspect ratio and zero skew have 9 degrees of freedom (3 intrinsic and 6 extrinsic), making the pose estimation problem minimal with 4.5

[1]Note that here we only consider the first 3×3 block of the camera matrix. For the full camera matrix the dimension would be 10.

points. Computing the three dimensional nullspace to the projection equations allow us to parametrize the camera matrix with three unknowns

$$P = \alpha_1 P_1 + \alpha_2 P_2 + \alpha_3 P_3. \tag{6.9}$$

The scale can be fixed by setting $\alpha_3 = 1$. Inserting the first 3×3 block of (6.9) into the constraints from Section 6.1.2 we get 2 equations of degrees 4 and 5 equations of degree 5 in the two unknowns α_1 and α_2. Using the automatic generator from Chapter 1, we constructed a polynomial solver with template size 11×21.

If we only use the two original constraints, (6.5) and (6.6), the automatic generator from Chapter 1 returns a polynomial solver with template size 20×36 and if we employ the automatic saturation technique from Chapter 2 to saturate the determinant we get a template of size 34×50.

6.1.4 Unknown Aspect Ratio - P5Pfuva

In the case of unknown aspect ratio and zero skew we only have a single constraint (6.5) on the camera matrix. For this ideal no additional constraints are yielded when we saturate the determinant. Cameras with zero skew have 10 degrees of freedom (4 intrinsic and 6 extrinsic) and the pose estimation problem becomes minimal with 5 point correspondences. The linear constraints from 5 points have a 2 dimensional nullspace, allowing us to parameterize the camera using only a single unknown,

$$P = \alpha_1 P_1 + P_2. \tag{6.10}$$

Inserting into the constraint (6.5) yields a single quartic equation in α_1 that can be efficiently solved.

6.1.5 Implementation Details

For the 4.5 point solver from Section 6.1.3 we can get 9 linear constraints from 5 point correspondences by ignoring a coordinates for one of the image points. The ignored coordinate can then be used to filter solutions. Another approach is to consider all 5 points (10 linear constraints) and compute an approximate 3 dimensional nullspace. In experiments we found that this approach is less sensitive to noise, however the runtime is slightly longer due to the need for computing an SVD to find the approximate nullspace.

For the zero-skew 5 point solver from Section 6.1.4 we need to find the roots to a quartic polynomial. This can be done by either computing the eigenvalues of the companion matrix, or using the closed form solution for the quartic. In experiments we found that these have similar accuracy if some care is taken to avoid cancellation errors when implementing the closed form solver.

Implemented in C++ the runtimes on a standard desktop computer are $\approx$ 120 µs (P4.5PFUV) and 5 µs (P5PFUVA).

6.2 Radial Distortion with Unknown Center

Radial distortion adds an extra non-linearity to the projections which makes pose estimation more difficult. The problem is further complicated if the center of distortion (typically the principal point) is unknown. In this case the projection equations can be written as

$$\begin{pmatrix} \boldsymbol{x} - \boldsymbol{x}_0 \\ 1 + k\,\|\boldsymbol{x} - \boldsymbol{x}_0\|^2 \end{pmatrix} \simeq \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} (R\boldsymbol{X} + \boldsymbol{t}). \tag{6.11}$$

The problem contains 10 degrees of freedom and thus becomes minimal with 5 points. However the minimal problem is extremely difficult and we have not found any tractable formulation.

6.2.1 Seven Point Relaxation - P7PFRUV

To tackle this problem we instead consider a non-minimal relaxation using 7 points. The idea is to consider only the first two equations of (6.11),

$$\begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} \simeq f \begin{bmatrix} R_1\boldsymbol{X} + t_1 \\ R_2\boldsymbol{X} + t_2 \end{bmatrix} \tag{6.12}$$

These equations constrain the projections to lie on the lines passing through the distortion center (u_0, v_0) and the image point (u, v). This constraint is independent of the focal length and the radial distortion parameter, since they just move the projections along these lines.

In (6.12) we can of course ignore the focal length since it is non-zero for any interesting solution. Using 2×2 determinants we can rewrite the equations as

$$(u - u_0)(R_2\boldsymbol{X} + t_2) - (v - v_0)(R_1\boldsymbol{X} + t_1) = 0 \tag{6.13}$$

This relaxed problem has 7 degrees of freedom, and since each point yields a single constraint the problem becomes minimal with 7 points. Of course solving (6.13) only gives the orientation R, distortion center (u_0, v_0) and the first two components of the translation t_1 and t_2. The remaining unknowns in (6.11) can be solved for linearly, see Section 6.2.4.

This relaxation is similar to the one made in [125] where they solved the P4PFR problem using five points instead of the minimal four. In [217] Tsai used a similar approach but did not enforce the constraints on the rotation and simply solved for the unknowns linearly using more points. However, in both these works the principal point is assumed to be known.

6.2.2 Simplifying the Equations

To solve the equations in (6.12) we start by translating the 2D and 3D coordinate systems such that

$$\boldsymbol{x}_1 = (u_1, v_1)^T = (0,0)^T, \quad \boldsymbol{X}_1 = (0,0,0)^T. \tag{6.14}$$

The equations from the first point then reduce to

$$\begin{bmatrix} -u_0 \\ -v_0 \end{bmatrix} \simeq \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \tag{6.15}$$

If we let R be a scaled rotation, we can instead fix the scale of the camera matrix by setting $t_1 = u_0$ and $t_2 = v_0$. This eliminates two unknowns and has the additional benefit that when we insert this into the equations in (6.13), the mixed quadratic terms in (u_0, v_0, t_1, t_2) cancel and we are left with equations which only depend linearly on (u_0, v_0).

Using the hidden variable trick [42] we can eliminate the distortion center (u_0, v_0) from our equations. Rewrite (6.13) as

$$M(R) \begin{pmatrix} u_0 \\ v_0 \\ 1 \end{pmatrix} = 0 \tag{6.16}$$

where $M(R)$ is a 6×3 matrix depending on the rotation R. Requiring all 3×3 determinants of this matrix to vanish, we get 20 equations of degree 3 in the elements of R.

Finally using quaternions we parameterize the scaled rotation matrix, i.e.

$$R(\boldsymbol{q}) = \begin{bmatrix} q_1^2 + q_2^2 - q_3^2 - q_4^2 & 2q_2q_3 - 2q_1q_4 & 2q_1q_3 + 2q_2q_4 \\ 2q_1q_4 + 2q_2q_3 & q_1^2 - q_2^2 + q_3^2 - q_4^2 & 2q_3q_4 - 2q_1q_2 \\ 2q_2q_4 - 2q_1q_3 & 2q_1q_2 + 2q_3q_4 & q_1^2 - q_2^2 - q_3^2 + q_4^2 \end{bmatrix}$$

This yields 20 equations of degree 6 in $\boldsymbol{q} = (q_1, q_2, q_3, q_4)$.

Studying this equation system in Macaulay2 [75] we find that it has 88 solutions. However 16 of these are false solution introduced in the hidden variable trick (6.16).

6.2.3 Removing Symmetries

Using the quaternion parametrization we introduce a 2-fold symmetry into our problem since $R(\boldsymbol{q}) = R(-\boldsymbol{q})$. For this problem there is another symmetry corresponding to changing the sign of the first two rows of the rotation matrix. Note that this symmetry is also present in the original (non-relaxed) problem where the focal length also changes sign.

This type of symmetry also occurred in the WPNP problem from Larsson et al. [133] (see Chapter 3) and in the P3.5PF formulation of [227]. In [133] we handled the symmetry by doing the following linear change of variables in the quaternion,

$$\boldsymbol{q} = \begin{bmatrix} 0 & -i & -i & 0 \\ -1 & 0 & 0 & 1 \\ i & 0 & 0 & i \\ 0 & -1 & 1 & 0 \end{bmatrix} \hat{\boldsymbol{q}}, \tag{6.17}$$

which reduces the symmetry into two sign symmetries in $(\hat{q}_1, \hat{q}_2)$ and $(\hat{q}_3, \hat{q}_4)$. Removing the symmetries collapses the 88 solutions into 22.

Using the automatic generator from Chapter 1 (which automatically handles these sign symmetries as described in Chapter 3) we were able to construct a polynomial solver with template size 124×162.

6.2.4 Recovering the Full Solutions

The solver we created only returns the rotation R. To recover the remaining parameters we first solve linearly for u_0, v_0, t_1 and t_2 from (6.13). Since these are part of the relaxed problem we have an exact solution to this system.

To recover the remaining parameters f, k and t_3 we rewrite (6.11) as

$$(R_3\boldsymbol{X} + t_3)(\boldsymbol{x} - \boldsymbol{x}_0) = f(1 + k\,\|\boldsymbol{x} - \boldsymbol{x}_0\|^2)\begin{bmatrix} R_1\boldsymbol{X} + t_1 \\ R_2\boldsymbol{X} + t_2 \end{bmatrix} \tag{6.18}$$

where we can solve linearly for f, fk and t_3. Note that in general there is no exact solution satisfying all $7 \times 2 = 14$ equations since we did not solve the true minimal problem. So instead we solve the linear equations in a least squares sense.

Since this does not minimize any meaningful geometric error we can refine the solutions by performing a few iterations of local optimization. Note that this can be done very quickly since we have very few unknowns and residuals. Since the division model's inverse transform is quite messy, we minimize

$$\sum_{i=1}^{7} \left\| (\boldsymbol{x}_i - \boldsymbol{x}_0) - \frac{f(1 + k\,\|\boldsymbol{x}_i - \boldsymbol{x}_0\|^2)}{(R_3\boldsymbol{X}_i + t_3)} \begin{bmatrix} R_1\boldsymbol{X}_i + t_1 \\ R_2\boldsymbol{X}_i + t_2 \end{bmatrix} \right\|^2 \tag{6.19}$$

instead of the true reprojection error. Empirically we have seen that this approximation works well.

6.3 Experiments

We experimentally evaluate our new solvers on both synthetic and real image data. For P4.5PFUV we compare both using the exact nullspace from 4.5 points (computed using QR), as well as the approximate nullspace from 5 points (computed using SVD). For P5PFUVA we compare solving the quartic equation both using the closed form solution and by computing the eigenvalues to the companion matrix. For solvers returning multiple focal length estimates (i.e. non-unit aspect ratio) we compute the focal length error by comparing against the geometric mean $f = \sqrt{f_1 f_2}$. For the P7PFRUV solver the results are without the extra non-linear refinement proposed in Section 6.2.4 unless otherwise noted.

For the solver from Triggs [214] we added some normalization of the coordinate systems (scaling and shifting) since it was not available in the original implementation available from the author. Experimentally we have observed that this improves the performance drastically in the presence of noise.

For some experiments we only show the errors in the focal length since the other errors are qualitatively similar.

6.3.1 Stability

In this section we evaluate the numerical stability of the proposed polynomial solvers. We generated random but feasible noise-free synthetic problem instances. To generate the scene we uniformly sample five 3D-points from the box $[-2, 2] \times [-2, 2] \times [2, 8]$ in the camera's local coordinate system. These are then transformed with a random rotation and translation. The focal length was drawn uniformly from the interval $[200, 2000]$ and the principal point was placed randomly 500 px from the origin.

Figure 6.1 shows the distribution of the $\log_{10}$ relative focal length errors for 10,000 instances, and we can see that all solvers are quite stable.

We ran a similar experiment but where we added radial distortion to the image points. The distortion parameter was drawn uniformly from the interval $[-0.4, 0]$. The results for the P7PFRUV solver with and without the non-linear refinement is shown in Figure 6.2. On a small number of instances the P7PFRUV solver had issues with numerical stability. However, these solutions were still a good enough starting guess for the non-linear refinement (Section 6.2.4).

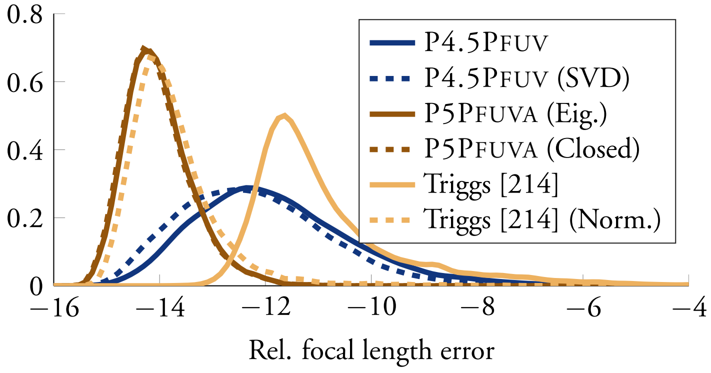


Figure 6.1: Relative focal length error $\frac{|f - f_{gt}|}{f_{gt}}$ for 10,000 random synthetic instances.

6.3.2 Varying Noise

Next we evaluate the sensitivity to image noise. We use a similar setup as in Section 6.3.1 but where we fixed the focal length $f_{gt} = 1000$ and added Gaussian noise with varying standard deviation to the image points. Figure 6.3 shows the

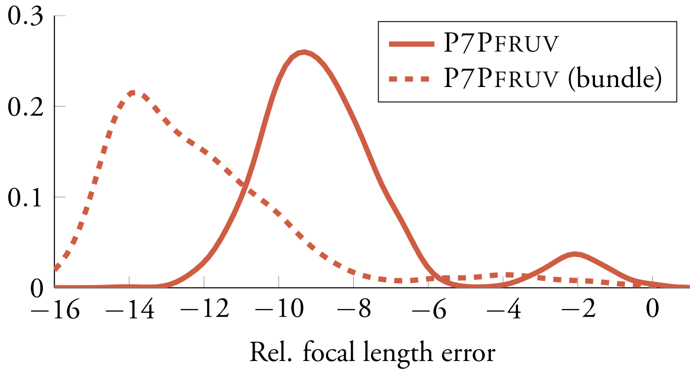


Figure 6.2: Relative focal length error $\frac{|f-f_{gt}|}{f_{gt}}$ for 10,000 random synthetic instances for image points with added radial distortion.

median relative focal length error against the noise level and Figure 6.4 shows the error distribution for $\sigma = 2$ px.

Our new solvers, especially P4.5PFUV (SVD), performs the best in the presence of image noise. For the solver from Triggs [214], the accuracy degrades heavily for noisy image points. Additionally we can see that normalization of the image and 3D coordinate systems is essential.

Again we ran a similar experiment but with radial distortion added to the image points. The distortion parameter was fixed at $k_{gt} = -0.2$. The median relative focal length errors are shown in Figure 6.5.

Comparison to 6p DLT

We also did a comparison with standard 6 point DLT [81]. To get a fair comparison we use 6 points to compute the approximate nullspace in our P4.5PFUV (SVD) solver. Figure 6.6 shows that we are able to get better results by enforcing the correct constraints on the intrinsic parameters.

6.3.3 Varying Principal Point

In this section we study the effects of varying principal point. We compare our new solvers to the state-of-the-art P3.5PF and P4PFR solvers from Chapter 5 ([137]), which assume that the principal point is in the center of the image. We generate synthetic scenes where the distance from the origin to the principal point

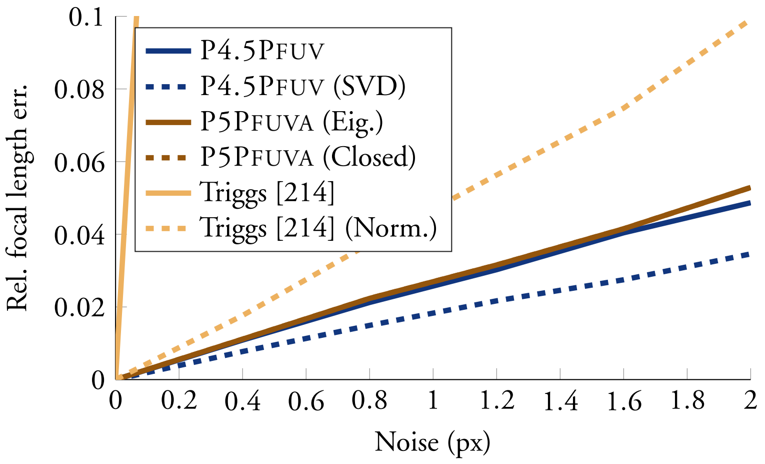


Figure 6.3: Median relative focal length error for varying noise.

is varied. The ground truth focal length was $f_{gt} = 1000$ and we added small Gaussian noise ($\sigma = 0.1$ px) to the image coordinates. For the distortion solvers we also add radial distortion to the image points with $k_{gt} = -0.2$. The median errors in the focal length and rotation are shown in Figure 6.7. As expected ignoring the principal point makes the pose estimation significantly worse.

6.3.4 Varying Radial Distortion

In this section we compare the performance of the new solvers when we add varying degrees of radial distortion to the image points. We generated synthetic scenes similarly to Section 6.3.2 and added varying radial distortion. The ground truth focal length was $f_{gt} = 1000$ and the principal point was chosen randomly at distance 500 px from the origin. We also added some small Gaussian noise ($\sigma = 0.1$ px) to the image coordinates. Figure 6.8 shows the median relative focal length and rotation errors for different radial distortion parameters. For the solvers which do not model the radial distortion the error increases drastically when distortion is added. We can also see that the P7PFRUV solver has slightly worse performance on image data with little or no radial distortion.

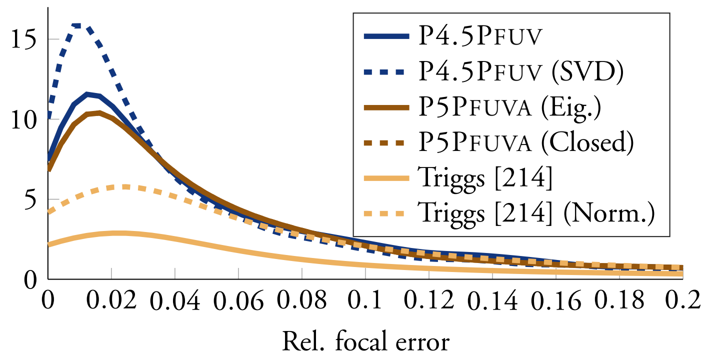


Figure 6.4: Distribution of relative focal length error for 2 px noise.

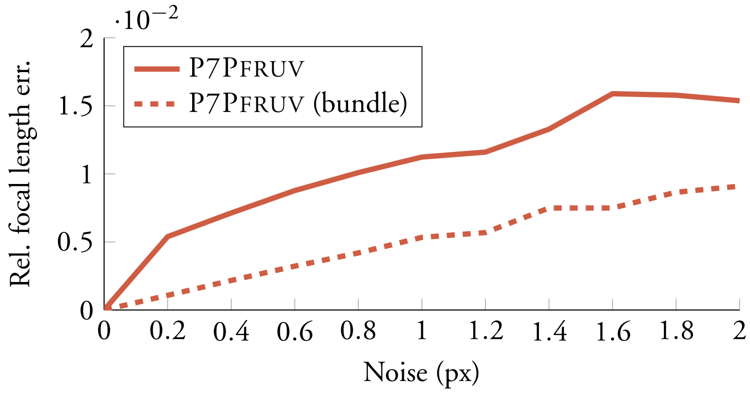


Figure 6.5: Median relative focal length error for varying noise with radial distortion.

6.3.5 Real Data

We evaluated all proposed solvers on real image data and compared them with the current state-of-the-art solvers. We downloaded 101 images of the Notre Dame cathedral from the Internet. All downloaded images have a square resolution varying from 800 px $\times$ 800 px to 3000 px $\times$ 3000 px. Since the images have a square resolution, there was a higher probability that some of these images were edited or cropped and that their principal points are not in the center of the image. Some example images are shown in Figure 6.9.

Using the RealityCapture software [2] we built a 3D reconstruction of the scene. Since the dataset is quite challenging and it contains many manually

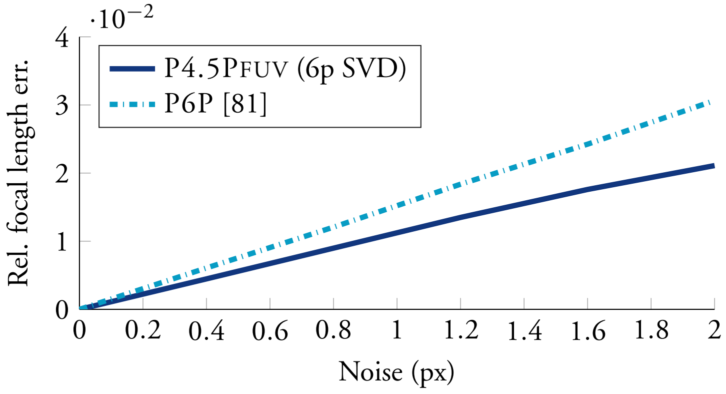
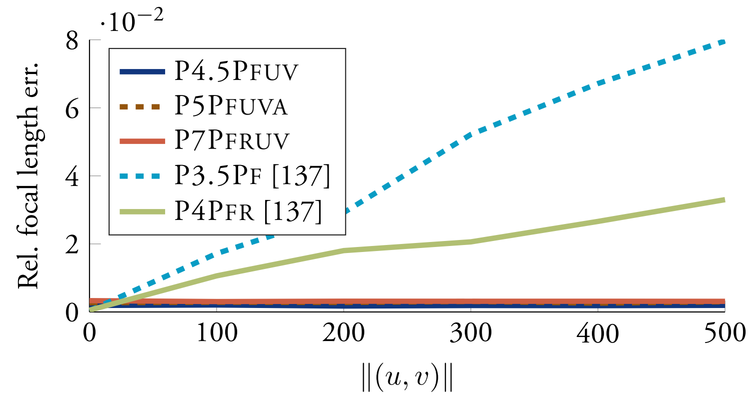


Figure 6.6: Comparison with 6 point DLT. The graph shows the median relative focal length error for varying noise levels. Note that we use 6 points in our P4.5PFUV solver as well.

edited images, images taken at different conditions, or images with a small overlap, the RealityCapture was only able to register 81 of the 101 images. The Notre Dame reconstruction contains 24762 3D points, the average reprojection error was 0.6277 pixels and the maximum 1.997 pixels over 73543 image points. According to the principal point estimates returned by RealityCapture, 32 out of 81 images have the principal point shifted by more than 6% of the image size.

We used the 3D model and 2D-to-3D correspondences returned by RealityCapture to estimate the pose of each image using the solvers in a simple RANSAC framework with the number of RANSAC iterations fixed to 1000. We used the camera intrinsic and extrinsic parameters obtained from RealityCapture as the ground truth for the experiment. Table 6.1 shows the errors for the focal length, the radial distortion, the camera pose, as well as the ratio of inliers obtained by different solvers for all 81 registered images. Table 6.2 shows the same errors for 32 images with the principal point shift larger than 6% of the image size. The results with non-linear refinement for P7PFRUV, as described in Section 6.2.4, are marked with (b). Note that this is performed using only the 7 sampled point correspondences.

Overall, the errors are quite small and the new solvers (marked bold) perform the best. It is visible that solvers which assume that the principal point in the center of the image (P3.5PF, P4PFR) perform significantly worse than the solvers with unknown principal point, especially on images with larger principal



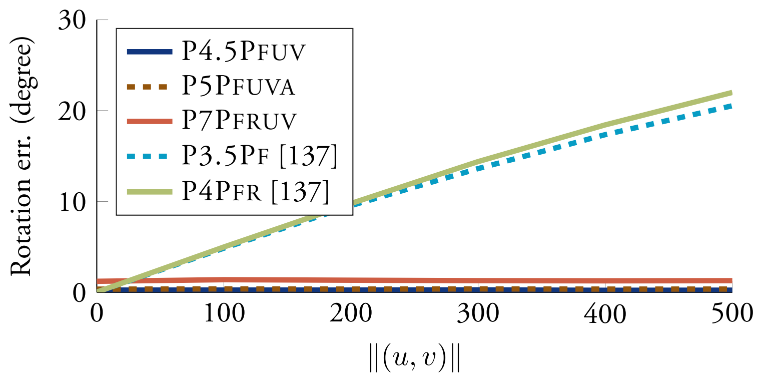


Figure 6.7: Comparison to standard solvers with for scenes with varying principal point. *Top:* Relative focal length error $\frac{|f-f_{gt}|}{f_{gt}}$. *Bottom:* Rotation error in degrees.

point shift (Table 6.2). Even though the images from the Notre Dame dataset do not have a significant radial distortion, the new radial distortion solver helps to improve the results.

6.3.6 Real Images with Radial Distortion

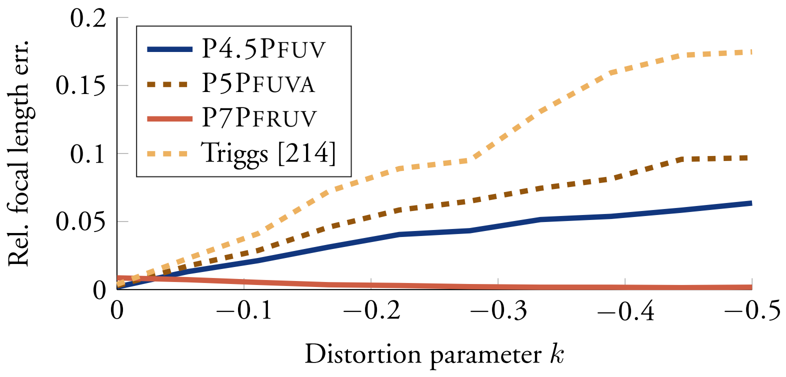
Finally, we evaluate our solvers on real images which have both a shifted principal point and significant radial distortion. Since it is difficult to generate reliable ground truth for this problem we took the following approach. We started with the *Rotunda* dataset which was used in [127, 137]. The dataset contains images captured with a wide-angle cameras with large radial distortion and a 3D reconstruction with cameras' intrinsic and extrinsic parameters (see Chapter 5 for more details). We used these camera parameters as a ground truth data. For each image in the dataset we cropped out 80% of the image, starting from each of the four corners. See Figure 6.10 for an example. This gave us a new dataset with $4 \times 62 = 248$ images which have both radial distortion and shifted principal points.

The results for running 1000 iterations of RANSAC are shown in Table 6.4. The best results are obtained using the new P7PFRUV solver which can model both radial distortion and shifted principal point.

6.4 Conclusions

In this chapter, we have revisited the camera pose estimation problem with unknown principal point. We proposed effective polynomial constraints to avoid trivial rank-deficient solutions, and successfully developed the first exactly minimal solver for the case of unknown principal point and focal length by using four and a half point correspondences (P4.5PFUV), as well as an extremely efficient variant using five point correspondences in the presence of unknown aspect ratio (P5PFUVA). We have also explored the extremely challenging case of unknown principal point and radial distortion, and developed the first practical non-minimal solver by using seven point correspondences (P7PFRUV). The applicability of these new solvers has been verified on both synthetic data and real images.

The high non-linearity in the case of both unknown principal point and radial distortion prevents us from developing an exactly minimal solver. We will continue to explore the feasibility of the exactly minimal problem in the future.



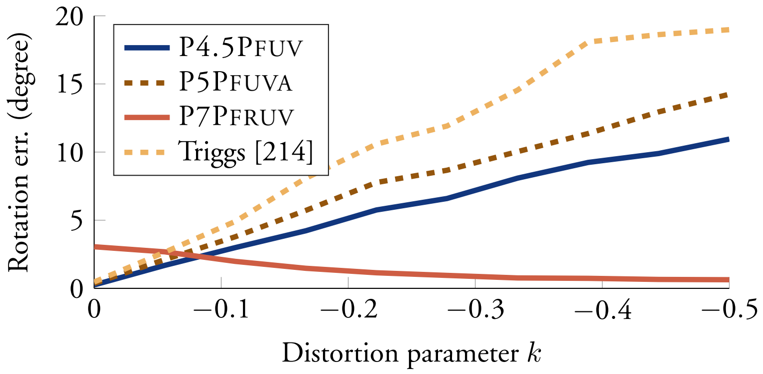


Figure 6.8: Comparison of the solvers for scenes with varying radial distortion. *Top:* Relative focal length error $\frac{|f-f_{gt}|}{f_{gt}}$. *Bottom:* Rotation error in degrees.

Figure 6.9: Example images from the *NotreDame* internet dataset.

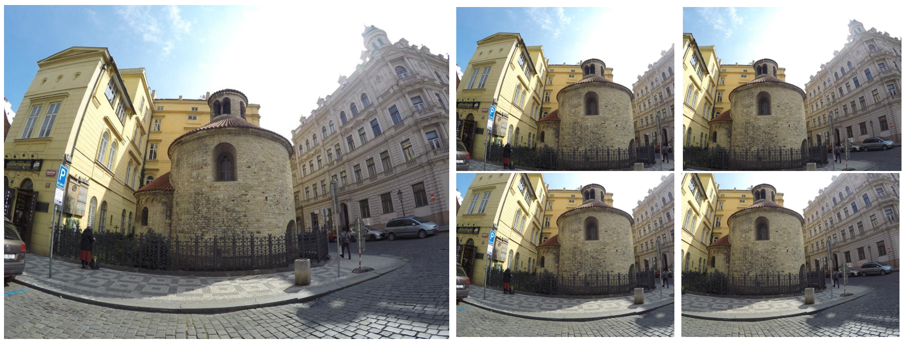


Figure 6.10: Example images. *Left:* Image from the original *Rotunda* dataset [127]. *Right:* Cropped images from the *Rotunda* dataset used for the experiment in Section 6.3.6.

		P3.5PF [137]	P6P [81]	P5PFUV [214]	**P4.5PFUV**	**P4.5PFUV (SVD)**	**P4.5PFUV (6pt)**	**P5PFUVA**	P4PFR [137]	**P7PFRUV**	**P7PFRUV (b)**
Focal length	mean	0.1256	0.0242	0.0234	0.0209	0.0189	**0.0142**	0.0243	0.1012	0.0211	0.0037
	median	0.0458	0.0137	0.0138	0.0142	0.0138	**0.0096**	0.0159	0.0264	0.0165	0.0024
	max	1.3042	0.1342	0.1123	0.1449	0.0962	**0.0535**	0.1176	0.9307	0.0807	0.0186
Rotation	mean	2.9495	1.2067	1.2559	1.2544	1.2564	**1.1228**	1.2322	3.0043	0.9005	0.3166
	median	1.6891	0.8814	0.9778	1.0582	1.1260	0.9255	**0.8546**	1.6816	0.7009	0.2085
	max	19.5789	5.1510	4.1978	3.9726	3.8889	4.0754	4.5883	21.7933	**2.7497**	1.9464
Translation	mean	0.6196	0.1163	0.1249	0.0894	0.0849	**0.0714**	0.1200	0.4941	0.1016	0.0188
	median	0.2052	0.0522	0.0569	0.0586	0.0629	**0.0414**	0.0640	0.1315	0.0629	0.0097
	max	5.4442	0.6711	1.6183	**0.5100**	0.5809	0.7289	0.7050	4.1039	0.5929	0.1665
Principal point	mean	-	0.0319	0.0317	0.0335	0.0335	0.0296	0.0324	-	**0.0227**	0.0089
	median	-	0.0226	0.0223	0.0239	0.0236	0.0207	0.0224	-	**0.0168**	0.0054
	max	-	0.2263	0.2829	0.4484	0.4411	0.3877	0.3491	-	**0.1735**	0.0808
Distortion	mean	-	-	-	-	-	-	-	0.0868	**0.0474**	0.0165
	median	-	-	-	-	-	-	-	0.0290	**0.0173**	0.0071
	max	-	-	-	-	-	-	-	**1.0509**	1.9341	0.3840
Inlier (%)	mean	74.9315	88.0265	86.8770	87.8971	88.6420	89.4039	87.5297	82.1422	**93.2453**	98.3413
	median	79.6975	90.6261	90.3534	91.7344	91.8065	92.6474	90.3015	89.3571	**94.6934**	98.3413
	max	**99.6805**	99.2391	99.0581	99.3407	99.4610	99.6711	99.3856	99.6805	99.3610	100.0000

Table 6.1: Full results for the *NotreDame* dataset: Comparison of different solvers on 81 images downloaded from the Internet. The table shows the relative errors except for the rotation errors which are in degrees. For the principal point the error is relative to the image size. The best results (excluding P7PFRUV with bundle adjustment) are marked bold.

		P3.5PF [137]	P6P [81]	P5PFUV [214]	**P4.5PFUV**	**P4.5PFUV (SVD)**	**P4.5PFUV (6pt)**	**P5PFUVA**	P4PFR [137]	**P7PFRUV**	**P7PFRUV (b)**
	mean	0.2615	0.0295	0.0228	0.0185	0.0186	**0.0123**	0.0286	0.2177	0.0192	0.0043
Focal length	median	0.1409	0.0150	0.0138	0.0139	**0.0089**	0.0093	0.0224	0.1363	0.0163	0.0031
	max	1.3042	0.1342	0.0809	0.0572	0.0762	**0.0437**	0.1176	0.9307	0.0807	0.0186
	mean	5.6991	1.6485	1.5334	1.3566	1.3602	1.3361	1.5225	5.9204	**0.8759**	0.3942
Rotation	median	4.6145	1.1001	1.0628	1.0924	1.1260	1.0800	1.1426	4.6211	**0.6414**	0.2657
	max	19.5789	5.1510	4.1978	3.9726	3.8889	4.0754	4.5883	21.7933	**2.7497**	1.9127
	mean	1.2483	0.1679	0.1528	0.1004	0.0978	**0.0699**	0.1646	1.0641	0.1140	0.0283
Translation	median	0.7351	0.0941	0.0735	0.0745	0.0759	**0.0601**	0.1105	0.7020	0.0797	0.0135
	max	5.4442	0.6711	1.6183	0.3328	0.3557	**0.2896**	0.6612	4.1039	0.4998	0.1665
	mean	-	0.0413	0.0373	0.0333	0.0334	0.0319	0.0396	-	**0.0214**	0.0108
Principal point	median	-	0.0291	0.0277	0.0300	0.0270	0.0258	0.0286	-	**0.0163**	0.0071
	max	-	0.1559	0.0971	0.0965	0.1114	0.0907	0.1375	-	**0.0669**	0.0572
	mean	-	-	-	-	-	-	-	0.1647	**0.0240**	0.0121
Distortion	median	-	-	-	-	-	-	-	0.1102	**0.0176**	0.0066
	max	-	-	-	-	-	-	-	1.0509	**0.1082**	0.1137
	mean	61.7820	89.4821	87.9500	89.3505	90.0521	90.8913	88.6629	66.8490	**93.0248**	98.2320
Inlier (%)	median	68.0180	91.7491	90.3534	92.2825	92.1621	92.7361	90.9337	70.3435	**94.9796**	98.2320
	max	94.9309	99.1837	99.0581	99.3407	99.3563	**99.6711**	99.3856	96.7742	99.1031	100.0000

Table 6.2: Full results for the *NotreDame* dataset: Comparison of different solvers on images with principal point shift $> 6\%$. The table shows the relative errors except for the rotation errors which are in degrees. For the principal point the error is relative to the image size. The best results (excluding P7PFRUV with bundle adjustment) are marked bold.

		P3.5PF [137]	P6P [81]	P5PFUV [214]	**P4.5PFUV**	**P4.5PFUV (SVD)**	**P4.5PFUV (6pt)**	**P5PFUVA**	P4PFR [137]	**P7PFRUV**	**P7PFRUV (b)**
Focal length	mean	0.0351	0.0206	0.0238	0.0225	0.0191	**0.0155**	0.0214	0.0236	0.0223	0.0034
	median	0.0240	0.0124	0.0136	0.0142	0.0157	**0.0096**	0.0150	0.0159	0.0166	0.0015
	max	0.2235	0.0920	0.1123	0.1449	0.0962	**0.0535**	0.1137	0.0870	0.0757	0.0173
Rotation	mean	1.1164	**0.9121**	1.0709	1.1863	1.1872	0.9806	1.0387	1.0603	0.9169	0.2649
	median	0.9137	0.8007	0.9329	0.9518	1.1145	0.8911	**0.7650**	0.9135	0.7830	0.1866
	max	3.2030	2.5408	3.0633	3.4579	3.6908	2.6247	4.2505	2.7917	**2.3552**	1.9464
Translation	mean	0.2005	0.0818	0.1064	0.0820	0.0763	**0.0724**	0.0902	0.1142	0.0934	0.0126
	median	0.0861	0.0361	0.0478	0.0533	0.0542	**0.0356**	0.0481	0.0568	0.0553	0.0072
	max	2.2931	**0.4828**	0.5347	0.5100	0.5809	0.7289	0.7050	1.2441	0.5929	0.0772
Principal point	mean	-	0.0257	0.0280	0.0336	0.0335	0.0280	0.0275	-	**0.0236**	0.0077
	median	-	0.0191	0.0204	0.0213	0.0231	0.0188	**0.0152**	-	0.0187	0.0043
	max	-	0.2263	0.2829	0.4484	0.4411	0.3877	0.3491	-	**0.1735**	0.0808
Distortion	mean	-	-	-	-	-	-	-	**0.0349**	0.0631	0.0195
	median	-	-	-	-	-	-	-	**0.0136**	0.0161	0.0091
	max	-	-	-	-	-	-	-	**0.6777**	1.9341	0.3840
Inlier (%)	mean	83.6978	87.0561	86.1617	86.9281	87.7019	88.4123	86.7742	92.3377	**93.3923**	98.4141
	median	85.4583	89.7185	90.3243	90.7809	90.5397	92.1720	90.1257	93.8954	**94.5743**	98.4141
	max	**99.6805**	99.2391	99.0415	99.2063	99.4610	99.3976	98.8903	**99.6805**	99.3610	100.0000

Table 6.3: Full results for the *NotreDame* dataset: Comparison of different solvers on images with principal point shift $< 6\%$. The table shows the relative errors except for the rotation errors which are in degrees. For the principal point the error is relative to the image size. The best results (excluding P7PFRUV with bundle adjustment) are marked bold.

		P3.5PF [137]	P6P [81]	P5PFUV [214]	**P4.5PFUV**	**P4.5PFUV (SVD)**	**P4.5PFUV (6pt)**	**P5PFUVA**	P4PFR [137]	**P7PFRUV**	**P7PFRUV** (b)
Focal length	mean	0.3696	0.2199	0.1667	0.1699	0.1666	0.1696	0.1928	0.0845	**0.0012**	0.0008
	median	0.2629	0.1564	0.1386	0.1319	0.1290	0.1368	0.1494	0.0540	**0.0010**	0.0007
	max	4.1083	0.5135	0.5066	0.4187	0.4461	0.4217	0.5450	0.5896	**0.0040**	0.0053
Rotation	mean	14.6800	10.7499	8.4318	8.4168	8.3194	8.2880	9.7119	13.6371	**0.2558**	0.1837
	median	13.3526	10.0873	7.8776	8.1739	7.8868	7.6738	9.4980	13.3771	**0.2191**	0.1588
	max	176.5515	24.4273	22.8580	20.9626	22.7519	22.4078	27.0481	21.1274	**0.7516**	0.7629
Translation	mean	0.4538	0.1687	0.1446	0.1436	0.1434	0.1450	0.1535	0.2181	**0.0043**	0.0031
	median	0.2673	0.1218	0.1175	0.1170	0.1136	0.1129	0.1211	0.2062	**0.0038**	0.0026
	max	5.9461	0.8867	0.6083	0.6217	0.6804	0.7358	0.6327	0.7070	**0.0138**	0.0138
Principal point	mean	-	0.0635	0.0619	0.0616	0.0614	0.0608	0.0655	-	**0.0022**	0.0016
	median	-	0.0633	0.0536	0.0556	0.0539	0.0535	0.0624	-	**0.0018**	0.0012
	max	-	0.1710	0.2004	0.1857	0.2008	0.1764	0.1798	-	**0.0080**	0.0080
Distortion	mean	-	-	-	-	-	-	-	0.2042	**0.0067**	0.0045
	median	-	-	-	-	-	-	-	0.1271	**0.0054**	0.0034
	max	-	-	-	-	-	-	-	2.9758	**0.0334**	0.0232
Inlier (%)	mean	21.26	40.46	34.73	36.04	36.81	36.93	37.45	33.18	**96.78**	97.49
	median	18.67	38.55	31.97	32.73	33.58	33.98	35.22	29.93	**97.06**	97.80
	max	55.03	67.12	63.42	62.06	64.12	65.65	65.76	64.50	**98.93**	99.19

Table 6.4: Full results for the cropped *Rotunda* dataset: Comparison of different solvers on 248 images with radial distortion and shifted principal point. The table shows the relative errors except for the rotation errors which are in degrees. For the principal point the error is relative to the image size. The best results (excluding P7PFRUV with bundle adjustment) are marked bold.

Chapter 7

Convex Relaxations for Low Rank Matrix Approximation

Low rank approximation is an important tool in many applications. Given an observed matrix with elements corrupted by noise it is possible to find the best approximating matrix in Frobenius sense of a given rank through singular value decomposition. However, due to the non-convexity of the formulation it is not possible to incorporate any additional knowledge of the sought matrix without resorting to heuristic optimization techniques.

In this chapter we propose a convex formulation that is more flexible in that it can be combined with any other convex constraints and penalty functions. The formulation uses the so called convex envelope, which is the provably best possible convex relaxation. We show that for a general class of problems the envelope can be efficiently computed and may in some cases even have a closed form expression. We test the algorithm on a number of real and synthetic datasets and show state-of-the-art results.

This chapter is based on the papers [136, 139, 140].

7.1 Introduction

Low rank approximation procedures such as PCA are important tools for various dimensionality reduction problems. The fundamental result that makes these approaches possible is due to [53] which shows that a closed form solution to the problem

$$\min_{\text{rank}(X)=r_0} \|X - X_0\|_F^2, \tag{7.1}$$

can be obtained from the singular value decomposition of X_0. In this chapter we consider the more general formulation

$$\min_{\text{rank}(X)=r_0} \|X - X_0\|_F^2 + \mathcal{C}(X). \tag{7.2}$$

Here the function $\mathcal{C}(X)$ represents any additional prior knowledge that we may have on the entries of X. The only requirement we make is that $\mathcal{C}$ is convex.

Adding a convex function may seem to be a very minor change. Unfortunately, the problem in (7.1) is not convex and any modification makes it much more difficult since the SVD approach is no longer applicable.

In order to make the formulation more flexible it is desirable to replace the complicated rank constraint with a convex surrogate. Typically a soft penalty on the singular values is adopted resulting in

$$\min_X \mu\|X\|_* + \|X - X_0\|_F^2, \tag{7.3}$$

where the so called nuclear norm is given by

$$\|X\|_* = \sum_{i=1}^{n} \sigma_i(X), \tag{7.4}$$

and n is the number of singular values. The rationale behind this choice is that penalizing the sum of singular values is likely to produce results where many of them are set to zero, thus resulting in a solution of low rank. The connection to the rank function is given in [63] where it is shown that the nuclear norm is the convex envelope of the rank function on the set $\{X; \sigma_1(X) \leq 1\}$, where $\sigma_1(X)$ is the largest singular value of X. The constraint $\sigma_1(X) \leq 1$ is artificial and only added since the convex envelope of the rank function on the whole domain is simply the zero function.

The nuclear norm approach is very general and can in principle be applied to a wide range of problems as long as the remaining terms are convex. On the downside, since all singular values are penalized rather than just the small ones, the formulation introduces an unwanted bias to solutions with small elements.

There is for example no choice of μ such that the relaxed problem in (7.3) will give the correct solution to (7.1) (unless X_0 is already of low rank).

In contrast, in this chapter we derive a convex formulation that has the same optimizer as (7.1). The key observation is that if one considers not just the rank function but also the data term $\|X - X_0\|_F^2$ one can derive a significantly more accurate convex approximation of (7.1), without the need for adding artificial constraints. Compared to the nuclear norm constraint $\{X; \sigma_1(X) \leq 1\}$, the term $\|X - X_0\|_F^2$ effectively translates the feasible region to a neighborhood around X_0 instead of the origin. Therefore our convex envelope can penalize smaller singular values harder than larger ones, while the nuclear norm tries to force all singular values to be zero. Specifically, we show that it is possible to compute the convex envelope of functions of the type

$$f_g(X) = g(\text{rank}(X)) + \|X - X_0\|_F^2. \tag{7.5}$$

The only assumption we make is that the function g can be written

$$g(k) = \begin{cases} g_0 & \text{if} \quad k = 0 \\ g_0 + \sum_{i=1}^{k} g_i & \quad \text{otherwise} \end{cases}, \tag{7.6}$$

where the sequence g_i is non-negative and non-decreasing for $1 \leq i \leq n$. It is easy to see that this is possible if g is convex and non-decreasing on $\mathbb{R}$. Furthermore, we will assume that $g_0 = 0$ since subtracting a constant from the objective function does not affect the minimizers (and only subtracts a constant from the convex envelope).

In particular we are interested in two special cases of g. With a slight abuse of notation we define

$$f_\mu(X) = \mu \, \text{rank}(X) + \|X - X_0\|_F^2 \tag{7.7}$$

and

$$f_{r_0}(X) = \mathbb{I}(\text{rank}(X) \leq r_0) + \|X - X_0\|_F^2, \tag{7.8}$$

where $\mathbb{I}(\text{rank}(X) \leq r_0) = 0$ if $\text{rank}(X) \leq r_0$ and ∞ otherwise. The first formulation consists of a trade-off between rank and measurement fit using a soft rank penalty. In many practical applications it is however not interesting to lower the rank further than a predefined value r_0. In such cases the second formulation is more appropriate. Note that in this case we select $g_k = 0$ for $k \leq r_0$ and $g_k = \infty$ for $k > r_0$.

For many applications any of the two formulations can be used. Given a value of r_0 it is for example possible to find a value of μ such that f_μ and f_{r_0} have the same optimizers and vice versa. For applications involving the ranks of multiple interdependent matrices searching for the right parameters can become a difficult task. In such cases it is of interest to directly specify the correct prior knowledge. In applications with multiple matrices we will typically assume that the rank is known a priori and therefore we often prefer to use f_{r_0}. On the other hand the convex envelope of f_μ is mathematically easy to handle since it has a closed form expression. Additionally, it is intuitively easy to understand and compare to existing approaches.

Minimizing the convex envelope of f_g gives the same result as minimizing f_g itself. The significant advantage of using the envelope instead is that it is convex and therefore can be combined with other convex constraints and functions. For example, in the presence of missing data we propose to utilize it as illustrated in Figure 7.13 on page 193. More specifically, our convex relaxation is applied to sub-blocks of the matrix with no missing entries rather than the nuclear norm of the entire matrix X. In effect, this can be seen as minimizing the rank of each sub-block separately, and due to the convexity of our approximation, it is possible to enforce that the sub-blocks agree on their overlap. Furthermore, we show that under mild assumptions it possible to extract a solution to the full matrix X that has rank equal to the largest rank of the sub-blocks. We present an ADMM [18] based approach for obtaining a solution which only requires to compute SVDs of the sub-blocks rather than the whole matrix resulting in an efficient implementation. To enable efficient optimization we show how to compute the proximal operators for our convex relaxations.

7.1.1 Related Work

The ability to find the best fixed rank approximation of a matrix has been proven useful in applications such as rigid and non rigid Structure-from-Motion, photometric stereo and optical flow [211, 20, 228, 72, 17, 71]. The rank of the approximating matrix typically describes the complexity of the solution. For example, in non-rigid Structure-from-Motion the rank measures the number of basis elements needed to describe the point motions [20]. Under the assumption of gaussian noise the rank approximation problem can be solved optimally in a least squares sense using SVD [53], but the strategy is limited to problems without missing data and outliers.

In the presence of missing data the general problem becomes much more difficult, some versions even NP-hard [73]. In [10] it is shown that replacing the Frobenious norm with the spectral norm yields a closed form solution if the missing data forms a Young pattern and thus the globally optimal solution can be computed. For general problems [10] proposes an alternation over Young patterns.

A recent heuristic that has been shown to work well is to replace the rank function with the nuclear norm [180, 37, 72, 175, 9]. It can be shown [180, 37] that if the location of the missing entries are random then the nuclear norm approach provides a good approximation. However, in many applications such as Structure-from-Motion, where missing entries are highly correlated, they have been shown to perform poorly (e.g. [136, 175]).

In an effort to strengthen the nuclear norm formulation, [9] adds prior information using the generalized trace norm. The formulation is related to (7.3) by a reweighing of the data term and can incorporate knowledge such as smoothness priors. The availability of such information can improve the estimation, however the formulation still uses the nuclear norm for regularization. On a high level our approach is similar to [9] in that it also attempts to find a stronger convex relaxation by considering not just the rank function but also the data. However, in contrast to [9] we do not add priors to the problem but simply use the information in the available measurements.

If the rank of the sought matrix is known, bilinear parametrizations with local optimization are often employed. Buchanan and Fitzgibbon [23] showed that alternating methods often exhibit very slow convergence. Instead they proposed a damped gauss-newton update that jointly optimizes over both factors. In [35] it was observed that the damping factor can be seen as a nuclear norm regularization term thereby unifying the use of bilinear parametrization and nuclear norm minimization. In [172] it was illustrated that the Wiberg elimination strategy [225] is very robust to local minima.

In [106] the ℓ_1 norm is used to address outliers. The proposed alternating approach is shown to converge slowly in [58]. Instead [58, 203, 233] propose generalizations of the Wiberg approach designed to handle the non differenetiable objective function while jointly updating the two factors. While experimental results indicate robustness to suboptimal solutions, these methods are still local in that updates are computed using a local approximation of the objective function.

In sparse prediction it has been observed that the standard ℓ_1 approach shrinks too many of the variables in settings where these are highly correlated. This has motivated the use of elastic nets [236] and more recently the so called k-support norm [11, 132]. This norm is a convex relaxation of a combination of cardinality and the ℓ_2 norm. In [159, 59] these results are generalized to a combination of rank and Frobenious norm for the matrix setting.

The work most similar to ours is perhaps [101] where the convex envelope of a vector version of our problem (the cardinality function and the ℓ_2-norm) is computed. The underlying idea of the above line of work is that approximating the cardinality/rank function without regard for the remaining objective functions may yield relaxations unsuitable for the problem at hand. Rather, it is desirable to consider as much of the objective function as possible when computing the convex envelopes. Specifically, in our work, we propose a problem specific relaxation that considers as much of the measurement data as possible. In the above works the additional ℓ_2-norm is centered around the origin, and will actually serve to penalize large non-zero components hard, which is the opposite of what we aim for here. In our framework the use of a non-centered Frobenious-norm is an essential component required to avoid penalizing large singular values.

In [76] Grussler et al. present another derivation of the convex envelope of Frobenious loss together with the indicator function for rank r matrices, which is a special case of the convex envelopes presented in this chapter. In [76] they present some results for the relaxation gaps, as well as an SDP-representation for the convex envelope which can be useful for optimization in some settings. They also present an extension where they allow non-integer values for the rank to allow for stronger penalties for cases where the envelope is not tight.

From an algorithmic point of view there are several methods that similar to us have employed proximal operators for inference in rank approximation problems. In the context of nuclear norm optimization [36, 150, 158, 37, 62] employ these techniques in order to avoid costly semidefinite programming problems associated with the nuclear norm.

7.1.2 Notation

Throughout the chapter we use $\sigma_i(X)$, $i = 1, ..., n$ to denote the ith singular value of a matrix X. Here n denotes the number of singular values and for notational convenience we will also define $\sigma_0(X) = \infty$ and $\sigma_{n+1}(X) = 0$.

The vector of all singular values is denoted $\boldsymbol{\sigma}(X)$. With some abuse of notation we write the SVD of X as $U \operatorname{diag}(\boldsymbol{\sigma}(X))V^T$. For ease of notation we do not explicitly indicate the dependence of U and V on X since this will be clear from the context. The scalar product is defined as $\langle X, Y \rangle = \operatorname{tr}(X^T Y)$, where tr is the trace function, and the Frobenius norm $\|X\|_F = \sqrt{\langle X, X \rangle} = \sqrt{\sum_{i=1}^n \sigma_i^2(X)}$. Truncation at zero is denoted $[a]_+$, that is, $[a]_+ = 0$ if $a < 0$ and a otherwise. Elementwise multiplication is denoted $\odot$.

7.2 Convexification

In this section we show that it is possible to compute the convex envelope of f_g in (7.5). To compute the envelope we will use the Fenchel conjugate [182]

$$f_g^*(Y) = \sup_X \ \langle Y, X \rangle - f_g(X). \tag{7.9}$$

The convex envelope is then given by computing the biconjugate f^{**}. See the Introduction chapter for more details.

7.2.1 The Conjugate Function

The first step in finding the convex envelope is the computation of the conjugate function. By inserting (7.5) in (7.9) and completing squares, the conjugate can be written

$$\max_k \sup_{\operatorname{rank}(X)=k} \|Z\|_F^2 - \|X_0\|_F^2 - \|X - Z\|_F^2 - \sum_{i=1}^k g_i, \tag{7.10}$$

where

$$Z = \frac{1}{2}Y + X_0. \tag{7.11}$$

Note that the first two terms are independent of X and can be considered as constants in the maximization over X and k. In addition k is fixed in the inner maximization. For a fixed k, the maximizing X is given by the best rank k approximation of $Z = \frac{1}{2}Y + X_0$ which can be obtained from an SVD of Z by setting all singular values but the k largest to zero [53]. Inserting into (7.10) we get

$$\max_k \|Z\|_F^2 - \|X_0\|_F^2 - \sum_{i=k+1}^{n} \sigma_i^2(Z) - \sum_{i=1}^{k} g_i. \tag{7.12}$$

Recall that the sequence g_i is non-decreasing by assumption and the singular values $\sigma_i(Z)$ are non-increasing. To select the best k we note that the largest value is achieved when k fulfills

$$\sigma_k^2(Z) \geq g_k \text{ and } g_{k+1} \geq \sigma_{k+1}^2(Z). \tag{7.13}$$

For this maximizing k the last two sums can be written

$$\sum_{i=k+1}^{n} \sigma_i^2(Z) + \sum_{i=1}^{k} g_i = \sum_{i=1}^{n} \min\left(g_i,\ \sigma_i^2(Z)\right). \tag{7.14}$$

Therefore we get the conjugate function

$$f_g^*(Y) = \left\| \frac{1}{2}Y + X_0 \right\|_F^2 - \|X_0\|_F^2 - \sum_{i=1}^{n} \min\left(g_i,\ \sigma_i^2(\frac{1}{2}Y + X_0)\right). \tag{7.15}$$

7.2.2 The Convex Envelope

We next proceed to compute the bi-conjugate of (7.5). To keep the notation simple we again change variables to $Z = \frac{1}{2}Y + X_0$ and maximize over Z instead. Inserting (7.15) into the definition of the biconjugate, we get $f^{**}(X) =$

$$\sup_Z \ 2\langle X, Z - X_0 \rangle - \|Z\|_F^2 + \|X_0\|_F^2 + \sum_{i=1}^{n} \min(g_i,\ \sigma_i^2(Z)). \tag{7.16}$$

The first three terms can, by completing squares, be simplified into $\|X - X_0\|_F^2 - \|Z - X\|_F^2$. Furthermore, since $\|X - X_0\|_F^2$ does not depend on Z we get

$$f_g^{**}(X) = \mathcal{R}_g(X) + \|X - X_0\|_F^2, \tag{7.17}$$

where

$$\mathcal{R}_g(X) = \max_Z \left(\sum_{i=1}^{n} \min\left(g_i,\ \sigma_i^2(Z)\right) - \|Z - X\|_F^2 \right). \tag{7.18}$$

The sum in (7.18) only depends on the singular values of Z and is therefore unitarily invariant. We also note that $-\|Z-X\|_F^2 = -\|Z\|_F^2 + 2\langle Z, X\rangle - \|X\|_F^2$. The term $\|Z\|_F^2$ is unitarily invariant and by von Neumann's trace inequality, we know that $\langle Z, X\rangle \leq \sum_{i=1}^n \sigma_i(Z)\sigma_i(X)$. Equality is achieved if X and Z have SVDs with the same U and V. Hence, for Z to maximize (7.18), its SVD should be of the form $Z = U\operatorname{diag}(\boldsymbol{\sigma}(Z))V^T$ if $X = U\operatorname{diag}(\boldsymbol{\sigma}(X))V^T$. This reduces the maximization in (7.18) to

$$\max_{\boldsymbol{\sigma}(Z)} \left(\sum_{i=1}^n \min\left(g_i, \sigma_i^2(Z)\right) - \sum_{i=1}^n (\sigma_i(Z) - \sigma_i(X))^2 \right). \tag{7.19}$$

Note that the elements of $\boldsymbol{\sigma}(Z)$ have to fulfill

$$\sigma_1(Z) \geq \sigma_2(Z) \geq \ldots \geq \sigma_n(Z) \geq 0, \tag{7.20}$$

since these are singular values.

What is left now is to determine the singular values of Z. For the general case in (7.5) there does not seem to be any closed form solution. None the less, the problem has some properties that allow us to efficiently enumerate and search the possible sets of maximizing singular values. The concave maximization problem could in principle be solved using an equivalent second order cone formulation with standard solvers like [1, 204]. However, as we shall see the particular properties of this problem allow us to solve it much more efficiently.

Properties of the Optimal $\boldsymbol{\sigma}(Z)$

Considering each singular value $\sigma_i(Z)$ separately they should solve a program of the type

$$\max_s \quad \min(g_i, s^2) - (s - \sigma_i(X))^2 \tag{7.21}$$

$$\text{s.t.} \quad \sigma_{i+1}(Z) \leq s \leq \sigma_{i-1}(Z) \tag{7.22}$$

Note that for $i = 1$ there is no upper bound on s and for $i = n$ there is no positive lower bound since we use the convention that $\sigma_0(Z) = \infty$ and $\sigma_{n+1}(Z) = 0$.

We first consider the unconstrained objective function. This function is the pointwise minimum of the two concave functions $g_i - (s - \sigma_i(X))^2$ (for $s \geq \sqrt{g_i}$) and $s^2 - (s - \sigma_i(X))^2 = 2s\sigma_i(X) - \sigma_i^2(X)$. The function is concave and attains its optimum in $s = \sigma_i(X)$ if $\sigma_i(X) \geq \sqrt{g_i}$ and in $s = \sqrt{g_i}$ otherwise (see Figure 7.1). In case $\sigma_i(X) = 0$ the optimum is not unique. For simplicity we will assume that $\sigma_i(X) > 0$ in what follows. The solution we create will still be valid if $\sigma_i(X) = 0$ but might not be unique. Let s_i be the individual

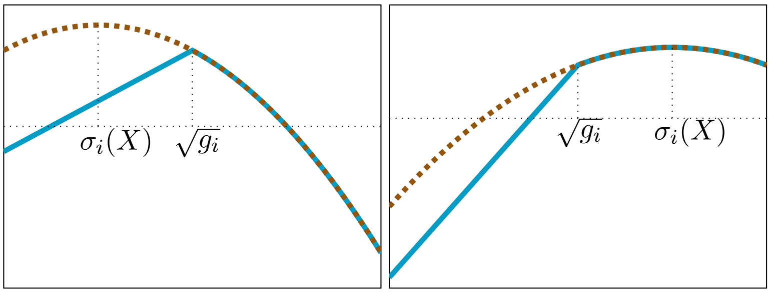


Figure 7.1: The objective function in (7.21) for $\sigma_i(X) \leq \sqrt{g_i}$ and $\sigma_i(X) \geq \sqrt{g_i}$.

unconstrained optimizers of (7.21), i.e.

$$s_i = \max(\sqrt{g_i}, \sigma_i(X)). \tag{7.23}$$

Note that this sequence is decreasing when $\sigma_i(X)$ is larger than $\sqrt{g_i}$ and increasing after that. We choose p such that s_p is the smallest value in the sequence s_i.

We now consider the constrained problem (7.21)-(7.22). Since the cost function is separable and concave the optimal $\sigma_i(Z)$ must be either the unconstrained optimum s_i or equal to one of its neighbors $\sigma_{i+1}(Z)$ and $\sigma_{i-1}(Z)$. The search over singular values can be further reduced using the following lemma.

Lemma 7.1. *If $Z^\star$ is an optimal solution to* (7.19) *then there is a value $s_p \leq s \leq s_n$ such that*

$$\sigma_i(Z^\star) = \max(s, s_i), \qquad \text{if } i \leq p, \tag{7.24}$$

$$\sigma_i(Z^\star) = s, \qquad \text{if } i \geq p. \tag{7.25}$$

The above result is a consequence of Theorem 7.5 (with $q = n$) which we prove in Appendix 7.A.2. It essentially says that we can limit our search to a 1-parameter family of vectors parametrized by s. For the decreasing part of the sequence $\{s_i\}$ the optimal singular values are selected by truncation at s and for the increasing part we let all singular values be s. In the next section we show how to find the optimal s through a simple linear search.

Finding the Optimal $\boldsymbol{\sigma}(Z)$

Finding the optimal $Z^\star$ amounts to a one dimensional search over the unknown parameter s. We let $\boldsymbol{\sigma}(s)$ be given by (7.24) and (7.25) and the cost of each singular value

$$c_i(s) = \min(g_i, \sigma_i^2(s)) - (\sigma_i(s) - \sigma_i(X))^2. \tag{7.26}$$

For $i \geq p$, we have $\sigma_i(s) = s$ and therefore c_i is concave in s with a non-differentiable point at $s = \sqrt{g_i}$. For $i < p$ we have $\sigma_i(s) \geq s_i \geq \sqrt{g_i}$ due to (7.23), which gives the cost function

$$c_i(s) = g_i - [s - \sigma_i(X)]_+^2. \tag{7.27}$$

This function is concave and differentiable for $s \neq \sigma_i(X)$. (In fact it is differentiable everywhere since the derivative of $(s - \sigma_i(X))^2$ is zero at $s = \sigma_i(X)$, but our algorithm does not utilize this.) The resulting objective function $c(s) = \sum_{i=1}^n c_i(s)$ is therefore concave on $s \geq s_p$ and piecewise differentiable.

Our general algorithm for maximizing this function consists of three steps:

1. Compute the sequence of unconstrained maximizers s_i and determine p.
2. Sort the breakpoints $s_p, \sqrt{g_{p+1}}, \sqrt{g_{p+2}}, ..., \sqrt{g_n}, \sigma_1(X), \sigma_2(X), ..., \sigma_{p-1}(X)$, in ascending order. Let $p_1, ..., p_m$ be the sorted points.
3. Maximize $c(s)$ on each subinterval $[p_i, p_{i+1}]$.

The last step requires evaluating at most two points for each interval (a possible stationary point and a boundary point). Furthermore, the sequences $\sqrt{g_i}$ and $\sigma_i(X)$ are already sorted and can therefore be merged in linear time. Hence given an SVD of the matrix X it is easy to see that the steps of the algorithm are linear in the number of singular values.

The Case of f_μ^{**}

In the special case of (7.7) we are able to determine the maximizing $\boldsymbol{\sigma}(Z)$ without iteration. Here we have $g(k) = \mu k$ and therefore $g_k = \mu$ for all $k > 0$. In this case $\{s_k\}$ will be decreasing for all k. The constraints (7.22) are automatically fulfilled and the sequence

$\{\max(\sqrt{\mu}, \sigma_k(X))\}$ will therefore contain the optimal singular values. Inserting into (7.17) gives, after some simplifications,

$$f_\mu^{**}(X) = \mathcal{R}_\mu(X) + \|X - X_0\|_F^2, \tag{7.28}$$

where

$$\mathcal{R}_\mu(X) = \sum_{i=1}^{n} \left(\mu - [\sqrt{\mu} - \sigma_i(X)]_+^2\right). \tag{7.29}$$

In [223] the authors propose a rank regularizer which for some parameter choices is equivalent to $\mathcal{R}_\mu$. However, they make no connection to the convex envelope of f and simply minimize it in a non-convex framework.

Figure 7.2 shows a one dimensional version of (7.28). To the left is the term $\mu - \left[\sqrt{\mu} - \sigma\right]_+^2$ which is in itself not convex. For singular values larger than $\sqrt{\mu}$ it gives a constant penalty. When the quadratic term σ^2 is added the result is a convex penalty, see the middle graph in Figure 7.2. For $\sigma < \sqrt{\mu}$ the function has a linear shape (red dashed curve) similar to the nuclear norm, while for $\sigma \geq \sqrt{\mu}$ it behaves like the quadratic function $\mu + \sigma^2$. Note that the one dimensional version of f_μ is identical to $\mu + \sigma^2$ everywhere except for $\sigma = 0$. In the right image we plotted the graphs of $\mu - \left[\sqrt{\mu} - \sigma\right]_+^2 + (\sigma - \sigma_0)^2$ for $\sigma_0 = 0, 1, 2$. If σ_0 is large enough the function will not try to force σ to be zero.

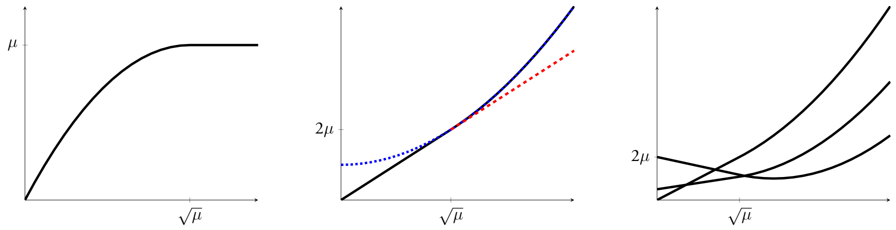


Figure 7.2: One dimensional visualizations of (7.28) for $\mu = 2$. Left: The graph of $\mu - \left[\sqrt{\mu} - \sigma\right]_+^2$. Middle: The graph of $\mu - \left[\sqrt{\mu} - \sigma\right]_+^2 + \sigma^2$. If μ is large its shape resembles the nuclear norm. Right: The graphs of $\mu - \left[\sqrt{\mu} - \sigma\right]_+^2 + (\sigma - \sigma_0)^2$ for $\sigma_0 = 0, 1, 2$.

The Case of $f_{r_0}^{**}$.

If $g_i = 0$ for $i \leq r_0$ and ∞ for $i > r_0$ we get that the sequence s_i (7.23) is minimized at $p = r_0$. Lemma 7.1 then shows that the optimizing $\boldsymbol{\sigma}(Z)$ is of the form

$$\sigma_i(Z) = \begin{cases} \max(\sigma_i(X), s), & i \leq r_0 \\ s, & i \geq r_0. \end{cases} \tag{7.30}$$

As a function of the unknown parameter s the cost for each singular value can therefore be written

$$c_i(s) = -[s - \sigma_i(X)]_+^2, \tag{7.31}$$

for $i \leq r_0$ and

$$c_i(s) = 2s\sigma_i(X) - \sigma_i^2(X), \tag{7.32}$$

for $i > r_0$. The resulting objective function $c(s) = \sum_{i=1}^n c_i(s)$ is concave and differentiable on $s \geq 0$. Furthermore, Lemma 7.1 shows that the optimal s fulfills $s \geq s_p$. Since $s_n = \infty$, the maximizing s will therefore be a stationary point of $c(s)$ in $[s_p, \infty)$. Note that the function is quadratic on the sub intervals

$$[\sigma_{r_0-i}(X), \sigma_{r_0-i-1}(X)], \quad i = 1, ..., r_0 - 1, \tag{7.33}$$

where $\sigma_0(X) = \infty$. Searching for stationary points in each interval amounts to solving a linear equation. Our algorithm therefore consists of a loop over these intervals. Note that when we have found a stationary point we can terminate the algorithm. Figure 7.3 shows a 3D illustration of the regularizer in the case of $f_{r_0}^{**}$.

7.3 Optimization and Performance Bounds

In this section we address the problem of minimizing $f_g(X) + \mathcal{C}(X)$. To achieve a convex formulation we replace f_g with its convex envelope f_g^{**}. Note that while any minimizer of f_g is also a minimizer of f_g^{**} the same may not hold when adding the function $\mathcal{C}$. However, the relaxed regularizer illustrated in Figure 7.2

Algorithm 2: Finding maximizing Z for $\mathcal{R}_{r_0}(X)$.

```
Data: $X, r_0$
Result: $\boldsymbol{\sigma}(Z^*)$
for $i = 0 : r_0 - 1$ do
    For the interval $[\sigma_{r_0-i}(X), \sigma_{r_0-i-1}(X)]$, compute $s^*$ by solving
      $\frac{d}{ds}c(s) = 0$;
    if $s^* \in [\sigma_{r_0-i}(X), \sigma_{r_0-i-1}(X)]$ then
        $\sigma_i(Z^*) := \max(\sigma_i(X), s^*), \quad \forall i \leq r_0$;
        $\sigma_i(Z^*) := s^*, \quad \forall i \geq r_0$;
        break;
    end
end
```

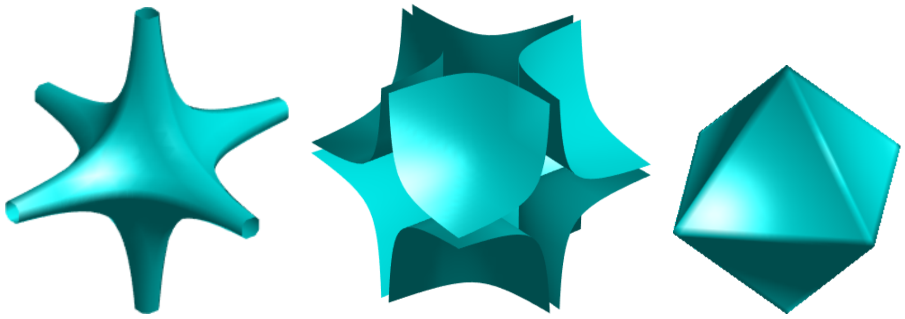


Figure 7.3: Level set surfaces $\{X \mid \mathcal{R}_{r_0}(X) = \alpha\}$ for $X = \operatorname{diag}(x_1, x_2, x_3)$ with $r_0 = 1$ (*Left*) and $r_0 = 2$ (*Middle*). Note that when $r_0 = 1$ the regularizer promotes solutions where only one of x_k is non-zero. For $r_0 = 2$ the regularlizer instead favors solutions with two non-zero x_k. For comparison we also include the level set of the nuclear norm. (*Right*)

still makes sense since it will penalize small singular values proportionally harder than large ones. Furthermore, since

$$f_g^{**}(X) + \mathcal{C}(X) \leq f_g(X) + \mathcal{C}(X), \tag{7.34}$$

for all X, we may determine if an optimal solution $X^\star$ of $f_g^{**}(X) + \mathcal{C}(X)$ is also optimal in $f_g(X) + \mathcal{C}(X)$ by comparing the objective values. Note that in contrast the nuclear norm formulation is not a proper lower bound on the whole domain and therefore it can not be used for verifying optimality in this way.

It is possible to make some simple theoretical estimates of the tightness of the relaxation f_g^{**} by analyzing $\mathcal{R}_g$. The relaxation gap is given by

$$\begin{aligned}0 &\leq f_g(X) - f_g^{**}(X) = g(\text{rank}(X)) - \mathcal{R}_g(X) \\ &= g(\text{rank}(X)) - \max_Z \sum_{i=1}^{n} \min(g_i, \sigma_i^2(Z)) - \|Z - X\|_F^2 \\ &\leq g(\text{rank}(X)) - \sum_{i=1}^{n} \min(g_i, \sigma_i^2(X))\end{aligned}$$

where the last inequality follows from setting $Z = X$ in the maximization. This can be further simplified by noting that for $i > \text{rank}(X)$ we must have $\min(g_i,\ \sigma_i^2(X)) = 0$. Thus the relaxation gap is bounded by

$$\sum_{i=1}^{\text{rank}(X)} g_i - \min(g_i, \sigma_i^2(X)) = \sum_{i=1}^{\text{rank}(X)} \left[g_i - \sigma_i^2(X)\right]_+ . \qquad (7.35)$$

Hence any matrix X whose non-zero singular values fulfill $\sigma_i^2(X) \geq g_i$ will have $f_g(X) = f_g^{**}(X)$.

In the special case of f_μ we have $g_i = \mu$, $\forall i$ and therefore any matrix with non-zero singular values larger than $\sqrt{\mu}$ has no gap. This is also is easily seen from Figure 7.2.

Similarly for f_{r_0} we have $g_i = 0$, for $i \leq r_0$ and $g_i = \infty$ for $i > r_0$. It then follows that

$$f_{r_0}(X) = f_{r_0}^{**}(X) \quad \forall X \text{ with } \text{rank}(X) \leq r_0. \qquad (7.36)$$

For optimization we employ the popular ADMM [18] approach. This is essentially a splitting scheme that uses two copies of the X and enforces them to be equal using dual variables. We formulate an augmented Lagrangian as

$$L(X, Y, \Lambda) = f_g^{**}(X) + \rho\|X - Y + \Lambda\|_F^2 + \mathcal{C}(Y) - \rho\|\Lambda_i\|_F^2. \qquad (7.37)$$

In each iteration t of ADMM the variable updates are given by

$$\begin{aligned}X_{t+1} &= \arg\min_X f_g^{**}(X) + \rho\|X - Y_t + \Lambda_t\|_F^2, && (7.38)\\ Y_{t+1} &= \arg\min_Y \rho\|X_{t+1} - Y + \Lambda_t\|_F^2 + \mathcal{C}(Y), && (7.39)\\ \Lambda_{t+1} &= \Lambda_t + X_{t+1} - Y_{t+1}. && (7.40)\end{aligned}$$

The updates in equations (7.38) and (7.39) are computed using the proximal operators of f_g^{**} and $\mathcal{C}$. In the next section we will show how to evaluate the proximal operator of f_g^{**}. The approach will be similar to the one presented in Section 7.2.2 and for the case of (7.7) we will show that the proximal operator admits a closed form solution.

7.3.1 The Proximal Operator

To evaluate the proximal operator of f_g^{**} we need to be able to efficiently solve a minimization problem of the following type

$$\min_X f_g^{**}(X) + \rho \|X - M\|_F^2 = \tag{7.41}$$

$$\min_X \mathcal{R}_g(X) + \|X - X_0\|_F^2 + \rho \|X - M\|_F^2 \tag{7.42}$$

Due to (7.18) this can be seen as convex-concave min-max problem (with an outer minimization over X and an inner maximization over Z). The key observation in order to be able to solve it efficiently is that we can switch the order of the maximization and minimization[1]. Performing the minimization in X first gives

$$X = M + \frac{X_0 - Z}{\rho} = \frac{(\rho+1)Y - Z}{\rho}, \tag{7.43}$$

where $Y = \frac{X_0 + \rho M}{1+\rho}$. Inserting into the objective function gives that the remaining maximization in Z is (ignoring constants)

$$\max_Z \sum_{i=1}^{n} \min\left(g_i, \sigma_i^2(Z)\right) - \frac{\rho+1}{\rho} \|Z - Y\|_F^2 . \tag{7.44}$$

Similarly to Section 7.2.2 we can reduce the search to the singular values of Z by noting that the first term is unitarily invariant and the second term is maximized when Z has the same U and V in its SVD as Y.

Each singular value $\sigma_i(Z)$ must then solve the following program

$$\max_s \ \min\left(g_i,\ s^2\right) - \frac{\rho+1}{\rho}\left(s - \sigma_i(Y)\right)^2 \tag{7.45}$$

$$\text{s.t. } \sigma_{i+1}(Z) \le s \le \sigma_{i-1}(Z) \tag{7.46}$$

[1]Since it is possible to restrict the minimization in X to a compact set the existence of a saddle point can be guaranteed (see [182] for details).

The objective function can be seen as the pointwise minimum of two quadratic functions,

$$q_1(s) = g_i - \frac{\rho+1}{\rho}\left(s - \sigma_i(Y)\right)^2, \tag{7.47}$$

$$q_2(s) = s^2 - \frac{\rho+1}{\rho}\left(s - \sigma_i(Y)\right)^2. \tag{7.48}$$

Since $\frac{\rho+1}{\rho} > 1$ both of these are concave quadratics and achieve their maximum at $s = \sigma_i(Y)$ and $s = (\rho+1)\sigma_i(Y)$ respectively. For the pointwise minimum three cases can occur (see also Figure 7.4):

1. The maximum occurs in the region where $s > \sqrt{g_i}$, that is, where $q_1(s) < q_2(s)$. In this case the maximum has to be that of q_1 which occurs at $s = \sigma_i(Y)$. Therefore we get that if $\sigma_i(Y) > \sqrt{g_i}$ then $\sigma_i(Y)$ is optimal.
2. The maximum occurs in the region where $s < \sqrt{g_i}$, that is, where $q_1(s) > q_2(s)$. In this case the maximum has to be that of q_2 which occurs at $s = (\rho+1)\sigma_i(Y)$. Therefore we get that if $(1+\rho)\sigma_i(Y) < \sqrt{g_i}$ then $(1+\rho)\sigma_i(Y)$ is optimal.
3. The maximum is in $s = \sqrt{g_i}$. This case occurs when $\sigma_i(Y) \leq \sqrt{g_i} \leq (\rho+1)\sigma_i(Y)$ or equivalently $\frac{1}{1+\rho}\sqrt{g_i} \leq \sigma_i(Y) \leq \sqrt{g_i}$.

Note that cases 1 and 2 are mutually exclusive since only one of $\sigma_i(Y) > \sqrt{g_i}$ and $(1+\rho)\sigma_i(Y) < \sqrt{g_i}$ can hold (with $\rho > 0$). (Having strict local maxima at both $\sigma_i(Y)$ and $(1+\rho)\sigma_i(Y)$ would also contradict the concavity of the objective function.)

Summarizing we get that the sequence of unconstrained maximizers s_i for each singular value is given by

$$s_i = \begin{cases} \sigma_i(Y), & \sqrt{g_i} \leq \sigma_i(Y) \\ \sqrt{g_i}, & \frac{1}{1+\rho}\sqrt{g_i} \leq \sigma_i(Y) \leq \sqrt{g_i} \\ (1+\rho)\sigma_i(Y), & \sigma_i(Y) \leq \frac{1}{1+\rho}\sqrt{g_i} \end{cases}. \tag{7.49}$$

Note that the sequence $\{s_i\}$ is first non-increasing with $\sigma_i(Y)$ up to some index p, then non-decreasing (with $\sqrt{g_i}$) from p to some index q and then again non-increasing (with $(1+\rho)\sigma_i(Y)$) from q to n. In what follows we will assume that $s_p < s_q$. If this is not the case the sequence $\{s_i\}$ will be non-increasing for all i

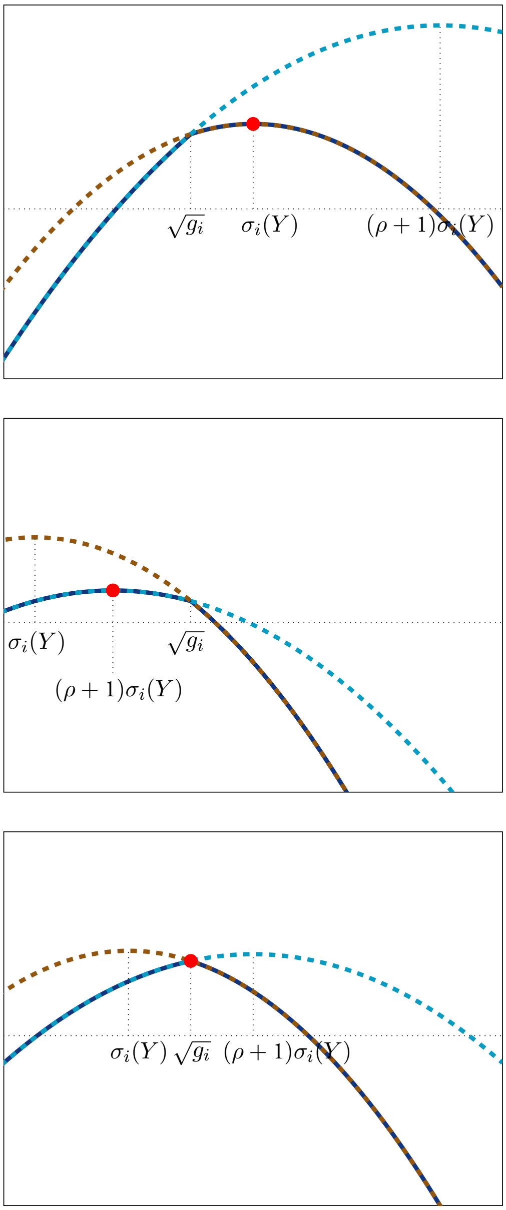


Figure 7.4: The three possible configurations for the maximum of $\min(q_1,\ q_2)$.

thus making it an optimal sequence of singular values for Z. Figure 7.5 shows an illustration of what the sequence can look like with the above definitions of p and q. Note that around both p and q the sequence can be constant, taking the values $s_p = \max(\sqrt{g_p}, \sigma_p(Y))$ and $s_q = \min(\sqrt{g_q}, (\rho+1)\sigma_q(Y))$. In Lemma 7.3 (of Appendix 7.A.1, page 201) we gives further details and proofs.

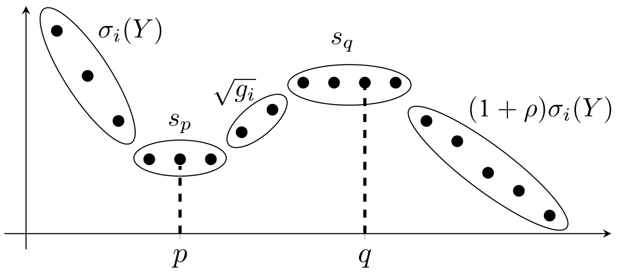


Figure 7.5: An example of sequence $\{s_i\}$ and the definition of p and q.

We now consider the constraints (7.46). Similarly to Section 7.2.2 we can reduce the maximization over the singular values to a one-dimensional search over a single parameter. Theorem 7.5 of Appendix 7.A.2 (page 204) shows that when $s_p < s_q$ the optimal set of singular values are of the form

$$\sigma_i(Z) = \max(s_i, s), \quad \text{if } i \leq p \tag{7.50}$$

$$\sigma_i(Z) = s, \quad \text{if } p \leq i \leq q \tag{7.51}$$

$$\sigma_i(Z) = \min(s_i, s), \quad \text{if } i \geq q. \tag{7.52}$$

We let $\boldsymbol{\sigma}(s)$ be given by (7.50)-(7.52) and the cost of each singular value

$$c_i(s) = \min(g_i, \sigma_i^2(s)) - \frac{\rho+1}{\rho}(\sigma_i(s) - \sigma_i(Y))^2. \tag{7.53}$$

Similarly to Section 7.2.2 the ith cost $c_i(s)$ will be concave and possibly non-differentiable at the breakpoints $s = \sqrt{g_i}$. Our algorithm for maximizing $\sum_{i=1}^n c_i(s)$ therefore consists of the following three steps:

1. Compute the sequence of unconstrained maximizers s_i and determine p and q.

2. Sort the breakpoints
$s_1, ..., s_n, \sqrt{g_{p+1}}, \sqrt{g_{p+2}}, ..., \sqrt{g_n},$ in ascending order. Remove doubles

and the ones that are either smaller than s_p or larger than s_q. Let $p_1, ..., p_m$ be the sorted points.

3. Maximize $c(s)$ on each subinterval $[p_i, p_{i+1}]$ and choose the best maximizer.

For each interval, the maximization consists of checking a boundary point and searching for a feasible stationary point for a quadratic objective function. The number of intervals that needs to be considered is linear in the number of singular values, and therefore the complexity of the search is linear.

The Proximal Operator of f_μ^{**}

For the special case of f_μ^{**} we have that all $g_i = \mu$. The unconstrained optimizers s_i from (7.49) will then be a non-increasing sequence. This implies that the optimal Z has $\sigma_i(Z) = s_i$. Inserting into (7.43) gives the singular values for the optimal X as

$$\sigma_i(X) = \begin{cases} \sigma_i(Y), & \sqrt{\mu} \le \sigma_i(Y) \\ \frac{(1+\rho)\sigma_i(Y) - \sqrt{\mu}}{\rho}, & \frac{1}{1+\rho}\sqrt{\mu} \le \sigma_i(Y) \le \sqrt{\mu} \\ 0, & \sigma_i(Y) \le \frac{1}{1+\rho}\sqrt{\mu} \end{cases}. \tag{7.54}$$

The proximal operators for the rank-function and the nuclear norm are hard and soft thresholding respectively. We note that the above operator is a mixture of both of these. Singular values larger than $\sqrt{\mu}$ are not truncated which is similar to hard thresholding. Whereas the middle case $\frac{1}{1+\rho}\sqrt{\mu} \le \sigma_i(Y) \le \sqrt{\mu}$ is similar to soft thresholding.

The Proximal Operator of $f_{r_0}^{**}$

For $f_{r_0}^{**}$ we have $g_i = 0$ if $i \le r_0$ and ∞ if $i > r_0$. This gives the sequence of unconstrained minimizers

$$s_i = \begin{cases} \sigma_i(Y), & i \le r_0 \\ (1+\rho)\sigma_i(Y), & i > r_0 \end{cases}, \tag{7.55}$$

and $p = r_0$, $q = r_0 + 1$. Theorem 7.5 now shows that the optimal singular values must be of the form

$$\sigma_i(Z) = \begin{cases} \max(\sigma_i(Y), s), & i \leq r_0 \\ \min((\rho+1)\sigma_i(Y), s), & i > r_0 \end{cases}, \tag{7.56}$$

and the optimal s is in $[\sigma_{r_0}(Y), (\rho+1)\sigma_{r_0+1}(Y)]$. For $i \leq r_0$ we now get the cost

$$c_i(s) = -\frac{\rho+1}{\rho}[s - \sigma_i(Y)]_+^2, \tag{7.57}$$

which is concave and differentiable everywhere. For $i \geq r_0 + 1$ we get after some simplifications that

$$c_i(s) = -\frac{[(\rho+1)\sigma_i(Y) - s]_+^2}{\rho} + (\rho+1)\sigma_i^2(Y). \tag{7.58}$$

This function is also concave and differentiable everywhere. Since there are no non-differentiable points, we simply search for any stationary point in $[\sigma_{r_0}(Y), (\rho+1)\sigma_{r_0+1}(Y)]$. The algorithm can be terminated when we find one. Algorithm 3 summarizes the approach.

Algorithm 3: Finding maximizing Z for $\mathcal{R}_{r_0}(X)$.

```
**Data:** $Y, r_0$
**Result:** $\boldsymbol{\sigma}(Z^*)$
Compute and sort the values
  $\sigma_1(Y), ..., \sigma_{r_0}(Y), (\rho+1)\sigma_{r_0+1}(Y), ..., (\rho+1)\sigma_n(Y)$
Let $p_1, ..., p_m$ be the sorted values in $[\sigma_{r_0}(Y), (\rho+1)\sigma_{r_0+1}(Y)]$.
**for** $i = 1 : m - 1$ **do**
    For the interval $[p_i, p_{i+1}]$, compute $s^*$ by solving $\frac{d}{ds}c(s) = 0$;
    **if** $s^* \in [p_i, p_{i+1}]$ **then**
        $\sigma_i(Z^*) := \max(\sigma_i(Y), s^*), \quad \forall i \leq r_0$;
        $\sigma_i(Z^*) := \min((\rho+1)\sigma_i(Y), s^*), \quad \forall i \geq r_0 + 1$;
        return;
    **end**
**end**
```

Note that the cost of evaluating the proximal operator is dominated by the computation of the SVD. To illustrate this we ran a small synthetic experiment (with $r_0 = 5$) where we computed the proximal operator for random matrices X_0 and M and with elements drawn from a normal distribution with unit variance. The average runtime for each matrix size can be seen in Figure 7.6. The average runtimes over all instances were 56 ms for computing the SVD and 0.1 ms for finding the stationary point.

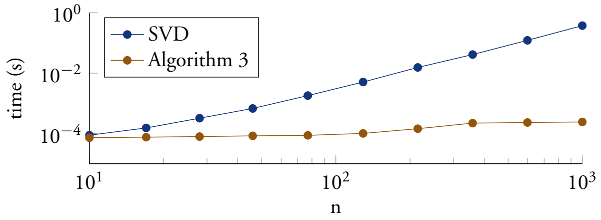


Figure 7.6: The average runtime for each matrix size. Note the logarithmic scales.

7.3.2 Relationship between the Relaxations.

It is easy to see that minimizing f_g can be done by finding a particular μ and minimizing f_μ instead. In this section we will show that the same holds for the convex relaxations $f_g^{**}(X)$ and $f_\mu^{**}(X)$. First we show that for each M there exist a μ such that the proximal operators are equal. Specifically, we show that when the unknown parameter s in (7.50)-(7.52) has been determined the singular values of the minimizer X of (7.42) can be written

$$\sigma_i(X) = \begin{cases} \sigma_i(Y), & s \leq \sigma_i(Y) \\ \frac{(1+\rho)\sigma_i(Y)-s}{\rho}, & \frac{1}{1+\rho}s \leq \sigma_i(Y) \leq s \\ 0, & \sigma_i(Y) \leq \frac{1}{1+\rho}s \end{cases} . \tag{7.59}$$

Therefore the problem can also be solved using f_μ^{**} with $\mu = s^2$.

Inserting (7.50)-(7.52) into (7.43) yields three cases

$$\sigma_i(X) = \begin{cases} \frac{(\rho+1)\sigma_i(Y)-\max(s,s_i)}{\rho}, & i \leq p \\ \frac{(\rho+1)\sigma_i(Y)-s}{\rho}, & p \leq i \leq q \\ \frac{(\rho+1)\sigma_i(Y)-\min(s,s_i)}{\rho}, & q \leq i \end{cases} . \tag{7.60}$$

To show that (7.60) is equivalent to (7.59) we use the properties of the sequence $\{s_i\}$ derived in Lemma 7.3, Appendix 7.A.1 (see also Figure 7.5). We first consider the case $i \leq p$ in (7.60). Since $s_i = \max(\sigma_i(Y), s_p)$ for $i \leq p$, due to (7.83), and $s \geq s_p$ we see that $\max(s, s_i) = \max(s, \sigma_i(Y))$. Therefore the terms $i \leq p$ can be written

$$\sigma_i(X) = \begin{cases} \frac{(\rho+1)\sigma_i(Y) - \sigma_i(Y)}{\rho} = \sigma_i(Y), & s \leq \sigma_i(Y) \\ \frac{(\rho+1)\sigma_i(Y) - s}{\rho}, & s \geq \sigma_i(Y) \end{cases}. \tag{7.61}$$

We also note that $s \leq s_q \leq (1+\rho)\sigma_q(Y) \leq (1+\rho)\sigma_i(Y)$ for all $i \leq q$ and therefore (7.61) can be written as (7.59).

For $i \geq q$ we have $s_i = \min(s_q, (\rho+1)\sigma_i(Y))$ by (7.83) and $s \leq s_q$, which shows that $\min(s, s_i) = \min(s, (\rho+1)\sigma_i(Y))$. Therefore the third case in (7.60) reduces to

$$\sigma_i(X) = \begin{cases} \frac{(\rho+1)\sigma_i(Y) - s}{\rho}, & s \leq (\rho+1)\sigma_i(Y) \\ 0, & s \geq (\rho+1)\sigma_i(Y) \end{cases}. \tag{7.62}$$

Additionally, $s \geq s_p \geq \sigma_p(Y) \geq \sigma_i(Y)$ for all $i \geq p$ which shows that (7.62) can be written as (7.59).

The second case $p \leq i \leq q$ in (7.60) is already of the right form. Furthermore, as we have seen all the terms in $p \leq i \leq q$ fulfill $\sigma_i(Y) \leq s \leq (\rho+1)\sigma_i(Y)$, which shows that (7.59) and (7.60) are equivalent.

The above argument establishes that for any M we can always find a μ such that the proximal operators give the same solution. Since this also holds at any optimizer of $f_g^{**} + \mathcal{C}$ we can easily show the following theorem.

Theorem 7.2. *There exist a μ such that*

$$\arg\min_X f_g^{**}(X) + \mathcal{C}(X) = \arg\min_X f_\mu^{**}(X) + \mathcal{C}(X). \tag{7.63}$$

Proof. A necessary and sufficient condition for $X^\star$ to optimize $f_g^{**} + \mathcal{C}$ is that there exist Λ such that

$$\begin{cases} 0 \in \partial f_g^{**}(X^\star) - \Lambda \\ 0 \in \partial \mathcal{C}(X^\star) + \Lambda \end{cases} \tag{7.64}$$

From the first condition we get

$$0 \in \partial f_g^{**}(X^\star) - \Lambda \iff \tag{7.65}$$

$$0 \in \partial f_g^{**}(X^\star) + 2\rho(X^\star - (X^\star + \frac{1}{2\rho}\Lambda)) \tag{7.66}$$

which is the first order condition for $X^\star$ to be optimal in (7.42) with $M = X^\star + \frac{1}{2\rho}\Lambda$. Since we can always find μ such that the proximal operators of f_g^{**} and f_μ^{**} are equal we get $0 \in \partial f_\mu^{**}(X^\star) - \Lambda$, which shows that $X^\star$ is also an optimal point for $f_\mu^{**} + \mathcal{C}$. □

The above result shows that for problem formulations involving f_g^{**} we can get an equivalent formulation with f_μ^{**} by finding a specific μ. In a sense the algorithm for the proximal operator of f_g^{**} from Section 7.3.1 can be seen as an efficient way of finding the correct μ. Note, that in case the problem formulation involves multiple matrices searching for the right combination of coefficients can be difficult. In such cases it is of great benefit to be able to have an efficient algorithm for doing this.

7.4 Single Matrix Applications

In this section we evaluate our convex envelopes on a couple of low rank estimation problems. We consider one application with linear constraints and one application with additional convex objective terms. In both cases we consider applications where a single matrix is sought.

7.4.1 Hankel Matrix Estimation

Hankel matrices are commonly occurring in various engineering applications. The rank of the Hankel matrix is often connected to the complexity of the system. For example, in the context of linear dynamical systems, [63] shows that if f is an impulse response of an order r_0 system, then the corresponding Hankel matrix $H(f)$ is of rank r_0. If f is a linear combination of r_0 complex exponentials (with arbitrary frequencies) then $H(f)$ is of rank r_0 [8], which is of importance in signal processing applications.

In this section we consider the problem of estimating a low rank Hankel matrix from a noisy measurement matrix X_0. We seek to solve

$$\min_{H \in \mathcal{H}} \mathbb{I}(\operatorname{rank}(H) \leq r_0) + \|H - X_0\|_F^2 \tag{7.67}$$

where $\mathcal{H}$ is the set of Hankel matrices. Note that as Hankel matrices are constant along anti-diagonals the constraint $X \in \mathcal{H}$ corresponds to linear constraints. Moreover the proximal operator of $\mathbb{I}(X \in \mathcal{H})$ corresponds to projection onto $\mathcal{H}$ and amounts to taking average values of the anti-diagonals.

In this experiment we generated Hankel matrices of rank $r_0 = 8$ by randomly sampling damped sinusoids according to

$$f(t) = \sum_{i=1}^{4} e^{d_i(t-t_i)} \cos(\phi_i(t - t_i)), \tag{7.68}$$

where d_i and t_i are sampled from a uniform distribution on $[-1, 1]$ and ϕ_i from $[-20\pi, 20\pi]$. Note that each term of the sum can be realized with 2 complex exponentials and therefore the corresponding matrix $H(f)$ will be of rank 8. We added Gaussian noise with varying standard deviation σ and tried to recover the true matrix by solving (7.67). Figure 7.7 shows the generated signal, its corresponding Hankel matrix and the added noise for one instance of the problem. Figure 7.8 shows the results of using $f_{r_0}^{**}(H)$ on the data of Figure 7.7. For comparison we also plot the results of ignoring the Hankel constraint and simply taking the SVD of X_0 as well as ignoring the low rank objective and only projecting X_0 on the closest Hankel matrix.

Figure 7.9 shows the result of optimizing the three relaxations $f_\mu^{**}(H)$, $f_{r_0}^{**}(H)$ and $\mu\|H\|_* + \|H - X_0\|_F^2$ with the Hankel constraint. For the nuclear norm and f_μ^{**} formulations we use a bisection approach for finding the best value of μ. We try to find μ such that 99.99% of the matrix is described by the first r_0 singular values. Given an upper bound μ_u and a lower bound μ_l we test their average $\mu = (\mu_u + \mu_l)/2$ and reduce the upper bound if the criteria is met and alternatively increase the lower bound if not. Note that there is no guarantee that this approach will work since the singular values may depend nonlinearly on μ. In Figure 7.9 we varied the noise level and measured the distance between the obtained solution and the ground truth data. Here we averaged over 100 problem instances for each noise level. In theory, there should be a μ such that f_μ^{**} gives the same result as $f_{r_0}^{**}$. However the performance of the f_μ^{**} formulation is worse

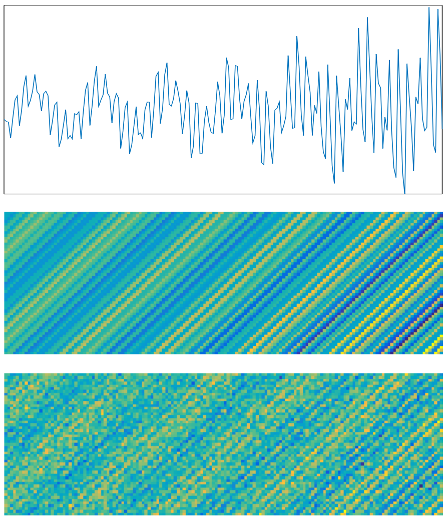
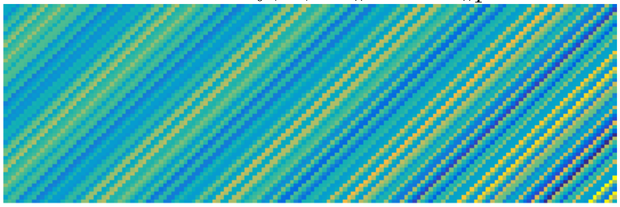


Figure 7.7: Data from one instance of the Hankel experiment. Top - Signal generated form (7.68). Middle - Ground truth Hankel matrix. Bottom - Matrix with noise.

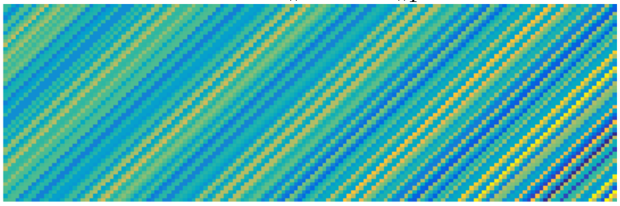
due to the difficulty of finding this μ. The nuclear norm formulation has the additional disadvantage that all singular values are shrunk in order to get the correct rank, further degrading the result. For comparison we also plotted the result obtained when only doing SVD of the matrix (ignoring the Hankel constraint) and when computing the mean over the anti-diagonals of the measurement matrix. While the mean operation gives a relatively good estimation it does not give a low rank solution. To illustrate this we plot the average sum of the leading r_0 singular values $\sum_{i=1}^{r_0} \sigma_i(H)$ normalized by $\|H\|_*$ versus the noise level in Figure 7.10.

$\min_{H\in\mathcal{H}} \mathcal{R}_{r_0}(H) + \|H-M\|_F^2$



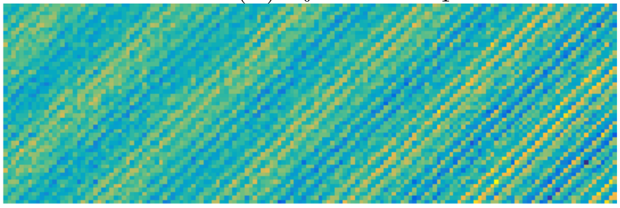
$\|H - H(f)\|_F^2 = 6.08 \quad \sum_{i=r_0+1}^{n} \sigma_i(H) = 3.69 \cdot 10^{-5}$

$\min_{H\in\mathcal{H}} \|H-M\|_F^2$



$\|H - H(f)\|_F^2 = 23.0 \quad \sum_{i=r_0+1}^{n} \sigma_i(H) = 133$

$\min_{\text{rank}(H)=r_0} \|H-M\|_F^2$



$\|H - H(f)\|_F^2 = 67.0 \quad \sum_{i=r_0+1}^{n} \sigma_i(H) = 0$

Figure 7.8: Results for the data in Figure 7.7 using our relaxation $f_{r_0}^{**}$ (top), projection onto closest Hankel matrix (middle), SVD without Hankel constraint (bottom).

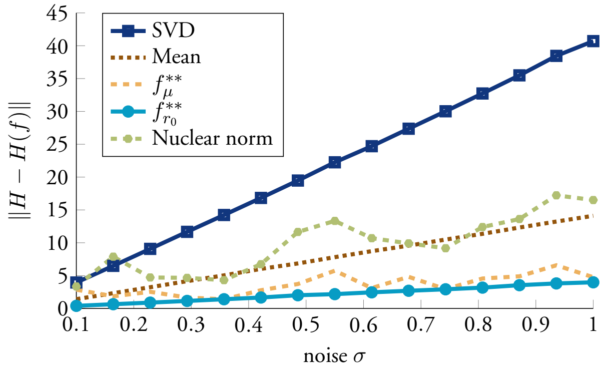


Figure 7.9: Average distance to ground truth $\|H - H(f)\|$ vs. noise level σ for the Hankel experiment.

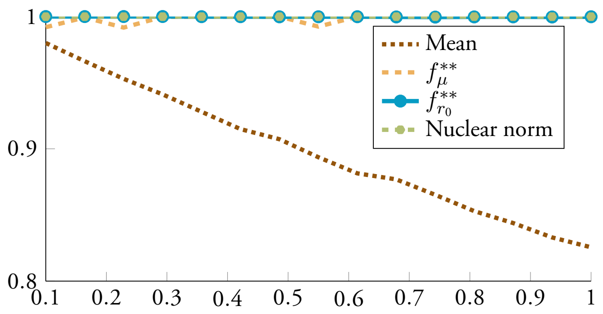


Figure 7.10: Average size of leading r_0 singular values $\sum_{i=1}^{r_0} \sigma_i(H)/\|H\|_*$ vs. noise level σ for the Hankel experiment.

7.4.2 Smooth Linear Shape Basis Model

In this experiment we show an example of how the proposed convex rank penalty can be integrated into other convex frameworks. In [72] an alternating framework for estimating the cameras and non-rigid 3D-shape is presented. To ensure a smooth 3D shape they minimize an energy penalizing the total variation of the 3D point coordinates in each frame. Here we consider a simplified version of their shape update by minimizing

$$f_N(S) = \|S - S_0\|_F^2 + \mu \|P(S)\|_* + \tau \mathrm{TV}(S), \tag{7.69}$$

where $S \in \mathbb{R}^{3F \times N}$ are the 3D coordinates in each frame stacked on top of each other and $P : \mathbb{R}^{3F \times N} \to \mathbb{R}^{F \times 3N}$ is simply the linear map which stacks the coordinates in each frame. The function TV is the total variation norm, see [72] for more information. We compare this to minimizing

$$f_{\mathcal{R}}(S) = \|S - S_0\|_F^2 + \mathcal{R}_\mu(P(S)) + \tau \mathrm{TV}(S). \tag{7.70}$$

The two methods were evaluated on the synthetic face data used in Garg et al. [72]. The data consists of 10 synthetic face basis shapes which we used to generate 50 faces by forming random convex combinations. To the generated faces we then added Gaussian noise. Since the faces are all linear combinations of the 10 original base faces the rank of $P(S)$ should be 10.

For different noise levels we solved both (7.69) and (7.70) using ADMM and then measured the error to the ground truth after projecting to the correct rank. For each instance the parameter μ was chosen such that the correct rank was attained for $\tau = 0$ (i.e. without TV regularization). The error to the ground truth shape is shown in Figure 7.11. In Figure 7.12 one of the faces are shown. While our reconstruction error is lower the result is visually similar to the nuclear norm and therefore we only plot our reconstruction.

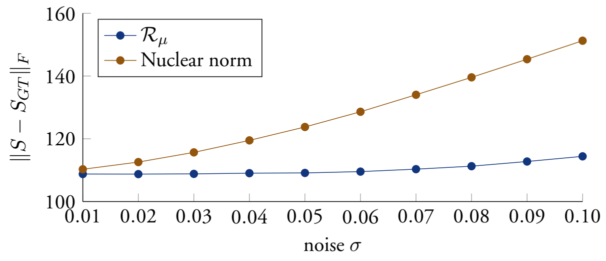


Figure 7.11: The average reconstruction error $\|S - S_{GT}\|_F$ over 50 instances for different noise levels.

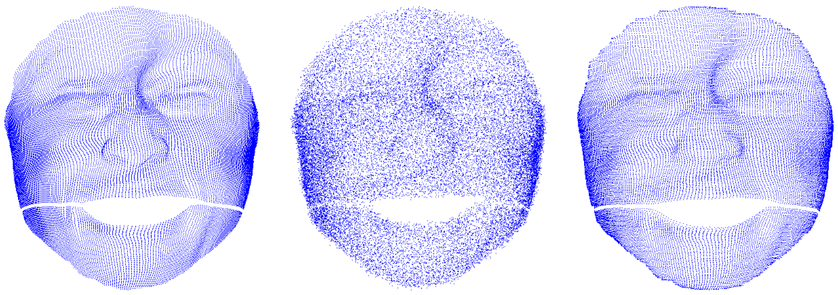


Figure 7.12: *Left* Ground truth *Middle:* Added noise. ($\sigma = 0.1$) *Right:* Reconstruction by minimizing (7.70).

7.5 Applications with Multiple Matrices

In previous sections we considered problems where we are searching for a single matrix of low rank. The flexibility of the framework makes it possible to search for several low rank matrices at once. This gives us a way of addressing problems with missing data by considering completely observed sub-matrices.

The missing data problem can be formulated as

$$\min_X \|W \odot (X - M)\|_F^2 \quad \text{s.t.} \quad \operatorname{rank}(X) = r_0, \tag{7.71}$$

where $W_{ij} \in \{0, 1\}$ indicates if M_{ij} was observed. While our convex envelope is not directly applicable to the energy we will in this section show how to perform low rank approximations for some structured data patterns. The assumption we make is that the observed entries contain a set of sufficiently large overlapping submatrices, such that each column and row of the matrix has at least one intersection with one of the submatrices. This is common in vision where structured data patterns occur naturally. In Figure 7.13 we show the observed entries from the Oxford dinosaur sequence and some overlapping blocks covering each row and column. Note that in this example the blocks are formed from contiguous sets of rows and columns. This is not necessary and other block configurations with non-contiguous submatrices can cover a much larger percentage of the data.

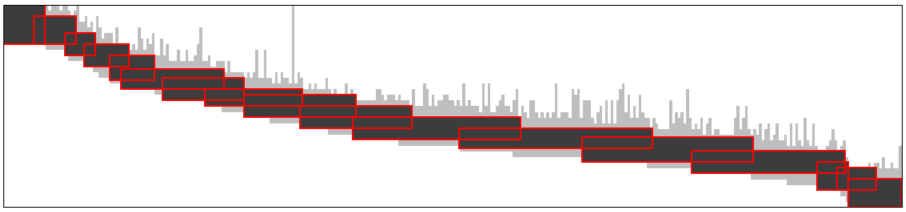


Figure 7.13: Missing data pattern for a subset of the Oxford dinosaur sequence. The gray elements correspond to observed data. The blocks are highlighted in red. Note that the blocks contain no missing elements.

To find a low rank approximation in the presence of missing data we minimize our convex relaxation on each complete submatrix simultaneously. Let $\mathcal{P}_k : \mathbb{R}^{m\times n} \to \mathbb{R}^{m_k\times n_k}$ be the operator which extracts the k:th submatrix. The energy we minimize is then

$$E(X) = \sum_k \mathcal{R}_g(\mathcal{P}_k(X)) + \|\mathcal{P}_k(X) - \mathcal{P}_k(M)\|_F^2\,, \tag{7.72}$$

where $M \in \mathbb{R}^{m\times n}$ is the matrix containing the data. Note that we assumed that the blocks were chosen such that $\mathcal{P}_k(M)$ contains no missing data. The minimization problem can be reformulated as

$$\begin{aligned} \min_{X,\, X_k} \quad & \sum_k \mathcal{R}_g(X_k) + \|X_k - \mathcal{P}_k(M)\|_F^2\,, \\ \text{s.t.} \quad & X_k = \mathcal{P}_k(X). \end{aligned} \tag{7.73}$$

Since the constraints are linear this is a convex problem which can be solved efficiently using ADMM. Note that the solution obtained is only defined on blocks. In the Appendix 7.A.3 we show that the solution can be extended to the full matrix without increasing the rank. Under mild assumptions this extension will be unique.

To select the blocks we use a simple heuristic. For every kth row we consider l consecutive rows. We form a block from the columns which are observed in all l rows. The block is then extended by adding all rows which observe all of the selected columns. The parameters k and l are chosen manually for each dataset.

7.5.1 Evaluation of the Convex Relaxation

Next we empirically evaluate the relaxation in the framework for missing data from Section 7.5. For evaluation the non-convex energy we consider is

$$E(X) = \sum_{i=1}^{K} \mu \operatorname{rank}(\mathcal{P}_i(X)) + \|\mathcal{P}_i(X) - \mathcal{P}_i(M)\|_F^2, \tag{7.74}$$

for a fix parameter μ.[2] Using the proposed convex envelope a convex relaxation is

$$E_{\mathcal{R}}(X) = \sum_{i=1}^{K} \mathcal{R}_\mu(\mathcal{P}_i(X)) + \|\mathcal{P}_i(X) - \mathcal{P}_i(M)\|_F^2. \tag{7.75}$$

Since each term in $E_{\mathcal{R}}$ is a lower bound for the corresponding term in E we must have $E_{\mathcal{R}}(X) \leq E(X)$ for all X. Let $X_{\mathcal{R}}^\star = \arg\min_X E_{\mathcal{R}}(X)$ then if

$$E_{\mathcal{R}}(X_{\mathcal{R}}^\star) = E(X_{\mathcal{R}}^\star) \tag{7.76}$$

it follows that $X^\star$ must be a global minimizer to the non-convex energy E. Note if $K = 1$ then $E_{\mathcal{R}}$ is simply the convex envelope of E and we necessarily have

$$\min_X E(X) = \min_X E_{\mathcal{R}}(X). \tag{7.77}$$

We evaluate on synthetic instances with $K = 10$ and varying levels of noise added to M. Using ADMM we minimize (7.75) and then compare $E_{\mathcal{R}}(X_{\mathcal{R}}^\star)$ and $E(X_{\mathcal{R}}^\star)$.

We also include the results found using the nuclear norm as a surrogate for the rank, i.e.

$$E_N(X) = \sum_{i=1}^{K} \mu \|\mathcal{P}_i(X)\|_* + \|\mathcal{P}_i(X) - \mathcal{P}_i(M)\|_F^2. \tag{7.78}$$

Note that this is only a lower bound on E on the set $\{X \mid \sigma_1(X) \leq 1\}$. In Figure 7.14 the function values are plotted against the added noise level.

[2]Note that we choose $g(k) = \mu k$ here to allow for a comparison with the nuclear norm. In general when we solve the missing data problem we use $g(k) = \mathbb{I}(k \leq r_0)$.

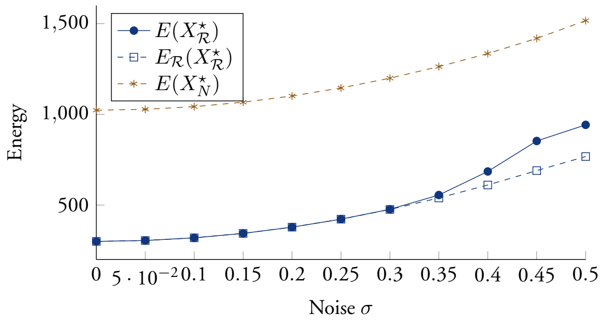


Figure 7.14: Evaluation of the convex relaxation. The relaxation $E_\mathcal{R}$ is a lower bound for E. If $E_\mathcal{R}(X^\star_\mathcal{R}) = E(X^\star_\mathcal{R})$ a global optimum to E has been found.

7.5.2 Comparison to Non-Convex Methods

Next we compare the performance of the proposed method to three non-convex methods; OptSpace [107], Truncated Nuclear Norm Regularization [92] and Damped Wiberg-L2 [173]. In contrast to the proposed approach these methods are local in nature and their result is dependent on initialization.

In the previous experiment we used the regularization term $\mathcal{R}_\mu$ because of its simplicity. Here we instead use $\mathcal{R}_{r_0}$. The reason is that we are searching for many matrices of the same rank. Doing this with $\mathcal{R}_\mu$ would require iteration over the μ (one parameter for each block) which quickly becomes tedious.

The measurement matrix was chosen as $M = UV^T + N$ where $U, V \in \mathbb{R}^{100\times5}, N \in \mathbb{R}^{100\times100}$ and $U_{ij}, V_{ij} \sim \mathcal{N}(0,1)$ and $N_{ij} \sim \mathcal{N}(0,\sigma)$. If σ is small then M will be approximately rank 5. The observation matrix W consisted of overlapping blocks along the diagonal and had 72% missing data. In Figure 7.15 we show the average of $||W \odot (X - M)||_F$ over 100 instances. Note that while on most instances the non-convex methods converged to the optimal solution they sometimes find local-minima which raise their average error. The performance of the proposed method and Damped Wiberg-L2 is very similar on this data. To illustrate the benefit of the proposed method we also performed an experiment on another family of instances generated by replacing the fifth column of V by $10^3\mathbb{1}$. This essentially makes M have one very dominant singular value which is common in applications. The averaged result for these instances can be seen in the bottom graph in Figure 7.15.

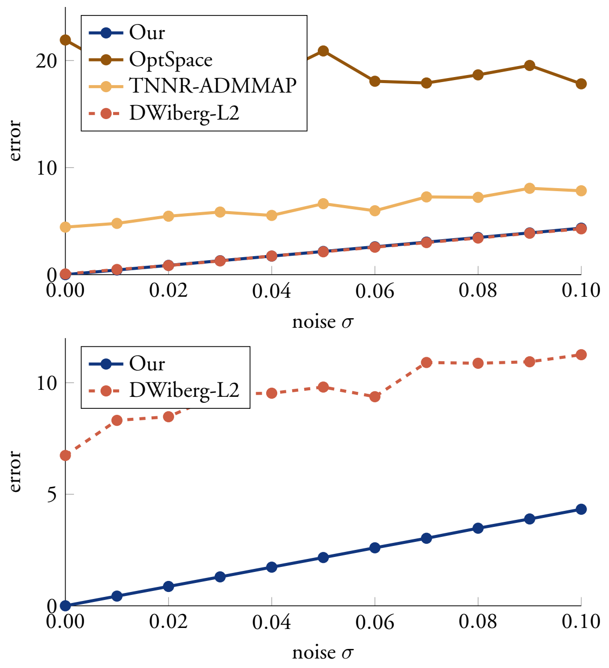


Figure 7.15: Comparison with non-convex methods. *Top:* Initial experiment. (Note that the errors for our approach and DWiberg-L2 are very similar). *Bottom:* Experiment with adjusted row-mean.

7.5.3 Linear Shape Basis with Missing Data

A common assumption when dealing with non-rigid deformation is that the points move in some low-dimensional subspace. The idea is that the points in each frame can be written as a linear combination of a shared shape basis. This shape basis assumption leads to a low rank prior on the matrix containing the points from all frames.

We performed an experiment where we used a standard KLT tracker on two video sequences. Due to tracking failure most of the points could not be tracked throughout the entire sequence. Figure 7.16 shows the pattern for the observed data for the sequences and Table 7.1 shows further information about the sequences.

Using the block approach we found a low rank approximation of the partially

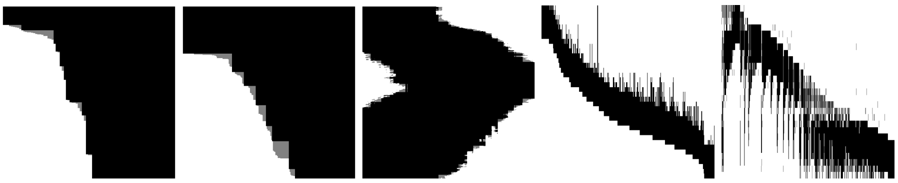


Figure 7.16: Missing data patterns for the experiments with real data. The observed entries are shown as black if they are contained in a block and gray otherwise. From left to right : *Book*, *Hand*, *Banner*, *Oxford*, *Cathedral*.

filled in matrix with the tracked point coordinates from each frame. The results can be seen in Figure 7.17 and Figure 7.18. Here blue points have been correctly detected in the image whereas yellow point positions correspond to missing entries that have been filled in using the estimated model. For the book sequence we used $f_{r_0}^{**}$ with $r_0 = 3$ and for the hand sequence $r_0 = 5$.

We also performed a similar experiment where we instead filmed a piece of cloth with a Kinect camera. During filming a wave like motion was created in the cloth by moving one of the corners. We tracked points on the cloth while panning the camera up and down such that cloth was only partially visible at any time. See Figure 7.16 and Table 7.1. Using our method we found a low rank approximation of the partially observed matrix containing the 3D points. The results can be seen in Figure 7.19 and Figure 7.20. Here we used $r_0 = 9$. In Figure 7.20 we also compare our results obtained with the standard nuclear norm formulation

$$\min_X \mu\|X\|_* + \|W \odot (X - M)\|_F^2. \tag{7.79}$$

Here μ was selected so that the rank of X was 9. The nuclear norm bias towards small singular values manifests itself by stretching of the solution towards the origin when there are large areas of missing data.

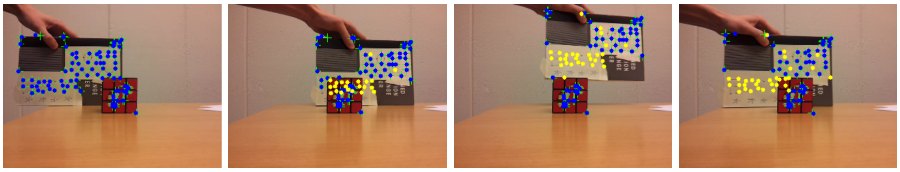


Figure 7.17: The reconstructed points for four frames from the *book* sequence from [136]. The green crosses show the observed data. The reconstruction of the observed and unobserved entries are indicated by the blue and yellow dots respectively.

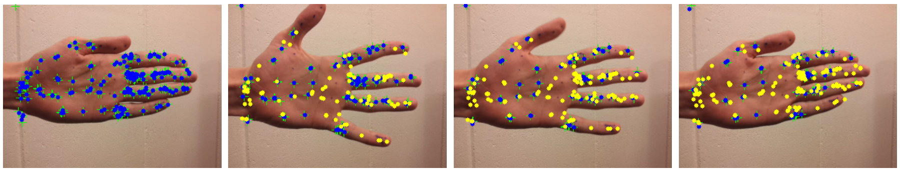


Figure 7.18: The reconstructed points for four frames from the *hand* sequence from [136]. The green crosses show the observed data. The reconstruction of the observed and unobserved entries are indicated by the blue and yellow dots respectively.

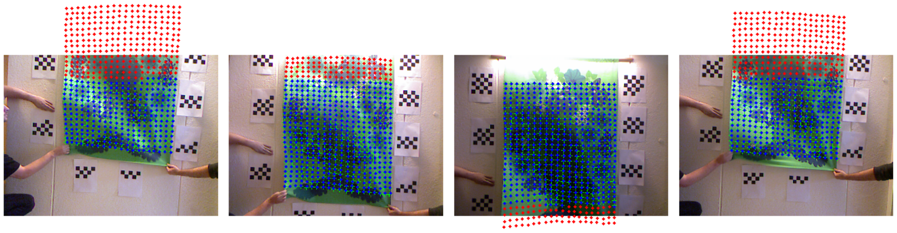


Figure 7.19: The reprojection for four frames from the *banner* sequence from [136]. The green crosses show the observed entries. The reconstruction of the observed and unobserved entries are indicated by the blue and red dots respectively.

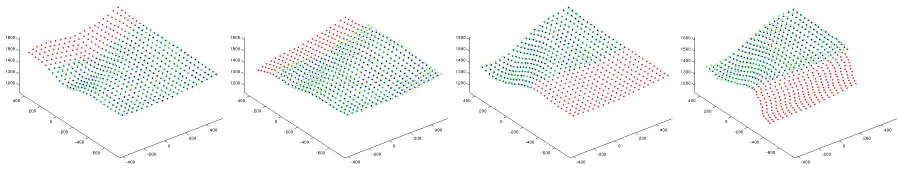


Figure 7.20: *From left to right:* Our solution (frame 329), nuclear norm solution (frame 329), our solution (frame 650), nuclear norm solution (frame 650).

7.5.4 Affine Structure-from-Motion

In affine Structure-from-Motion the assumption is that the image points are formed as

$$M = PX + T\mathbb{1}^T, \tag{7.80}$$

where $P \in \mathbb{R}^{2F\times 3}$ are the cameras, $X \in \mathbb{R}^{3\times N}$ the structure and $T \in \mathbb{R}^{2F}$ the translation. This implies that $\text{rank}(M) \leq 4$ and that $\mathbb{1}^T$ lie in the row space of the matrix.

If all data is observed the cameras and structure can be recovered using standard factorization approaches [211]. But due to tracking failures it is rare to observe each point in each frame. Using our block approach to missing data we can reconstruct the full measurement matrix M. To handle the constraint on the row space we add an additional row to each block in (7.73) which we through linear constraint enforce to be constant. This will ensure that $\mathbb{1}^T$ lies in the row space of each block. The modified optimization problem then becomes

$$\min_{Z,Z_k} \sum_k \mathcal{R}_{r_0}(Z_k) + \left\| Z_k - \begin{bmatrix} \mathcal{P}_k(M) \\ \mathbb{1}^T \end{bmatrix} \right\|_F^2 \tag{7.81}$$

$$\text{s.t. } Z_k = \begin{bmatrix} \mathcal{P}_k(Z) \\ \mathbb{1}^T \end{bmatrix}. \tag{7.82}$$

The recovered Z will then approximate M on the observed data and can be factorized into $Z = PX + T\mathbb{1}^T$.

Figure 7.21 shows the results for the well-known Oxford sequence compared to simply finding a rank 4 approximation as in Section 7.5. We also included a solution found by minimizing the nuclear norm formulation (7.79). The same tendency to stretch trajectories towards the origin is clearly visible.

We also include the result from a short image sequence of Lund Cathedral. To minimize the perspective effects of the camera we selected a subset of 2480 points where the projective depth is approximately the same in all images. Figure 7.22 shows the recovered reprojections in three of the frames.

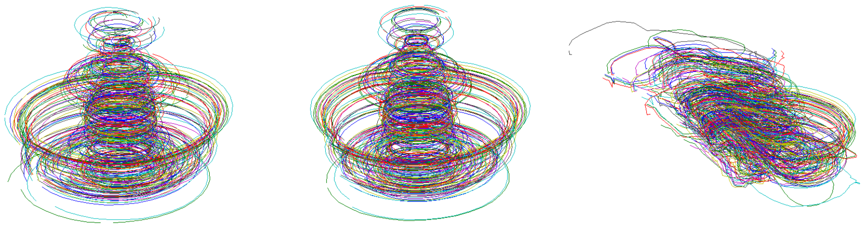


Figure 7.21: Results on the Oxford dinosaur sequence. *Left:* Rank 4. *Middle:* Rank 4 with a constant vector forced into the row space. *Right:* Solution found using the nuclear norm approach.

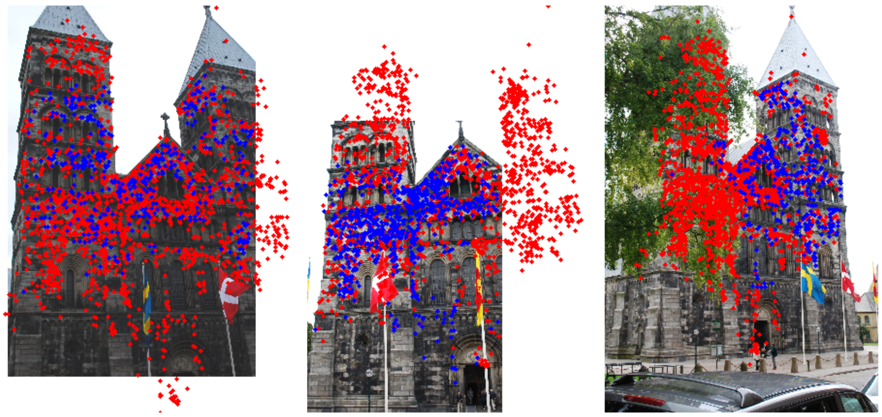


Figure 7.22: Three frames from the Lund cathedral sequence. Red points correspond to tracks which were not observed in the frame.

	Size	Missing	Blocks	Coverage
Hand	742×203	32.4%	9	96.8%
Book	1336×85	35.4%	16	98.8%
Banner	2052×560	26.9%	31	97.9%
Oxford	72×319	76.9%	32	100%
Cathedral	54×2420	68.8%	44	89.5%

Table 7.1: Statistics for the image sequences used in the experiments.

7.6 Conclusions

In this chapter we have proposed a convex method for low-rank approximation of matrices with additional convex constraints. The approach is based on a new convex relaxation - the strongest one possible - of the rank function and a least squares data term. Unlike the nuclear norm, it is able to avoid penalizing large singular values. Our experiments clearly show the benefits of being able to do so in a convex framework. It should be noted that the presented results are the outputs of our approach without refinement. In cases where the relaxation is not tight, the solution can be used as a starting point for local optimization to obtain even better results.

For the missing data problem the proposed method only has to compute the SVD of small sub-matrices, therefore it has potential to tackle large-scale problems. Furthermore, the ADMM approach allows to perform computations in a parallel and distributed manner.

A limitation of the formulation is that in its current form it is sensitive to outliers. The issue has received a lot of attention lately, for example, using the arguably more robust ℓ_1-norm [58, 203, 233] and it is something that we intend to address in the near future.

7.A Appendix

7.A.1 The Sequence of Unconstrained Minimizers

In this section we prove that with the definitions of p and q as in Section 7.3.1 the sequence of unconstrained minimizers defined by equation (7.49) will have the shape illustrated in Figure 7.5.

Lemma 7.3. *If p and q are selected such that the sequence $\{s_i\}$ defined by (7.49) is non-increasing for $i \leq p$ and $q \leq i$, non-decreasing for $p \leq i \leq q$ and $s_p < s_q$ then*

$$s_i = \begin{cases} \max(\sigma_i(Y), s_p), & i \leq p \\ \min(\max(\sqrt{g_i}, s_p), s_q) & p \leq i \leq q \\ \min((\rho+1)\sigma_i(Y), s_q), & i \geq q \end{cases}. \tag{7.83}$$

Proof. We first note that (7.49) can equivalently be written

$$s_i = \min\Big(\max(\sqrt{g_i}, \sigma_i(Y)), (\rho+1)\sigma_i(Y)\Big). \tag{7.84}$$

We begin with the case $i \leq p$. Since

$$s_p < s_q \leq (\rho+1)\sigma_q(Y) \leq (\rho+1)\sigma_p(Y), \tag{7.85}$$

we have by (7.84) that $s_p = \max(\sqrt{g_p}, \sigma_p(Y)) < (\rho+1)\sigma_i(Y)$ and therefore $\sqrt{g_p} \leq (\rho+1)\sigma_p(Y)$. Furthermore, since g_i is non-decreasing and $\sigma_i(Y)$ is non-increasing $\sqrt{g_i} \leq (\rho+1)\sigma_i(Y)$ for all $i \leq p$. For $i < p$ we now get that if $s_i > s_p$ then $s_i > \sqrt{g_p} \geq \sqrt{g_i}$ and therefore by (7.84) $s_i = \sigma_i(Y)$.

For $i \geq q$ we similarly have

$$s_q > s_p \geq \sigma_p(Y) \geq \sigma_q(Y), \tag{7.86}$$

which together with (7.84) gives that $s_q = \min(\sqrt{g_p}, (\rho+1)\sigma_q(Y))$ and $\sqrt{g_q} \geq \sigma_q(Y)$. Moreover, since g_i is non-decreasing and $\sigma_i(Y)$ is non-increasing $\sqrt{g_i} \geq \sigma_i(Y)$ for all $i \geq q$. For $i > q$ we now get that if $s_i < s_q$ then $s_i < \sqrt{g_q} \leq \sqrt{g_i}$ and therefore by (7.84) $s_i = (\rho+1)\sigma_i(Y)$.

For the final case $p \leq i \leq q$ we note that $s_p \leq s_i \leq s_q$ since s_i is non-decreasing between p and q. If $s_i > s_p$ then

$$s_i > s_p \geq \sigma_p(Y) \geq \sigma_i(Y). \tag{7.87}$$

If $s_i < s_q$ then

$$s_i < s_q \leq (\rho+1)\sigma_q(Y) \leq (\rho+1)\sigma_i(Y). \tag{7.88}$$

Therefore if $s_p < s_i < s_q$ then $s_i = \sqrt{g_i}$. $\square$

7.A.2 Properties of Feasible Minimizers

In this section we give a result that enables us to efficiently search for optimal sequences of singular values. The key observation is that for concave costs the optimum is either in a stationary point (determined by minimizing each singular value separately) or constrained by one of its neighboring singular values. Using this information it is possible to single out a 1-parameter family of singular value configurations guaranteed to contain the optimal one.

We let s_i be a sequence of non-negative numbers. For $i \leq p$ we require that the sequence is non-increasing, for $p \leq i \leq q$ non-decreasing and $q \leq i$ non-increasing. Note that due to the definition s_q and s_p will be local extreme points of the sequence ($s_{p-1} \geq s_p \leq s_{p+1}$ and $s_{q-1} \leq s_q \geq s_{q+1}$).

Lemma 7.4. *Let $\{s_i\}$ be the unconstrained maximizers of $f_i(s)$, where f_i are concave (with unique unconstrained maximizers). Then the maximizer of $g(\boldsymbol{\sigma}) = \sum_i f_i(\sigma_i)$, such that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$, fulfills*

$$\sigma_i = \max(s_i, \sigma_{i+1}), \quad 1 \leq i \leq p \tag{7.89}$$

$$\sigma_i = \sigma_{i+1}, \quad p \leq i \leq q-1 \tag{7.90}$$

$$\sigma_i = \min(s_i, \sigma_{i-1}), \quad i \geq q \tag{7.91}$$

Proof. Since each f_i is concave and $\sigma_{i+1} \leq \sigma_{i-1}$ the optimization over σ_i can be limited to three choices

$$\sigma_i = \begin{cases} s_i & \text{if} \quad \sigma_{i+1} \leq s_i \leq \sigma_{i-1} \\ \sigma_{i-1} & \text{if} \quad \sigma_{i-1} < s_i \\ \sigma_{i+1} & \text{if} \quad s_i < \sigma_{i+1} \end{cases} . \tag{7.92}$$

Using induction we first prove the recursion

$$\sigma_i = \max(s_i, \sigma_{i+1}) \quad \text{for } i \leq p. \tag{7.93}$$

For $i = 1$ we see from (7.92) that s_1 is the optimal choice if $s_1 > \sigma_2$ otherwise σ_2 is optimal. Therefore $\sigma_1 = \max(s_1, \sigma_2)$. Next assume that $\sigma_{i-1} = \max(s_{i-1}, \sigma_i)$ for some $i \leq p$. Then

$$\sigma_{i-1} \geq s_{i-1} \geq s_i, \tag{7.94}$$

therefore we can ignore the second case in (7.92), which proves the recursion (7.93).

Next we show that the sequence $\{\sigma_i\}$ is constant in $p \leq i \leq q$. We consider σ_i for some $p \leq i \leq q-1$. If $\sigma_i > s_i$ it must have been bounded from below in (7.92), i.e. $\sigma_i = \sigma_{i+1}$. If instead $\sigma_i \leq s_i$ we have $\sigma_{i+1} \leq \sigma_i \leq s_i \leq s_{i+1}$. Then similarly σ_{i+1} is bounded from above in (7.92) which implies $\sigma_{i+1} = \sigma_i$.

For the final part we consider $i \geq q$ and show that

$$\sigma_i = \min(s_i, \sigma_{i-1}) \quad \text{for} \quad i \geq q. \tag{7.95}$$

It is clear from (7.92) that this holds for $i = n$. We continue using induction by assuming $\sigma_{i+1} = \min(s_{i+1}, \sigma_i)$ holds. Then

$$\sigma_{i+1} \leq s_{i+1} \leq s_i, \tag{7.96}$$

since s_i are non-increasing for $i \geq q$. This means that for σ_i we can ignore the third case in (7.92). Thus it follows that $\sigma_i = \min(s_i, \sigma_{i-1})$. So (7.95) holds for all $i \geq q$. □

Theorem 7.5. *Then the maximizer* $\boldsymbol{\sigma}$ *can be written*

$$\sigma_i = \max(s_i, s), \quad 1 \leq i \leq p \tag{7.97}$$
$$\sigma_i = s, \quad p \leq i \leq q-1 \tag{7.98}$$
$$\sigma_i = \min(s_i, s), \quad i \geq q, \tag{7.99}$$

where s *fulfills* $s_p \leq s \leq s_q$.

Proof. We first consider $i \leq p$. Assume $\sigma_i \neq s_i$ for some $i < p$. From (7.93) it follows that

$$\sigma_i = \sigma_{i+1} > s_i. \tag{7.100}$$

But s_i is non-increasing for $i \leq p$ which implies that $\sigma_{i+1} > s_{i+1}$. By repeating the argument it follows that

$$\sigma_i = \sigma_{i+1} = \sigma_{i+2} = ... = \sigma_p. \tag{7.101}$$

We let $s = \sigma_p$ and note that due to (7.93) $s \geq s_p$. By Lemma 7.4 we also have

$$\sigma_p = \sigma_{p+1} = \sigma_{i+2} = ... = \sigma_q. \tag{7.102}$$

Therefore $s = \sigma_q$ and by (7.95) we get $s \leq s_q$.

Now assume that for some $i \geq q$ we have $\sigma_i(Z) \neq s_i$. By (7.95) we must have that

$$\sigma_i(Z) = \sigma_{i-1}(Z) < s_i \leq s_{i-1}. \tag{7.103}$$

By repeating the argument we get

$$\sigma_i(Z) = \sigma_{i-1}(Z) = \sigma_{i-2}(Z) = ... = \sigma_q(Z). \tag{7.104}$$

and the result follows. □

7.A.3 Extension Outside the Blocks

In this section we show how to extend a partial low rank solution computed on overlapping blocks of the matrix to a complete solution. The approach hinges on the following result.

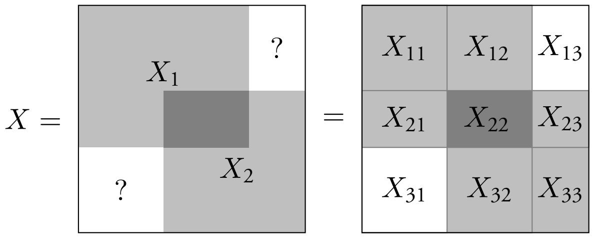


Figure 7.23: Two overlapping blocks X_1 and X_2. The goal of the extension is to find the unknown X_{13} and X_{31} such that the rank is not increased, i.e. $\mathrm{rank}(X) = \max(\mathrm{rank}(X_1), \mathrm{rank}(X_2))$.

Lemma 7.6. *Let X_1 and X_2 be two overlapping blocks such that they agree on the overlap X_{22} (in the notation from Figure 7.23). If the overlap satisfies*

$$\mathrm{rank}(X_{22}) = \min(\mathrm{rank}(X_1), \mathrm{rank}(X_2)) \tag{7.105}$$

then there exist X_{13} and X_{31} such that

$$\mathrm{rank}(X) = \max(\mathrm{rank}(X_1),\ \mathrm{rank}(X_2)), \tag{7.106}$$

Furthermore, if $\mathrm{rank}(X_1) = \mathrm{rank}(X_2)$ *the extension is unique.*

Proof. Without loss of generality assume that $\mathrm{rank}(X_{22}) = \mathrm{rank}(X_2) \leq \mathrm{rank}(X_1)$. Then the column space of X_2 must be spanned by $\begin{bmatrix} X_{22} \\ X_{32} \end{bmatrix}$ and similarly the row space by $\begin{bmatrix} X_{22} & X_{23} \end{bmatrix}$. There exist coefficient matrices C_1 and C_2 such that

$$\begin{bmatrix} X_{22} \\ X_{32} \end{bmatrix} C_1 = \begin{bmatrix} X_{23} \\ X_{33} \end{bmatrix} \quad \text{and} \quad C_2 \begin{bmatrix} X_{22} & X_{23} \end{bmatrix} = \begin{bmatrix} X_{32} & X_{33} \end{bmatrix}. \tag{7.107}$$

For the extension we can then take

$$X_{13} := X_{12}C_1 \quad \text{and} \quad X_{31} := C_2 X_{21}. \tag{7.108}$$

To see that this does not increase the rank we note that

$$\begin{bmatrix} X_{12} \\ X_{22} \\ X_{32} \end{bmatrix} C_1 = \begin{bmatrix} X_{13} \\ X_{23} \\ X_{33} \end{bmatrix}, \tag{7.109}$$

and similarly for the rows. This means that the number of linearly independent columns and rows have not increased and the rank must be preserved.

Now assume that $\operatorname{rank}(X_1) = \operatorname{rank}(X_2)$. We prove uniqueness by means of contradiction. Assume there exist two different extensions

$$X_{13} = X_{12}C_1 \quad \text{and} \quad \tilde{X}_{13} = X_{12}\tilde{C}_1. \tag{7.110}$$

To be extensions which preserve the rank C_1 and $\tilde{C}_1$ must satisfy

$$\begin{bmatrix} X_{23} \\ X_{33} \end{bmatrix} = \begin{bmatrix} X_{22} \\ X_{32} \end{bmatrix} C_1 = \begin{bmatrix} X_{22} \\ X_{32} \end{bmatrix} \tilde{C}_1. \tag{7.111}$$

Which implies that $C_1 - \tilde{C}_1$ lies in the nullspace of $\begin{bmatrix} X_{22}^T & X_{32}^T \end{bmatrix}^T$. But by assumption we have

$$\operatorname{rank}(\begin{bmatrix} X_{22} \\ X_{32} \end{bmatrix}) = \operatorname{rank}(\begin{bmatrix} X_{13} \\ X_{22} \\ X_{32} \end{bmatrix}). \tag{7.112}$$

This implies that $C_1 - \tilde{C}_1$ must also lie in the nullspace of X_{12}, i.e.

$$X_{12}(C_1 - \tilde{C}_1) = 0 \Leftrightarrow X_{13} = \tilde{X}_{13}, \tag{7.113}$$

which is a contradiction. $\square$

The previous lemma showed that each pair of overlapping blocks has an extension which preserves the rank. If we assume that the blocks are chosen to be connected (in a graph sense) we can iterate this construction to find an extension to the whole matrix.

Chapter 8

Compact Matrix Factorization

Traditional low rank matrix factorization methods approximate high dimensional data by fitting a low dimensional subspace. This imposes constraints on the matrix elements which allow for estimation of missing entries. A lower rank provides stronger constraints and makes estimation of the missing entries less ambiguous at the cost of measurement fit.

In this chapter we propose a new factorization model that further constrains the matrix entries. Our approach can be seen as a unification of traditional low-rank matrix factorization and the more recent union-of-subspace approach. It adaptively finds clusters that can be modelled with low dimensional local subspaces and simultaneously uses a global rank constraint to capture the interactions between clusters. For inference we use an energy that penalizes a trade-off between data fit and degrees-of-freedom of the resulting factorization. We show qualitatively and quantitatively that regularizing both local and global dynamics yields significantly improved missing data estimation.

This chapter is based on the paper [141].

8.1 Introduction

Matrix factorization is a an important tool in many engineering applications. The assumption that data belongs to a low dimensional subspace has been proven useful in numerous computer vision applications, e.g. non-rigid and articulated structure from motion [20, 5, 228], photometric stereo [17], optical flow [71], face recognition [226, 191] and texture reparation [149].

Given an $m \times n$ matrix M containing m-dimensional measurements a low dimensional approximation $X \approx M$, where $\text{rank}(X) = r_0$, can be found using singular value decomposition (SVD). Since $\text{rank}(X) = r_0$ the matrix X can be written

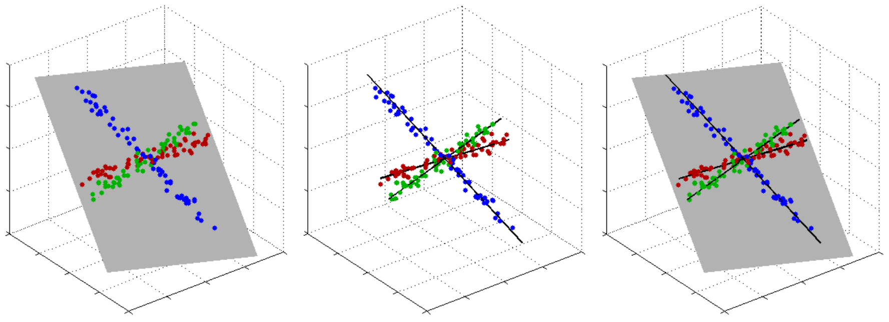
$$X = BC^T, \tag{8.1}$$

where B is $m \times r_0$ and C is $n \times r_0$. The columns of B constitute a basis for the column-space of X. The matrix C contains coefficients used to form the columns of X from the basis. Alternatively one may think of the rows of X as n-dimensional data, C as a basis for the row-space and B as the coefficients. In both cases the data is approximated by an r_0-dimensional subspace, as illustrated in Figure 8.1(a).

In a sense the factorization BC^T can be seen as a compressed representation of M where the mn elements have been reduced to $(m + n - r_0)r_0$ degrees of freedom (see Section 8.3.1). It is therefore possible to compute the factorization even if only a subset of the elements of M are known, by solving $W \odot M \approx W \odot (BC^T)$. Here $\odot$ denotes element-wise multiplication and the matrix W has elements $w_{ij} = 1$ for known data and 0 for missing data. Note that once computed, BC^T contains estimates of both known and missing data. In this way it is theoretically possible to "predict" at most $mn - (m + n - r_0)r_0$ missing elements.

In the presence of missing data the low rank approximation problem becomes very difficult, some variations of the problem even NP-hard [73]. However, due to its practical importance a lot of research have been directed at finding good algorithms. In [10] it is shown that under the spectral norm a closed form solution exist if the missing data forms a so called Young pattern. A recent trend has been to replace the rank function with convex surrogates, such as the nuclear norm [180, 37, 175]. However, in many applications such as structure from motion, where missing entries are highly correlated, this approach has been shown to perform poorly (see e.g. the experiments in Chapter 7).

If the rank of the sought matrix is known, the bilinear parametrization (8.1) can be locally optimized. Buchanan and Fitzgibbon [23] showed that alternating methods often exhibit very slow convergence and proposed a damped Gauss-Newton update. In [172] it was illustrated that the Wiberg elimination strategy [225] is very robust to local minima. For a recent comparison of different approaches to minimize the bilinear formulation see [93]. In [106] the ℓ_1 norm is used to address outliers. The proposed alternating approach is shown to converge



(a) (b) (c)

Figure 8.1: 3D illustration of subspace representations. (a) - A 2D subspace is fitted to all the data (global model). (b) - A union of independent 1D subspaces is fitted to clustered data (local models). (c) - Our unified approach. 1D subspaces are fitted to clustered data and restricted to lie in a 2D subspace. (For this data $m = 3$, $n = 100$, $r_0 = 2$ and $r_k = 1$, see Section 8.2 for definitions.)

slowly in [58]. Instead [58, 203] use generalizations of the Wiberg approach designed to handle the non-differentiable objective function while jointly updating the two factors.

Despite numerous recent developments in rank optimization missing data is still a problem plaguing vision algorithms. Dai et al. [47] argue that researchers have focused too much on optimization and ignored modeling issues. While the rank constraint provides a compact model representation it is limited by only measuring the overall complexity of the matrix even though individual sub-blocks may be less complex. Hence, there is no incentive to use fewer basis columns for sub-blocks than what the total rank admits. A relatively high overall model complexity is a particular problem when missing data needs to be estimated. As noted in [156, 74, 67] the availability of too many basis elements causes methods only optimizing a global rank constraint to over-fit giving very poor results.

A related model used in clustering is the union-of-subspace approach [235, 229]. Here data is clustered into similar groups that can be represented with independent low dimensional subspaces, see Figure 8.1(b). We refer to these as local subspaces since they are local to a particular cluster. In [152, 111] these are used to cluster frames into groups that allow simple deformation models. In principle these could also be used to address the missing data problem. In contrast to the global rank constraint, which constrains the whole matrix, each cluster has its own set of basis vectors and can only be constructed from these. This gives a data representation that is often (but not always, see Section 8.3.1) more compact. The overall idea of dividing the matrix into less complex parts and treating them separately is shared with the multi-body factorization methods [221, 46, 230] which typically perform clustering on the trajectories.

In this chapter we address the missing data problem by presenting a new compact factorization formulation. Our approach unifies the local and global subspace approaches leveraging the benefits of them both. Our method adaptively clusters the data and fits local subspaces, but also enforces a low rank on the entire data matrix. This ensures that any potential interactions between clusters are identified by the model which increases the prediction capability. For example, if clusters correspond to rigid parts of an object, similar to [183], our model can predict occluded parts if a motion dependency exists. In contrast the union-of-subspace approach lacks the ability to learn global scene dependence since subspaces are treated independently. Figure 8.1(c) illustrates our approach for a simple 3D example.

The contributions in this chapter are:

- We analyze the performance of global and local models with respect to different types of missing data.
- We present a new factorization that incorporates both a global rank constraint and local subspace constraints and show how this reduces model complexity.
- For computing the factorization we propose an energy-based model fitting framework that is able to perform joint clustering and adaptive model selection.
- We show in experiments that this model is well suited for recovering missing data in tasks involving multi-body and non-rigid image point trajectories.

8.2 A Dependent Subspace Model

In this section we present our model. We make two assumptions on the data matrix; that the entire scene is explained well by a low rank model, and that it can be partitioned into clusters that are explained by simpler rank models. Let X be an $m \times n$ matrix. The model can then (possibly after column permutations) be written as

$$\begin{gathered} X = \begin{bmatrix} X_1 & X_2 & \dots & X_K \end{bmatrix} \\ \text{rank}(X) = r_0, \quad \text{rank}(X_k) = r_k \end{gathered} \tag{8.2}$$

where each X_k is an $m \times n_k$ matrix that contains the data points of a cluster. It is clear that $r_0 \geq r_k$ and typically we try to have $r_0 \ll \sum_{k=1}^{K} r_k$ since we want to model the dependence between the clusters. Here we have divided the matrix columns into clusters. Note however that the same model can be applied to the rows by transposing.

Since X_k is of rank r_k it can be factorized into $X_k = B_k C_k^T$, where B_k is $m \times r_k$ and C_k is $n_k \times r_k$. The matrix B_k contains a basis for the subspace spanned by the columns of X_k. The full matrix X can thus be written

$$X = \begin{bmatrix} B_1 C_1^T & B_2 C_2^T & \dots & B_K C_K^T \end{bmatrix}. \tag{8.3}$$

Note that if the global rank constraint $\text{rank}(X) = r_0$ is ignored then $B_1, B_2, ..., B_K$ are assumed to be independent and this expression constitutes a union of subspace representation of X.

Now, assuming $r_0 < \sum_{k=1}^{K} r_k$ there is a dependence between the cluster subspaces. Since the columns of X are spanned by the columns of $\begin{bmatrix} B_1 & B_2 & \dots & B_K \end{bmatrix}$ this matrix must also be of rank r_0. Therefore we may factor it into

$$\begin{bmatrix} B_1 & B_2 & \dots & B_K \end{bmatrix} = B \begin{bmatrix} U_1 & U_2 & \dots U_K \end{bmatrix}, \tag{8.4}$$

where B is $m \times r_0$ and U_k is $r_0 \times r_k$. Here B is a basis of the column space of $\begin{bmatrix} B_1 & B_2 & \dots & B_K \end{bmatrix}$ and therefore also of X. Inserting into (8.3) gives our model

$$X = B \begin{bmatrix} U_1 C_1^T & U_2 C_2^T & \dots & U_K C_K^T \end{bmatrix}. \tag{8.5}$$

We can think of the $r_0 \times r_k$ matrices U_k as selecting a r_k-dimensional basis within the r_0-dimensional space spanned by the columns of B. While the union-of-subspace model (8.3) treat subspaces independently by allowing arbitrary selection of the bases $B_1, B_2, ..., B_k$ our model forces these to be selected in the shared global subspace spanned by B. Figure 8.2 shows an example of the three model factorizations when $r_0 = 5$ and $r_k = 3$ for $k = 1, 2, 3$.

In the above description of our model we have assumed that the subspaces are linear. Note however that it is easy to use affine subspaces by restricting the last row of C_k^T to be all ones. If $B_k = \begin{bmatrix} A & t \end{bmatrix}$ and $C_k^T = \begin{bmatrix} C^T \\ \mathbb{1}^T \end{bmatrix}$ then $B_k C_k^T = AC^T + t\mathbb{1}^T$, which is an affine function in C.

8.3 Benefits of Dependent Models

In this section we discuss the benefits of using both local and global subspace constraints. We compare three formulations: the *global model* (8.1), *local models* (8.3) and our *unified model* (8.5).

8.3.1 Degrees of Freedom

We first compute the degrees of freedom (DOF) of the three models. Note that it is clear that the unified model will have fewer DOF than both the local and the global models since (8.5) is a special case of both (8.1) and (8.3). Having an accurate model with few DOF makes matrix completion more well posed and reduces the space of feasible matrices.

Linear Subspace Models Under the global model the data matrix X can be factorized as in (8.1). The matrices B and C have mr_0 and nr_0 elements respectively. However due to the gauge freedom $X = BC^T = BGG^{-1}C^T$, where G is an unknown invertible $r_0 \times r_0$ matrix the DOF for the global model are

$$mr_0 + nr_0 - r_0^2. \tag{8.6}$$

For cluster k in (8.3) the matrices B_k and C_k have mr_k and $n_k r_k$ elements respectively. Similarly to the global model B_k and C_k are only determined up to an invertible $r_k \times r_k$ matrix G_k. We therefore get

$$\sum_{k=1}^{K} mr_k + n_k r_k - r_k^2 \tag{8.7}$$

DOF for the local models.

For the unified model we first consider the term $BU_kC_k^T$. Since B is $m \times r_0$, U_k is $r_0 \times r_k$ and C_k $n_k \times r_k$ this term has $mr_0 + r_0 r_k + n_k r_k$ elements. However, since

$$X_k = BU_kC_k^T = BGG^{-1}U_kG_kG_k^{-1}C_k^T, \tag{8.8}$$

there are two ambiguities here. The first subtracts r_0^2 DOF once and the second r_k^2 DOF for each cluster. Summing over k we thus get

$$mr_0 - r_0^2 + \sum_{k=1}^{K} r_0 r_k + r_k n_k - r_k^2. \tag{8.9}$$

Note that for independent clusters this reduces to (8.7). However when $r_0 < \sum_k r_k$ (and typically $r_0 \ll \sum_k r_k$) it is easy to see that the unified model is at least as compact as the local model. To compare to the global model we note that $\sum_k n_k = n$ and subtract (8.9) from (8.6). This gives

$$nr_0 - \sum_{k=1}^{K}(r_0 r_k + r_k n_k - r_k^2) = \sum_{k=1}^{K}(r_0 - r_k)(n_k - r_k). \tag{8.10}$$

Since we can't form clusters with fewer columns than their rank both terms of the product are positive, which confirms that the unified model is always at least as compact as the global model.

Affine Subspace Models In our applications we will typically use affine subspaces since it better models the affine camera projection. See Section 7.5.4 in Chapter 7 for a comparison of using affine and linear subspaces. In this case the matrix C_k^T is required to have one row of all ones, which reduces the DOF in this matrix to $n_k(r_k - 1)$. Furthermore, this requires the last row of G_k^{-1} to be $\begin{bmatrix} 0 & 0 & \dots & 1 \end{bmatrix}$ which therefore has $r_k(r_k - 1)$ DOF. The unified model then has

$$mr_0 - r_0^2 + \sum_{k=1}^{K} r_0 r_k + (r_k - 1)n_k - (r_k - 1)r_k \tag{8.11}$$

DOF. Note that the dimension of the affine subspace is $r_k - 1$ while the rank of its matrix $BU_kC_k^T$ is still r_k.

8.3.2 Predicting Missing Data

In this section we discuss the prediction capabilities of the unified model and illustrate how the global and local models complement each other when recovering missing data. To gain some intuition about the model we first consider the situation where a new column is added to each of the three factorizations, see Figure 8.2. In SfM this corresponds to estimation of a point track from a motion model. To generate a new column we need to specify coefficients in the C and C_k matrices (for some $k \in \{1, ..., K\}$), that is, the elements marked with c in Figure 8.2. In this example the global model needs to determine 5 parameters

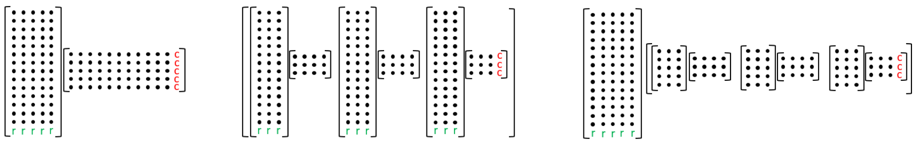


Figure 8.2: Three factorizations: *Left* - global model. *Middle* - union of subspace model. *Right* - unified model. Here $r_0 = 5$ and $r_i = 3$, $i = 1, 2, 3$. The r and c markers highlight elements that need to be estimated when adding a new row or column.

and therefore requires at least 5 known elements in the new column. For the local and unified models we only have 3 unknowns. (Additionally we may need a 4th known element to determine which cluster the new column belongs to.) Hence, in this situation the local and unified models require less data than the global model to predict missing elements.

Interestingly, when we consider rows instead of columns (see Figure 8.2) the relation is different. In SfM this situation corresponds to estimating a new scene shape from a shape model. For the global and the unified models there are 5 coefficients that needs to be determined. For the local model there are 9 since the cluster bases are independent. Hence the global and unified models can recover the entire row using 5 available measurements while the local model requires 9. Furthermore, note that the local model needs at least three measurements for each cluster since these are estimated independently. In contrast, the unified model could theoretically predict the entire row from measurements in a subset of the clusters. Specifically, if X_{new} is the new row (with missing data) we want to find a row B_{new} by solving

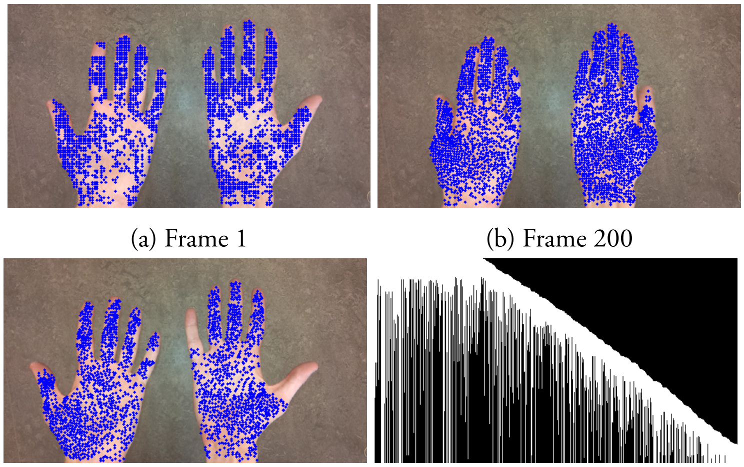
$$X_{\text{new}} = B_{\text{new}} \begin{bmatrix} U_1 C_1^T & U_2 C_2^T & \dots & U_K C_K^T \end{bmatrix} \tag{8.12}$$

(possibly in a least squares sense). This is possible if the columns of

$$\begin{bmatrix} U_1 C_1^T & U_2 C_2^T & \dots & U_K C_K^T \end{bmatrix}$$

that correspond to known data entries of X_{new} span an r_0-dimensional space. In the example of Figure 8.2 each $U_i V_i^T$ is of at most rank 3 hence it is not possible to completely determine B_{new} from only one cluster. However two clusters could be enough if their columns span the entire column space of B.

Next we show an example with data from real images that illustrates the benefits of using the unified model. The sequence consists of images containing two hands flexing, see Figure 8.3. Using the method of [207] we tracked points on the hands throughout the sequence. The dataset contains 7899 point trajectories in 441 frames with 67% missing data due to tracking failures. Figure 8.3 shows three of the 441 images together with the tracked points as well as the missing data pattern.



(a) Frame 1

(b) Frame 200

(c) Frame 441

(d) The missing data pattern.

Figure 8.3: Frames 1, 200 and 441 of the hand sequence. Note that in the last frame the right thumb has no tracks. Bottom right shows the missing data pattern. The observed entries of the measurement matrix are shown in white.

The point trajectories were manually partitioned into 14 approximately rigid components (see Figure 8.4d). Since each rigid component essentially only undergoes planar rotation and translation we restrict each cluster to a two-dimensional affine subspace (i.e. $r_k = 3$). For the global model we used $r_0 = 5$.

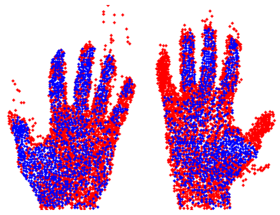
Figure 8.4 shows the result for the last frame of the sequence. In this frame the right thumb has almost no point trajectories due to tracking failures. Using

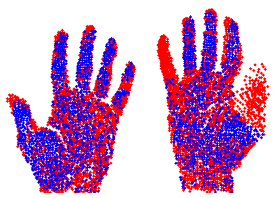
Dataset	Hand	Paper	Back	Heart
Global model	43880	3311	166824	547576
Local models	52758	2750	63472	163134
Unified model	20309	1686	43908	138814

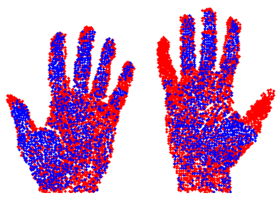
Table 8.1: DOF for the three type of models for various datasets used in the experiments (see Section 8.5.2 for more information.)

only the global model (Figure 8.4a) we can successfully recover the unobserved thumb but each rigid part is over-parameterized leading to over-fitting and noisy tracks. Table 8.1 (first column) shows the number of parameters for the three alternatives.

Using only the local models (Figure 8.4b) it is difficult to recover the correct track locations at the thumb when there are only a few visible tracks. Combining both the global and local models (Figure 8.4c) allows us to deal with the missing observations without over-parameterizing each rigid part.


(a) Global model


(b) Local models


(c) Unified model

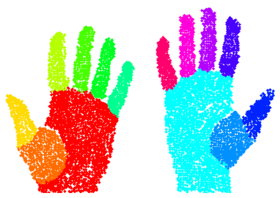

(d) Partitioning for local models.

Figure 8.4: The reconstructed tracks in the last frame of the sequence. The tracks which have observations in the current frame are shown in blue.

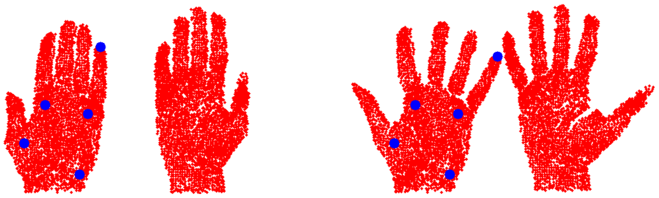


Figure 8.5: The constraint $r_0 = 5$ allows us to generate new shapes. Here the position of the five blue points where specified while the red points where predicted by the model.

Figure 8.5 illustrates how the unified model can estimate new poses (rows) from only 5 known point positions (since $r_0 = 5$). Note that since the hands move together throughout the sequence the learned model can infer the pose of the right hand (for which there are no measurements) from the left. If the clusters were treated independently each cluster would need at least 3 measurements for successful estimation. On the other hand our model would not fit well to a new image where for example the distance between the two hands is significantly different from what has been previously observed.

8.4 Model Fitting

In this section we present an energy-based optimization framework for computing compact factorizations. Given a measurement matrix M we seek a factorization

$$W \odot M \approx W \odot \left(B \left[U_1 C_1^T \quad \dots \quad U_K C_K^T\right] P\right), \tag{8.13}$$

where P is a permutation matrix that switches the order of columns and W is a binary matrix with element $w_{ij} = 1$ if m_{ij} is known and 0 otherwise. Changing column order using P corresponds to assigning a column of M to a particular cluster. Note that the overall rank r_0 (and thereby the size of B) is assumed to be known (otherwise it is possible that rank estimation methods similar to [98] could be used). However, the cluster number K, the ranks r_k and assignments are estimated by penalizing a trade-off between data fit and complexity.

For a fixed B determining the factorization can be seen as a model fitting problem where we assign affine subspaces to the columns of M. In the discrete setting, it is well known that these problems are NP hard [95]. However, [95, 49] have demonstrated that move making approaches such as α-expansion typically provide good solutions.

8.4.1 Energy Formulation

The approach we take essentially follows [95, 49] which generate a large but finite number of proposal subspaces and fuse them into a complete clustering by optimizing a discrete labeling energy using α-expansion [19].

Let ℓ be a labeling of the matrix columns. Then given a finite set of proposal subspaces $\{BU_k\}$, letting $l_p = k$ corresponds to assigning column p to cluster k. Note that once a column is assigned to a local subspace the coefficients C_k can be determined solving a simple least squares problem.

From the proposals we compute the cluster assignment by minimizing the discrete function

$$E(l) = \sum_p D_p(l_p) + \sum_k h_k \delta_k(l). \tag{8.14}$$

The data term D_p consists of two components. The first is a standard least squares term that measures the fit to the measurement matrix. The second component counts the number of elements required for representing the column in the factorization. Specifically, we use

$$D_p(k) = \min_c \|W_p \odot (M_p - BU_k c)\|_F^2 + \lambda(r_k - 1), \tag{8.15}$$

where W_p and M_p denote the p:th column of W and M respectively. Summing over the columns in the cluster the second term contributes $\lambda n_k(r_k - 1)$, which is the DOF in the C_k matrix of (8.11) times a weight λ. The weight λ controls the trade-off between data-fit and DOF.

The second term in (8.14) is a label cost term which we use to encode the remaining part of the model-complexity in (8.11) by setting

$$h_k = \lambda \left(r_0 r_k - (r_k - 1) r_k \right). \tag{8.16}$$

The function δ_k returns one if any of the columns is assigned to proposal k and zero otherwise. Thus using both the data term and the label cost we can achieve an adaptive penalization of the complexity of the factorization. Since we assume that r_0 is known the first term of (8.11) is constant and ignored.

Note that a pairwise Potts terms $V_{pq}(l_p, l_q)$ [19] can easily be introduced to (8.14) to add geometric context. From a practical point of view this can help to resolve ambiguous assignments in the vicinity of subspace intersections and therefore typically yields visually more appealing clusters. However it requires a neighborhood system and a number of additional parameters. For the experiments in the following section we therefore only use (8.14). In Section 8.6 we perform experiments with the pairwise term.

8.4.2 Optimization

It is clear from [49] that the above energy yields submodular α-expansions. Note however that the dimensionality of the search space is typically very large, which makes efficient proposal generation difficult. For example, to compute a affine subspace of dimension three we need to specify the elements of four columns, that is $4m$ elements, where m is the number of rows of M. Furthermore, because of missing data we cannot expect to be able to sample complete columns directly from M. To address this issue we maintain estimates of the B, U_k and C_k matrices and use these to fill in the measurement matrix. Using the completed measurement matrix we sample subsets of columns M_s and use these to estimate new U_k such that $M_s \approx BU_k$. If there is no application specific prior on the dimension of the local subspaces, the number of sampled columns is also selected at random in order to ensure that subspaces of different dimensions are generated. We employ the above proposal generation with α-expansion as outlined in [95]. In each iteration re-estimation is performed individually for the B, U_k and C_k matrices by solving the corresponding linear least squares problems. For initialization we find one r_0-dimensional subspace for the whole matrix using local optimization.

8.5 Experiments

In this section we will evaluate the performance of our method both quantitatively and qualitatively on different datasets and compare to several state-of-the-art methods. In order to obtain ground truth data we use a number of publicly available data sets and remove random entries from these. Figure 8.6 shows the data patterns that we consider. In the left pattern entries were discarded with a uniform probability. It is well known from compressed sensing that nuclear norm optimization works well (and even has performance guarantees [37]) for this kind of data. We argue that this setup is of limited interest since it does not occur in tracking based applications and further results in easier problem instances. Therefore we only test this type of data in Section 8.5.3 for completeness.

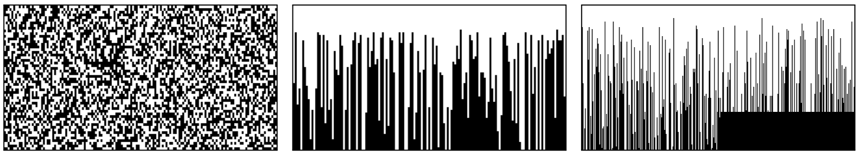


Figure 8.6: Examples of synthetic missing data patterns used for the experiments. Observed entries are shown in white. *Left:* Uniformly missing entires. *Middle:* Trajectories exhibiting tracking failure. *Right:* Tracking failure and occlusion.

To construct more realistic patterns we simulate tracking failure by randomly selecting (with uniform probability) if a track should have missing data. We then select (with uniform probability after the first few frames) in which image tracking failure occurs. No track is restarted after it has been lost. This results in data patterns such as the on displayed in the middle of Figure 8.6. In Section 8.5.2 we further simulate occlusion by removing a complete block of the matrix, see the right pattern of Figure 8.6.

8.5.1 Effects of the Trade-off Parameter λ

Our energy contains one parameter λ that controls the trade-off between model fit and DOF. To evaluate the behavior of our energy for different λ we use one instance from the CMU Motion Capture dataset. We used subject 10, which contains 5 sequences of a person kicking a soccer ball and one sequence of walking. These were selected since they were approximately the same size and they all provided about 330 point trajectories. The 3D points were projected into an orthographic camera slowly rotating around the subject. Some example frames can be seen in Figure 8.7. We generated missing data patterns as displayed in the middle image of Figure 8.6. Figure 8.8 shows how the resulting errors on the observed and missing data as well as the model complexity varies for different λ. For low values of λ the model fit term dominates the energy giving almost perfect fit to the available measurements. On the other hand model complexity is high which limits the ability to accurately predict missing data. The best results are achieved for mid range values of λ (in this case between 1 and 10). When λ is high the DOF of the model becomes too low to be able to capture the full scene

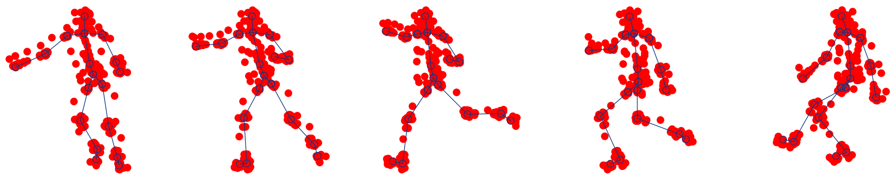


Figure 8.7: Some example frames from one of the soccer kick instances. Blue skeleton added for visualization.

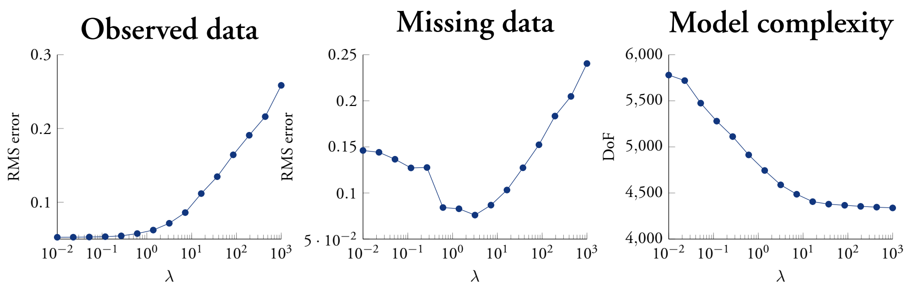


Figure 8.8: For different values of λ the plots show; *Left:* the RMS error on the observed data, *Middle:* the RMS error on the missing data, *Right:* the degrees of freedom in the resulting model.

dynamics resulting in poor prediction.

8.5.2 Occlusion and Tracking Failures

In this section we show some result on trajectories from three public image sequences: The *paper sequence* [185] containing 340 points in 70 frames, the *back sequence* [72] containing 20561 points in 150 frames and the *heart sequence* [72] containing 68295 points in 80 frames. To these we generated missing data as illustrated in the right image of Figure 8.6. (For occlusion we remove all trajectories in one half of the image for the last 25% of the frames.) Since these sequences are roughly locally planar we only sample affine rank 3 subspaces. We used $\lambda = 500$ in all three cases. Figure 8.9 shows the obtained clusterings and one frame from each sequence with the visible (blue) points and the reconstructed (red) points. Here we compare the local, global and unified models. For the local model we used the clustering computed by our method. Note that clusters that do not have any visible points due to occlusion are not reconstructed by the local

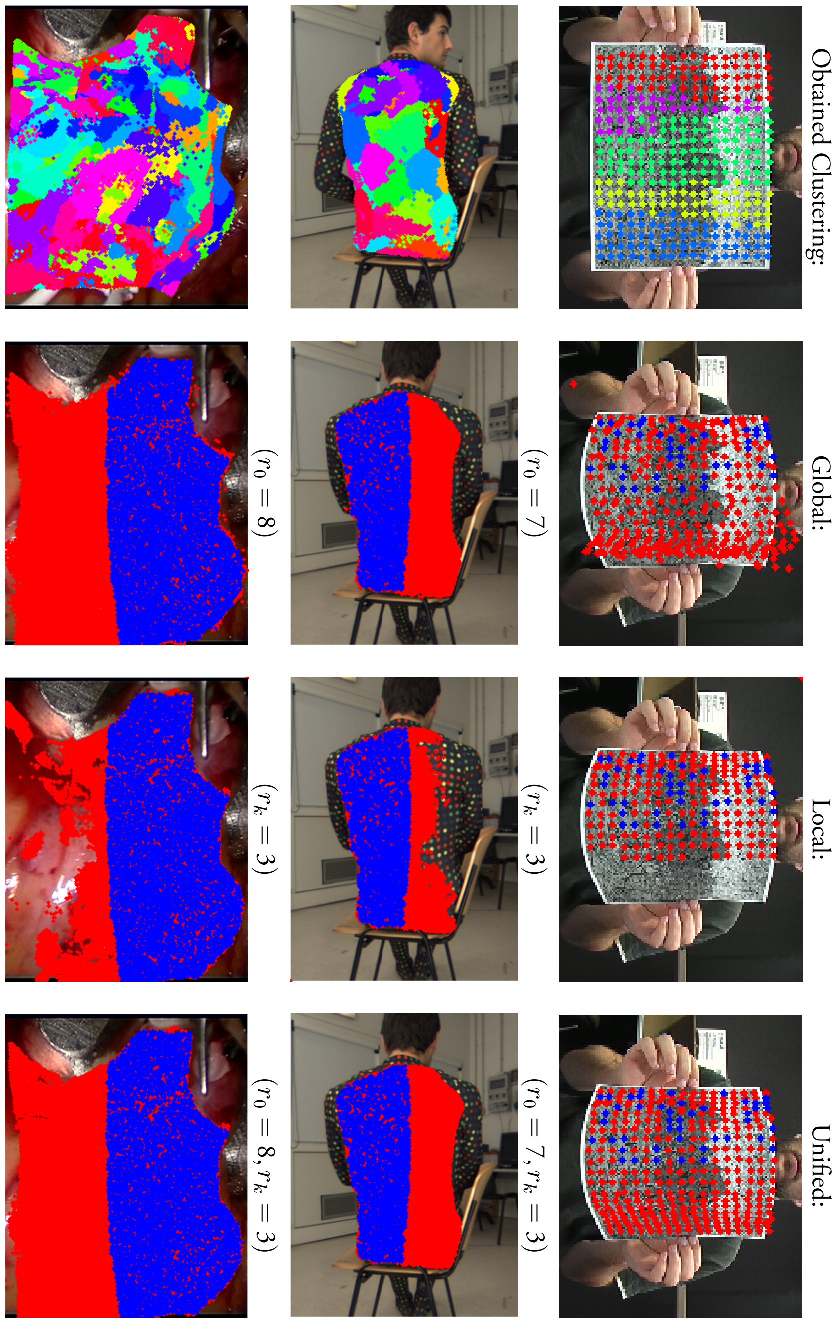


Figure 8.9: Paper, Back and Heart sequences from [185, 72]. The left column shows the clustering obtained using our method. The rest of the columns show the visible points (blue) and the reconstructed points (red) in one frame.

method. Table 8.2 shows the reconstruction errors for both missing and visible points. Table 8.1 shows the DOF of the resulting factorizations.

	Observed data			Missing data		
	Paper	Back	Heart	Paper	Back	Heart
Global	**3.95e1**	5.3e2	1.1e3	3.27e3	1.9e3	4.4e3
Local	1.04e2	**4.9e2**	**1.1e3**	2.79e4	1.1e5	2.2e5
Unified	1.07e2	6.9e2	1.5e3	**2.86e2**	**1.4e3**	**3.6e3**

Table 8.2: Reconstruction errors for visible and missing data. For each column the smallest errors highlighted in bold.

8.5.3 Quantitative Comparisons

Next we compare our approach to a number of state-of-the-art methods. We test four methods that are based on a single global rank model:

- **LM-r0** and **Wiberg**: Fitting a rank r_0 matrix by minimizing $f(B,C) = \|W \odot (BC^T - M)\|_F^2$ using Levenberg-Marquardt and the damped Wiberg method from [173] respectively.
- **CSF** and **CSF-DCT**: The column space fitting method from [74], both with and without using the DCT basis.
- **NN**: Nuclear norm minimization $f(X) = \lambda \|X\|_* + \|W \odot (X - M)\|_F^2$ using ADMM [18].

We also test using two approaches from [229] for clustering the columns followed by fitting local models to each cluster. For these methods we use rank 4 affine models since these correspond to rigid 3D objects.

- **SSC-EZWF+LM-r4**: The Entry-wise Zero-Fill method from [229], followed by fitting rank 4 affine models to each cluster.
- **NN+SSC+LM-r4**: Nuclear norm minimization (as in **NN**) followed by regular SSC [54]. Affine rank 4 models are fitted to the resulting clusters.

For the competing methods the available parameters were tuned for each dataset to give the best results. In our comparisons we measure the fraction of matrix elements that have reconstruction error less than a given threshold. This is because

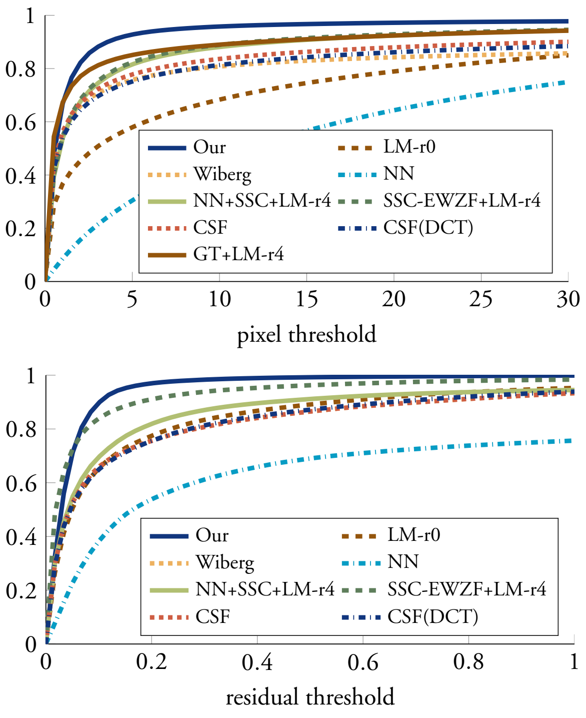


Figure 8.10: The fraction of residuals across all instances less than a threshold for the Hopkins (left) and MOCAP data (right).

over-fitting to noise may lead to highly unstable tracks which results in unpredictable ℓ_2 errors.

Hopkins155 and CMU Motion Capture. We first consider the Hopkins155 dataset [216] which is commonly used for motion segmentation. It contains 155 sequences with multiple rigidly moving objects. Since the dataset contains a ground truth clustering of the trajectories, we include a comparison with fitting local models to this partition. This is denoted **GT+LM-r4** in the results. For the methods using a global rank constraint we use $r_0 = 4K$ where K is the number of scene motions.

Here we generated both uniform missing entries and random tracking failure (left and middle of Figure 8.6). Table 8.3 shows the number of instances where more than 90% of the missing data was reconstructed with less than a 10

	Missing data	
Method	Uniform	Tracking failure
Our	**155 (100.0%)**	**148 (95.5%)**
LM-r0	143 (92.3%)	26 (16.8%)
Wiberg	152 (98.1%)	75 (48.4%)
NN	**155 (100.0%)**	6 (3.9%)
NN+SSC+LM-r4	**155 (100.0%)**	97 (62.6%)
SSC-EWZF+LM-r4	**155 (100.0%)**	93 (60.0%)
CSF	118 (76.1%)	70 (45.2%)
CSF(DCT)	154 (99.4%)	54 (34.8%)
GT+LM-r4	**155 (100.0%)**	88 (56.8%)

Table 8.3: Number of instances where 90% of the missing entries have less than 10px error.

pixel error. In Figure 8.10 we vary the pixel threshold and show the fraction of residuals, across all instances, that are reconstructed with a lower error. (Here we did not consider the uniform missing entry pattern.) It is a bit surprising that our approach outperforms competing methods, even **GT+LM-r4**, on this dataset since the ground truth clusters are typically independently moving objects. Closer inspection reveals that our method often splits objects into smaller dependent models, whose interactions are captured by the global rank constraint. Thus a likely explanation is that affine rank 4 models with ground truth clusters is still an over-parametrization (due to degenerate motions/planar structure). To the right in Figure 8.10 we show the results obtained when performing the same experiment on subject 10 of the MOCAP data set.

8.6 Spatial Smoothness for Labeling

In this section we consider the incorporation of a pair-wise Potts penalty [19] in our energy based model fitting framework. This regularization term, typically referred to as the smoothness term, adds spatial context to the formulation and can resolve ambiguous cluster assignments. This is particularly effective in areas where several good subspaces are available, e.g. in the vicinity of transitions between models.

	r_0	r_k	λ	μ	Size of M
hands	5	3	5000	100	882×7899
paper	7	3	500	100	140×340
back	8	3	500	100	300×20561
heart	8	$\{3,4\}$	500	100	160×68295

Table 8.4: Parameters used for the real image sequences. The column r_k contains the dimensions used for the local subspaces.

We use the formulation

$$E(l) = \sum_p \left(D_p(l_p) + \sum_{q \in \mathcal{N}(p)} \mu S_{pq}(l_p, l_q) \right) + \sum_k h_k \delta_k(l). \tag{8.17}$$

Recall that the data terms $D_p(l_p)$ and the label costs $h_k\delta_k(l)$ jointly encode a trade-off between data fit and model complexity. The additional term $S_{pq}(l_p, l_q)$ penalizes cluster assignments where neighboring trajectories are assigned different clusters. The function S_{pq} assumes the values 0 or 1 and the parameter μ controls the penalty strength.

For the neighborhood system we use the k nearest neighbors. We define the distance between column k and column ℓ as the maximum distance between the elements in their overlapping rows, i.e.

$$\text{dist}\,(M_k, M_\ell) = \max_{W_{ik}=W_{i\ell}=1} |M_{ik} - M_{i\ell}|. \tag{8.18}$$

If the two columns do not have any overlapping observations we define the distance to be infinite. In the following experiments we used the 8 nearest neighbors.

We applied the method with the smoothness term to the four image sequences; *hands*, *paper*, *back* and *heart*. The results are shown in Figure 8.11. The reconstruction error, both with and without the pairwise term, are reported in Table 8.5 for the cases when the ground truth is known. Note that while the labels are more visually pleasing the reconstruction quality is very similar. The problem sizes and parameters used for the experiments are shown in Table 8.4.

	Observed data			Missing data		
	Paper	Back	Heart	Paper	Back	Heart
$\mu = 0$	**1.07e2**	**6.86e2**	**1.50e3**	**2.86e2**	**1.44e3**	**3.67e3**
$\mu \neq 0$	1.50e2	7.61e2	1.79e3	3.63e2	**1.44e3**	3.70e3

Table 8.5: Comparison of reconstruction error for the three datasets where ground truth is available.

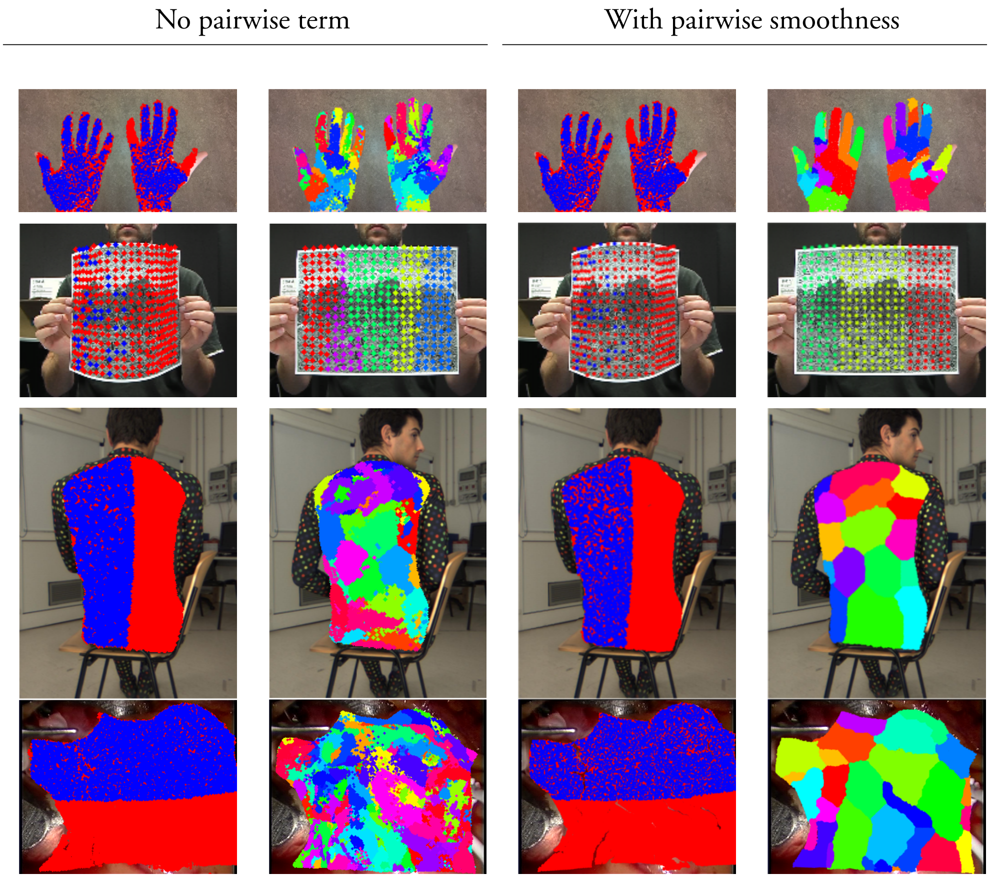


Figure 8.11: Estimated point positions and obtained clusters using the energy (8.17) with and without the smoothness term S_{pq}.

8.7 Conclusions

In this chapter we have presented an extension of the traditional low rank matrix factorization. In addition to requiring the matrix to be of low rank, we further constrain the rank of sub-matrices. This provides a more compact factorization of the matrix. We have shown in experiments that this model is well suited for task involving multi-body and non-rigid image point trajectories.

References

[1] *The MOSEK optimization toolbox for MATLAB manual.*

[2] RealityCapture. `www.capturingreality.com`.

[3] C. Aholt, S. Agarwal, and R. Thomas. A qcqp approach to triangulation. In *European Conference on Computer Vision (ECCV)*, 2012.

[4] C. Aholt and L. Oeding. The ideal of the trifocal variety. *Mathematics of Computation*, 83(289):2553–2574, 2014.

[5] I. Akhter, Y. A. Sheikh, S. Khan, and T. Kanade. Nonrigid structure from motion in trajectory space. In *Neural Information Processing Systems*, 2008.

[6] C. Albl, Z. Kukelova, A. Fitzgibbon, J. Heller, M. Smid, and T. Pajdla. On the two-view geometry of unsynchronized cameras. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[7] C. Albl, Z. Kukelova, and T. Pajdla. R6p-rolling shutter absolute camera pose. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2292–2300, 2015.

[8] F. Andersson, M. Carlsson, J.-Y. Tourneret, and H. Wendt. A new frequency estimation method for equally and unequally spaced data. *Signal Processing, IEEE Transactions on*, 62(21):5761–5774, 2014.

[9] R. Angst, C. Zach, and M. Pollefeys. The generalized trace-norm and its application to structure-from-motion problems. In *International Conference on Computer Vision (ICCV)*, 2011.

[10] P. M. Q. Aquiar, M. Stosic, and J. M. F. Xavier. Spectrally optimal factorization of incomplete matrices. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[11] A. Argyriou, R. Foygel, and N. Srebro. Sparse prediction with the k-support norm. In *Advances in Neural Information Processing Systems*, 2012.

[12] E. Ask, O. Enqvist, and F. Kahl. Optimal geometric fitting under the truncated l2-norm. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1722–1729, 2013.

[13] E. Ask, Y. Kuang, and K. Åström. Exploiting p-fold symmetries for faster polynomial equation solving. In *International Conference on Pattern Recognition (ICPR)*, 2012.

[14] E. Ask, Y. Kuang, and K. Åström. Exploiting p-fold symmetries for faster polynomial equation solving. In *International Conference on Pattern Recognition (ICPR)*, Tsukuba, Japan, 2012.

[15] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the solution of algebraic eigenvalue problems: a practical guide*. SIAM, 2000.

[16] J. P. Barreto and K. Daniilidis. Fundamental matrix for cameras with radial distortion. In *International Conference on Computer Vision (ICCV)*, volume 1, pages 625–632, 2005.

[17] R. Basri, D. Jacobs, and I. Kemelmacher. Photometric stereo with general, unknown lighting. *International Journal of Computer Vision (IJCV)*, 72(3):239–257, May 2007.

[18] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, Jan. 2011.

[19] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 23(11):1222–1239, 2001.

[20] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *Computer Vision and Pattern Recognition (CVPR)*, 2000.

[21] J. H. Brito, C. Zach, K. Koeser, M. Ferreira, and M. Pollefeys. One-sided radial-fundamental matrix estimation. In *British Machine Vision Conference (BMVC)*, 2012.

[22] M. Brown, R. I. Hartley, and D. Nistér. Minimal solutions for panoramic stitching. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2007.

[23] A. M. Buchanan and A. W. Fitzgibbon. Damped newton algorithms for matrix factorization with missing data. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[24] B. Buchberger. A theoretical basis for the reduction of polynomials to canonical forms. *ACM SIGSAM Bulletin*, 10(3):19–29, 1976.

[25] M. Bujnak. *Algebraic solutions to absolute pose problems*. PhD thesis, Czech Technical University, Prague., 2012.

[26] M. Bujnak, Z. Kukelova, and T. Pajdla. A general solution to the p4p problem for camera with unknown focal length. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008.

[27] M. Bujnak, Z. Kukelova, and T. Pajdla. 3d reconstruction from image collections with a single known focal length. In *International Conference on Computer Vision (ICCV)*, pages 1803–1810. IEEE, 2009.

[28] M. Bujnak, Z. Kukelova, and T. Pajdla. New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion. In *Asian Conference on Computer Vision (ACCV)*, pages 11–24. Springer, 2010.

[29] M. Bujnak, Z. Kukelova, and T. Pajdla. Making minimal solvers fast. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[30] S. Burgess, Y. Kuang, and K. Åström. Pose estimation from minimal dual-receiver configurations. In *International Conference on Pattern Recognition (ICPR)*, 2012.

[31] M. Byröd, M. Brown, and K. Åström. Minimal solutions for panoramic stitching with radial distortion. In *British Machine Vision Conference (BMVC)*, 2009.

[32] M. Byröd, K. Josephson, and K. Åström. Fast optimal three view triangulation. In *Asian Conference on Computer Vision (ACCV)*, 2007.

[33] M. Byröd, K. Josephson, and K. Åström. A column-pivoting based strategy for monomial ordering in numerical gröbner basis calculations. In *European Conference on Computer Vision (ECCV)*, 2008.

[34] M. Byröd, K. Josephson, and K. Åström. Fast and stable polynomial equation solving and its application to computer vision. *International Journal of Computer Vision (IJCV)*, 2009.

[35] R. Cabral, F. de la Torre, J. Costeira, and A. Bernardino. Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition. In *International Conference on Computer Vision (ICCV)*, 2013.

[36] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM J. on Optimization*, 20(4):1956–1982, 2010.

[37] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *J. ACM*, 58(3):11:1–11:37, June 2011.

[38] O. Chum and J. Matas. Matching with prosac-progressive sample consensus. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[39] O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. In *Joint Pattern Recognition Symposium*, pages 236–243. Springer, 2003.

[40] D. Claus and A. W. Fitzgibbon. A rational function lens distortion model for general cameras. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 213–219. IEEE, 2005.

[41] R. M. Corless, K. Gatermann, and I. S. Kotsireas. Using symmetries in the eigenvalue method for polynomial systems. *Journal of symbolic computation*, 44(11):1536–1550, 2009.

[42] D. Cox, J. Little, and D. O'Shea. *Ideals, Varities and Algorithms*. Springer-Verlag, New York, NY, USA, 1992.

[43] D. Cox, J. Little, and D. O'Shea. *Ideals, Varities and Algorithms*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, NY, USA, 2007.

[44] D. A. Cox, J. Little, and D. O'Shea. *Using algebraic geometry*, volume 185 of *Graduate Texts in Mathematics*. Springer-Verlag New York, 2005.

[45] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher. SfM with MRFs: Discrete-continuous optimization for large-scale structure from motion. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 35(12):2841–2853, December 2013.

[46] N. da Silva and J. Costeira. The normalized subspace inclusion: Robust clustering of motion subspaces. In *International Conference on Computer Vision (ICCV)*, pages 1444–1450, 2009.

[47] Y. Dai and H. Li. Rank minimization or nuclear-norm minimization: Are we solving the right problem? In *DICTA*, 2014.

[48] Y. Dai, H. Li, and M. He. A simple prior-free method for non-rigid structure-from-motion factorization. *International Journal of Computer Vision (IJCV)*, 107(2):101–122, 2014.

[49] A. Delong, A. Osokin, H. Isack, and Y. Boykov. Fast approximate energy minimization with label costs. *International Journal of Computer Vision (IJCV)*, 96:1–27, Jan. 2012.

[50] M. Demazure. *Sur deux problemes de reconstruction*. PhD thesis, INRIA, 1988.

[51] P. Denis, J. H. Elder, and F. J. Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In *European Conference on Computer Vision (ECCV)*, 2008.

[52] F. Devernay and O. D. Faugeras. Automatic calibration and removal of distortion from scenes of structured environments. In *SPIE's 1995 International Symposium on Optical Science, Engineering, and Instrumentation*, pages 62–72. International Society for Optics and Photonics, 1995.

[53] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[54] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 35(11):2765–2781, 2013.

[55] M. Elkadi and B. Mourrain. *Some applications of bezoutians in effective algebraic geometry*. PhD thesis, INRIA, 1998.

[56] I. Z. Emiris. On the complexity of sparse elimination. *Journal of Complexity*, 12(2):134–166, 1996.

[57] O. Enqvist, E. Ask, F. Kahl, and K. Åström. Tractable algorithms for robust model estimation. *International Journal of Computer Vision (IJCV)*, 112(1):115–129, 2015.

[58] A. Eriksson and A. Hengel. Efficient computation of robust weighted low-rank matrix approximations using the L_1 norm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(9):1681–1690, 2012.

[59] A. Eriksson, T. Thanh Pham, T.-J. Chin, and I. Reid. The k-support norm and convex envelopes of cardinality and rank. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[60] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint.* MIT press, 1993.

[61] J.-C. Faugere. A new efficient algorithm for computing gröbner bases (f4). *Journal of pure and applied algebra*, 139(1-3):61–88, 1999.

[62] P. Favaro, R. Vidal, and A. Ravichandran. A closed form solution to robust subspace estimation and clustering. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.

[63] M. Fazel, H. Hindi, and S. P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference*, 2001.

[64] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 1981.

[65] A. W. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–I. IEEE, 2001.

[66] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[67] K. Fragkiadaki, M. Salas, P. Arbelaez, and J. Malik. Grouping-based low-rank trajectory completion and 3d reconstruction. In *Advances in Neural Information Processing Systems 27*, 2014.

[68] F. Fraundorfer, P. Tanskanen, and M. Pollefeys. A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles. *European Conference on Computer Vision (ECCV)*, 2010.

[69] K. Fukuda, A. Jensen, and R. Thomas. Computing gröbner fans. *Math. Comput.*, 76(260):2189–2212, 2007.

[70] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 25(8):930–943, 2003.

[71] R. Garg, A. Roussos, and L. Agapito. A variational approach to video registration with subspace constraints. *International Journal of Computer Vision (IJCV)*, 104(3):286–314, 2013.

[72] R. Garg, A. Roussos, and L. de Agapito. Dense variational reconstruction of non-rigid surfaces from monocular video. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.

[73] N. Gillis and F. Glinuer. Low-rank matrix approximation with weights or missing data is np-hard. *SIAM Journal on Matrix Analysis and Applications*, 32(4), 2011.

[74] P. F. Gotardo and A. M. Martinez. Non-rigid structure from motion with complementary rank-3 spaces. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.

[75] D. R. Grayson and M. E. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at http://www.math.uiuc.edu/Macaulay2/.

[76] C. Grussler, A. Rantzer, and P. Giselsson. Low-rank optimization with convex constraints. *IEEE Transactions on Automatic Control*, 2018.

[77] S. Haner and K. Åström. Absolute pose for cameras under flat refractive interfaces. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1428–1436, 2015.

[78] R. Hartley. Projective reconstruction from line correspondences. In *Computer Vision and Pattern Recognition (CVPR)*, pages 903–907. IEEE Computer Society Press, 1994.

[79] R. Hartley. Lines and points in three views and the trifocal tensor. *International Journal of Computer Vision (IJCV)*, 22(2):125–140, March 1997.

[80] R. Hartley and F. Schaffalitzky. L_{inf} minimization in geometric reconstruction problems. In *Computer Vision and Pattern Recognition (CVPR)*, 2004.

[81] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[82] R. I. Hartley. Projective reconstruction and invariants from multiple images. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 16(10):1036–1041, 1994.

[83] R. I. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding (CVIU)*, 68(2):146–157, 1997.

[84] J. Hedborg, A. Robinson, and M. Felsberg. Robust three-view triangulation done fast. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 152–157, 2014.

[85] G. Hee Lee, M. Pollefeys, and F. Fraundorfer. Relative pose estimation for a multi-camera system with known vertical direction. In *Computer Vision and Pattern Recognition (CVPR)*, pages 540–547, 2014.

[86] J. Heikkila. Using sparse elimination for solving minimal problems in computer vision. In *International Conference on Computer Vision (ICCV)*, pages 76–84, 2017.

[87] J. A. Hesch and S. I. Roumeliotis. A direct least-squares (dls) method for pnp. In *International Conference on Computer Vision (ICCV)*, pages 383–390. IEEE, 2011.

[88] A. Heyden. *Geometry and Algebra of Multipe Projective Transformations*. PhD thesis, Lund Institute of Technology, Sweden, 1995.

[89] J.-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of convex analysis*. Springer Science & Business Media, 2012.

[90] H. Homeier. On newton-type methods with cubic convergence. *Journal of computational and applied mathematics*, 176(2):425–432, 2005.

[91] D. Hook and P. McAree. Using sturm sequences to bracket real roots of polynomial equations. In *Graphics gems*, pages 416–422. Academic Press Professional, Inc., 1990.

[92] Y. Hu, D. Zhang, J. Ye, X. Li, and X. He. Fast and accurate matrix completion via truncated nuclear norm regularization. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 35(9):2117–2130, 2013.

[93] J. Hyeong Hong and A. Fitzgibbon. Secrets of matrix factorization: Approximations, numerics, manifold optimization and random restarts. In *International Conference on Computer Vision (ICCV)*, pages 4130–4138, 2015.

[94] T. M. W. Inc. *MATLAB Reference Guide*. The Math Works, Inc., 1992.

[95] H. Isack and Y. Boykov. Energy-based geometric multi-model fitting. *International Journal of Computer Vision (IJCV)*, 97(2):123–147, 2012.

[96] A. N. Jensen. Gfan, a software system for Gröbner fans. http://home.imf.au.dk/ajensen/software/gfan/gfan.html.

[97] A. N. Jensen. A presentation of the gfan software. In *Mathematical Software - ICMS 2006, Second International Congress on Mathematical Software,*

Castro Urdiales, Spain, September 1-3, 2006, Proceedings, pages 222–224, 2006.

[98] P. Ji, M. Salzmann, and H. Li. Shape interaction matrix revisited and robustified: Efficient subspace clustering with corrupted and incomplete data. In *International Conference on Computer Vision (ICCV)*, pages 4687–4695, 2015.

[99] F. Jiang, Y. Kuang, J. E. Solem, and K. Åström. A minimal solution to relative pose with unknown focal length and radial distortion. In *Asian Conference on Computer Vision (ACCV)*, pages 443–456. Springer, 2014.

[100] H. Jin. A three-point minimal solution for panoramic stitching with lens distortion. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008.

[101] V. Jojic, S. Saria, and D. Koller. Convex envelopes of complexity controlling penalties: the case against premature envelopment. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[102] K. Josephson and M. Byröd. Pose estimation with radial distortion and unknown focal length. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2419–2426. IEEE, 2009.

[103] F. Kahl, S. Agarwal, M. K. Chandraker, D. Kriegman, and S. Belongie. Practical global optimization for multiview geometry. *International Journal of Computer Vision (IJCV)*, 79(3):271–284, 2008.

[104] K. Kanatani, Y. Sugaya, and H. Niitsuma. Triangulation from two views revisited: Hartley-sturm vs. optimal correction. In *British Machine Vision Conference (BMVC)*, 2008.

[105] D. Kapur, T. Saxena, and L. Yang. Algebraic and geometric reasoning using dixon resultants. In *Proceedings of the international symposium on Symbolic and algebraic computation*, pages 99–107. ACM, 1994.

[106] Q. Ke and T. Kanade. Robust l1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[107] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Trans. Inf. Theory*, 56(6):2980–2998, 2010.

[108] L. Kneip. Polyjam: Toolbox for symbolic polynomial computations and automatic groebner basis solver generation in c++. `https://github.com/laurentkneip/polyjam`. Accessed: 2018-03-28.

[109] L. Kneip, H. Li, and Y. Seo. Upnp: An optimal o (n) solution to the absolute pose problem with universal applicability. In *European Conference on Computer Vision (ECCV)*, pages 127–142, 2014.

[110] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2969–2976. IEEE, 2011.

[111] C. Kong and S. Lucey. Prior-less compressible structure from motion. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[112] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[113] Y. Kuang and K. Åström. Numerically stable optimization of polynomial solvers for minimal problems. In *European Conference on Computer Vision (ECCV)*, 2012.

[114] Y. Kuang and K. Åström. Pose estimation with unknown focal length using points, directions and lines. In *International Conference on Computer Vision (ICCV)*, pages 529–536, 2013.

[115] Y. Kuang and K. Åström. Stratified sensor network self-calibration from tdoa measurements. In *21st European Signal Processing Conference 2013*, 2013.

[116] Y. Kuang, S. Burgess, A. Torstensson, and K. Åström. A complete characterization and solution to the microphone position self-calibration problem. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3875–3879. IEEE, 2013.

[117] Y. Kuang, M. Oskarsson, and K. Åström. Revisiting trifocal tensor estimation using lines. In *International Conference on Pattern Recognition (ICPR)*, pages 2419–2423, 2014.

[118] Y. Kuang, J. E. Solem, F. Kahl, and K. Åström. Minimal solvers for relative pose with a single unknown radial distortion. In *Computer Vision and Pattern Recognition (CVPR)*, pages 33–40. IEEE, 2014.

[119] Y. Kuang, Y. Zheng, and K. Åström. Partial symmetry in polynomial systems and its applications in computer vision. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.

[120] Z. Kukelova. *Algebraic methods in computer vision*. PhD thesis, Czech Technical University, Prague., 2013.

[121] Z. Kukelova, M. Bujnak, J. Heller, and T. Pajdla. Singly-bordered block-diagonal form for minimal problem solvers. In *Asian Conference on Computer Vision (ACCV)*, pages 488–502. Springer, 2014.

[122] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. In *European Conference on Computer Vision (ECCV)*, pages 302–315. Springer, 2008.

[123] Z. Kukelova, M. Bujnak, and T. Pajdla. Polynomial eigenvalue solutions to the 5-pt and 6-pt relative pose problems. In *British Machine Vision Conference (BMVC)*, Leeds, UK, 2008.

[124] Z. Kukelova, M. Bujnak, and T. Pajdla. Polynomial eigenvalue solutions to minimal problems in computer vision. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 34(7):1381–1393, 2012.

[125] Z. Kukelova, M. Bujnak, and T. Pajdla. Real-time solution to the absolute pose problem with unknown radial distortion and focal length. In *International Conference on Computer Vision (ICCV)*, pages 2816–2823, 2013.

[126] Z. Kukelova, M. Byröd, K. Josephson, T. Pajdla, and K. Åström. Fast and robust numerical solutions to minimal problems for cameras with radial distortion. *Computer Vision and Image Understanding (CVIU)*, 114(2):234–244, 2010.

[127] Z. Kukelova, J. Heller, M. Bujnak, A. Fitzgibbon, and T. Pajdla. Efficient solution to the epipolar geometry for radially distorted cameras. In *International Conference on Computer Vision (ICCV)*, pages 2309–2317, 2015.

[128] Z. Kukelova, J. Heller, M. Bujnak, and T. Pajdla. Radial distortion homography. In *Computer Vision and Pattern Recognition (CVPR)*, pages 639–647, 2015.

[129] Z. Kukelova, J. Kileel, B. Sturmfels, and T. Pajdla. A clever elimination strategy for efficient minimal solvers. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[130] Z. Kukelova and T. Pajdla. A minimal solution to the autocalibration of radial distortion. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–7. IEEE, 2007.

[131] Z. Kukelova, T. Pajdla, and M. Bujnak. Fast and stable algebraic solution to l2 three-view triangulation. In *International Conference on 3D Vision (3DV)*, 2013.

[132] H. Lai, Y. Pan, C. Lu, Y. Tang, and S. Yan. Efficient k-support matrix pursuit. In *European Conference on Computer Vision (ECCV)*, volume 8690, 2014.

[133] V. Larsson and K. Åström. Uncovering symmetries in polynomial systems. In *European Conference on Computer Vision (ECCV)*, 2016.

[134] V. Larsson, K. Åström, and M. Oskarsson. Efficient solvers for minimal problems by syzygy-based reduction. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[135] V. Larsson, K. Åström, and M. Oskarsson. Polynomial solvers for saturated ideals. In *International Conference on Computer Vision (ICCV)*, 2017.

[136] V. Larsson, E. Bylow, C. Olsson, and F. Kahl. Rank minimization with structured data patterns. In *European Conference on Computer Vision (ECCV)*, 2014.

[137] V. Larsson, Z. Kukelova, and Y. Zheng. Making minimal solvers for absolute pose estimation compact and robust. In *International Conference on Computer Vision (ICCV)*, pages 2335–2343. IEEE, 2017.

[138] V. Larsson, Z. Kukelova, and Y. Zheng. Camera pose estimation with unknown principal point. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[139] V. Larsson and C. Olsson. Convex envelopes for low rank approximation. In *Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, 2015.

[140] V. Larsson and C. Olsson. Convex low rank approximation. *International Journal of Computer Vision (IJCV)*, 120(2):194–214, 2016.

[141] V. Larsson and C. Olsson. Compact matrix factorization with dependent subspaces. In *Computer Vision and Pattern Recognition (CVPR)*, pages 280–289, 2017.

[142] V. Larsson, M. Oskarsson, K. Åström, A. Wallis, Z. Kukelova, and T. Pajdla. Beyond gröbner bases: Basis selection for minimal solvers. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[143] T.-L. Lee, T.-Y. Li, and C.-H. Tsai. Hom4ps-2.0: a software package for solving polynomial systems by the polyhedral homotopy continuation method. *Computing*, 83(2-3):109, 2008.

[144] B. Li, L. Heng, G. H. Lee, and M. Pollefeys. A 4-point algorithm for relative pose estimation of a calibrated camera with a known relative rotation angle. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1595–1601. IEEE, 2013.

[145] H. Li. A simple solution to the six-point two-view focal-length problem. In *European Conference on Computer Vision (ECCV)*, pages 200–213. Springer, 2006.

[146] H. Li and R. Hartley. A non-iterative method for correcting lens distortion from nine-point correspondences. In *OmniVision'05, ICCV-workshop*, 2005.

[147] H. Li and R. Hartley. Five-point motion estimation made easy. In *International Conference on Pattern Recognition (ICPR)*, 2006.

[148] T.-Y. Li. Numerical solution of multivariate polynomial systems by homotopy continuation methods. *Acta numerica*, 6:399–436, 1997.

[149] X. Liang, X. Ren, Z. Zhang, and Y. Ma. Repairing sparse low-rank texture. In *European Conference on Computer Vision (ECCV)*, 2012.

[150] Z. Lin, M. Chen, and Y. Ma. The augmented lagrange multiplier method for exact recovery of corrupted low rank matrices. *Mathematical Programming*, 2010.

[151] P. Lindstrom. Triangulation made easy. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[152] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 35(1):171–184, 2013.

[153] F. Lu and R. Hartley. A fast optimal algorithm for l 2 triangulation. In *Asian Conference on Computer Vision (ACCV)*, 2007.

[154] F. Macaulay. Some formulae in elimination. *Proceedings of the London Mathematical Society*, 1(1):3–27, 1902.

[155] L. Magerand and A. Del Bue. Practical projective structure from motion (p2sfm). In *Computer Vision and Pattern Recognition (CVPR)*, pages 39–47, 2017.

[156] M. Marques and J. Costeira. Estimating 3d shape from degenerate sequences with missing data. *Computer Vision and Image Understanding (CVIU)*, 113(2):261 – 272, 2009.

[157] S. Maybank. *Theory of reconstruction from image motion*, volume 28. Springer Science & Business Media, 2012.

[158] R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *J. Mach. Learn. Res.*, 11:2287–2322, 2010.

[159] A. M. McDonald, M. Pontil, and D. Stamos. Spectral k-support norm regularization. In *Advances in Neural Information Processing Systems*, 2014.

[160] Y. Min. L-infinity norm minimization in the multiview triangulation. In *International Conference on Artificial Intelligence and Computational Intelligence*, pages 488–494. Springer, 2010.

[161] L. Mirsky. A trace inequality of john von neumann. *Monatshefte für mathematik*, 79(4):303–306, 1975.

[162] F. M. Mirzaei and S. I. Roumeliotis. Optimal estimation of vanishing points in a manhattan world. In *International Conference on Computer Vision (ICCV)*, 2011.

[163] H. M. Möller and H. J. Stetter. Multivariate polynomial equations with multiple zeros solved by matrix eigenproblems. *Numerische Mathematik*, 70(3):311–329, 1995.

[164] M. B. Monagan, K. O. Geddes, K. M. Heal, G. Labahn, S. M. Vorkoetter, J. McCarron, and P. DeMarco. *Maple 10 Programming Guide*. Maplesoft, Waterloo ON, Canada, 2005.

[165] F. Mora and L. Robbiano. The Gröbner fan of an ideal. *Journal of Symbolic Computation*, 6(2-3):183–208, 1988.

[166] P. Moulon, P. Monasse, and R. Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *International Conference on Computer Vision (ICCV)*, pages 3248–3255. IEEE, 2013.

[167] G. Nakano. Globally optimal dls method for pnp problem with cayley parameterization. In *British Machine Vision Conference (BMVC)*, 2015.

[168] O. Naroditsky and K. Daniilidis. Optimizing polynomial solvers for minimal geometry problems. In *International Conference on Computer Vision (ICCV)*, 2011.

[169] O. Naroditsky, X. S. Zhou, J. Gallier, S. I. Roumeliotis, and K. Daniilidis. Two efficient solutions for visual odometry using directional correspondence. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 34(4):818–824, 2012.

[170] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 2004.

[171] V. Noferini and A. Townsend. Numerical instability of resultant methods for multidimensional rootfinding. *SIAM Journal on Numerical Analysis*, 54(2):719–743, 2016.

[172] T. Okatani and K. Deguchi. On the wiberg algorithm for factorization with missing data. *International Journal of Computer Vision (IJCV)*, 72(3):329–337, 2007.

[173] T. Okatani, T. Yoshida, and K. Deguchi. Efficient algorithm for low-rank matrix factorization with missing components and performance comparison of latest algorithms. In *International Conference on Computer Vision (ICCV)*, 2011.

[174] C. Olsson and O. Enqvist. Stable structure from motion for unordered image collections. In *Scandinavian Conference on Image Analysis (SCIA)*, 2011.

[175] C. Olsson and M. Oskarsson. A convex approach to low rank matrix approximation with missing data. In *Scandinavian Conference on Image Analysis (SCIA)*, 2009.

[176] M. Oskarsson, A. Zisserman, and K. Åström. Minimal projective reconstruction for combinations of points and lines in three views. *Image and Vision Computing (IVC)*, 22(10):777–785, 2004.

[177] S. Petitjean. Algebraic geometry and computer vision: Polynomial systems, real and complex roots. *Journal of Mathematical Imaging and Vision (JMIV)*, 10(3):191–220, 1999.

[178] J. Pritts, Z. Kukelova, V. Larsson, and O. Chum. Radially-distorted conjugate translations. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[179] L. Quan. Invariants of six points and projective reconstruction from three uncalibrated images. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 17(1):34–46, 1995.

[180] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Rev.*, 52(3):471–501, Aug. 2010.

[181] L. Robbiano. Term orderings on the polynominal ring. In *Proc. of EUROCAL '85, European Conference on Computer Algebra*, 1985.

[182] R. Rockafellar. *Convex Analysis*. Princeton University Press, 1997.

[183] C. Russell, R. Yu, and L. Agapito. Video pop-up: Monocular 3d reconstruction of dynamic scenes. In *European Conference on Computer Vision (ECCV)*, 2014.

[184] M. Salzmann. Continuous inference in graphical models with polynomial energies. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1744–1751. IEEE, 2013.

[185] M. Salzmann, R. Hartley, and P. Fua. Convex optimization for deformable surface 3-d tracking. In *International Conference on Computer Vision (ICCV)*, pages 1–8. IEEE, 2007.

[186] O. Saurer, M. Pollefeys, and G. H. Lee. A minimal solution to the rolling shutter pose estimation problem. In *Intelligent Robots and Systems (IROS)*, pages 1328–1334. IEEE, 2015.

[187] O. Saurer, P. Vasseur, C. Demonceaux, and F. Fraundorfer. A homography formulation to the 3pt plus a common direction relative pose problem. In *Asian Conference on Computer Vision*, pages 288–301. Springer, 2014.

[188] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[189] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[190] A. Segers. *Algebraic Attacks from a Gröbner Basis Perspective*. PhD thesis, Eindhoven University of Technology, 2004.

[191] Q. Shi, A. Eriksson, A. Van Den Hengel, and C. Shen. Is face recognition really a compressive sensing problem? In *Computer Vision and Pattern Recognition (CVPR)*, pages 553–560. IEEE, 2011.

[192] Z. Simayijiang, S. Burgess, Y. Kuang, and K. Åström. Minimal solutions for dual microphone rig self-calibration. In *European Signal Processing Conference (EUSIPCO)*, 2014.

[193] Z. Simayijiang, S. Burgess, Y. Kuang, and K. Åström. Toa-based self-calibration of dual-microphone array. *IEEE Journal on Selected Topics in Signal Processing*, 9(5):791–801, 2015.

[194] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM transactions on graphics (TOG)*, volume 25, pages 835–846. ACM, 2006.

[195] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision (IJCV)*, 80(2):189–210, 2008.

[196] A. M.-C. So and Y. Ye. Theory of semidefinite programming for sensor network localization. *Mathematical Programming*, 109(2-3):367–384, 2007.

[197] P. Stefanovic. Relative orientation–a new approach. *ITC Journal*, 3:417–448, 1973.

[198] H. Stewénius. *Gröbner Basis Methods for Minimal Problems in Computer Vision*. PhD thesis, Lund University, 2005.

[199] H. Stewénius, C. Engels, and D. Nistér. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284–294, 2006.

[200] H. Stewénius, D. Nistér, F. Kahl, and F. Schaffalitzky. A minimal solution for relative pose with unknown focal length. *Image and Vision Computing (IVC)*, 26(7):871–877, 2008.

[201] H. Stewénius, D. Nistér, M. Oskarsson, and K. Åström. Solutions to minimal generalized relative pose problems. In *Workshop on Omnidirectional Vision*, Beijing China, OCT 2005.

[202] H. Stewenius, F. Schaffalitzky, and D. Nister. How hard is 3-view triangulation really? In *International Conference on Computer Vision (ICCV)*, 2005.

[203] D. Strelow. General and nested Wiberg minimization. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[204] J. F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12:625–653, 1999.

[205] B. Sturmfels. On the newton polytope of the resultant. *Journal of Algebraic Combinatorics*, 3(2):207–236, 1994.

[206] B. Sturmfels. *Gröbner Bases and Convex Polytopes*. University Lecture Series. American Mathematical Society, Providence, RI, USA, 1996.

[207] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *European Conference on Computer Vision (ECCV)*, pages 438–451. Springer, 2010.

[208] L. Svarm, O. Enqvist, F. Kahl, and M. Oskarsson. City-scale localization for cameras with known vertical direction. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 2016.

[209] C. Sweeney, J. Flynn, and M. Turk. Solving for relative pose with a partially known rotation is a quadratic eigenvalue problem. In *International Conference on 3D Vision (3DV)*, volume 1, pages 483–490. IEEE, 2014.

[210] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[211] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision (IJCV)*, 9(2):137–154, 1992.

[212] P. H. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding (CVIU)*, 78(1):138–156, 2000.

[213] P. H. S. Torr. Bayesian model estimation and selection for epipolar geometry and generic manifold fitting. *International Journal of Computer Vision (IJCV)*, 50(1):35–61, 2002.

[214] B. Triggs. Camera pose and calibration from 4 or 5 known 3d points. In *International Conference on Computer Vision (ICCV)*, volume 1, pages 278–284. IEEE, 1999.

[215] B. Triggs. Routines for relative pose of two calibrated cameras from 5 points. 2000. Technical Report.

[216] R. Tron and R. Vidal. A benchmark for the comparison of 3D motion segmentation algorithms. In *Computer Vision and Pattern Recognition (CVPR)*, 2007.

[217] R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, 1987.

[218] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. A minimal solution to the generalized pose-and-scale problem. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.

[219] J. Verschelde. *Homotopy continuation methods for solving polynomial systems*. PhD thesis, University of California, Berkeley, 1996.

[220] J. Verschelde. Algorithm 795: Phcpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software (TOMS)*, 25(2):251–276, 1999.

[221] R. Vidal, R. Tron, and R. Hartley. Multiframe motion segmentation with missing data using powerfactorization and gpca. *International Journal of Computer Vision (IJCV)*, 79(1):85–105, 2008.

[222] J. von Neumann. Some matrix-inequalities and metrization of matrix-space. *Tomsk University Review. vol 1*, 1937.

[223] S. Wang, D. Liu, and Z. Zhang. Nonconvex relaxation approaches to robust matrix recovery. In *International Joint Conference on Artificial Intelligence*, 2013.

[224] S. Weerakoon and T. Fernando. A variant of newton's method with accelerated third-order convergence. *Applied Mathematics Letters*, 13(8):87–93, 2000.

[225] T. Wiberg. Computation of principal components when data are missing. In *Proc. Second Symp Computational Statistics*, 2013.

[226] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 31(2):210–227, 2009.

[227] C. Wu. P3.5p: Pose estimation with unknown focal length. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2440–2448, 2015.

[228] J. Yan and M. Pollefeys. A factorization-based approach for articulated nonrigid shape, motion and kinematic chain recovery from video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(5):865–877, 2008.

[229] C. Yang, D. Robinson, and R. Vidal. Sparse subspace clustering with missing entries. In *International Conference on Machine Learning (ICML)*, 2015.

[230] L. Zappella, A. D. Bue, X. Lladó, and J. Salvi. Joint estimation of segmentation and structure from motion. *Computer Vision and Image Understanding (CVIU)*, 117(2):113 – 129, 2013.

[231] E. Zheng, K. Wang, E. Dunn, and J.-M. Frahm. Minimal solvers for 3d geometry from satellite imagery. In *International Conference on Computer Vision (ICCV)*, December 2015.

[232] Y. Zheng, Y. Kuang, S. Sugimoto, K. Åström, and M. Okutomi. Revisiting the pnp problem: A fast, general and optimal solution. In *International Conference on Computer Vision (ICCV)*, 2013.

[233] Y. Zheng, G. Liu, S. Sugimoto, S. Yan, and M. Okutomi. Practical low-rank matrix approximation under robust L_1-norm. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[234] Y. Zheng, S. Sugimoto, I. Sato, and M. Okutomi. A general and simple method for camera pose and focal length determination. In *Computer Vision and Pattern Recognition (CVPR)*, pages 430–437, 2014.

[235] Y. Zhu, D. Huang, F. De La Torre, and S. Lucey. Complex non-rigid motion 3d reconstruction by union of subspaces. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.

[236] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005.