Low Rank Matrix Factorization and Relative Pose Problems in Computer Vision

Jiang, Fangyuan

2015

Low Rank Matrix Factorization and Relative Pose Problems in Computer Vision

Fangyuan Jiang

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 2015:6
ISSN 1404-0034

ISBN 978-91-7623-363-4 (print)
ISBN 978-91-7623-364-1 (digital)
LUTFMA-1054-2015

Printed in Sweden by MediaTryck, Lund 2015

Preface

This thesis is focused on geometric computer vision problems. The first part of the thesis aims at solving one fundamental problem, namely low-rank matrix factorization. We provide several novel insights into the problem. In brief, we characterize, generate, parametrize and solve the minimal problems associated with low-rank matrix factorization. Beyond that, we give several new algorithms based on the minimal solvers when the measurement matrix is either sparse, noisy or with outliers. The cost function and the algorithm can easily be adapted to several robust norms, for example, the L_1-norm and the truncated L_1-norm. We demonstrate our approach on several geometric computer vision problems. Another application is in sensor network calibration, which is also explored.

The second part of the thesis deals with the relative pose problem. We solve the minimal problem of estimating the relative pose with unknown focal length and radial distortion. Beyond that, we also propose a brute force approach, which does not suffer from common algorithmic degeneracies. Further, the algorithm achieves a globally optimal solution up to a discretization error and it is easily parallelizable. Finally, we look into the problem of object detection with unknown pose.

The contents of the thesis are based on the following papers.

Main papers

- Fangyuan Jiang, Magnus Oskarsson and Kalle Åström (2015) "On the Minimal Problems of Low-rank Matrix Factorization" Accepted by *IEEE Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, United States, 2015

- Fangyuan Jiang, Olof Enqvist and Fredrik Kahl (2015) "A Combina-

torial Approach to L_1-Matrix Factorization" Published in *Journal of Mathematical Imaging and Vision (JMIV)*, 2015

- Fangyuan Jiang, Yubin Kuang, Jan-Erik Solem and Kalle Åström (2014) "A Minimal Solution to Relative Pose with Unknown Focal Length and Radial Distortion" Published at *Asian Conference on Computer Vision (ACCV)*, Singapore, 2014

- Fangyuan Jiang, Olof Enqvist and Fredrik Kahl (2013) "Improved Object Detection and Pose Using Part-Based Models" Published at *Scandinavian Conference on Image Analysis (SCIA)*, Espoo, Finland, 2013

- Fangyuan Jiang, Yubin Kuang and Kalle Åström (2013) "Time Delay Estimation for TDOA Self-calibration using Truncated Nuclear Norm Regularization" Published at *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada 2013

- Olof Enqvist, Fangyuan Jiang and Fredrik Kahl " A Brute-force Algorithm for Reconstructing a Scene from Two Projections" Published at *IEEE Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, CO, United States, 2011

Subsidiary papers

- Dennis Medved, Fangyuan Jiang, Peter Exner, Magnus Oskarsson, Pierre Nugues and Kalle Åström "Combining Text Semantics and Image Geometry to Improve Scene Interpretation" Published at *International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, Angers, France, 2014.

- Agnes Tegen, Rebecka Weegar, Linus Hammarlund, Magnus Oskarsson, Fangyuan Jiang, Dennis Medved, Pierre Nugues and Kalle Åström "Image Segmentation and Labeling using Free-form Semantic Annotation" Published at *International Conference on Pattern Recognition (ICPR)*, Stockholm, Sweden, 2014.

Acknowledgements

I would like to express my very deep gratitude to my supervisor, Fredrik Kahl, for all the inspirations and wisdom you have shared through many fruitful discussions. Also the great gratitude extends to my co-supervisors, Kalle Åström for tremendous support and infinite patience, and Olof Enqvist for many good ideas, interesting discussions and careful reading with several manuscripts. Without you, the thesis could not have been finished. I would also like to thank all my co-authors and colleagues in the vision group, especially Yubin Kuang for collaboration and contribution in two papers.

For everything else, I would like to thank my Mom and Dad for unconditional love and support, as always.

Berlin, March 2015

Contents

Chapter 1
Introduction

Bilinearity is an essential relationship for problems in geometric computer vision. In affine structure-from-motion, this bilinear relation comes from the interaction between the camera motion and the 3D scene points. In photometric stereo, the bilinearity stems from the interaction between the different light sources and the surfaces of the 3D objects. In the non-rigid structure-from-motion, the underlying linear shape basis and cameras give the bilinear model. In sensor network calibration, the positions of the transmitters and receivers also form a bilinear relation in the relative distance matrix. When one collects the observations, for example, the image coordinates in affine structure-from-motion, the intensity values in photometric stereo or the point tracks in the non-rigid structure-from-motion, and stack them in a matrix, the underlying bilinear relations lead to a low-rank property on the measurement matrix.

To recover the camera motion and the 3D scene points in affine structure-from-motion, or to figure out the directions of the light sources together with the surface normal of an object in photometric stereo, or to estimate the camera motion and the 3D shape in non-rigid structure-from-motion, it requires us to find a low-rank matrix factorization given an observation matrix. Without missing data or outliers, the problem can be solved optimally using Singular Value Decomposition (SVD). However, the missing data are very common due to occlusion or loss of point tracks. Outliers cannot be ignored in many applications, which makes the low-rank matrix factorization a hard problem.

State-of-the-art approaches either use a bilinear formulation and alternatively optimize between the two factor matrices given an initial guess, or use a nuclear-norm based convex relaxation technique. As we will show in the thesis, the bilinear formulation suffers from the local minima problem, and finding a good initialization is almost as hard as the original problem in

presence of missing values and outliers. On the other hand, the performance of nuclear norm-based method is affected by the amount of missing data. When the measurement matrix is very sparse, those methods will eventually fail.

In the second part of the thesis, another key problem in geometric computer vision, namely, the relative pose problem is studied. Given two images of a scene, one interesting problem is to estimate the relative motion between the two cameras. A minimal problem seeks a solution using a minimal set of point correspondences. In the thesis, we provide a minimal solution to the relative pose problem when camera has unknown radial distortion and focal length. RANSAC bundled with minimal solvers achieves robust estimation. However, RANSAC does not guarantee the optimality and the minimal solvers can suffer from algorithmic degeneracy. In the thesis, a brute-force algorithm is proposed to estimate the relative orientation problem in a globally optimal way which can handle degenerated or restricted motions. Incorporating the 3D pose estimation in the objection detection is also explored at the end of the thesis.

1.1 Thesis Overview

The thesis is divided into the following chapters.

Chapter 2. Some preliminary knowledge and background are provided to understand the models and the algorithms used in the thesis.

Chapter 3. The low-rank matrix factorization problem under the L_1-norm is studied in this chapter. If each column of a matrix X is treated as a point in $\mathbb{R}^m$, then saying that the matrix X is of low rank is equivalent to that all the data points lie in a low-dimensional linear subspace. The problem can equivalently be treated as to find a low-dimensional linear subspace such that when all the data points are projected onto the subspace, the projection error is minimized.

In this chapter, we first give an optimal L_1-projection algorithm assuming that the subspace is given. To find the optimal subspace, the special case of hyperplane fitting is discussed first and an optimal algorithm is given. For the general case, we explore the zero patterns in the residual matrix and focus on the cases with sufficient zeros in the residual matrix

such that the problem is well constrained and directly solvable from the entries in the observation matrix which correspond to zeros in the residual matrix. For low-dimensional problems, we show empirically the optimal solution is very likely (more than 90%) to be one of those cases. We propose both an exhaustive search and a random search algorithm to search among the cases that are linearly solvable. The algorithm is evaluated on affine structure-from-motion and photometric stereo problems.

Author contribution: The initial idea to investigate the problem is from Fredrik Kahl. My contributions to this chapter include co-developing the theory, implementing most of the algorithms and performing the experiments. Each author of the paper contributes roughly the same to writing the article.

Chapter 4. In this chapter, we further look into the minimal problems of low-rank matrix factorization. The minimal problems are characterized using an index matrix W to indicate if a certain element is missing or present. The inspiration comes from the Laman graph in rigid graph theory, which describes a family of minimal rigid systems represented by vertices and edges of a planar graph. Using analogy of the Henneberg extension in a Laman graph, we propose several extension schemes, namely Henneberg-k extensions to generate all minimal problems of a certain rank iteratively. We further propose algorithmic solvers for different Henneberg extensions, either linear or based on a simple polynomial solver. We demonstrate the minimal solvers in both synthetic data and real applications, for example, affine structure-from-motion and linear shape basis estimation. Our method outperforms the state of the art in several cases when the measurement matrix is sparse or contains outliers.

Author contribution: Kalle Åström proposed the idea and we collaborated on the theory and implementation of the solvers. I implemented the block-partition algorithm and performed all the experiments.

Chapter 5. In this chapter, we illustrate one application of rank minimization in sensor network calibration. In this problem, the receivers are calibrated, which means the actual time when the signal arrives at a certain receiver is measured. However, the transmitters are uncalibrated, in other words, the time of transmission is unknown. If we could estimate the unknown time of transmission, then the time duration the signal travels

between a pair of transmitter and receiver is directly obtained, which can be expressed as a distance measure to recover the location of each sensor. In this chapter, we focus on estimating the unknown time of transmission by encoding it into the measurement matrix. The low-rank constraint is derived for the measurement matrix with unknowns. The problem is formulated as a truncated nuclear norm regularization (TNNR) and optimized using the alternating direction method of multipliers (ADMM).

Author contribution: My contributions to this chapter include improving the implementation and performing some experiments. Most of the theoretical part and implementation were done by Yubin Kuang.

Chapter 6. Another key problem in geometric computer vision, the relative pose problem is studied in this chapter. The problem is to estimate the relative rotation and the translation between two views given the feature correspondences. If the camera is calibrated, that is, the intrinsic parameters are known, then the well-known 5-point algorithm in Nistér (2004) would solve the problem. If the camera is partially calibrated with only an unknown focal length, then the minimal solver in Stewénius et al. (2005) solves the problem. However, the case when both cameras have the same unknown radial distortion and focal length still remains unsolved. Introducing both the radial distortion and focal length as unknowns leads to a complicated polynomial system, with higher degrees and more unknowns to solve. However, with the recent progress in Gröbner basis methods, solving such a complicated polynomial system becomes possible.

Author contribution: For this chapter, I implemented and performed the experiments and wrote the article. Yubin Kuang contributed to the idea and implementation of the solvers.

Chapter 7. Minimal solvers with RANSAC provide a robust way to estimate the relative pose between two views. However, RANSAC does not guarantee the global optimality. Besides, minimal solvers are known to suffer from the algorithmic degeneracy. In this chapter, a brute force method is proposed to estimate the relative pose between two views. With a non-standard epipolar parametrization, the epipole of a camera is represented as a vector on a unit sphere. By an exhaustive search on a discretized unit sphere, it is possible to find the globally optimal solution up to a discretized error. The algorithm does not suffer from any degeneracy and could handle several

restricted motions, like planar motion. A parallel algorithm is proposed and implemented using CUDA on a graphical card. It achieves state-of-the-art result in the Hopkins 155 dataset for the motion segmentation problem.

Author contribution: My contributions in this chapter include developing and implementing the parallel algorithm on a graphical card using CUDA. I have also contributed to the motion segmentation experiments with spatial priors. Fredrik Kahl and Olof Enqvist developed the main theory.

Chapter 8. In this chapter, we incorporate pose estimation in the object detection problem. We annotate each side of a block-shaped object, instead of a single bounding box on the whole object. Using richer annotations, we can directly estimate the 3D pose of an object. With the estimated poses, we could rectify the object and train an individual deformable part-based model using Felzenszwalb et al. (2010b) for each of its distinctive aspects. A small set of typical poses are also selected from the training set by formulating it as a vertex-cover problem. The test images are transformed using the typical poses, and detected with different aspect detectors. By combining the detections, we simultaneously detect the object and recover the object pose.

Author contribution: I did most of the work in this chapter, including developing the algorithms, implementing the whole detection pipeline and performing the experiments. Olof Enqvist contributed to the algorithms and also writing.

Chapter 2
Preliminaries

This chapter aims to provide the background and preliminary knowledge necessary for the subsequent chapters.

2.1 Camera Model

A camera maps the 3D world to a 2D image. Such a linear mapping can be represented by a matrix of size 3×4, which maps a 3D point to a 2D point on the image plane, both in homogeneous coordinates. Most camera models considered in geometric computer vision are specialized versions of projective cameras.

In this thesis, we mainly consider two types of cameras: one is the pinhole camera, which has a finite camera center; the other is the affine camera, whose camera center is at infinity.

Pinhole camera model The pinhole camera model is based on central projection. Consider a pinhole camera with *camera center* C at the origin, that is $C = (0, 0, 0)^{\mathrm{T}}$. We denote the *image plane* π. The *principal axis* is the line from the camera center C perpendicular to the image plane π. The *principal point* is the point where the principal axis intersects with the image plane (see Figure 2.1).

To illustrate the central projection, consider a 3D point $X = (x, y, z)^{\mathrm{T}}$, its viewing ray $X - C$ from the camera center C to X intersects with the image plane π at point $\mathbf{x}$. If we use f to denote the focal length, that is the distance from the camera center to the image plane, then a simple equation captures the essential relationship of the pinhole camera model

$$(x, y, z)^{\mathrm{T}} \mapsto (fx/z, fy/z)^{\mathrm{T}}. \tag{2.1}$$

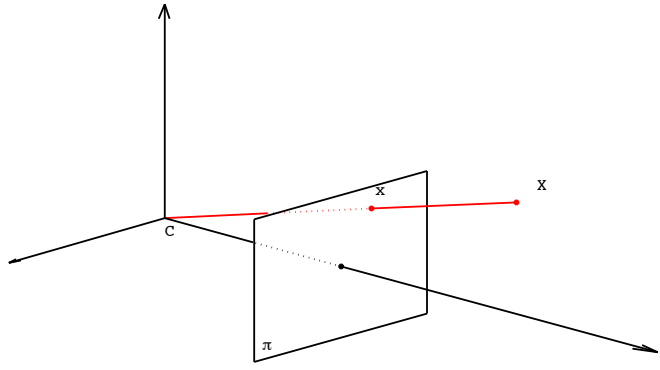


Figure 2.1: The central projection

Using homogeneous coordinates, we obtain the following equation

$$\begin{bmatrix} fx/z \\ fy/z \\ 1 \end{bmatrix} \sim \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{2.2}$$

The matrix in (2.2) can be rewritten as $\mathtt{P} = \mathrm{diag}(f, f, 1)[\mathtt{I}|0]$, which represents a simple pinhole camera that maps a world point at $(x, y, z, 1)^{\mathrm{T}}$ to an image point at $(fx/z, fy/z, 1)^{\mathrm{T}}$ in homogeneous coordinates. In general, a camera matrix P is a 3×4 matrix that maps a world point X to its projection $\mathbf{x}$ on the image plane using the following camera equation

$$\lambda \mathbf{x} = PX, \tag{2.3}$$

where both X and $\mathbf{x}$ are in homogeneous coordinates and λ is the depth, that is the distance between the camera center and the world point in the direction of the principal axis. Note that the camera center C is the unique null space of the camera matrix P since $PC = 0$.

Intrinsic parameters In (2.2), we assume that the origin of the image plane is at the principal point. As this might not be the case in practice, we should use the following general mapping that accounts for the offset of the

principal point

$$(x, y, z)^{\mathrm{T}} \mapsto (fx/z + p_x, fy/z + p_y)^{\mathrm{T}}. \tag{2.4}$$

Using homogeneous coordinates and rewriting the general mapping in a matrix form, we have

$$\begin{bmatrix} fx/z + p_x \\ fy/z + p_y \\ 1 \end{bmatrix} \sim \begin{bmatrix} fx + p_x z \\ fy + p_y z \\ z \end{bmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}. \tag{2.5}$$

If we separate the focal length f and the principal point (p_x, p_y) from the matrix in 2.5, this gives us the *intrinsic matrix*, or called the *calibration matrix* of a camera as

$$\mathtt{K} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}. \tag{2.6}$$

The complete calibration matrix also contains two extra parameters. One is *skew* s, which is zero for most cameras with the x-axis and y-axis perpendicular to each other. The other parameter is the *aspect ratio* α, which is one for cameras with the same scale in both the x and y directions. This gives the calibration matrix of a camera as

$$\mathtt{K} = \begin{bmatrix} f & s & p_x \\ & \alpha f & p_y \\ & & 1 \end{bmatrix}. \tag{2.7}$$

In this thesis, we assume that camera models have zero skew and a unit aspect ratio, that is $s = 0$ and $\alpha = 1$.

Extrinsic parameters In the above example, we assume that the coordinate framework of the camera and the world coincide, that is, the camera center lies at the origin and the principal axis points toward the z-axis.

However, in the general case, we need to define a mapping from the world coordinate framework to the camera coordinate framework. This mapping contains a rotation matrix, $\mathtt{R} \in \mathbb{R}^{3\times3}$, and a translation vector, $\mathtt{t} \in \mathbb{R}^3$, and is known as the *extrinsic parameters* of a camera.

Assume that the camera center is C in the world coordinate framework; to map a point from its world coordinates $\tilde{X}$ to its camera coordinates $\tilde{X}_{cam}$ we have

$$\tilde{X}_{cam} = \mathtt{R}(\tilde{X} - C), \tag{2.8}$$

where both $\tilde{X}_{cam}$ and $\tilde{X}$ are inhomogeneous coordinates. If we denote $\mathtt{t}$ as the translation vector, that is $\mathtt{t} = -\mathtt{R}C$, then we have the following equation

$$\tilde{X}_{cam} = \mathtt{R}\tilde{X} + \mathtt{t}. \tag{2.9}$$

Using homogeneous coordinates, we have

$$\lambda \mathbf{x} = \mathtt{K}[\mathtt{R}\ \mathtt{t}]X. \tag{2.10}$$

Radial distortion By now, we assume a linear camera model, that is, the world point, the image point, and the camera center are collinear. However, this collinearity does not hold for cameras with *radial distortion*. In that case, the image points are distorted depending on their distance from the center of distortion.

More specifically, radial distortion is modeled using the following equation

$$\mathbf{x}_d = L(r_u)\mathbf{x}_u, \tag{2.11}$$

where $\mathbf{x}_d$ and $\mathbf{x}_u$ are inhomogeneous coordinates of the distorted and the undistorted image points, respectively, and r_u is the radial distance of x_u from the center of radial distortion. $L(r_u)$ is a distortion factor that depends only on the radial distance, r_u, which could be modeled using a polynomial as follows

$$L(r) = 1 + \lambda_1 r + \lambda_2 r^2 + \lambda_3 r^3 + ... \tag{2.12}$$

In practice, it is unnecessary to include higher-order terms in (2.12) as the accuracy required in the feature matching is of the order of a pixel (see Fitzgibbon (2001)). A common practice is to expand $L(r)$ using Taylor expansion, keeping only the first nonlinear even term, which gives

$$\mathbf{x}_d = (1 + \lambda r_u^2)\mathbf{x}_u. \tag{2.13}$$

To further render $L(r)$ tractable for inverse transformation, a division model is proposed by Fitzgibbon (2001) for estimating the undistorted image points given the distorted ones, that is,

$$\mathbf{x}_u = \frac{1}{1 + \lambda r_d^2} \mathbf{x}_d. \tag{2.14}$$

This division model is widely used in the sequel work and is also adopted in the thesis.

Affine camera In Chapter 3, we will use an affine camera model, that is the model of a camera whose center is at infinity. This models the limit case when one moves the camera center away from the object along the principal axis, while simultaneously zooming in with the lens, which increases the focal length to keep the object of interest the same size. When both the focal length and the distance from the object increase, the image remains the same size but the perspective effect is reduced (see Hartley and Zisserman (2004)).

An affine camera is defined such that the third row of the camera matrix, P, is $(0, 0, 0, 1)$. It is straightforward to demonstrate that points at infinity are mapping to points at infinity for affine cameras.

2.2 Epipolar Geometry

The most interesting and fundamental problem in multi-view geometry stems from the two-view case, in other words, *epipolar geometry*.

Consider two cameras with camera centers C_1 and C_2 as in Figure 2.2; 3D world point X is projected onto the image plane of two cameras at $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. The line going through C_1 and C_2 intersect with the image planes π_1 and π_2 at points $\mathbf{e}_1$ and $\mathbf{e}_2$, respectively, which are called *epipoles*. Point $\mathbf{x}_1$ in one image corresponds to an *epipolar line* that goes through $\mathbf{e}_2$ in the other image.

Fundamental matrix Corresponding points from two views are related using a 3×3 matrix, called a *fundamental matrix*. In other words, for a pair of feature correspondences $(\mathbf{x}_1, \mathbf{x}_2)$ in homogeneous coordinates, we have the following constraint

$$\mathbf{x}_2^{\mathrm{T}} \mathbf{F} \mathbf{x}_1 = 0, \tag{2.15}$$

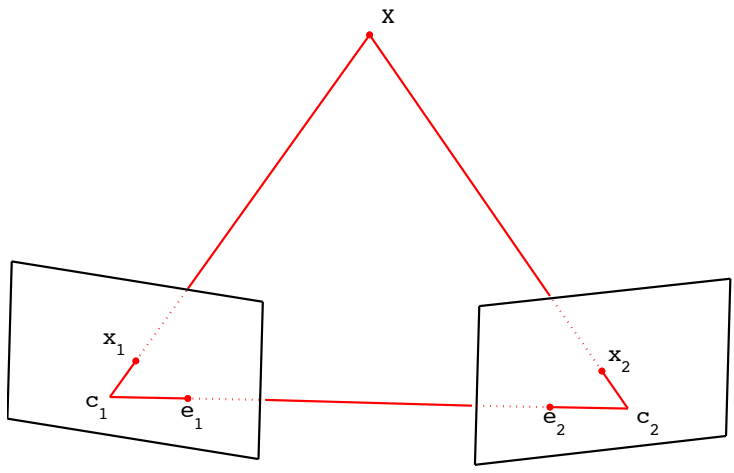


Figure 2.2: Epipolar geometry

where $\mathtt{F}$ is the fundamental matrix with zero determinant, that is,

$$\det(\mathtt{F}) = 0. \tag{2.16}$$

In Longuet-Higgins (1981), an eight-point algorithm is proposed for estimating the fundamental matrix. In practice, the normalized eight-point algorithm presented in Hartley (1997) is often a better choice as it offers improved numerical stability.

Essential matrix If two cameras are calibrated, that is, the calibration matrices $\mathtt{K}_1$ and $\mathtt{K}_2$ are known, an *essential matrix* $\mathtt{E}$ is used to represent the epipolar constraint as follows

$$\tilde{\mathbf{x}}_2^{\mathrm{T}} \mathtt{E} \tilde{\mathbf{x}}_1 = 0, \tag{2.17}$$

where $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$ are normalized coordinates as in $\tilde{\mathbf{x}}_1 = \mathtt{K}_1^{-1}\mathbf{x}_1$ and $\tilde{\mathbf{x}}_2 = \mathtt{K}_2^{-1}\mathbf{x}_2$. The essential matrix $\mathtt{E}$ has a determinant of zero

$$\det(\mathtt{E}) = 0, \tag{2.18}$$

and has two equal non-zero singular values. This constraint is equivalently formulated using trace as follows

$$\mathtt{E}\mathtt{E}^{\mathrm{T}}\mathtt{E} - \frac{1}{2}\mathrm{trace}(\mathtt{E}\mathtt{E}^{\mathrm{T}})\mathtt{E} = 0. \tag{2.19}$$

Once the essential matrix $\mathtt{E}$ is estimated, one can recover the relative rotation $\mathtt{R}$, and the translation $\mathtt{t}$, from the following constraint

$$\mathtt{E} = [\mathtt{t}]_{\times}\mathtt{R}, \tag{2.20}$$

where $[\,]_{\times}$ is the cross-product form of a vector. The essential matrix, $\mathtt{E}$, has only five degrees of freedom since both $\mathtt{R}$ and $\mathtt{t}$ have three degrees of freedom and there is an overall scale ambiguity. This means that we need at least five point correspondences to estimate $\mathtt{E}$.

However, due to the trace constraint, (2.19), solving $\mathtt{E}$ using a minimum of five points involves some nonlinear algebraic equations, which makes it a more difficult problem than solving $\mathtt{F}$. In Nistér (2004), an efficient five-point algorithm is proposed for estimating the coefficients and finding the roots of a tenth-degree polynomial in a closed form.

In Chapter 6, we also use an indirect way to find the essential matrix $\mathtt{E}$ from the fundamental matrix $\mathtt{F}$ and the calibration matrices $\mathtt{K}_1$ and $\mathtt{K}_2$. The relationship between $\mathtt{F}$ and $\mathtt{E}$ is given as

$$\mathtt{E} = \mathtt{K}_2^{\mathrm{T}}\mathtt{F}\mathtt{K}_1. \tag{2.21}$$

2.3 A Polynomial Solver

Minimal problems In this thesis, we frequently deal with minimal problems in geometric computer vision. A minimal problem is formulated using the minimum required constraints. One motivation for studying and solving a minimal problem is that a smaller set of correspondences is more likely to be outlier free. However, using minimal configurations usually introduces complexity into a problem. For example, solving $\mathtt{E}$ using a minimal five-point algorithm requires solving a tenth-degree polynomial, whereas a non-minimal eight-point algorithm for estimating $\mathtt{F}$ only entails solving a linear system. Fortunately, recent progress in the Gröbner basis method (see Byröd et al. (2009)) makes it tractable and efficient to deal with relatively large algebraic systems, meaning that some problems previously considered unsolvable are now tractable.

System of polynomials In the following, we only consider polynomial systems with a finite number of solutions. Assume we have a polynomial system H comprising the following equations

$$\begin{aligned} f_1(\mathbf{x}) &= 0, \\ &\vdots \\ f_n(\mathbf{x}) &= 0, \end{aligned} \tag{2.22}$$

where $\mathbf{x} = \{x_1, x_2, ..., x_m\}$ and $f_i(\mathbf{x}) \in \mathbb{C}[x_1, \cdots, x_m]$ are polynomials in the following form

$$f_i(\mathbf{x}) = \sum_{\alpha} c_\alpha \mathbf{x}^\alpha, \tag{2.23}$$

that is, a linear combination of *monomials*. Each monomial $\mathbf{x}^\alpha$ is in the form of a product

$$x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_m^{\alpha_m}, \tag{2.24}$$

where α_i is a non-negative integer for $i = 1, \ldots, m$. We use $\mathbb{C}[x_1, \cdots, x_m]$ or simply $\mathbb{C}[\mathbf{x}]$ to denote the set of polynomials in $\{x_1, \cdots, x_m\}$ with coefficients in the complex domain $\mathbb{C}$

In algebraic geometry (see Cox et al. (2005)), the above polynomial equations together generate an *ideal* denoted as

$$I =< f_1, \cdots, f_n >= \{\sum_{i=1}^{n} p_i f_i \mid p_i \in \mathbb{C}[\mathbf{x}]\}, \tag{2.25}$$

where $\{f_1, f_2, \cdots, f_n\}$ is called a *generating set* or *generator* of I. The ideal I in (2.25) is a generalization of the polynomial equations in (2.22). One reason we study ideals is due to the following property: A point, $\mathbf{x}$, is a zero of (2.22) if and only if it is a zero of I defined in (2.25).

An ideal is not uniquely generated. One could find another polynomial system, H', that generates the same ideal I but is different from H. Since the zero set of a polynomial system is determined by the zero set of the ideal it generates, this implies a way to solve polynomial system H by finding another generator, H', that generates the same ideal but is simpler to solve. The Gröbner basis is such a generator.

Gröbner basis One basic problem in algebraic geometry is that of determining whether a polynomial, f, is an element of a given ideal, $I =< f_1, \cdots, f_n >$. This requires a polynomial division algorithm to divide f by I. If the remainder of the division is zero, then f is in the ideal I.

Polynomial division in the univariate case is trivial and well-defined. However, in the multivariate case, several difficulties exist. One first needs to define an ordering of monomials for division to proceed. A simple choice is the *lexicographical ordering*. For example, the lexicographic ordering of $\{x, y, z\}$ is defined as $x > y > z$. For more choices of monomial ordering, see Cox et al. (2005). Another difficulty is that under a fixed order of monomials, the division of a polynomial, f, by an ideal, I, is generally not well-defined in the sense that the remainder depends on the generator one chooses.

As long ago as 1965, Bruno Buchberger in his PhD thesis (originally in German; see Buchberger (2006) for a recent English version), proposed a special generator, called the *Grönber basis* after his PhD advisor, Wolfgang Gröbner, together with an algorithm, the Buchberger algorithm, to compute it. The Gröbner basis, denoted $\mathcal{G}$, is a special generator of a given ideal, I, with the property that the multivariate division of any polynomial f by $\mathcal{G}$ is well-defined. In other words, the division of f by $\mathcal{G}$ has a zero remainder if and only if f is in the ideal I generated by $\mathcal{G}$. To compute using the Gröbner basis, one also needs to specify the order of the monomials. It turns out that using the *degree reverse lexicographic order*, (see Cox et al. (2005)) gives the most efficient Gröbner basis computation.

Buchberger's algorithm provides an approach to computing the Gröbner basis of an ideal, I, starting from any generating set, H. Briefly stated, it works by successively eliminating the leading terms of a pair of polynomials in H. It provides a theoretical way to compute $\mathcal{G}$ in exact arithmetic terms. However, successive elimination of leading terms could pose some serious numerical challenges. In practice, the process quickly becomes numerically unstable in floating-point arithmetic due to the round-off errors.

Action matrix Before further describing the action matrix, we first define an equivalent relationship in a set of polynomials. We say that two polynomials, f and g, are *equivalent modulo* I if $f - g \in I$. This equivalent relationship naturally groups a set of polynomials into equivalent classes,

denoted by the *coset* $[f]$ as

$$[f] = f + I = \{f + h : h \in I\}. \tag{2.26}$$

All these equivalent classes together form a quotient space, denoted $\mathbb{C}[\mathbf{x}]/I$. Using the quotient space, one can find a compact representation of any polynomial $f \in \mathbb{C}[\mathbf{x}]$. More specifically, using the multivariate division, any polynomial f divided by the Gröbner basis $\mathcal{G} = \{g_1, \cdots, g_t\}$ could be represented as

$$f = h_1 g_1 + \cdots + h_t g_t + \bar{f}^{\mathcal{G}}, \tag{2.27}$$

where $\bar{f}^{\mathcal{G}}$ denotes the remainder of f divided by $\mathcal{G}$. One can demonstrate that the equivalent class, or the coset $[f]$, is in one-to-one correspondence with the remainder $\bar{f}^{\mathcal{G}}$. So one can use the remainder $\bar{f}^{\mathcal{G}}$ as a representative of its coset $[f]$ in $\mathbb{C}[\mathbf{x}]/I$. It can easily be demonstrated that the sum of two remainders, $\bar{f}^{\mathcal{G}}$ and $\bar{g}^{\mathcal{G}}$, yields remainder $\overline{f+g}^{\mathcal{G}}$ and that one can multiply a remainder by a constant. In other words, quotient space $\mathbb{C}[\mathbf{x}]/I$ is a vector space. If we only consider polynomial systems with a finite number of solutions, then $\mathbb{C}[\mathbf{x}]/I$ is of finite dimensions: it has r dimensions, where r is the number of solutions to the ideal generated by $\mathcal{G}$

Given a polynomial, $p(\mathbf{x})$, consider the operation $T_p : f(\mathbf{x}) \mapsto p(\mathbf{x})f(\mathbf{x})$, where both $p(\mathbf{x})$ and $f(\mathbf{x}) \in \mathbb{C}[\mathbf{x}]$. The operation defines a mapping from $\mathbb{C}[\mathbf{x}]/I$ to itself. As the quotient space $\mathbb{C}[\mathbf{x}]/I$ is a finite-dimensional vector space, this mapping can be represented as matrix M_p, which is called the *action matrix*. In the following, we only consider a single variable, x_i, that acts on $f(\mathbf{x})$; that is, $p(\mathbf{x}) = x_i$ and x_i is referred to as the *action variable*. We know that each polynomial $f(\mathbf{x})$ can be represented as $f(\mathbf{x}) = \mathbf{c}^{\mathrm{T}}\mathbf{b}(\mathbf{x})$, where $\mathbf{c}$ is the coefficient vector and $\mathbf{b}(\mathbf{x})$ is the vector of basis monomials. Because the action of $p(\mathbf{x})$ on f is independent of the coefficient $\mathbf{c}$, one can rewrite this as

$$p(\bar{\mathbf{x}})\mathbf{b}(\bar{\mathbf{x}}) = M_p^{\mathrm{T}}\mathbf{b}(\bar{\mathbf{x}}), \tag{2.28}$$

where $\bar{\mathbf{x}}$ is in the zero set of f, that is $f(\bar{\mathbf{x}}) = 0$. From (2.28), one can recognize an eigenvalue problem of M_p^{T}, the eigenvalues and eigenvectors of which are $p(\mathbf{x})$ and $\mathbf{b}(\mathbf{x})$, respectively, both evaluated on the zero set of $f(\mathbf{x})$.

Now the remaining problem is to find the *basis monomials*. Looking back at (2.27), one can easily see that the remainder, $\bar{f}^{\mathcal{G}}$, contains no leading terms of the ideal, I. Actually, remainders are linear combinations of monomials, $\mathbf{x}^{\alpha}$, that exclude the leading terms of I. The set of monomials is linearly independent and can be regarded as a basis of $\mathbb{C}[\mathbf{x}]/I$. This gives a way to generate basis monomials: One first computes the Gröbner basis $\mathcal{G}$, then collects all the monomials that are not the leading terms of $\mathcal{G}$, which forms a set of basis monomials.

In practice, any set that includes those collected monomials can be used as a basis set. A larger basis set usually gives better numerical stability. To construct the action matrix, one needs to express those monomials not in the basis set in terms of the basis monomials. This sometimes leads to a numerically difficult situation if we only use a minimal set of r basis monomials. The idea proposed in Byröd et al. (2009) is to move those "problematic" monomials, which might cause numerical problems when expressed using basis monomials, into the basis set to produce a redundant basis set. In that case, we trade off an improved numerical situation against the cost of solving a larger eigenvalue problem. The redundant set of basis monomials is referred to as the *permissible* set, from which one could use a column-pivoting strategy to select the minimal basis set. More details of this strategy are described in the following section.

Polynomial solver in practice In practice, given a system of polynomial equations, the first step is to use algebraic geometry software, such as Macaulay2 (see Grayson and Stillman (1993-2002)), to find the number, r, of solutions. The software defines the dimension of the quotient space, $\mathbb{C}[\mathbf{x}]/I$. The next step is to generate a redundant system of equations by multiplying each polynomial by various monomials. This can be done systematically to generate a polynomial system of a given degree. A redundant system is generated in order to obtain a sufficiently large set of monomials, from which one can select a basis set, $\mathcal{B}$, such that the monomials not in $\mathcal{B}$ can be represented using basis monomials; this is the premise for constructing the action matrix.

Once we generate a redundant polynomial system, $\bar{H}$, from the original system, H, it can be expressed in the following matrix form

$$C^{\mathrm{T}}\mathbf{x}_{\mathcal{M}} = 0, \tag{2.29}$$

where C is the coefficient matrix and $\mathbf{x}_{\mathcal{M}}$ is the vector of all monomials. The next step is to choose an action variable, denoted x_a, based on which we partition the set of monomials, $\mathcal{M}$, into three disjoint sets, namely, the *excessive set*, *reducible set*, and *permissible set*, denoted $\mathcal{E}$, $\mathcal{R}$, and $\mathcal{P}$, respectively. The partition is based on the following rules:

- Permissible set $\mathcal{P}$ contains monomials that remain in set $\mathcal{M}$ after multiplying them by the action variable, x_a, that is, $\mathcal{P} = \{\mathbf{x} | x_a\mathbf{x} \in \mathcal{M}\}$.
- Reducible set $\mathcal{R}$ contains monomials not in permissible set $\mathcal{P}$ but that are multiplications of the action variable by permissible monomials, that is, $\mathcal{R} = x_a\mathcal{P} \setminus \mathcal{P}$.
- Excessive set $\mathcal{E}$ contains monomials that are in neither permissible set $\mathcal{P}$ nor reducible set $\mathcal{R}$, that is, $\mathcal{E} = \mathcal{M} \setminus (\mathcal{P} \cup \mathcal{R})$.

Note that a permissible set, $\mathcal{P}$, is considered a redundant set that contains the basis set, $\mathcal{B}$. After reordering columns of the coefficient matrix C and the elements of the monomial vector $\mathbf{x}_{\mathcal{M}}$ based on the above partition, we can rewrite (2.29) in the following equation

$$\begin{bmatrix} C_{\mathcal{E}} & C_{\mathcal{R}} & C_{\mathcal{P}} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{E}} \\ \mathbf{x}_{\mathcal{R}} \\ \mathbf{x}_{\mathcal{P}} \end{bmatrix} = 0. \tag{2.30}$$

Putting everything in matrix form enables us to use theories and tools from numerical linear algebra. For example, eliminating the leading terms now becomes a row operation on the coefficient matrix. We use a column-pivoting strategy as in Byröd et al. (2009) to select a better-conditioned basis set, $\mathcal{B}$, as described below.

Since excessive monomials are not in a basis set, they can be eliminated by transforming $C_{\mathcal{E}}$ into row echelon form using LU factorization, which gives

$$\begin{bmatrix} U_{\mathcal{E}1} & C_{\mathcal{R}1} & C_{\mathcal{P}1} \\ 0 & C_{\mathcal{R}2} & C_{\mathcal{P}2} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{E}} \\ \mathbf{x}_{\mathcal{R}} \\ \mathbf{x}_{\mathcal{P}} \end{bmatrix} = 0, \tag{2.31}$$

where $U_{\mathcal{E}1}$ is an upper triangular matrix. By expressing $\mathbf{x}_{\mathcal{E}}$ using $\mathbf{x}_{\mathcal{R}}$ and $\mathbf{x}_{\mathcal{P}}$, one can remove the top rows, which involve $\mathcal{E}$, and obtain a simplified system. Another LU factorization of the simplified system gives

$$\begin{bmatrix} U_{\mathcal{R}2} & C'_{\mathcal{P}2} \\ 0 & C_{\mathcal{P}3} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{R}} \\ \mathbf{x}_{\mathcal{P}} \end{bmatrix} = 0. \tag{2.32}$$

One now needs to select r basis monomials from $\mathcal{P}$. In Byröd et al. (2009), a column-pivoting QR is applied to $C_{\mathcal{P}3}$ to find a numerically stable basis set. Assume that permutation matrix Π is introduced and applied to $C_{\mathcal{P}3}$; this gives a reordering of $\mathbf{x}_{\mathcal{P}}$ and the last r monomials are selected as the basis monomials, that is $\mathbf{x}_{\mathcal{P}} = [\mathbf{x}_{\mathcal{P}'}\ \mathbf{x}_{\mathcal{B}}]$. This gives

$$\begin{bmatrix} U_{\mathcal{R}2} & C''_{\mathcal{P}2} & C_{\mathcal{B}2} \\ 0 & U_{\mathcal{P}3} & C_{\mathcal{B}3} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{R}} \\ \mathbf{x}_{\mathcal{P}'} \\ \mathbf{x}_{\mathcal{B}} \end{bmatrix} = 0. \tag{2.33}$$

Now both monomials in $\mathcal{R}$ and $\mathcal{P}'$ can be linearly expressed using monomials in $\mathcal{B}$ in the following form

$$\begin{bmatrix} \mathbf{x}_{\mathcal{R}} \\ \mathbf{x}'_{\mathcal{P}} \end{bmatrix} = - \begin{bmatrix} U_{\mathcal{R}2} & C''_{\mathcal{P}2} \\ 0 & U_{\mathcal{P}3} \end{bmatrix}^{-1} \begin{bmatrix} C_{\mathcal{B}2} \\ C_{\mathcal{B}3} \end{bmatrix} \mathbf{x}_{\mathcal{B}}, \tag{2.34}$$

from which the action matrix M_p can be easily extracted. Eigenvalue decomposition on M_p as in (2.28) will give us the values of the basis monomials evaluated at the zero set, from which the solution to the polynomial system H is estimated.

Chapter 3

Low-Rank Matrix Factorization: L_1-Norm and Truncated L_1-Norm

Low-rank matrix factorization problems have a wide range of applications even outside computer vision. One application is data representation and compression, where a large low-rank matrix can be represented as two smaller factor matrices. Another application is recommendation systems. In Koren et al. (2009), it is demonstrated that the matrix factorization method is superior to the classic nearest neighbor method for predicting users' ratings of movies in the well-known Netflix Prize competition (see Bennett and Lanning (2007)).

The following two chapters are devoted to the low-rank matrix factorization problems. We focus on applications mainly in geometric computer vision, for example, affine structure-from-motion, photometric stereo and linear shape basis estimation. State-of-the-art approaches are either based on alternating optimization using a bilinear formulation that depends on an initial solution, or based on minimizing a convex relaxation of a rank function, for example, the nuclear norm of a matrix. However, the performance of these methods either is affected by an increasing number of missing data or depends on initial solutions, which in many cases are non-trivial to find. In the following two chapters, we instead provide several novel insights into the problem, together with algorithms that (1) handle a large number of missing data, (2) are easily adapted to robust cost functions, and (3) require no initial solutions.

3.1 Introduction

Given an observation matrix $X \in \mathbb{R}^{m \times n}$, we are interested in finding a rank-r approximation $\hat{X}$ of X. This can be formulated as

$$\begin{aligned} \underset{\hat{X}}{\text{minimize}} \quad & \|X - \hat{X}\| \\ \text{subject to} \quad & \text{rank}(\hat{X}) = r. \end{aligned} \tag{3.1}$$

This is equivalently saying that the matrix $\hat{X}$ can be factorized into two matrices $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{r \times n}$, which gives the following equivalent formulation

$$\underset{U,V}{\text{minimize}} \quad \|X - UV\|. \tag{3.2}$$

Related works The matter of missing data in low-rank matrix factorization was originally addressed in Wiberg (1976), then under the L_2-norm. An algorithm independent of initialization was given in Jacobs (2001), but the method is highly sensitive to noise. Still, it is suitable as an initialization method if followed by an iterative, refinement technique. Similar approaches to the structure-from-motion problem are studied in Tardif et al. (2007); Kahl and Heyden (1999).

An early work that aims for robustness to outliers is Aanaes et al. (2002), which uses iteratively reweighted least squares to optimize a robust error function. A limitation is that the method requires a good initial solution, which is often difficult to obtain. The theory of robust subspace learning is further developed in Torre and Black (2003). In Buchanan and Fitzgibbon (2005), a damped Newton method is proposed to solve the problem of missing data. In Bue et al. (2012), a bilinear model is formulated under the L_2-norm with the constraint that the factor matrices should lie in a certain manifold. The model is solved using the augmented Lagrange multiplier method. In Ke and Kanade (2005), alternating optimization is proposed for both the Huber norm and the L_1 norm. Yet another iterative approach proposed in Eriksson and Hengel (2012) can be seen as extending the method of Wiberg (1976), except for the L_1 norm. This approach has been further generalized in Strelow (2012) to handle the projective structure-from-motion problem. In Okatani et al. (2011), a damping factor is incorporated into the Wiberg method. It is also experimentally

demonstrated in Okatani et al. (2011) that Newton-family minimization techniques using a damping factor lead to excellent global convergence performance. The method presented in Zheng et al. (2012a) first solves the affine factorization in the L_2 norm by adding an extra mean vector to the formulation. Another recent algorithm, presented in Zheng et al. (2012b), adds orthogonal constraints to columns of U and a nuclear norm regularizer to V; used with the augmented Lagrangian multiplier method, it has achieved rapid convergence. All of these algorithms are based on local optimization, and hence risk becoming stuck in local minima. The cost function may indeed exhibit several local optima, as exemplified in Figure 3.3. One notable attempt to solve the problem in a globally optimal way is proposed in Chandraker and Kriegman (2008), which uses a branch and bound method and proves that the globally optimal solution is obtained. However, in practice, the method is restricted to simple problems for which the number of variables in either U or V is very small; for example, there are only nine variables in U in one of the experiments in Chandraker and Kriegman (2008).

Alternative approaches to tackling the low-rank factorization or low-rank approximation problems include minimizing a convex surrogate of the rank function, for example, the nuclear norm. In Candès and Recht (2008), the solution turns out to be very pleasing, as only a convex optimization problem needs to be solved. The nuclear norm formulation entails solving an SDP, which the methods in Candès et al. (2009); Lin et al. (2009) try to do efficiently. These approaches can handle application problems when the rank is not known a priori, for example, segmentation in Cheng et al. (2011), background modeling in Candès et al. (2009), and tracking in Xiong et al. (2012). However, when applied to problems with known rank, the performance of the methods based on the nuclear norm formulation is typically worse than that of methods based on the bilinear formulation (see Cabral et al. 2013). These methods assume that the missing data are sparse and that the locations of missing data are random. However, for many applications these assumptions are generally not fulfilled. For example, in the structure-from-motion problem, the missing data are neither sparse nor randomly located, but rather distributed densely in the off-diagonal chunks. In Olsson and Oskarsson (2011), it is also noted that the convex factorization approaches may break down due to violation of the sparsity assumption in structure from motion.

3.2 Problem Formulation

If we consider the problem in (3.2) under the L_2-norm and assume X is a complete matrix, then the problem is solved optimally by computing a Singular Value Decomposition (SVD) of X. However, in practice, it usually contains missing data and outliers in X, in which case, some robust norms, for example, L_1-norm or the truncated L_1-norm are more adequate choices.

L_1-Norm The L_1-norm of a matrix is defined as follows

$$\|X\|_1 = \sum_{ij} |x_{ij}|, \tag{3.3}$$

Note that the norms used in the thesis, if not explicitly specified, are all referred to the "entry-wise" norm. It is different from the "induced" p-norm defined using the vector norm which is

$$\|X\|_p = \sup_{\substack{\mathbf{y}\in\mathbb{R}^n \\ \mathbf{y}\neq 0}} \frac{\|X\mathbf{y}\|_p}{\|\mathbf{y}\|_p}, \tag{3.4}$$

where the norm on the right-hand side of equation (3.4) is the vector p-norm. Under the L_1-norm, the problem in (3.2) can simply be stated as

$$\underset{U,V}{\text{minimize}} \quad \sum_{i,j} |x_{ij} - \sum_k u_{ik} v_{kj}|, \tag{3.5}$$

Note in presence of missing data, the cost function in (3.5) should only be summed over the indices (i, j) that have measured values in X. We make no requirement that the full observation matrix is available.

Truncated L_1-Norm Using L_1-norm, the penalty on the measurements grows linearly, instead of quadratically with L_2-norm. A more robust way is to assign a bounded penalty to the measurements, which introduce the truncated L_1-norm defined as

$$\|X\|_{1*} = \sum_{ij} \min(|x_{ij}|,\ \epsilon), \tag{3.6}$$

where ϵ is a given threshold for truncation. Under the truncated L_1-norm, the formulation is slightly modified as follows

$$\underset{U,V}{\text{minimize}} \quad \sum_{i,j} \min(|x_{ij} - \sum_k u_{ik} v_{kj}|, \epsilon). \tag{3.7}$$

The maximum cost for an outlier measurement is ϵ under this model.

Subspace estimation To shed further light on the factorization problem, one can view it as estimation of a low-dimensional subspace. Given data $\mathbf{x}_i \in \mathbb{R}^m$, $i = 1, \cdots, n$, the problem in (3.5) can be treated as that of finding an optimal subspace

$$S = \{\mathbf{x} \in \mathbb{R}^m | \mathbf{x} = U\mathbf{v}, \mathbf{v} \in \mathbb{R}^r\}, \tag{3.8}$$

defined as a matrix $U \in \mathbb{R}^{m \times r}$ such that when all the data is projected onto S, the sum of projection error $\sum_i \|\mathbf{x}_i - U\mathbf{v}_i\|$ is minimized. With this formulation, we always get a linear subspace containing the origin. In many applications though, one is interested in finding an affine subspace

$$S = \{\mathbf{x} \in \mathbb{R}^m | \mathbf{x} = U\mathbf{v} + \mathbf{t}, \mathbf{v} \in \mathbb{R}^r, \mathbf{t} \in \mathbb{R}^m\}, \tag{3.9}$$

which is defined by $U \in \mathbb{R}^{m \times r}$ and $\mathbf{t} \in \mathbb{R}^m$. For observations with no missing entries or outliers, the translational component $\mathbf{t}$ is optimally estimated as the mean of the observation vector $\mathbf{x}_i$ under the L_2-norm. This is clearly not a good estimator in the presence of missing data or outliers or under the L_1-norm. In analogy to the formulation in (3.5), the affine subspace problem can be formulated as

$$\underset{U,V,t}{\text{minimize}} \sum_{i,j} |x_{ij} - t_i - \sum_{k=1}^{r} u_{ik} v_{kj}|, \tag{3.10}$$

or in matrix notation

$$\underset{U,V,\mathbf{t}}{\text{minimize}} \left\| X - \begin{bmatrix} U & \mathbf{t} \end{bmatrix} \begin{bmatrix} V \\ 1 \cdots 1 \end{bmatrix} \right\|, \tag{3.11}$$

where $X \in \mathbb{R}^{m \times n}$, $U \in \mathbb{R}^{m \times r}$, $V \in \mathbb{R}^{r \times n}$ and $\mathbf{t} \in \mathbb{R}^m$.

The residual matrix $R \in \mathbb{R}^{m \times n}$, which will be used later, is defined here as

$$R = \left| X - \begin{bmatrix} U & \mathbf{t} \end{bmatrix} \begin{bmatrix} V \\ 1 \dots 1 \end{bmatrix} \right|. \tag{3.12}$$

In summary, we are considering two different cost functions for the factorization problem, one based on the L_1-norm (3.5) and one based on the truncated L_1-norm (3.7), as well as two different versions, one viewed as a linear subspace estimation problem (3.8) and one viewed as an affine subspace problem (3.9).

3.3 L_1-Projections

In this section, we will give some general results concerning the L_1-projections.

Theorem 3.1. *For a given point $\mathbf{x} \in \mathbb{R}^m$ and a given r-dimensional affine subspace S defined by a matrix $U \in \mathbb{R}^{m \times r}$ and $\mathbf{t} \in \mathbb{R}^m$, the L_1-projection of $\mathbf{x}$ onto S occurs only along $m - r$ directions.*

This is equivalently saying that the remaining r directions are error free, that is, r components of the residual vector $\mathbf{x} - U\mathbf{v}$ are always zero. To understand the theorem, we illustrate two examples in 2D and 3D space in Figure 3.1. In the left figure, a 2D point $\mathbf{x}$ is projected onto a one-dimensional subspace, that is, a line ($m = 2, r = 1$). We can see that the L_1-ball of $\mathbf{x}$ meets the line on one of its vertices $\mathbf{x}_p$, which means the projection only occur along one directions, that is, y-axis in this case. The right figure shows a 3D case ($m = 3, r = 1$). The L_1-ball of x usually intersects the line on one of its edges, which indicates that the projection occurs along two directions, that is, the y and z-axis in this case.

The above result is well-known in Mangasarian (1997); Brooks and Dulá (2013). It can be formally proved using linear programming theory. However, it should intuitively be clear that the theorem is true. Writing the cost function explicitly, we have

$$\min_{v} \sum_{i=1}^{m} |x_i - t_i - \sum_{k=1}^{r} u_{ik} v_k|, \tag{3.13}$$

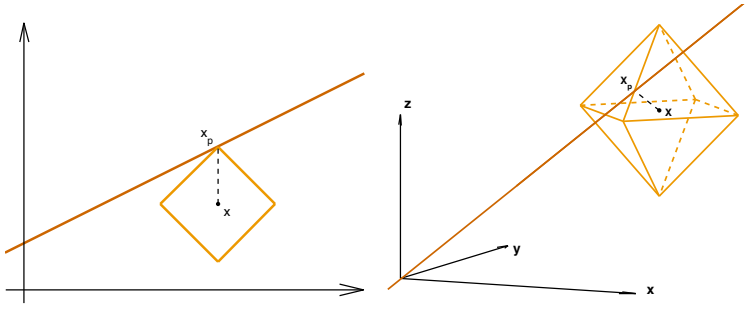


Figure 3.1: L_1-projection in 2D and 3D.

from which we see that it is a piecewise linear function of the v_k's. Furthermore, as the column vectors $\mathbf{u}_k$ for $k = 1, \ldots, r$ form a basis for an r-dimensional subspace they are all linearly independent. Hence the cost tends to infinity as $u \to \infty$ and the minimum must be attained at a corner point, that is, where the derivative is not defined in any direction. So, at least r elements are zero in the residual vector at optimum.

Assume, for a while, that we know the positions of the r zeros of Theorem 3.1. Since each zero gives a linear constraint on $\mathbf{v}$ we could easily compute the L_1-projection from this information. And even if the zero positions are unknown, this technique can be useful if an exhaustive search over the possible positions is performed[1]. A natural question is whether a similar approach can be used to solve the full problem.

3.4 Hyperplane Fitting

If the dimension of the subspace $r = m - 1$ then we are dealing with a hyperplane. According to Theorem 3.1, the projection of a given point $\mathbf{x} \in \mathbb{R}^m$ onto a hyperplane occurs along a single direction. Moreover, this direction depends only on the hyperplane - not on the point $\mathbf{x}$. This result is a direct consequence of Theorem 2.1 in Mangasarian (1997), but for clarity, we state it as a theorem.

Theorem 3.2. *Given a set of points* $\mathbf{x}_k \in \mathbb{R}^m$ *and an* $(m-1)$*-dimensional*

[1] This is not the most efficient way of computing an L_1-projection.

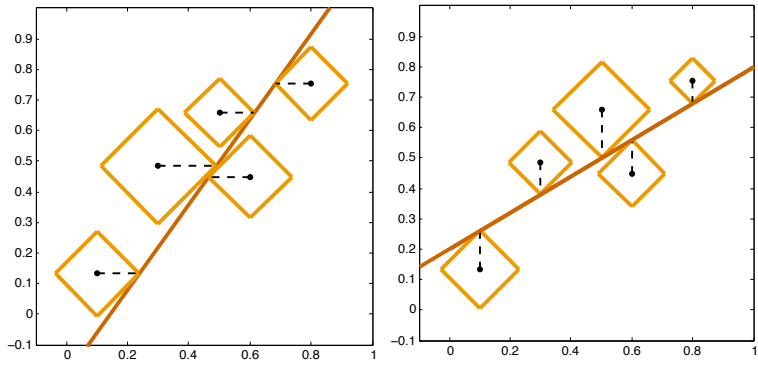


Figure 3.2: Hyperplane fitting in 2D.

affine subspace S, there exist optimal L_1-projections of $\mathbf{x}_k$ onto S such that all occur along a single axis.

We illustrate the hyperplane fitting in 2D ($m = 2$) in the Figure 3.2. It is obviously seen that the L_1-projection of all points occur along a single direction, either all along x-axis or all along the y-axis. The direction depends on the one-dimensional subspace. More specifically, it depends on the slope k of the line in the form of $y = kx + l$ in this case.

If we know this axis, then we can solve for the hyperplane using linear programming (LP). Hence optimal hyperplane fitting can be solved as a series of m LP problems. Another option is indicated by the following theorem.

Theorem 3.3. *For an optimal affine hyperplane, there will be $m - 1$ rows of zeros and one row with m zero elements in the residual matrix R in* (3.12)*. Provided we know the positions of these zeros, the hyperplane can be solved for in closed-form.*

Proof. According to Theorem 3.2, all the points will be projected along a single direction. This means that the residual matrix R will have $m - 1$ rows of zeros. Without loss of generality, we assume the top $m - 1$ rows of R are zeros, which gives a partition of $R = [\mathbf{0}, \hat{\mathbf{r}}]^{\mathrm{T}}$ where $\hat{\mathbf{r}}^{\mathrm{T}}$ is a row vector.

Applying the same partition to X, U and $\mathbf{t}$ leads to the following equation

$$\begin{bmatrix} \mathbf{0} \\ \hat{\mathbf{r}} \end{bmatrix} = \begin{bmatrix} \tilde{X} \\ \hat{\mathbf{x}} \end{bmatrix} - \begin{bmatrix} \tilde{U} & \tilde{\mathbf{t}} \\ \hat{\mathbf{u}} & \hat{t} \end{bmatrix} \begin{bmatrix} V \\ 1 \dots 1 \end{bmatrix}. \tag{3.14}$$

Note the partition of $R \in \mathbb{R}^{m \times n}$ yields a zero matrix $\mathbf{0} \in \mathbb{R}^{(m-1) \times n}$ and a row vector $\hat{\mathbf{r}} \in \mathbb{R}^n$.

There exists a coordinate ambiguity in the factorization as we can always reparametrize $[U, \mathbf{t}]$ using a matrix $Q = \begin{bmatrix} \tilde{Q} & \tilde{\mathbf{q}} \\ \mathbf{0} & 1 \end{bmatrix}$ since we have

$$\begin{bmatrix} U & \mathbf{t} \end{bmatrix} \begin{bmatrix} V \\ 1 \dots 1 \end{bmatrix} = \begin{bmatrix} U & \mathbf{t} \end{bmatrix} Q Q^{-1} \begin{bmatrix} V \\ 1 \dots 1 \end{bmatrix}, \tag{3.15}$$

which means we can always reparametrize (3.14) such that $\tilde{U} = I$ and $\tilde{t} = \mathbf{0}$. The reparametrization gives the solution $V = \tilde{X}$, that is,

$$\begin{bmatrix} \mathbf{0} \\ \hat{\mathbf{r}} \end{bmatrix} = \begin{bmatrix} \tilde{X} \\ \hat{\mathbf{x}} \end{bmatrix} - \begin{bmatrix} I & \mathbf{0} \\ \bar{\mathbf{u}} & \bar{t} \end{bmatrix} \begin{bmatrix} \tilde{X} \\ 1 \dots 1 \end{bmatrix}. \tag{3.16}$$

The remaining cost $\|\hat{\mathbf{r}}\|_1$ is now a function of $\bar{\mathbf{u}}$ and $\bar{t}$ which is piecewise linear. If the columns of $\tilde{X}$ span $\mathbb{R}^m$ then the cost tends to infinity as $\| \begin{bmatrix} \bar{\mathbf{u}} & \bar{t} \end{bmatrix} \|_1 \to \infty$. Hence the piecewise linear cost $\|\hat{\mathbf{r}}\|_1$ attains its minimum at a corner point with m zeros in the residual vector and if we know the zero positions, then we can estimate the unknowns in $\bar{\mathbf{u}}$ and $\bar{t}$ by solving linear equations.

If on the other hand the columns of $\tilde{X}$ do not span $\mathbb{R}^m$, then the complete data matrix X has to lie in a subspace of $\mathbb{R}^m$. So, for the optimal hyperplane, all residuals are zero. Using m of these we can compute an optimal hyperplane.

□

As an example, consider the case of line fitting to a set of points $\{\mathbf{x}_i, i = 1, \dots, n\}$ in $\mathbb{R}^2$. For an optimal line, the residual matrix $R \in \mathbb{R}^{2 \times n}$ has a full row of zeros in either x or y coordinates.

Note that the final estimation of $\bar{\mathbf{u}}$ and $\bar{t}$ could be solved more efficiently using LP, see Algorithm 1. However, this does not generalize to truncated L_1-norm. In that case, one needs to do an exhaustive search based on Theorem 3.3 or a random search as will be described later.

Algorithm 1 Optimal hyperplane fitting (HF)

Given an observation matrix X, solve for the optimal affine subspace $(U^*, \mathbf{t}^*)$ and the projection matrix V^*

1. Initialize the best error $\epsilon^* = \infty$
2. For $i = 1$ to m
3. Set the index set $\mathcal{P}$ for row partition as $\mathcal{P} = \{1, 2, \ldots, m\} \setminus \{i\}$
5. Let $\tilde{X} = X_{\mathcal{P}}$ and $\hat{\mathbf{x}} = X_{\{i\}}$ in (3.16)
6. Solve $\min_{\bar{\mathbf{u}}, \bar{t}} \|\hat{\mathbf{r}}\|_1$ in (3.16) using LP
7. Calculate the L_1-error ϵ
8. If $\epsilon < \epsilon^*$
9. $U^* = U, \mathbf{t}^* = \mathbf{t}, V^* = V$ and $\epsilon^* = \epsilon$
10. return $U^*, \mathbf{t}^*, V^*, \epsilon^*$

3.5 The General Case

A linear subspace defined by $U \in \mathbb{R}^{m \times r}$ has $d = (m - r)r$ degrees of freedom (mr parameters defined up to an $r \times r$ coordinate transformation). Similarly, an affine subspace defined by $U \in \mathbb{R}^{m \times r}$ and $\mathbf{t} \in \mathbb{R}^m$ has $d = (m - r)(r + 1)$ degrees of freedom. One can see that by fixing the gauge with a reparametrization in U and $\mathbf{t}$ as $U = \begin{bmatrix} I \\ \bar{U} \end{bmatrix}$ and $\mathbf{t} = \begin{bmatrix} \mathbf{0} \\ \bar{\mathbf{t}} \end{bmatrix}$, where $\bar{U} \in \mathbb{R}^{(m-r) \times r}$ and $\bar{\mathbf{t}} \in \mathbb{R}^{m-r}$, see (3.15). For example, when $r = m - 1$ as in the previous section, there are only m degrees of freedom of the affine subspace, and these m unknowns can be determined in closed-form from the m extra zeros in the residual matrix, see Theorem 3.3.

In the general case ($r < m-1$), there may be fewer zeros in the residual matrix than necessary to solve directly for the subspace. Moreover, even with sufficiently many zeros, the structure of the residual matrix might not allow us to linearly solve for the parameters. Despite these facts, similar ideas can be used to achieve state-of-the-art results and very often to find an optimal L_1-factorization. The basis will be the following type of points:

Definition 3.4. *A point $(U, \mathbf{t})$ representing an affine subspace in parameter space is a* principal stationary point *if the residual matrix has d extra zeros for the optimal V.*

By *extra* here is meant the additional zeros to the r zeros present in

every column of the residual matrix according to Theorem 3.1. Note that when $r = m - 1$, then there are always $d = m$ extra zeros and hence all optimal subspaces U^* to the L_1-factorization problem are principal stationary points (Theorem 3.3).

Empirically, we have made the following two observations concerning L_1-optimal factorizations:

- In practice, the optimal subspace for L_1-factorization is often a principal stationary point.

- Even if the optimal subspace is not a principal stationary point, there is often a principal stationary point which is close to the optimal one.

How common are principal stationary points? To give some insight into this question we considered a low-dimensional problem in order for brute-force search to be applicable. More precisely, we considered fitting of a r-dimensional subspace in $\mathbb{R}^m$.

To generate the data, we first randomly generate r orthonormal basis $\mathbf{u}_i$ for $i = 1, 2, \cdots, r$ in $\mathbb{R}^m$, which constitutes the columns of ground truth subspace U. The random data $\mathbf{x}_j$ in the subspace are generated using a linear combination of the basis, $\mathbf{x}_j = \sum_{i=1}^{j} a_i \mathbf{u}_i$ where the coefficients a_i are uniformly drawn from $[-1, 1]$. Gaussian noise from $\mathcal{N}(0, 0.02)$ are added to all the points. And 10% data are regarded as outliers by a random perturbation uniformly drawn from $[-1, 1]$.

Grid search is performed in the parameter space of U. We fix the gauge by setting the top r-by-r block of U to be an identity matrix I, and search for the remaining $(m - r)r$ variables of U. Since the ground truth of elements of U is generated between $[-1, 1]$. We perform the grid search in the slightly larger range of $[-2, 2]$ by dividing it into equally-sized intervals, with the length of each interval being 0.1. We estimate V by L_1-projection and refining the best solution using the method from Eriksson and Hengel (2012). The number of zeros in the residual matrix of the solution was counted to evaluate whether it was a principal stationary point or not.

Due to the exponentially increasing cost for the grid search with the number of variables in U, we only consider the test on low-dimensional problems. More specifically, we estimated the subspace with dimension $r = 1, 2, 3$ in the $\mathbb{R}^m$ space with varied m from 3 to 5. We skipped the hyperplane-fitting case ($r = m - 1$) here. We run 2000 random problems

	m=3	m=4	m=5
r=1	98.4%	96.6%	96.0%
r=2	-	92.0%	94.0%
r=3	-	-	95.0%

Table 3.1: Percentage of principal stationary points being optimal.

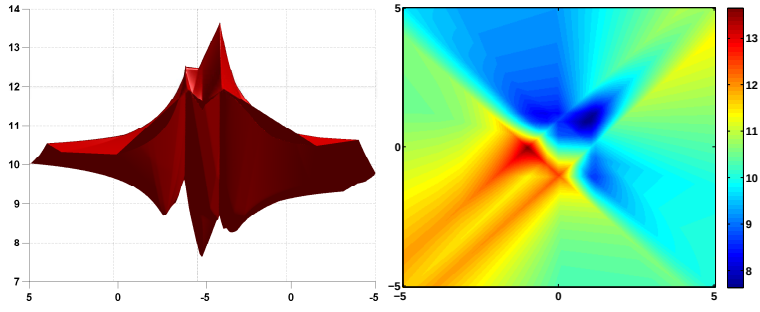


Figure 3.3: The L_1-cost for fitting a 1D-subspace to a set of points in $\mathbb{R}^3$ ($m = 3$ and $r = 1$). The cost function is varied over two dimensions of U and then the optimal solution is computed for the other variables. Note that there are at least three local minima.

for $r = 1$ cases and 200 random problems for $r = 2, 3$ cases due to the exponentially increasing time complexity. The percentage of the principal stationary point being the global optimal is summarized in Table 3.1

From Table 3.1, we observe that for more than 90% of random problems we tested on estimating the different low-dimensional subspaces, the principal stationary points are the globally optimal solutions. This observation motivates the following algorithms that consider only principal stationary points. Note the cost function for one example of the problem $m = 3, r = 1$ is plotted in Figure 3.3.

3.5.1 Searching for Principal Points

Our approach to general subspace fitting is based on searching for principal stationary points, that is, points that have d extra zeros in the residual matrix, allowing us to solve directly for the parameters. This suggests that

to estimate the subspace U we only need to consider a subset of columns with d extra zeros in the residual matrix. Once the subspace U is estimated, the projection coefficient V for the remaining columns of X can be solved either by a linear program or with the approach discussed in Section 3.3.

We first focus on estimating a subspace U. A zero at position (i, j) in the residual matrix gives a bilinear equation in the unknowns t_i, u_{ik} and v_{kj} as

$$x_{ij} - t_i - \sum_{k=1}^{r} u_{ik} v_{kj} = 0. \tag{3.17}$$

By Theorem 3.1, every column yields at least r such equations, but looking for principal stationary points we can assume that there are d extra zeros corresponding to the d degrees of freedom of the subspace. Hence we consider a subset of at most d columns and assume that there are $dr + d$ zeros in the corresponding residual matrix. For small problems an exhaustive search over the possible positions of these zeros might be tractable, but to handle larger problems, a randomized algorithm is necessary.

In principle, this approach can be viewed as applying RANSAC in Fischler and Bolles (1981) to low-rank matrix factorization, although our motivation was quite different. Just as in RANSAC a minimal set of data points are assumed to have zero errors and this assumption is used to find the model parameters, and then, the obtained parameters are evaluated on all data to measure the goodness of fit. Either we repeat this exhaustively for every possible minimal subset or for a fixed number of random subsets. More on this later.

3.5.2 Exhaustive Search

For small-sized problems it is tractable to search the space of all possible positions for the $d + dr$ zeros of a principal stationary point. For each possible zero pattern, we need to solve a set of $d + dr$ degree-2 polynomial equations, which can be expensive and might yield up to 2^{d+dr} solutions. Fortunately, the structure of these polynomial equations, for example, all the quadratic terms are bilinear, yields much fewer solutions. In fact, for low-dimensional problems, such as line fitting in $\mathbb{R}^3$, there is a unique and simple closed form solution, rendering a much more efficient algorithm.

Algorithm 2 Exhaustive Search (ES)

Given an observation matrix X, solve for the optimal affine subspace $(U^*, \mathbf{t}^*)$ and the projection matrix V^*.

1. Initialize the best error $\epsilon^* = \infty$
2. Generate all the column subsets $\{\mathcal{I}_i\}$ of size d
3. Generate all the residual patterns $\{R_j\}$ of size $\mathbb{R}^{m \times d}$
4. For each column subset $\mathcal{I}_i$
5. For each residual pattern R_j.
6. Compute U, $\mathbf{t}$ and $V_{\mathcal{I}_i}$ in closed form using $X_{\mathcal{I}_i}$, R_j
7. Compute the projection $V_{\{1,2,\ldots,n\}\setminus\mathcal{I}_i}$
8. Compute the L_1-error ϵ
9. If $\epsilon < \epsilon^*$
10. $U^* = U, \mathbf{t}^* = \mathbf{t}, V^* = V$ and $\epsilon^* = \epsilon$
11. return $U^*, \mathbf{t}^*, V^*, \epsilon^*$

Note that when generating the residual patterns, one should first make sure each column has at least r zeros, which follows from Theorem 3.1. Then d extra zeros should be arranged such that each row has at least r zeros, which followed by applying Theorem 3.1 to the transpose of the measurement matrix X. This makes sure that each row of U can be determined from the equation system.

3.5.3 Random Search

The exhaustive search algorithm quickly becomes infeasible with growing problem size, both because the number of possible zero patterns grows exponentially and because solving the system of quadratic equations gets increasingly more expensive, as for general d and r, there is no simple closed-form solution.

Simple patterns of zeros To work around these problems, we propose a random search algorithm only considering especially simple residual patterns. More precisely, we restrict the search to patterns that lead to linear equations and can hence be solved extremely fast. Note that even these simple patterns abound and in practice, we can always find such patterns despite missing entries as long as the factorization problem is well-posed.

Here we describe a family of general residual pattern that can be solved linearly. The motivation is that this family of residual pattern enables one to linearly solve the rows $\mathbf{u}_i^{\mathrm{T}}$ of U and the columns $\mathbf{v}_j$ of V in an alternating and iterative manner. Since both $\mathbf{u}_i^{\mathrm{T}}$ and $\mathbf{v}_j$ contains r unknowns, it generally require r zeros in the corresponding row or column of the residual matrix to solve for $\mathbf{u}_i^{\mathrm{T}}$ or $\mathbf{v}_j$.

The first assumption is that (possibly after some permutations on rows and columns) we have a $(r+1) \times r$ zeros in the top left of the residual matrix. Based on the degree of freedom in the subspace U, a coordinate system can always be chosen such that the first r rows of U become an identity matrix, for example the top r rows, that is

$$U = \begin{bmatrix} I_{r\times r} \\ \mathbf{u}_{r+1}^{\mathrm{T}} \\ \mathbf{u}_{r+2}^{\mathrm{T}} \\ \cdots \\ \mathbf{u}_{m}^{\mathrm{T}} \end{bmatrix}. \tag{3.18}$$

Now, from the definition of the residual matrix R in (3.12), we can immediately obtain the first r columns of V from

$$\tilde{X} - I \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_r \end{bmatrix} = \mathbf{0}, \tag{3.19}$$

where $\tilde{X}$ is the top left $r \times r$ sub-matrix of X. Then the last row of the $(r+1) \times r$ zeros in R gives us r linear constraints on $\mathbf{u}_{r+1}^{\mathrm{T}}$

$$\mathbf{x}_{r+1}^{\mathrm{T}} - \mathbf{u}_{r+1}^{\mathrm{T}} \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_r \end{bmatrix} = \mathbf{0}, \tag{3.20}$$

where $\mathbf{x}_{r+1}^{\mathrm{T}}$ are the elements of X corresponding to the last row of the zero block in R. As $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_r$ are already known from (3.19), we can compute $\mathbf{u}_{r+1}^{\mathrm{T}}$ linearly from (3.20). (We have r linear equations and r unknowns in $\mathbf{u}_{r+1}^{\mathrm{T}}$.)

To solve for another row of U, we need at least r zeros in that row of R. They can be either in new columns or in columns already used to compute

$\mathbf{u}_{r+1}^{\mathrm{T}}$. Let us assume that after permutations, the zeros are in columns $r+1$ to $2r$.[2] This yields

$$\mathbf{x}_{r+2}^{\mathrm{T}} - \mathbf{u}_{r+2}^{\mathrm{T}} \begin{bmatrix} \mathbf{v}_{r+1} & \mathbf{v}_{r+2} & \dots & \mathbf{v}_{2r} \end{bmatrix} = \mathbf{0}. \tag{3.21}$$

However, since the $\mathbf{v}_{r+1}, \dots, \mathbf{v}_{2r}$ are unknown, we first need to compute them. We can use the fact that the $\mathbf{u}_1^{\mathrm{T}}$, $\mathbf{u}_2^{\mathrm{T}}$, ... $\mathbf{u}_{r+1}^{\mathrm{T}}$ are all known. This means for the column $\mathbf{v}_i \in \mathbb{R}^r$ of V to be solvable, we need at least r zeros in the top $r+1$ rows of i^{th} column of R. Take solving $\mathbf{v}_{r+1}$ for example, assume $n_1, n_2, ..., n_r$ are the indices of rows, where those r zeros locate in the $(r+1)^{th}$ column of R. Taking the corresponding elements of X, that is, the rows $n_1, n_2, ..., n_r$ of the column $r+1$, we form a vector $\tilde{\mathbf{x}}$. Taking the corresponding rows of U, that is, the rows $n_1, n_2, ..., n_r$ of U, we form a sub-matrix $\tilde{U}$ of size $r \times r$. Then $\mathbf{v}_{r+1}$ is computed from

$$\tilde{\mathbf{x}} - \tilde{U}\mathbf{v}_{r+1} = \mathbf{0}. \tag{3.22}$$

Note this is similar to (3.19) except we have to solve the column of V separately as the zero position for each column might be varied now. Other columns, $\mathbf{v}_{r+2}, \dots, \mathbf{v}_{2r}$ are solved in the same way. With $\mathbf{v}_{r+1}, \dots, \mathbf{v}_{2r}$ solved, we can compute $\mathbf{u}_{r+2}^{\mathrm{T}}$ using (3.21).

So to solve $\mathbf{u}_{r+2}^{\mathrm{T}}$, we require a block of size $(r+2) \times r$, inside which the top $r+1$ rows contain at least r zeros in each column, and the last row contains only zeros. Note that apart from this the positions of the non-zero residuals in this block are arbitrary. One example for $r = 3$ is

$$R = \begin{bmatrix} 0 & 0 & 0 & \cdot & 0 & 0 & ... \\ 0 & 0 & 0 & 0 & \cdot & 0 & ... \\ 0 & 0 & 0 & 0 & 0 & 0 & ... \\ 0 & 0 & 0 & 0 & 0 & \cdot & ... \\ \cdot & \cdot & \cdot & 0 & 0 & 0 & ... \\ \dots & \dots & \dots & \dots & \dots & \dots & ... \end{bmatrix}. \tag{3.23}$$

The remaining rows $\mathbf{u}^{\mathrm{T}}$ of U can be solved sequentially in the same way. It is worth noting that in general, this will not compute all columns in V but only up to d of them. We will use $\mathcal{J}$ to denote the columns which

[2] If one or more are in the first r columns it only makes things easier as v_1 to v_r are already known.

are computed directly in this way. The remaining columns of V can be found using L_1-projections since U is already known.

This also leads to a simple strategy to handle missing data. When a random residual pattern is generated in Algorithm 3, we simply restrict it such that the zeros in R cannot be placed in a position corresponding to a missing element of X. Figure 3.4 shows an example residual pattern generated randomly for a structure from motion dataset. The zero positions are close to the main diagonal since that is where we have observed data.

Figure 3.4: A random generated residual pattern for $m = 20$ and $r = 3$ case. Each black patch is a zero in R, also corresponds to the sampled observation in X that used to solve for U.

Adaptive sampling To generate the residual pattern described above, we sample $d + dr$ elements from observation X, corresponding to the zeros in R. As noted earlier, this is basically the RANSAC approach to matrix factorization although motivated in a completely different way. Consequently, we can use any of the alternative sampling strategies that have been proposed for RANSAC. For example, in Tordoff and Murray (2002) it proposed an algorithm called guided-MLESAC for maximum likelihood estimation by RANSAC, which estimates the inlier probability of each match based on the proximity of matched features. In Chum and Matas (2005), it proposed PROSAC to measure the similarities of correspondences, and used sequential thresholding to form a sequence of progressively larger set of top-ranked correspondences. It is based on the mild assumption that correspondences with high similarity are more likely to be inliers. We chose the following PROSAC-like sampling strategy.

We initialize the probability of sampling x_{ij} of X to be $p_{ij} > 0$ if x_{ij} is observable, otherwise, we set $p_{ij} = 0$. In each iteration when a better solution is found, we check the residual $r_{ij} = |x_{ij} - \mathbf{u}_i^{\mathrm{T}}\mathbf{v}_j|$, if r_{ij} is large,

then we lower the probability p_{ij} of picking up x_{ij} in the next iteration, otherwise we increase p_{ij}. In practice, the strategy helps us to find a better solution using fewer sampling steps.

Algorithm 3 Random Search (RS)

Given an observation matrix X and a max iterations N, solve for the optimal affine subspace $(U^*, \mathbf{t}^*)$ and the projection matrix V^*.

1. Initialize the best error $\epsilon^* = \infty$
2. Initialize the probabily $p_{ij} = 1$ for $i = 1, \ldots, m, j = 1, \ldots, n$
3. While $k \leq N$
4. Randomly generate a simple residual pattern s.t. element (i, j) is included with probability $\approx p_{ij}$
5. Compute $U, \mathbf{t}$ and $V_{\mathcal{J}}$ linearly as described in text
6. Compute $V_{\{1,2,\ldots,n\}\setminus\mathcal{J}}$ using L_1-projection
7. Compute the L_1-error ϵ^i
8. If $\epsilon^k < \epsilon^{k-1}$
9. Update the probability p_{ij} based on $r_{ij} = |w_{ij} - \mathbf{u}_i^{\mathrm{T}}\mathbf{v}_j|$.
10. If $\epsilon^k < \epsilon^*$
11. $U^* = U, \mathbf{t}^* = \mathbf{t}, V^* = V$ and $\epsilon^* = \epsilon$
12. return $U^*, \mathbf{t}^*, V^*, \epsilon^*$

3.6 Truncated L_1-Factorization

Perhaps somewhat surprisingly, most of the results we have presented generalize easily to the truncated L_1-norm in (3.7). A brief sketch of the proof is as follows. Consider an optimal factorization with respect to the truncated L_1-norm. Now divide the measurements into inliers — having a residual smaller than ϵ — and outliers — having a residual larger than ϵ. Now apply Theorems 3.2 and 3.3 to the inliers only.

Algorithms 2 and 3 (but not Algorithm 1) can be used to optimize the truncated L_1-norm. The only required modification is to evaluate solutions using the *truncated* L_1-norm rather than the standard L_1-norm.

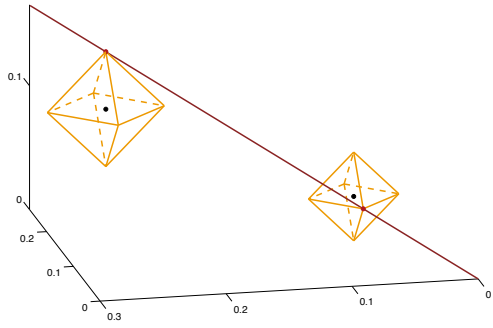


Figure 3.5: Line fitting in 3D. We know from Theorem 3.1 that L_1-projection of a point usually occur along $m - r = 2$ directions. However, as the line $\mathbf{u}$ has $d = (m - r)r = 2$ degrees of freedom, there are two columns in the residual matrix has an extra zero. So typically there are two points whose L_1-projection only occurs in a single direction as plotted. (The L_1-ball for other points are omitted)

3.7 Applications

3.7.1 Line-fitting

To view a line-fitting as a matrix factorization problem, let $\mathbf{x}_i \in \mathbb{R}^m$, $i = 1, \ldots, n$ be observed points in the plane ($m = 2$) or in space ($m = 3$). Then, a line through the origin can be parametrized by a direction vector $\mathbf{u} \in \mathbb{R}^m$. For each point there should be a parameter $v_i \in \mathbb{R}$ satisfying $\mathbf{x}_i \approx \mathbf{u}v_i$. In order to estimate the line parameters, one can solve the factorization problem

$$\min_{\mathbf{u},\mathbf{v}} \left\| \begin{bmatrix} \mathbf{x}_1 & \ldots & \mathbf{x}_n \end{bmatrix} - \mathbf{u} \begin{bmatrix} v_1 & \ldots & v_n \end{bmatrix} \right\|_1 . \tag{3.24}$$

For a line not necessarily going through the origin, it can be regarded as an affine subspace problem

$$\min_{\mathbf{u},\mathbf{v},\mathbf{t}} \left\| \begin{bmatrix} \mathbf{x}_1 & \ldots & \mathbf{x}_n \end{bmatrix} - \begin{bmatrix} \mathbf{u} & \mathbf{t} \end{bmatrix} \begin{bmatrix} v_1 & \ldots & v_n \\ 1 & \ldots & 1 \end{bmatrix} \right\|_1 . \tag{3.25}$$

See also Figure 3.5.

We first test our exhaustive search method for affine line fitting in $\mathbb{R}^3$. The purpose of this experiment is to investigate the local minima problem. More quantitative results are given in the following experiments. In this case, all the principal stationary points can be solved for in closed form. For each experiment, 20 3D points with coordinates in $[-1, 1]$ are generated on a line and perturbed with Gaussian noise with standard deviation 0.1. In addition, we perturb 80% of the points with uniform noise in $[-1, 1]$ to be outliers. 100 random examples are tested using both our exhaustive search algorithm with L_1-norm and L_1-Wiberg from Eriksson and Hengel (2012).

From Figure 3.6, we can see that our method performs better as a fairly large portion of errors (brown) fall into the interval between 0 and 0.1 while most errors for L_1-Wiberg (grey) lie between 0.1 and 0.2. In every single instance, our algorithm performs better (around 50% of the cases) or equally well compared to the L_1-Wiberg algorithm. This means that the L_1-Wiberg gets stuck in local optima roughly half of the instances. We have made similar observations for other settings by varying the inlier/outlier ratio and the dimensions, both for affine and non-affine cases. Running time is 2s for our methods and 0.03s for L_1-Wiberg.

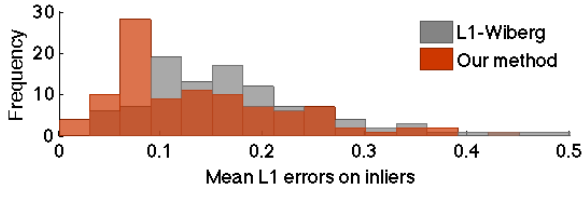


Figure 3.6: Results on the affine line fitting in $\mathbb{R}^3$.

3.7.2 Affine Structure-from-Motion

According to the affine camera model in Hartley and Zisserman (2004), a 3D point $\mathbf{v} \in \mathbb{R}^3$ is mapped to the image point $\mathbf{x} \in \mathbb{R}^2$ by $\mathbf{x} = U\mathbf{v} + \mathbf{t}$, where $U \in \mathbb{R}^{2\times 3}$ and $\mathbf{t} \in \mathbb{R}^2$ are the orientation and the translation of the camera, respectively. Given image points $\mathbf{x}_{ij}$ by projecting 3D points $\mathbf{v}_j$

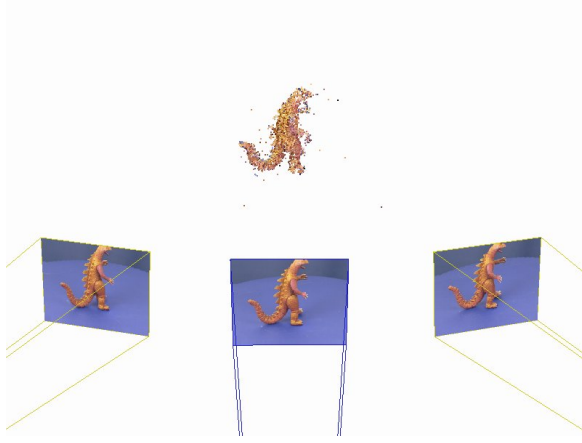


Figure 3.7: Affine Structure from Motion. Three images of a toy dinosaur and the corresponding 3D reconstruction.

onto image i, for $i = 1, \ldots, m$ and $j = 1, \ldots, n$, this can be written as

$$\begin{bmatrix} \mathbf{x}_{11} & \ldots & \mathbf{x}_{1n} \\ \vdots & \ddots & \vdots \\ \mathbf{x}_{m1} & \ldots & \mathbf{x}_{mn} \end{bmatrix} = \begin{bmatrix} U_1 & \mathbf{t}_1 \\ \vdots & \vdots \\ U_m & \mathbf{t}_m \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 & \ldots & \mathbf{v}_n \\ 1 & \ldots & 1 \end{bmatrix}.$$

This is the basis for the famous Tomasi-Kanade factorization algorithm in Tomasi and Kanade (1992) which first estimates the translation t_i by computing the mean of the observations in the corresponding rows, and then applies SVD to the (reduced) observation matrix in order to recover U and V. We will treat it as an affine subspace problem. See Figure 3.7 for an example.

We perform experiments of N-view structure from motion (SfM) using the Oxford dinosaur sequence. For each instance we use 300 points in N consecutive views, where N varies from 2 to 10. This creates a data matrix of size $2N \times 300$ with up to 75% missing data. Outliers with uniform noise on $[-50, 50]$ are added to 10% of the tracked points.

The 2-view SfM is exactly a hyperplane fitting problem, so we use Algorithm 1. For problems with more than two views, we use Algorithm 3 with 10^6 iterations. Running times are up to 3.8mins using our parallel OpenMP

implementation in C. As a comparison, we use the C++ implementation of L_1-Wiberg in Eriksson and Hengel (2012), General Wiberg in Strelow (2012) and Matlab code of the Regularized L_1-Augmented Lagrange Multiplier method (RegL_1-ALM) in Zheng et al. (2012b). In our experiments, L_1-Wiberg converges in no more than 50 iterations, which takes up to 30 s, while General Wiberg exhibits slower convergence. We set the maximum number of iterations to 500, which results in run-times up to 10.4mins. In a few cases, it does not even converge. The RegL_1-ALM converges in 0.3 s. L_1-Wiberg and General Wiberg are initialized using the truncated SVD, while the RegL_1-ALM is initialized with all zeros in U and V. All follows the settings in the original papers.

For each $N = 2, 3, \ldots, 10$, all possible instances with N consecutive views were tested. The Mean Absolute Error (MAE) of inliers for each N is shown in Fig. 3.8. The MAE of inliers is defined as

$$\text{MAE} = \frac{1}{|\mathcal{I}|} \sum_{(i,j)\in\mathcal{I}} |x_{ij} - \sum_{k=1}^{r} u_{ik} v_{kj}|, \tag{3.26}$$

where $\mathcal{I}$ is the set of inliers, and $|\mathcal{I}|$ is the cardinality of the set. We can see that our method clearly achieves lower error in all the experiments. As the dimension goes up, the percentage of missing data also rises, which heavily affects the performance for General Wiberg, but also for RegL_1-ALM.

To compare the methods with the same running time, we run the L_1-Wiberg and RegL_1-ALM with multiple random initializations. The General Wiberg is excluded in this comparison due to its weak performance both in accuracy and convergence. To generate random initializations, we first scale the data matrix X so that all the elements are around $[-1, 1]$. Then the elements of U and V are sampled uniformly from the interval $[-1, 1]$. We run the L_1-Wiberg with 10 different random initializations (including truncated SVD) and RegL_1-ALM with 1000 random initializations (including setting U and V to zeros). This leads to roughly the same running time for three methods. The comparison is given in Figure 3.9. It turns out that trying multiple random initializations lead to some improvement for both L_1-Wiberg and RegL_1-ALM. But the improvement is minor, especially for the RegL_1-ALM method, considering it runs with more random starts. It also verifies, as claimed in Zheng et al. (2012b), that it is hard to find a better solution using random initializations compared

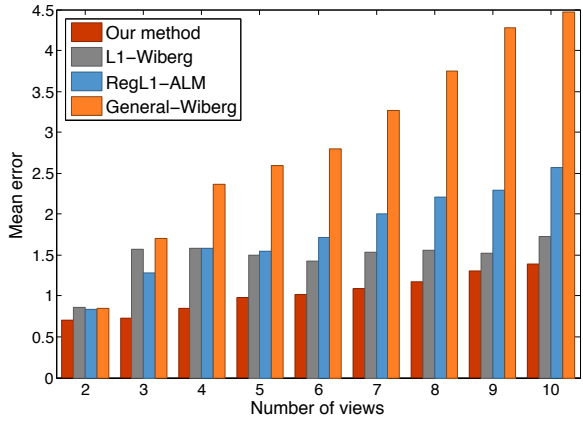


Figure 3.8: Results on Affine Structure from Motion with varying number of views. The L_1-Wiberg, General Wiberg and RegL_1-ALM run with truncated-SVD or all zeros initialization. Errors are given in pixels.

with the solution by setting U and V to be all zeros. One possible reason is that the algorithm first solves U with respect to V. In vision application, the number of elements in V is usually very large, leading to a huge search space of V. So 1000 random initializations on V might be relatively very few, from which it is hard to find a better solution. As seen from Figure 3.9, the other two methods do not gain much by trying different starting point given the same amount of running time. Our method still achieves lower errors.

To further examine the quality of our solutions, we run the random search algorithm 10 times, each with 10000 iterations, to check if the same optimal is obtained. Taking an example of 3-view SfM, we plot the result in Figure 3.10. We found that different random searches, although not leading to exactly the same solution, give very similar low errors. In this case, our random search has achieved lower error than our competitors in no more than 100 iterations.

We also tested on the 3-view data with different outlier ratios. The data setup is the same as above, but we add varied percentage of outliers to the data from 10% up to 50%. For our method, we use the random search with truncated L_1-norm. All the methods are affected by the increasing

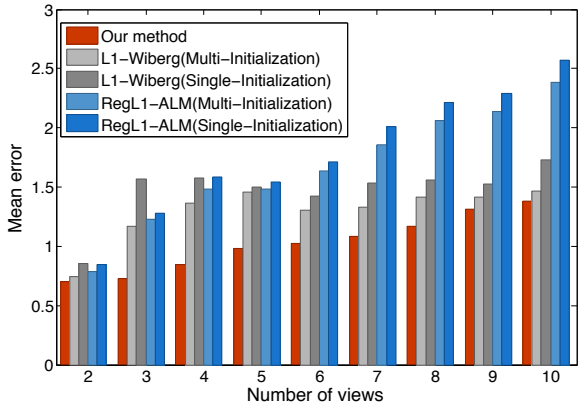


Figure 3.9: Results on Affine Structure from Motion with varying number of views. L_1-Wiberg and RegL_1-ALM are runned with single or multiple random initializations. Errors are given in pixels.

outliers, see Figure 3.11. The chance of our method to sample an outlier-free minimal set is decreased with increasing outlier ratio. Still we achieve smaller median errors of inliers.

3.7.3 Photometric Stereo

Assuming an orthographic camera viewing a Lambertian surface illuminated by a distant light source $\mathbf{v} \in \mathbb{R}^3$, the image intensity x of a surface element is given by

$$x = \mathbf{u}^{\mathrm{T}}\mathbf{v},$$

where $\mathbf{u} \in \mathbb{R}^3$ is the (unnormalized) surface normal. The length $\|\mathbf{u}\|$ gives the albedo (or the reflecting power) of the surface element. By varying the light source directions (and keeping the camera fixed), and by considering several image intensities, we end up in the factorization problem as (3.2). The measurement matrix $X \in \mathbb{R}^{m\times n}$ contains the intensities of m pixels in n images, $U \in \mathbb{R}^{m\times 3}$ the albedos and the surface normals of the m pixels and $V \in \mathbb{R}^{3\times n}$ the n light sources. Given the normals, it is possible to estimate a depth map by integration, see Yuille and Snow (1997).

In Alldrin et al. (2008), a set of 102 images (size 588×552 pixels) of a gourd was used to estimate the 3D shape and the surface reflectance

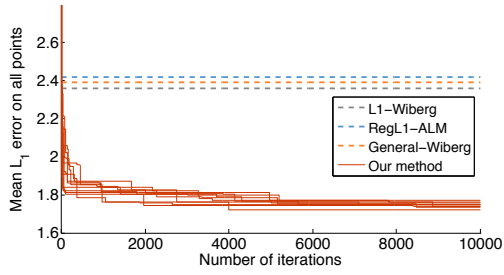


Figure 3.10: Quality of our solutions. The random search is run for 10 times with each red solid curve representing the current best error vs. the number of iterations. The grey, blue and orange dotted lines are the results for the other three methods.

properties using a sophisticated data-driven photometric stereo model. In contrast, we use a standard Lambertian model and a subset of eight random images to demonstrate that it is still possible to obtain a good estimate of 3D surface shape by using the truncated L_1-norm; see Fig. 3.12 for some example images. Note that in this example the surface is highly specular and do not concur with the Lambertian model at the specularities. Deviations from the Lambertian model such as specularities are handled by the robust choice of norm.

As the number of pixels is usually very large in the experiment, that is, of order 10^6, the computation of the L_1-projection, that is, estimate V given U for all the points in each iteration in Algorithm 3 are quite time consuming. Here we adopt a strategy that in each iteration, we compute the surface normal $\mathbf{v}$ for only a subset of points. In our experiment, we define this subset as the set of points on the down-sampled image with the down-sampled factor 4. It turned out the error on this subset of points is a good approximation of the error of all the points. And it gives a large speed up in the algorithm. Note that when estimating U we randomly sample the data from the original image. And the final solution is still for the image of the original size.

As a baseline, we compare with the RegL_1-ALM in Zheng et al. (2012b). We also tried the L_1-Wiberg of Eriksson and Hengel (2012) and General Wiberg of Strelow (2012), but none of those was possible to run on such a

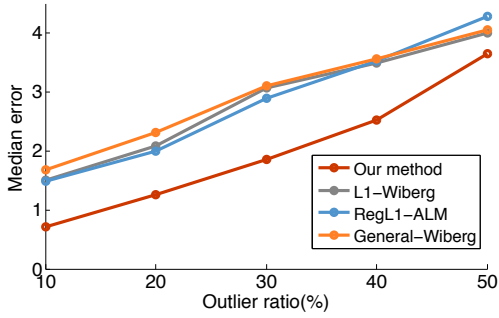


Figure 3.11: Results on 3-view Affine Structure from Motion with varying outlier ratios. Errors are given in pixels.

large problem. For our method, the truncation threshold is set to $\epsilon = 0.05$. It should be noted that each iteration of our method takes 0.3 s, and we use 15000 iterations. The running time for RegL_1-ALM is just 35 s. To make a fair comparison with respect to the running time, we also use the multiple random initializations for RegL_1-ALM and pick out the best solution. Here we run it with 100 different random starting points, which gives roughly the same running time.

The result of our 3D shape estimate is given in Figure 3.12 and the detected specularities are shown in Figure 3.13. Visual inspection shows that our solution basically captures the correct saturated points. The RegL_1-ALM has very similar visual results so we omit them here. If we calculate the average truncation error per pixel, our method achieves a mean absolute error of 0.0049 while the mean absolute error of RegL_1-ALM is 0.0052.

3.8 Conclusions

We have presented an alternative way of solving factorization problems under the L_1-norm that also works for the truncated L_1-norm. The method is independent of initialization, trivially parallelizable and as our empirical investigation on low-dimensional problems show, often the optimal solution is obtained. Compared to iterative methods based on local optimization, the quality of our solution is significantly better in terms of lower error. Our experimental results demonstrated that the local minima problem is

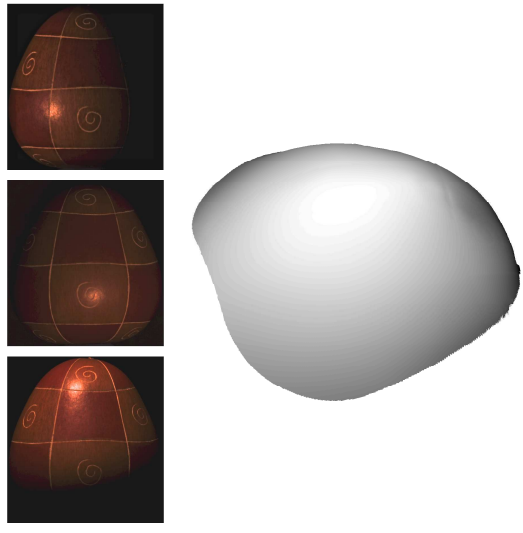


Figure 3.12: Photometric stereo. Three of eight images of a gourd and the corresponding 3D reconstruction viewed from the side. Images are courtesy of Alldrin et al. (2008)

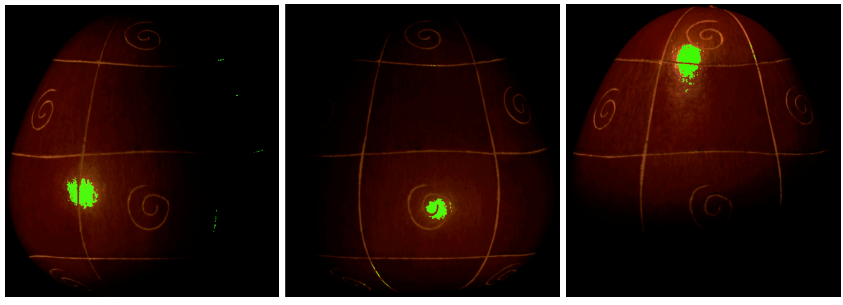


Figure 3.13: Results on photometric stereo. Three of eight input images where points with absolute residuals above $\epsilon = 0.05$ are marked in green.

not satisfactorily solved by the iterative methods.

Chapter 4

Low-Rank Matrix Factorization: Minimal Problems

In this chapter, we focus on the minimal problems in low-rank matrix factorization. One motivation is that with increasing sparsity of the measurement matrix, the nuclear norm based methods will eventually fail. Solving the minimal problems would help us recover a low-rank matrix with minimal observations. When the measurement matrix is dense, minimal solvers provide a robust estimation in presence of outliers.

As we will see, the principal stationary points in Definition 3.4 in the previous chapter belong to the minimal problems of low-rank matrix factorization. In this chapter, we present a more unified and general understanding of the minimal problems in low-rank matrix factorization. More specifically, we demonstrate how to characterize, generate, parameterize and solve these problems, restricting ourselves not only to the linearly solvable ones.

4.1 Introduction

We will begin by introducing a few definitions that will be used to characterize the minimal problems in low-rank matrix factorization.

Definition 4.1. *Given a matrix $X \in \mathbb{R}^{m\times n}$ with missing data, an* index matrix $W \in \{0,1\}^{m\times n}$ *is a binary valued matrix to indicate that the corresponding element at (i,j) is present if $w_{ij} = 1$ or missing if $w_{ij} = 0$*

We also define the following partial order relationship "$\leq$" between two index matrices W and W'.

Definition 4.2. *Given two index matrices W and W', we say that W is a* submatrix *of W', denoted as $W \leq W'$ if $w_{ij} = 1 \implies w'_{ij} = 1$.*

An index matrix W is said to be *rigid* if for general data, the low-rank matrix factorization problem is *locally well defined*.

Definition 4.3. *A low-rank matrix factorization problem with an index matrix W is said to be* locally well defined *if for a generic measurement matrix X of rank r, there are only a finite number of rank-r matrices $\tilde{X}$ that satisfy $W \odot (X - \tilde{X}) = 0$ where $\odot$ is the Hadamard product, that is the element wise product.*

The notion of rigidity is invariant under a permutation of rows or columns. We define the following equivalent relationship between two index matrices.

Definition 4.4. *Given two index matrices W and W', we say that W is equivalent to W' if $w_{ij} = w'_{P_1(i),P_2(j)}$ for all i, j where P_1, P_2 are permutations of rows and columns respectively.*

Before defining the minimal problems in matrix factorization, we first look at the degrees of freedom (DoF) for a low-rank matrix. For a rank-r factorization of a matrix $X \in \mathbb{R}^{m\times n}$, we have that U has mr DoF and V has nr DoF. There is a total coordinate ambiguity of size $r \times r$ as

$$X = U^{\mathrm{T}}V = U^{\mathrm{T}}QQ^{-1}V, \tag{4.1}$$

where $Q \in \mathbb{R}^{r\times r}$ is a full rank matrix. Thus a matrix $X \in \mathbb{R}^{m\times n}$ with rank r, has $mr + nr - r^2$ degrees of freedom, which means that we need at least $d = mr + nr - r^2$ measurements to recover a rank-r matrix X of size $m \times n$

A minimal problem for low-rank matrix factorization is characterized by a minimal index matrix, which is defined as below.

Definition 4.5. *An index matrix W for a rank-r problem is said to be* minimal *if it is rigid and satisfies $\sum_{ij} w_{ij} = mr + nr - r^2$. W is said to be* overdetermined *if $\sum_{ij} w_{ij} > mr + nr - r^2$.*

A minimal problem of low-rank matrix factorization is to find two factor matrices U and V that exactly solve the following equation

$$W \odot (X - U^{\mathrm{T}}V) = 0, \tag{4.2}$$

where W is a minimal index matrix and X is the measurement matrix. For the minimal problem characterized by a minimal index matrix W, there is a finite number $n_W > 0$ of solutions, where n_W only depends on the index matrix W.

4.1.1 Characterizing the Minimal Index Set

For each minimal index matrix W of size $m \times n$ there is a corresponding minimal index matrix W' of size $n \times m$ such that $w_{ij} = w'_{ji}$. Without loss of generality we may thus in the discussion assume that $n \geq m$. For each minimal index matrix, it has $mr + nr - r^2$ non-zero elements. Since by assumption $m \leq n$, we have at most $2nr - r^2$ non-zeros in W, which are distributed among n columns. Thus there are never enough non-zeros to fill up $2r$ positions of each column, in another word, there is at least one column which has fewer than $2r$ non-zeros. Furthermore it is obvious that for rank-r problems, the minimal index matrices must have at least r non-zero elements in each column, otherwise the corresponding column $\mathbf{v}$ of V has too few constraints to be solvable. So for the column with the smallest number k of non-zero elements we must have $r \leq k < 2r$.

We also notice that since one assumes $r < min(m, n)$, that is, $m - r \geq 1$. It means that for a minimal index matrix, the number of non-zeros $mr + nr - r^2 \geq nr + r$. If we distribute those non-zeros in n columns each of which has at least r zeros, then there is at least one column which has no less than $r + 1$ non-zeros.

4.2 Generating the Minimal Problems

4.2.1 Laman Graph and Henneberg Construction

The ideas and inspirations for generating the minimal problems of low-rank matrix factorization are from the rigid graph theory, especially from the Henneberg construction, see Lebrecht (1911) to generate minimally rigid graphs, also known as Laman graphs, see L. (1970). In this section, we will introduce some basic concepts and theories in rigid graphs, as a motivation for generating the minimal low-rank problems.

Laman graph If we consider to place some rods (edges) and joints (vertices) on a plane to form a planar graph, then for some configurations of

these rods and joints, there is no simultaneous motion of vertices, except for a Euclidean congruence, that can preserve the lengths of all edges. These configurations form a family of graph, which is known as *rigid graphs*.

Intuitively, given n vertices, if we keep adding edges to connect pairs of vertices, which reduces the degrees of freedom, it will eventually become a rigid graph. So it is interesting to study the minimal case, which requires a minimal number of edges to make a graph with n vertices rigid. Laman graphs are introduced to characterize a family of such minimally rigid planar graphs.

Definition 4.6. *A* Laman graph *is a graph on n vertices such that for $2 \leq k \leq n-1$, every subgraph with k vertices has at most $2k-3$ edges, and the whole graph has exactly $2n-3$ edges.*

To further understand the definition, we consider the degrees of freedom (DoF) in the problem. Given n vertices on a plane, there are totally $2n$ DoF to place the vertices since each vertex has two coordinates in 2D. However, a minimally rigid graph has only three DoF: two for coordinates of a certain vertex and one for a 2D rotation around that vertex. When we add an edge with fixed length to the graph, we reduce the DoF by one. So one needs $2n-3$ edges to reduce the DoF from $2n$ to 3

However, we need to avoid the cases that one puts excessive edges on a subgraph which is already rigid, otherwise some subgraphs might be under-constrained. Thus the condition in Definition 4.6 on subgraphs intuitively states that edges should be distributed evenly and each edge should contribute to reduce the overall degrees of freedom.

Henneberg construction In Lebrecht (1911), it shows that starting from a simple graph with two vertices and one edge, any minimally rigid graph can be generated by a sequence of the following two types of constructions,

- A *Type-1 Construction* adds a new vertex to the graph, together with edges connecting it to two previously existing vertices
- A *Type-2 Construction* subdivides an edge of the graph and add an edge connecting the newly formed vertex to a third previously existing vertex.

4.2.2 Henneberg Extensions

We now describe how to generate the minimal problems in low-rank matrix factorization. Inspired by Henneberg constructions in rigid graphs, the idea is that one could start with the smallest minimal index matrix and by a series of extensions every minimal index matrix could be generated. For example, for $r = 2$, the smallest index matrix is

$$W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}. \tag{4.3}$$

In the following we will distinguish between *constructive* extensions and *non-constructive* extensions. For a constructive extension from W to W', we could infer the number of solutions $n_{W'}$ from n_W and construct the solver, denoted by $f_{W'}$ from f_W. For a non-constructive extension, it can be shown that W is minimal if and only if W' is minimal. However, we could neither infer the number of solutions $n_{W'}$ from n_W nor derive a solver $f_{W'}$ from f_W. We propose the following extensions and reductions which are denoted as *Henneberg-k extensions/reductions*. Among these Henneberg-1 extensions are constructive, whereas Henneberg-k extensions for $k \geq 2$ are in general non-constructive.

Henneberg-1 extension Given a minimal index matrix W for a rank-r problem of size $m \times n$, an extended minimal index matrix W' of size $m \times (n+1)$ is formed by adding a column with exactly r indices, that is non-zero elements. The numbers of solutions for W and W' are identical, that is $n_W = n_{W'}$. Extending an algorithm from f_W to $f_{W'}$ is straightforward. A similar extension can be done by adding a row with r indices.

Henneberg-1 reduction Given a minimal index matrix W for a rank-r problem of size $m \times n$ where there is a column $\mathbf{w}_j$ with exactly r non-zeros, a reduced minimal index matrix W' of size $m \times (n-1)$ is formed by removing the column $\mathbf{w}_j$ from W. The number of solutions is preserved under Henneberg-1 reduction. The solution to W' can be obtained straightforward from the solution to W. A similar reduction can be done by removing a row from W with r non-zeros.

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \xrightarrow{\text{Hen I}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \xrightarrow{\text{Hen I}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \xrightarrow{\text{Hen II}} \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

Figure 4.1: An example of generating rank-2 index matrices of increasing size using a sequence of Henneberg-1 and Henneberg-2 extensions.

Henneberg-2 extension Given a minimal index matrix W for a rank-r problem of size $m \times n$, where there is a column $\mathbf{w}_j$ with at least $r+1$ non-zero elements at rows $i_1, \ldots, i_{r+1}$ (such a column must exist from Sec. 4.1.1) , an extended index matrix W' of size $m \times (n+1)$ can be formed by first adding a column $\mathbf{w}'$ with exactly $r+1$ non-zero elements at rows $i_1, \ldots, i_{r+1}$, then setting one of the non-zeros of $\mathbf{w}_j$ to be zero. The resulting index matrix W' is also minimal. A similar extension can be done for a row.

Similarly one can define Henneberg-2 reduction. This can be generalized to Henneberg-k extension and reduction for $k > 2$. We illustrate an example on how to generate an index matrix for a minimal problem of rank 2 using Henneberg-1 and Henneberg-2 extensions in Figure 4.1.

As shown in Lebrecht (1911), every minimally rigid graph can be constructed using a sequence of Henneberg-1 and Henneberg-2 construction in the context of rigid graphs. In the following we will show that every minimal index matrix can be generated using a series of Henneberg-k extension defined above for rank-1 and rank-2 problems. We also make a conjecture that this is the case for the general rank-r problems where $r > 2$.

Theorem 4.7. *Each minimal index matrix for rank-1 problems can be generated by a series of Henneberg-1 extensions from a* 1×1 *index matrix as the base case.*

Proof. The proof is by induction over size. If the matrix is of size 1×1 then we are done. Otherwise we assume that it is true for all matrices of size $m \times n$ with $m + n \leq K$. Now let us take a minimal index matrix of size $m \times n$ where $m + n = K + 1$. From Section 4.1.1, we know that there always exists a column with at least k non-zero elements where $r \leq k < 2r$. In this case, there is always a column with exactly one non-zero element. After a Henneberg-1 reduction on that column, we obtain a minimal index

matrix with $m + n = K$, which can be constructed as our assumption. So the original index matrix is a Henneberg-1 extension from a smaller index matrix that is in the assumption, which proves the theorem. Similar proof exists for a row-wise extension. □

Theorem 4.8. *Each minimal index matrix for rank-2 problems can be generated by a series of Henneberg-1 and Henneberg-2 extensions from a* 2×2 *minimal index matrix as the base case.*

Proof. The proof is similarly by induction over size. If the matrix is of size 2×2 then we are done. Otherwise we assume that it is true for all index matrices of size $m \times n$ with $m + n \leq K$. Now we take a minimal index matrix of size $m \times n$ where $m + n = K + 1$. From Section 4.1.1, we know that there is always a column with exactly k non-zero elements where $r \leq k < 2r$, that is $k = 2$ or 3 in this case. If the column has two non-zeros then the index matrix can be constructed using the Henneberg-1 extension from the one that is in the assumption. If the column has three non-zeros then it can be reduced to an index matrix of size $m + n \leq K$ using Henneberg-2 reduction. In either case, we have shown that the index matrix can be constructed using either Henneberg-1 or Henneberg-2 extension from a smaller index matrix that is in the assumption, which proves the theorem. One can show a similar proof for a row-wise extension. □

It is worth noting that in the above proofs we only show the *if* part that every minimal index can be generated by Henneberg extensions. The *only if* part that the index matrices generated by Henneberg extension are minimal is not proved. We conjecture that it is true and leave it for future work.

We have the following conjecture that it is possible to generate all the minimal index matrices using a sequence of Henneberg extensions.

Conjecture 4.9. *Each minimal index matrix for rank-*r *problems can be formed by a series of Henneberg-1 to Henneberg-*r *extensions.*

4.3 Minimal Solvers

In this section, we will describe the solvers to the minimal problems generated from different Henneberg extensions.

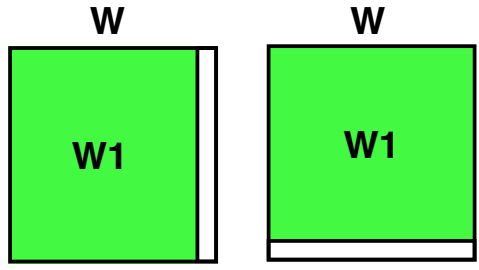


Figure 4.2: Henneberg-1 extension

4.3.1 Solvers for Henneberg-1 Extension

Suppose we are given a rank-r minimal problem with the index matrix W and the measurement matrix X. Assume the solution is given by U and V, which means

$$W \odot (X - U^{\mathrm{T}}V) = \mathbf{0}, \tag{4.4}$$

Now we apply Henneberg-1 extension to W as $W' = [W|\mathbf{w}]$ where the column vector $\mathbf{w}$ has r non-zeros. Correspondingly the measurement matrix is extended as $X' = [X|\mathbf{x}]$ where $\mathbf{x}$ has r observations. To find the solution to the extended minimal problem, it is obvious that we only need to solve for the extra column $\mathbf{v}$ of V such that the following equation is satisfied

$$W' \odot (X' - U^{\mathrm{T}} \left[V \,|\, \mathbf{v}\right]) = \mathbf{0}. \tag{4.5}$$

Now assume the positions of r observations of $\mathbf{x}$ is at $\mathtt{I} = \{n_1, \ldots, n_r\}$. Then we have the following equation

$$U_{\mathtt{I}}^{\mathrm{T}}\mathbf{v} = \mathbf{x}_{\mathtt{I}} \tag{4.6}$$

where $U_{\mathtt{I}} \in \mathbb{R}^{r \times r}$ and $\mathbf{x}_{\mathtt{I}} \in \mathbb{R}^r$ denotes taking the corresponding rows at $\mathtt{I}$ from U and $\mathbf{x}$ respectively. This is a linear system with a unique solution if U is of full rank. For the row-wise Henneberg-1 extension, we will keep V unchanged and solve for the extra column $\mathbf{u}$ of U using a similar strategy.

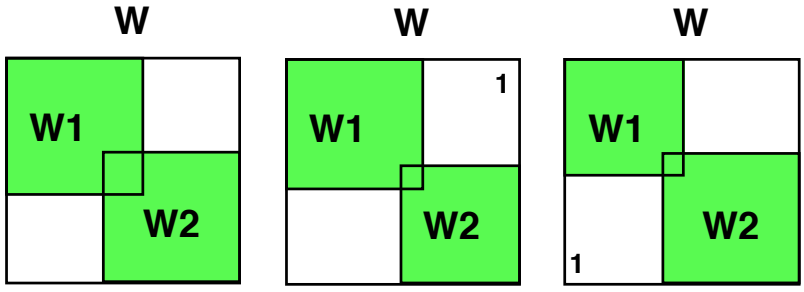


Figure 4.3: The other constructive extensions. (a) the overlap of size $r \times r$ gives sufficient constraints. (b) and (c) some extra constraints are needed when the overlap is smaller than $r \times r$

4.3.2 Solvers for Other Constructive Extension

Henneberg-k extensions for $k \geq 2$ are non-constructive, which means that for $W \rightarrow W'$ one cannot construct the solution to W' from the solution to W. However, we can define some other constructive extensions. The intuitive idea is that given two index matrices W_1 and W_2, one can construct a new index matrix W by "glueing" W_1 and W_2 together. By "glueing", we mean that W contains both W_1 and W_2 with overlapping rows and columns as illustrated in Figure 4.3.

In the following, we will first describe a general parametrization for these types of constructive extensions, which is independent of the rank r. We derive several constraints using the proposed parametrization to solve the problem. A few examples for rank-2 and rank-3 cases will be illustrated.

Parametrization Consider that we have a minimal problem with index matrix W and measurement X. W is constructed by "glueing" two index matrices W_1 and W_2 as in Fig. 4.3(a). We use $\mathtt{I}_1$ and $\mathtt{J}_1$ to denote the row and column indices of W_1 in W, similarly $\mathtt{I}_2$, $\mathtt{J}_2$ for W_2. Then we have $W_1 = W_{\mathtt{I}_1,\mathtt{J}_1}$ and $W_2 = W_{\mathtt{I}_2,\mathtt{J}_2}$ where $W_{\mathtt{I},\mathtt{J}}$ denotes the submatrix of W by taking the rows at $\mathtt{I}$ and the columns at $\mathtt{J}$. For measurement matrices X_1, X_2 associated with W_1 and W_2 respectively, we have $X_1 = X_{\mathtt{I}_1,\mathtt{J}_1}$ and $X_2 = X_{\mathtt{I}_2,\mathtt{J}_2}$. We also use $\mathtt{I}_{12}$ and $\mathtt{J}_{12}$ to denote the indices of overlapping

rows and columns as $\mathtt{I}_{12} = \mathtt{I}_1 \cap \mathtt{I}_2$ and $\mathtt{J}_{12} = \mathtt{J}_1 \cap \mathtt{J}_2$.

Assume the solutions to the sub-problems $\{W_1, X_1\}$ and $\{W_2, X_2\}$ are given by $\{U_1, V_1\}$ and $\{U_2, V_2\}$ respectively. To construct the solution to $\{W, X\}$, the idea is to find a transformation matrix $H \in \mathbb{R}^{r\times r}$ to transform the subspace U_2 to the same coordinate framework as the subspace U_1. Using this transformation we have

$$U_2^{\mathrm{T}} V_2 = U_2^{\mathrm{T}} H^{\mathrm{T}} H^{-\mathrm{T}} V_2 = (HU_2)^{\mathrm{T}} (H^{-\mathrm{T}} V_2), \tag{4.7}$$

Now HU_2 and $H^{-T}V_2$ are in the same coordinate framework as U_1 and V_1 respectively. The remaining problem is to solve for H. Apparently we have the following constraint which states U_1 and HU_2 should coincide for the overlapping columns as

$$(U_1)_{\mathtt{I}_1} = (HU_2)_{\mathtt{I}_2} = H(U_2)_{\mathtt{I}_2}, \tag{4.8}$$

where $\mathtt{I}_1$ and $\mathtt{I}_2$ denotes the indices of overlapping columns in U_1 and U_2 respectively and $U_{\mathtt{I}}$ denotes the submatrix of U by taking the columns given by $\mathtt{I}$. Similarly we have the overlapping constraints for V_1 and $H^{-\mathrm{T}}V_2$ as

$$(V_1)_{\mathtt{J}_1} = H^{-\mathrm{T}} (V_2)_{\mathtt{J}_2}, \tag{4.9}$$

where $\mathtt{J}_1$ and $\mathtt{J}_2$ denotes the indices of overlapping columns in V_1 and V_2 respectively. Or it can be written as

$$H^{\mathrm{T}} (V_1)_{\mathtt{J}_1} = (V_2)_{\mathtt{J}_2}, \tag{4.10}$$

which is linear with respect to H.

If we have enough constraints from (4.8) and (4.10), H can be solved linearly. In that case, the overlap should be of size $r \times r$ as in Figure 4.3(a). For the cases where the overlap does not give sufficiently many constraints, we need some extra constraints outside W_1 and W_2 to solve for the transformation matrix H as in Figure 4.3(b)(c).

To solve the case with extra constraints, we know that each extra measurement $x_{\mathtt{ij}}$ directly gives a constraint as

$$\mathbf{u}_{\mathtt{i}}^{\mathrm{T}} \mathbf{v}_{\mathtt{j}} - x_{\mathtt{ij}} = 0, \tag{4.11}$$

where $\mathbf{u}_{\mathtt{i}}$ and $\mathbf{v}_{\mathtt{j}}$ are the $\mathtt{i}$-th and the $\mathtt{j}$-th column of U and V respectively. We need to express the constraint in (4.11) using U_1, U_2, V_1 and V_2 from

which we assume U_1 and V_1 are in the same coordinate framework as U and V respectively. There are in general two cases. When $x_{\mathtt{ij}}$ is in the top-right as in Fig. 4.3(b), one needs to express (4.11) using U_1 and V_2 as

$$(U_1)_{\mathtt{i}_1}^{\mathrm{T}} H^{-\mathrm{T}} (V_2)_{\mathtt{j}_2} - x_{\mathtt{ij}} = 0, \tag{4.12}$$

where $\mathtt{i}_1$ denotes the local index of column $\mathtt{i}$ in U_1 and $\mathtt{j}_2$ the local index of column $\mathtt{j}$ in V_2, and $(U_1)_{\mathtt{i}_1}$ denotes taking the column $\mathtt{i}_1$ from U_1, similarly for notation $(V_2)_{\mathtt{j}_2}$. When $x_{\mathtt{ij}}$ is in the bottom-left as in Figure 4.3(c), one needs to rewrite (4.11) using U_2 and V_1 instead as

$$(H(U_2)_{\mathtt{i}_2})^{\mathrm{T}} (V_1)_{\mathtt{j}_1} - x_{\mathtt{ij}} = 0, \tag{4.13}$$

where $\mathtt{i}_2$ denotes the local index of column $\mathtt{i}$ in U_2 and $\mathtt{j}_1$ the local index of column $\mathtt{j}$ in V_1.

All these constraints from (4.8) (4.10) (4.12) (4.13) form either a linear system or a simple polynomial system, from which one can solve for the transformation H and thus yield a solution to the original problem. In the following we will illustrate a few examples for rank-2 and rank-3 problems, all of which are parametrized and solved using the similar strategy.

Examples of rank-2 constructive extension Here we will present a few constructive extensions for rank-2 problems, which are illustrated in Figure 4.4. The overlap for rank-2 problems is at most 2×2, otherwise the extension is non-minimal. For a 2×2 overlap, H can be solved linearly using only the overlap constraints. For a 1×1 overlap with an extra constraint as in Fig. 4.4(a), the overlap in U gives two equations in (4.8) and the overlap in V gives two equations in (4.10). Among the four equations one is redundant as it is automatically satisfied when the other three hold. So one can parametrize H using a single variable z, which is solved using a single extra constraint.

One could of course generate a minimal problem by "glueing" more index matrices. In Fig. 4.4(b) and (c), each submatrix needs a transformation, namely H_1, H_2 and H_3 of which one can fix the gauge by setting $H_2 = I$. Two extra constraint are needed to solve the problem. For the case in Fig. 4.4(d), four transformation H_1, H_2, H_3 and H_4 are needed. By setting $H_2 = I$, one can parametrize H_1, H_3 and $H_3^{-1} H_4$ using z_1, z_2 and z_3 respectively. Each of three extra constraints provide a quadratic equation in z_1, z_2 and z_3 respectively, which could be solved giving 3 solutions.

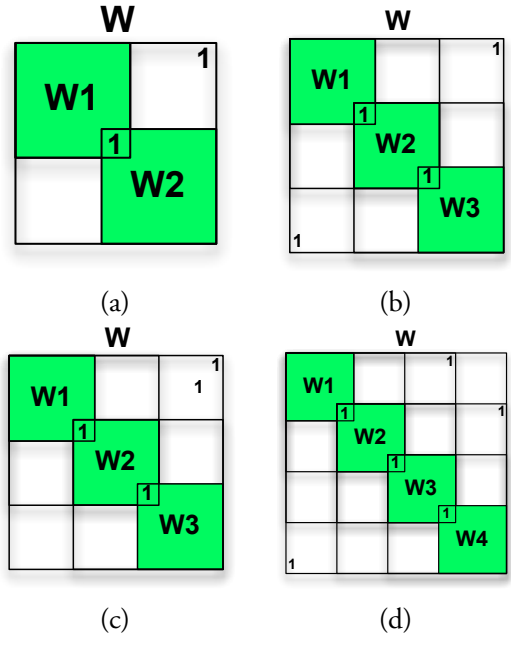
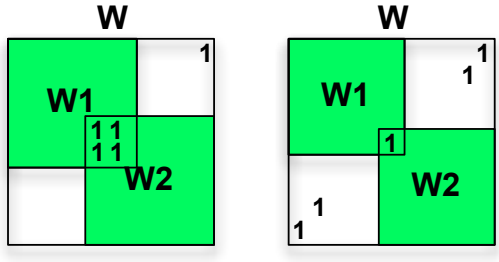


Figure 4.4: Examples of rank-2 constructive extensions. In each of these four examples several smaller minimal cases are glued together with a few extra constraints. For these extensions the multiplicity of solutions grows as follows: for (a) $n_W = n_{W_1} n_{W_2}$, for (b) $n_W = 2n_{W_1} n_{W_2} n_{W_3}$, for (c) $n_W = 2n_{W_1} n_{W_2} n_{W_3}$, and for (d) $n_W = 3n_{W_1} n_{W_2} n_{W_3} n_{W_4}$. For each of these extensions we have implemented the method of extending the algorithms, i.e. $\{f_{W_1}, f_{W_2}, \dots f_{W_n}\} \to f_W$.



(a) (b)

Figure 4.5: Examples of rank-3 constructive extensions.

Examples of rank-3 constructive extension For rank-3 problems, the maximum overlap is 3×3 which can be solved linearly. For the case with a 2×2 overlap illustrated in Figure 4.5(a), H can be parametrized using a single variable z, and can be solved linearly using one extra constraint. When the overlap is 1×1 as in Figure 4.5(b), the overlap provides three equations in (4.8) and three equations in (4.10). However one of them is redundant, which means it reduces the DoF of H from nine to four. Thus extra four constraints are needed to solve for H.

4.4 Algorithms for Structured Data Patterns

For real problems, the observations in the measurement matrix might be more dense than that for minimal problems. However, one could apply a random sampling strategy to sample the minimal configuration from the measurement matrix and solve one minimal problem in each iteration, while pick out the solution that gives the lowest error. It is a similar idea as the random search algorithm in Section 3.5.3.

Block partition for structured data patterns In applications where the locations of missing data are highly correlated and structured, for example, the affine structure-from-motion, we use a similar block-partition idea as in Larsson et al. (2014) to divide the measurement matrix into overlapping sub-blocks. The solution to each sub-block is estimated separately using the random sampling method. Once we obtain solutions to all the sub-blocks,

we use a null space matching method, see Olsen and Bartoli (2008) to combine those solutions and recover the full factorization.

However, block partition in our method differs from the strategy used in Larsson et al. (2014) in that using minimal solvers, missing data can be handled in sub-blocks, while in Larsson et al. (2014) each sub-block must be complete without missing data based on their convex formulation. As we will show in the experiment, being able to cope with missing data in sub-blocks leads to a more flexible way of partition that could cover all the observations.

4.5 Applications

We have conducted a number of experiments on both synthetic and real data. For synthetic data and affine structure-from-motion, we use the solvers both for the Henneberg-1 extension and the other constructive extensions from Section 4.3.2. The other constructive extensions are useful when the measurement matrix is sparse, especially when it is not possible to find a minimal configuration using only Henneberg-1 extensions. For the shape basis estimation, we use only the minimal solvers for the Henneberg-1 extension, as the matrix is more dense.

4.5.1 Synthetic Data

For synthetic data, we generate random rank-3 matrices of size 100×100 with entries uniformly drawn from $[-1, 1]$. All entries are then perturbed with noise drawn from a normal distribution $\mathcal{N}(0, \sigma)$ where σ takes $\{0, 10^{-3}, 10^{-2}\}$. The rank constraint is enforced using a truncated SVD.

A structured data pattern is formed by removing some measurements in the generated matrices. We are especially interested in the band-diagonal structure which appears in many affine structure-from-motion problems. A sparse matrix D is said to be band-diagonal with bandwidth k if

$$d_{ij} = 0 \quad \text{for} \quad j < i - k \quad \text{or} \quad j > i + k. \tag{4.14}$$

Here we generate random band-diagonal matrices of size 100×100 with varied bandwidth k from 20 down to 4. The corresponding proportion of missing data ranges from 64% up to 92%. For comparison, we consider two

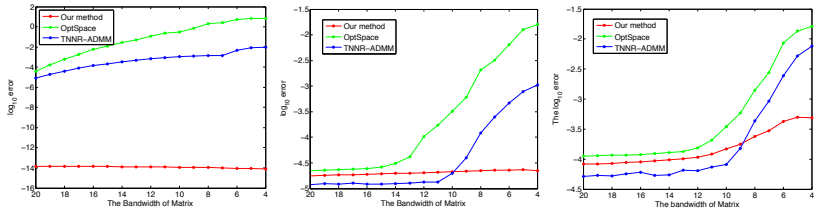


Figure 4.6: The synthetic data with band-diagonal structure. The bandwidth of matrix vs. The log_{10} L_2-error. Left: Noise-free case. Middle: Noise level $\sigma = 10^{-3}$, Right: Noise level $\sigma = 10^{-2}$.

state-of-the-art methods, namely Truncated Nuclear Norm Regularization (TNNR-ADMM) in Hu et al. (2013) and OptSpace in Keshavan et al. (2009). OptSpace was initialized using truncated SVD.

We plot $\log_{10}$ errors versus the bandwidth of the matrix in Figure 4.6, where the error is defined as $\|W \odot (X - U^{\mathrm{T}}V)\|_F$. As the bandwidth decreases (the missing data increases), the performance of both TNNR-ADMM and OptSpace are affected, especially when the bandwidth $k < 10$ which corresponds to around 80% missing data. In noise-free case, our method achieves significant lower error. With low level noise, that is $\sigma = 10^{-3}$, our method remains stable with respect to the rate of missing data. With medium level noise, that is $\sigma = 10^{-2}$, our method is also affected when $K < 10$, but still performs better compared with TNNR-ADMM and OptSpace.

4.5.2 Affine Structure-from-Motion

We evaluate our method on the well-known dinosaur sequence for affine structure-from-motion. The original dinosaur sequence contains 2D point tracks from the projection of in total 3184 3D points onto 36 cameras. Each 3D point is only visible in a few consecutive views and missing for the rest views due to self-occlusion. We consider a rank-4 factorization which does not use any initial estimate of the translation.

We take a subset of 3D points which are visible in at least 8 views. This forms an observation matrix X of size 72×116 with 87.8% miss data. We first compare with TNNR-ADMM and OptSpace, neither of which handles outliers. In this case, we divide the measurement matrix

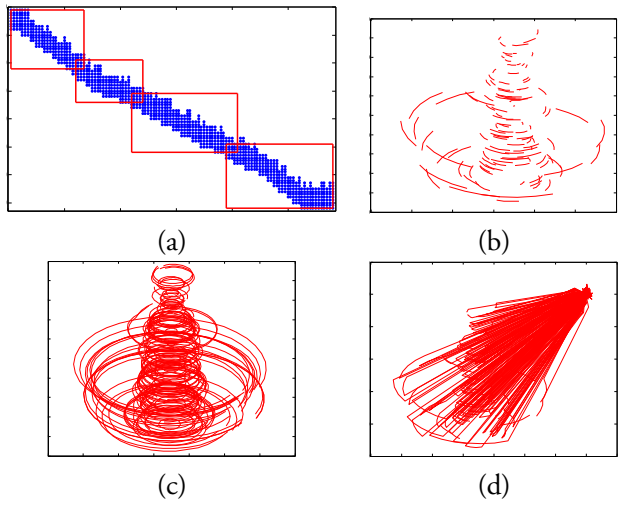


Figure 4.7: Affine structure from motion on the dinosaur sequence. (a) The band-diagonal structure and block partition. (b) The input 2D tracks. (c) The recovered 2D tracks using our method. (d) The recovered 2D tracks using TNNR-ADMM.

into 4 overlapping sub-blocks with varied size, see Figure 4.7(a). The partition of the measurement matrix is a trade-off between complexity and numerical accuracy. For a too large sub-block, the numerical error within the sub-block accumulates for Henneberg-1 extension solver. If we divide the matrix into too many sub-blocks, it unnecessarily increase the computational complexity. We ran our method with 1000 iterations followed by a non-linear least square optimization in the final step.

The results are shown in Figure 4.7, both TNNR-ADMM and OptSpace fails to recover the 2D tracks. We plot the recovered tracks for our method and TNNR-ADMM in Figure 4.7(c) and (d) respectively. When the data matrix is sparse, the recovered 2D tracks from TNNR-ADMM are stretched towards the origin (top right corner) of the image.

In presence of outliers, both TNNR-ADMM and OptSpace will fail. We thus compare with two L_1-based methods, namely Wiberg-L_1 in Eriksson and Hengel (2012) and Regularized L_1 augmented Lagrange multiplier method (RegL_1-ALM) in Zheng et al. (2012b). We add 10% outliers with

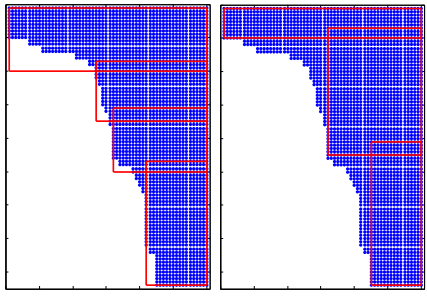


Figure 4.8: The missing data and block partition on *Book* dataset. Left: our method. Right: Larsson et al. (2014)

values in the range $[-50, 50]$ to the measurement matrix. The only change in our method is that in each iteration we compute the error in L_1-norm as $\|W \odot (X - U^{\mathrm{T}}V)\|_1$ instead of Frobenius norm. We run 1000 iterations using our minimal solver within each sub-block. The average L_1 error for our method is 1.314 pixels and 1.231 for Wiberg-L_1 and 2.223 for RegL_1-ALM.

4.5.3 Linear Shape Basis

For non-rigid structure-from-motion, a linear shape basis model is commonly used to model the shape of a non-rigid object. It assumes that any non-rigid deformation of an object can be represented as a linear combination of a set of shapes. Normally the size of the shape basis is assumed to be much smaller than either the number of frames or the number of tracked points, so the measurement matrix containing point tracks can be factorized into a coefficient matrix and a shape basis matrix.

Two datasets, namely *Book* and *Hand* from Larsson et al. (2014) are used. The image points are tracked using a standard *Kanade-Lucas-Tomasi* (KLT) tracker, see Lucas and Kanade (1981). Due to occlusions, the tracker fails after a number of frames for a subset of points, which leads to the missing data pattern shown in Figure 4.8 for *Book* dataset. In our experiments, we use a subset of 42 frames with 60 tracked points from *Book* and 38 frames with 203 points from the *Hand* dataset. Following the setup in Larsson et al. (2014), we seek for a rank-3 and rank-5 factorization on two

	Algorithm		10 % missing data	
Dataset	Larsson et al. (2014)	*Our*	Larsson et al. (2014)	*Our*
Book	0.3522	**0.1740**	8.0436	**0.1772**
Hand	0.8613	**0.6891**	1.5495	**0.7297**

Table 4.1: The result on linear shape basis estimation on the *Book* and *Hand* dataset, where the second experiment contains an extra 10% missing data. The numbers depict the Frobenius errors using our method compared to the method of Larsson et al. (2014).

datasets respectively.

The block partition of both our method and Larsson et al. (2014) are illustrated in Figure 4.8. Contrary to Larsson et al. (2014), missing data within blocks is handled in our method, giving a more flexible partition with wider coverage of measurements. We run our method with 1000 iterations for each block. To further show that our method is capable to handle random missing data, we conduct a second experiment by randomly adding an extra 10% of missing data in the measurement matrix and run both methods with the same setting as before.

We summarize the error $\|W \odot (X - U^{\mathrm{T}}V)\|_F$ in Table 4.1. Our method achieves smaller error in both dataset with or without adding extra missing data. When random missing data are added into the blocks, Larsson et al. (2014) fails with large errors. In Figure 4.9, the recovered tracked points using our method are plotted for both datasets with extra missing data. Without extra missing data, both our method and Larsson et al. (2014) achieve quite similar visual results, which are omitted. The running time for our method is 12s on *Book* and 28s on *Hand*. For Larsson et al. (2014), it is 2.5s on *Book* and 4.5s on *Hand*. However, the running time for our method can be further reduced as it is highly parallelizable.

4.6 Conclusions

In this chapter, we have introduced theory to characterize and generate the minimal problems of low-rank matrix factorization, inspired by Henneberg

extensions of rigid graph theory. We have demonstrated that for rank-1 and rank-2 problems, the proposed Henneberg extensions can generate all minimal problems. We conjecture that, by using additional Henneberg extensions, all minimal problems can be generated for any given rank. Several solvers are proposed for solving minimal problems using constructive extensions. With block partition and a random sampling scheme, minimal solvers can be used in a number of real applications when the data matrix is sparse and contains outliers.

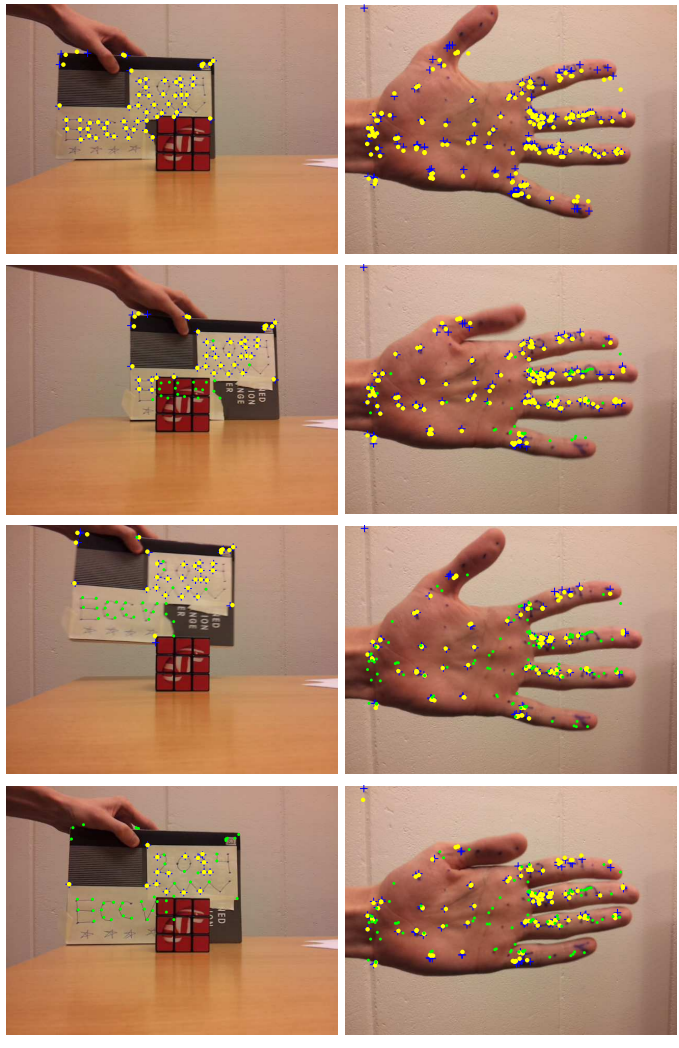


Figure 4.9: The results on *Book* (Left column) and *Hand* (Right column) dataset using our method with extra 10% missing data added in the blocks. From left to right: The 1^{st}, 10^{th}, 20^{th}, 30^{th} frame. The blue crossings are the measurements. The yellow dots are the recovered measurements (normally coincide with the actual measurements). The green dots are the recovered missing data.

Chapter 5

Rank Minimization in Sensor Network Calibration

In this chapter, we investigate the application of rank minimization in sensor network calibration. A sensor network contains several transmitters and receivers. A signal, for example, a sound, travels through the medium from a certain transmitter to a receiver. Receivers are usually calibrated, which means that for any receiver one could measure the time when it receives a signal from a certain transmitter. However, transmitters are not calibrated; in other words, the actual time of transmission is unknown.

The goal of this chapter is to estimate the unknown time of transmission in a sensor network. We derive a rank constraint on a measurement matrix with unknown time of transmission. The problem is formulated as a rank-minimization problem using truncated nuclear-norm regularization (TNNR) and is optimized using the alternating-direction method of multipliers (ADMM). We will demonstrate in synthetic experiments that our method recovers the time of transmission with good accuracy in the presence of noise and missing data. It is worth noting that in the case that the transmitters are calibrated instead of receivers, one can use a similar formulation and optimization to recover the unknown time of receiving.

5.1 Introduction

For a sensor network with m receivers and n transmitters, we denote the spatial coordinates of the receivers and transmitters $\{\mathbf{r}_i\}$ for $i = 1, \cdots, m$ and $\{\mathbf{s}_j\}$ for $j = 1, \cdots, n$, respectively. The measured time when a signal arrives at the i-th receiver from the j-th transmitter is denoted t_{ij}. The unknown time of transmission for the j-th transmitter is denoted t_j. If we

know the speed of the signal in the medium, denoted v, then we have

$$v(t_{ij} - t_j) = \|\mathbf{r}_i - \mathbf{s}_j\|_2 . \tag{5.1}$$

In the following we will use the distance measurements, instead of the time measurements. We have, namely the relative distance $f_{ij} = vt_{ij}$ and the delay distance $o_j = vt_j$. So (5.1) can be written as

$$f_{ij} - o_j = \|\mathbf{r}_i - \mathbf{s}_j\|_2 . \tag{5.2}$$

where f_{ij} are measurements and o_j are unknowns. For a sensor network with m calibrated receivers and n uncalibrated transmitters, we define two network calibration problems as following.

Problem 5.1. *(Delay distance estimation for a time-difference-of-arrival network) Given relative distance measurements f_{ij}, determine the delay distance, o_j, for unknown receiver positions $\mathbf{r}_i$ and unknown transmitter positions $\mathbf{s}_j$, such that $f_{ij} = \|\mathbf{r}_i - \mathbf{s}_j\|_2 + o_j$ for $i = 1, \cdots, m$ and $j = 1, \cdots, n$.*

Problem 5.2. *(Time-of-arrival network calibration) Given absolute distance measurements d_{ij} between the i-th receiver and j-th transmitter, determine the positions, $\mathbf{r}_i$, of receivers and the positions, $\mathbf{s}_j$, of transmitters such that $d_{ij} = \|\mathbf{r}_i - \mathbf{s}_j\|_2$ for $i = 1, \cdots, m$ and $j = 1, \cdots, n$.*

In this chapter, we focus on Problem 5.1 to estimate the delay distance o_j and treat it as a separate problem from reconstructing sensor locations $\{\mathbf{r}_i\}$ and $\{\mathbf{s}_j\}$. Once the delay distances o_j are estimated, Problem 5.1 can be recast as Problem 5.2 by setting $d_{ij} = f_{ij} - o_j$.

5.2 Problem Formulation

5.2.1 Rank Constraint

We assume that all the transmitters and receivers are located in a k-dimensional space, that is $\mathbf{r}_i, \mathbf{s}_j \in \mathbb{R}^k$. To solve the delay distance estimation problem for the time-difference-of-arrival (TDOA) network, we first introduce the rank constraint on the measurement matrix. From (5.2), we have

$$(f_{ij} - o_j)^2 = (\mathbf{r}_i - \mathbf{s}_j)^{\mathrm{T}}(\mathbf{r}_i - \mathbf{s}_j) = \mathbf{r}_i^{\mathrm{T}}\mathbf{r}_i - 2\mathbf{r}_i^{\mathrm{T}}\mathbf{s}_j + \mathbf{s}_j^{\mathrm{T}}\mathbf{s}_j. \tag{5.3}$$

By constructing the vectors $\hat{\mathbf{r}}_i$ and $\hat{\mathbf{s}}_j$ as

$$\hat{\mathbf{r}}_i = \begin{pmatrix} 1 \\ \mathbf{r}_i \\ \mathbf{r}_i^{\mathrm{T}}\mathbf{r}_i \end{pmatrix} \quad \text{and} \quad \hat{\mathbf{s}}_j = \begin{pmatrix} \mathbf{s}_j^{\mathrm{T}}\mathbf{s}_j - o_j^2 \\ -2\mathbf{s}_j \\ 1 \end{pmatrix}, \tag{5.4}$$

where $\hat{\mathbf{r}}_i, \hat{\mathbf{s}}_j \in \mathbb{R}^{k+2}$, we can rewrite (5.3) as

$$f_{ij}^2 - 2f_{ij}o_j = \hat{\mathbf{r}}_i^{\mathrm{T}}\hat{\mathbf{s}}_j. \tag{5.5}$$

By stacking the column vectors $\hat{\mathbf{r}}_i$ and $\hat{\mathbf{s}}_j$ into two matrices $\hat{R}$ and $\hat{S}$ respectively, we have

$$\hat{F} = \hat{R}^{\mathrm{T}}\hat{S}, \tag{5.6}$$

where $\hat{R} \in \mathbb{R}^{(k+2)\times m}$, $\hat{S} \in \mathbb{R}^{(k+2)\times n}$, and $\hat{F} \in \mathbb{R}^{m\times n}$ contain the elements $f_{ij}^2 - 2f_{ij}o_j$. This suggests that $\hat{F}$ is at most of rank $k+2$. Using this rank constraint and the structure of $\hat{R}$, a linear factorization technique is proposed in Pollefeys and Nister (2008) to solve for the unknown delay distances. This technique requires $m = 2(k+2)$ receivers and $n = k+2$ transmitters; for example, in 3D space where $k = 3$, one needs 10 receivers and five transmitters to solve the problem.

Here we present a slight modification of the linear scheme presented in Pollefeys and Nister (2008). The idea is that, by exploring the structure of $\hat{S}$ and $\hat{R}$, one can further reduce the rank constraint. To see this, we first right multiply $\hat{S}$ by matrix C_n of the following form

$$C_n = \begin{pmatrix} -1 & -1 & \cdots & -1 \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}, \tag{5.7}$$

where $C_n \in \mathbb{R}^{n\times(n-1)}$. Right multiplying by C_n is equivalent to subtracting the first column, $\hat{\mathbf{s}}_1$, from the remaining columns, $\hat{\mathbf{s}}_j$ ($j \geq 2$), of $\hat{S}$, which yields matrix $\bar{S}$ with all zeros in the last row. We could therefore remove the last row of zeros from the result matrix $\bar{S} = \hat{S}C_n$. Then the

last row of $\hat{R}$ can also be removed without changing the multiplication, $\hat{R}^{\mathrm{T}}\hat{S}$. Equivalently, this gives

$$\bar{F} = \hat{F}C_n = \bar{R}^{\mathrm{T}}\bar{S}, \tag{5.8}$$

where $\bar{R} \in \mathbb{R}^{(k+1)\times m}$, $\bar{S} \in \mathbb{R}^{(k+1)\times(n-1)}$ and the columns $\bar{\mathbf{s}}_j$ of $\bar{S}$ are of the form

$$\bar{\mathbf{s}}_j = \hat{\mathbf{s}}_{j+1} - \hat{\mathbf{s}}_1 = \begin{pmatrix} \mathbf{s}_{j+1}^{\mathrm{T}}\mathbf{s}_{j+1} - \mathbf{s}_1^{\mathrm{T}}\mathbf{s}_1 - o_{j+1}^2 + o_1^2 \\ -2(\mathbf{s}_{j+1} - \mathbf{s}_1) \end{pmatrix}, \tag{5.9}$$

the columns $\bar{\mathbf{r}}_i$ of $\bar{R}$ are of the form

$$\bar{\mathbf{r}}_i = \begin{pmatrix} 1 \\ \mathbf{r}_i \end{pmatrix}, \tag{5.10}$$

and the entries $\bar{f}_{ij}$ of $\bar{F}$ are obtained similarly by subtracting the first column from each of the remaining columns as

$$\bar{f}_{ij} = \hat{f}_{i,j+1} - \hat{f}_{i1} = f_{i,j+1}^2 - f_{i1}^2 - 2f_{i,j+1}o_{j+1} + 2f_{i1}o_1. \tag{5.11}$$

Since C_n removes one row from both $\hat{S}$ and $\hat{R}$, we call it a *compaction matrix*. This effectively reduces the rank constraint from $k+2$ to $k+1$.

We can further explore the structure of matrix $\bar{R}$ in a similar way. If one right multiplies $\bar{R}$ by the compaction matrix $C_m \in \mathbb{R}^{m\times(m-1)}$, the first column, $\bar{\mathbf{r}}_1$, of $\bar{R}$ is subtracted from the remaining columns, $\bar{\mathbf{r}}_i$ for $i \geq 2$, which yields a matrix with all zeros in the first row. This suggests that the first row of zeros can be removed from the result matrix, $\tilde{R} = \bar{R}C_m$. Correspondingly, the first row of $\bar{S}$ can also be removed. This is equivalent to left multiplying $\bar{F}$ by C_m^{T}, which subtracts the first row of $\bar{F}$ from the remaining rows. This gives

$$\tilde{F} = C_m^{\mathrm{T}}\bar{F} = (\bar{R}C_m)^{\mathrm{T}}\bar{S} = \tilde{R}^{\mathrm{T}}\tilde{S}, \tag{5.12}$$

where $\tilde{R} \in \mathbb{R}^{k\times(m-1)}$, $\tilde{S} \in \mathbb{R}^{k\times(n-1)}$ and the columns $\tilde{\mathbf{r}}_i$ of $\tilde{R}$ are of the form

$$\tilde{\mathbf{r}}_i = \mathbf{r}_{i+1} - \mathbf{r}_1, \tag{5.13}$$

and the columns $\tilde{\mathbf{s}}_j$ of $\tilde{S}$ are of the form

$$\tilde{\mathbf{s}}_j = -2\left(\mathbf{s}_{j+1} - \mathbf{s}_1\right). \tag{5.14}$$

The entries $\tilde{f}_{ij}$ of $\tilde{F}$ take the following form

$$\tilde{f}_{ij} = g_{ij} - g_{0j} - g_{i0} + g_{00}, \tag{5.15}$$

where g_{ij} is defined as

$$g_{ij} = f_{i+1,j+1}^2 - 2f_{i+1,j+1}o_{j+1}. \tag{5.16}$$

By using two compaction matrices C_m and C_n, the constraint that $\hat{F}$ is of rank $k+2$ is reduced to that $\tilde{F}$ is of rank k.

5.2.2 Problem Formulation

To enforce the rank constraint on $\tilde{F}$, we first rewrite $\tilde{F}$ as a linear combination of some constant matrices using the unknown variables o_j as

$$\tilde{F} = B_0 + \sum_{j=1}^{n} o_j B_j, \tag{5.17}$$

where $B_0, B_1, \cdots, B_n \in \mathbb{R}^{(m-1)\times(n-1)}$ are constant matrices derived from (5.15). More specifically, we have

$$\begin{aligned} B_0 &= C_m^{\mathrm{T}} F C_n, \\ B_1 &= \left(\mathbf{b}_1, \cdots, \mathbf{b}_1\right), \\ B_j &= \left(0, \cdots, \mathbf{b}_j, \cdots, 0\right) \text{ for } j \geq 2, \end{aligned} \tag{5.18}$$

where elements of F are equal to the squares of relative distance measurements, that is, f_{ij}^2. B_1 contains the same column, $\mathbf{b}_1$, with elements of the form $\{2(f_{i1} - f_{11})\}_{i\geq 2}$, that is,

$$\mathbf{b}_1 = 2\begin{pmatrix} f_{21} - f_{11} \\ f_{31} - f_{11} \\ \vdots \\ f_{m1} - f_{11} \end{pmatrix}, \tag{5.19}$$

and B_j is all zeros except for the j^{th} column, $\mathbf{b}_j$, with elements of the form $\{-2(f_{ij} - f_{1j})\}_{i\geq 2}$, that is,

$$\mathbf{b}_j = -2 \begin{pmatrix} f_{2j} - f_{1j} \\ f_{3j} - f_{1j} \\ \vdots \\ f_{mj} - f_{1j} \end{pmatrix}. \tag{5.20}$$

Now, we can formulate our problem as the following rank minimization problem

$$\begin{aligned} \underset{\mathbf{o},X}{\text{minimize}} \quad & \text{rank}(X) \\ \text{subject to} \quad & B_0 + \sum_{j=1}^{n} o_j B_j = X. \end{aligned} \tag{5.21}$$

As the rank function is generally difficult to cope with, a more feasible approach is instead to minimize the nuclear norm of the matrix, which gives the following formulation

$$\begin{aligned} \underset{\mathbf{o},X}{\text{minimize}} \quad & \|X\|_* \\ \text{subject to} \quad & B_0 + \sum_{j=1}^{n} o_j B_j = X, \end{aligned} \tag{5.22}$$

where $\|\cdot\|_*$ denotes the nuclear norm, which is the sum of the singular values of a matrix.

In the presence of noise in the measurement matrix, the constraint in (5.22) might not be strictly satisfied. We relax the equality constraint by adding a penalty in the cost function as

$$\underset{\mathbf{o},X}{\text{minimize}} \quad \|X\|_* + \frac{\mu}{2} \left\| B_0 + \sum_{j=1}^{n} o_j B_j - X \right\|_F^2, \tag{5.23}$$

where $\mu > 0$ is a scalar parameter and $\|\cdot\|_F$ is the Frobenius norm.

When minimizing the nuclear norm of X, in some cases the singular values are minimized evenly. It is undesirable to obtain a matrix with

small, widely distributed singular values. To avoid this, we apply the truncated nuclear norm regularization (TNNR) approach proposed in Hu et al. (2013). Instead of minimizing the nuclear norm, we could minimize the truncated nuclear norm, which is the sum of the smallest $n-k$ singular values, assuming that the rank of the matrix is k and the matrix is of size $n \times n$. In Hu et al. (2013), it is demonstrated that minimizing the nuclear norm plus an extra trace term is equivalent to minimizing the truncated nuclear norm. Using the TNNR formulation, we have

$$\underset{\mathbf{o},X,U,V}{\text{minimize}} \left\|X\right\|_* - \text{trace}(UXV^{\mathrm{T}}) + \frac{\mu}{2}\left\|B_0 + \sum_{j=1}^{n} o_j B_j - X\right\|_F^2, \tag{5.24}$$

where $U \in \mathbb{R}^{k\times(m-1)}$, $V \in \mathbb{R}^{k\times(n-1)}$, and $UU^{\mathrm{T}} = I$, $VV^{\mathrm{T}} = I$. Adding the extra trace term makes the formulation non-convex. However, we will illustrate in the next section how to find a local minimum for this problem.

5.3 Optimization using ADMM

5.3.1 Augmented Lagrangian and ADMM

We first introduce the augmented Lagrangian into a constrained optimization problem. Consider the following optimization problem with linear constraints

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & A\mathbf{x} + b = 0. \end{aligned} \tag{5.25}$$

The augmented Lagrangian of the constrained problem in (5.25) is defined as

$$L_\rho(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathrm{T}}(A\mathbf{x} - b) + \frac{\rho}{2}\left\|A\mathbf{x} - b\right\|_2^2. \tag{5.26}$$

When an objective function, $f(\mathbf{x})$, is naturally split into two functions, that is, $f(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x})$, the alternating direction method of multipliers (ADMM) can be used to break the problem of optimizing $f(\mathbf{x})$

into two subproblems using Lagrangian duality. Consider the following optimization problem with linear constraints

$$\begin{aligned} \underset{\mathbf{x},\mathbf{y}}{\text{minimize}} \quad & f_1(\mathbf{x}) + f_2(\mathbf{y}) \\ \text{subject to} \quad & A\mathbf{x} + B\mathbf{y} = \mathbf{c}, \end{aligned} \tag{5.27}$$

the augmented Lagrangian of the constrained problem in (5.27) is

$$\begin{aligned} L_\rho(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda}) =& f_1(\mathbf{x}) + f_2(\mathbf{y}) + \boldsymbol{\lambda}^{\mathrm{T}}(A\mathbf{x} + B\mathbf{y} - \mathbf{c}) \\ &+ \frac{\rho}{2} \left\| A\mathbf{x} + B\mathbf{y} - \mathbf{c} \right\|_2^2 . \end{aligned} \tag{5.28}$$

The ADMM algorithm performs the following updates iteratively

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \arg\min_{\mathbf{x}} L_\rho(\mathbf{x}, \mathbf{y}^{(k)}, \boldsymbol{\lambda}^{(k)}), \\ \mathbf{y}^{(k+1)} &= \arg\min_{\mathbf{y}} L_\rho(\mathbf{x}^{(k+1)}, \mathbf{y}, \boldsymbol{\lambda}^{(k)}), \\ \boldsymbol{\lambda}^{(k+1)} &= \boldsymbol{\lambda}^{(k)} + \rho(A\mathbf{x}^{(k+1)} + B\mathbf{y}^{(k+1)} - \mathbf{c}). \end{aligned} \tag{5.29}$$

In each iteration, one of $\mathbf{x}$, $\mathbf{y}$ and $\boldsymbol{\lambda}$ is updated while the other two are fixed.

5.3.2 An Iterative Scheme

Now consider the problem formulated as (5.24), in which an alternating optimization strategy is iteratively applied to two sets of variables, $\{U, V\}$ and $\{\mathbf{o}, X\}$, following Hu et al. (2013). More specifically, the variables are first initialized using the measurements. In each iteration, for a given X, we estimate U and V using singular value decomposition (SVD) of X. Then U and V are fixed and the unknown delay distances, $\mathbf{o}$ and X, are optimized using ADMM. The framework of the algorithm is summarized in Algorithm 4. We will describe each step of the algorithm in detail below.

Update U, V using SVD In the l-th iteration, we first estimate $U^{(l)}$, $V^{(l)}$ for a given $X^{(l)}$. Applying SVD on $X^{(l)}$ gives us

$$X^{(l)} = U\Sigma V^{\mathrm{T}}, \tag{5.30}$$

Algorithm 4 Delay Distance Estimation with TNNR

Given TDOA measurements $\{f_{ij}\}$ *of* m *receivers and* n *transmitters, a threshold* ϵ*, a maximum iterations* n_{max}*, estimate the delay distance* $\mathbf{o}$ *and the distance matrix* X

Construct the constant matrices B_j based on (5.18)
Initialize $\mathbf{o}^{(1)} = \mathbf{0}$, $X^{(1)} = B_0$ and $l = 1$
Repeat
  Step 1. Solve for $U^{(l)}$ and $V^{(l)}$ given $X^{(l)}$
    Using SVD gives $X^{(l)} = U\Sigma V^{\mathrm{T}}$
      where $U = (\mathbf{u}_1, \cdots, \mathbf{u}_{m-1})$, $V = (\mathbf{v}_1, \cdots, \mathbf{v}_{n-1})$,
    Let $U^{(l)} = (\mathbf{u}_1, \cdots, \mathbf{u}_k)^{\mathrm{T}}$ and $V^{(l)} = (\mathbf{v}_1, \cdots, \mathbf{v}_k)^{\mathrm{T}}$
  Step 2. Fix $U^{(l)}$, $V^{(l)}$, solve the following problem using ADMM
    $\{X^{l+1}, \mathbf{o}^{l+1}\} = \arg\min \|X\|_* - \text{trace}(UXV^{\mathrm{T}}) +$
      $\frac{\mu}{2}\left\|B_0 + \sum_{j=1}^{n} o_j B_j - X\right\|_F^2$.
  $l = l + 1$
Until $\sqrt{\left\|X^{(l+1)} - X^{(l)}\right\|_F^2 + \left\|\mathbf{o}^{(l+1)} - \mathbf{o}^{(l)}\right\|_F^2} < \epsilon$ or $l \geq n_{max}$

where $U = (\mathbf{u}_1, \cdots, \mathbf{u}_{m-1}) \in \mathbb{R}^{(m-1)\times(m-1)}$ and $V = (\mathbf{v}_1, \cdots, \mathbf{v}_{n-1}) \in \mathbb{R}^{(n-1)\times(n-1)}$. Then $U^{(l)}$ and $V^{(l)}$ can be formed by collecting the first k column vectors of U and V respectively, that is,

$$\begin{aligned} U^{(l)} &= (\mathbf{u}_1, \cdots, \mathbf{u}_k)^{\mathrm{T}}, \\ V^{(l)} &= (\mathbf{v}_1, \cdots, \mathbf{v}_k)^{\mathrm{T}}. \end{aligned} \tag{5.31}$$

Update o, X using ADMM For fixed $U^{(l)}$ and $V^{(l)}$, We use ADMM to estimate $\mathbf{o}^{(l+1)}$ and $X^{(l+1)}$. We first introduce a new variable $\hat{X}$ to split the objective function in (5.24) into two parts, one associated with X and the other with $\hat{X}$. This gives

$$\begin{aligned} \underset{\mathbf{o},X,\hat{X}}{\text{minimize}} \quad & \|X\|_* - \text{trace}(U\hat{X}V^{\mathrm{T}}) + \frac{\mu}{2}\left\|B_0 + \sum_{j=1}^{n} o_j B_j - \hat{X}\right\|_F^2 \\ \text{subject to} \quad & X = \hat{X}. \end{aligned} \tag{5.32}$$

Note that U and V are fixed as $U^{(l)}$ and $V^{(l)}$ respectively. The augmented Lagrangian of (5.32) is written as

$$L_\rho(\mathbf{o}, X, \hat{X}, Y) = \|X\|_* - \text{trace}(U\hat{X}V^{\mathrm{T}}) + \frac{\mu}{2}\left\|B_0 + \sum_{j=1}^{n} o_j B_j - \hat{X}\right\|_F^2 + \text{trace}(Y^{\mathrm{T}}(X - \hat{X})) + \frac{\rho}{2}\left\|X - \hat{X}\right\|_F^2. \tag{5.33}$$

Using ADMM, we alternate the optimization of three sets of variables, namely X, $\{\mathbf{o}, \hat{X}\}$, and Y. In each step, we optimize over one set of variables while fixing the others. To distinguish between the inner iterations within ADMM and the outer iterations that involve U and V, we use a subscript, for example, X_k, to denote variables in the k-th inner iteration using ADMM, while a superscript within parentheses, for example, $X^{(l)}$, is used to denote variables in the l-th outer iteration.

Starting from the initial solution, that is,

$$\begin{aligned} \mathbf{o}_1 &= \mathbf{0}, \\ X_1 = \hat{X}_1 &= Y_1 = B_0, \end{aligned} \tag{5.34}$$

where B_0 is defined in (5.18), we perform the following steps to update three sets of variables for $(k+1)$-th iteration.

- Update X_{k+1} Given $\mathbf{o}_k$, $\hat{X}_k$ and Y_k, we minimize the augmented Lagrangian $L(\mathbf{o}_k, X, \hat{X}, Y)$ over X. By ignoring constants, we have

$$X_{k+1} = \arg\min_X \|X\|_* + \text{trace}(Y_k^{\mathrm{T}}(X - \hat{X}_k)) + \frac{\rho}{2}\left\|X - \hat{X}_k\right\|_F^2, \tag{5.35}$$

which is equivalent to

$$X_{k+1} = \arg\min_X \|X\|_* + \frac{\rho}{2}\left\|X - (\hat{X}_k - \frac{1}{\rho}Y_k)\right\|_F^2. \tag{5.36}$$

It is demonstrated in Cai et al. (2010) that the problem in the form of (5.36) can be solved by applying the singular value thresholding theorem.

- Update $\mathbf{o}_{k+1}$, $\hat{X}_{k+1}$ Fix X_{k+1} and Y_k, we can optimize $\mathbf{o}_{k+1}$ and $\hat{X}_{k+1}$ as follows

$$\begin{aligned}\{\mathbf{o}_{k+1}, \hat{X}_{k+1}\} =& \arg\min_{\mathbf{o},\hat{X}} \frac{\mu}{2} \left\| B_0 + \sum_{j=1}^{n} o_j B_j - \hat{X} \right\|_F^2 \\ &+ \frac{\rho}{2} \left\| X_{k+1} - (\hat{X} - \frac{1}{\rho} Y_k) \right\|_F^2,\end{aligned} \tag{5.37}$$

where the objective function is the sum of two quadratic functions and can be solved by finding $\mathbf{o}, \hat{X}$ such that $\partial L_\rho(\mathbf{o}, X_{k+1}, \hat{X}, Y_k) = 0$.

- Update Y_{k+1} Y can be updated as

$$Y_{k+1} = Y_k + \rho(X_{k+1} - \hat{X}_{k+1}). \tag{5.38}$$

The above three steps of ADMM are performed iteratively until X_k converges. Then we leave the inner loops of ADMM and let $X^{(l+1)} = X_k$ to update $U^{(l+1)}$, $V^{(l+1)}$ in the outer loops and keep iterating. The algorithm stops when either $\{X^{(l)}, \mathbf{o}^{(l)}\}$ converges, that is,

$$\sqrt{\left\| X^{(l+1)} - X^{(l)} \right\|_F^2 + \left\| \mathbf{o}^{(l+1)} - \mathbf{o}^{(l)} \right\|_F^2} < \epsilon, \tag{5.39}$$

or a maximum number of iterations is reached.

The above algorithm can easily be modified to handle the missing data in the TDOA measurements. We only need to replace the term $\left\| B_0 + \sum_{j=1}^{n} o_j B_j - \hat{X} \right\|_F^2$ with $\left\| (B_0)_\Omega + \sum_{j=1}^{n} o_j (B_j)_\Omega - \hat{X}_\Omega \right\|_F^2$, where Ω is the set of available measurements. This modification will only slightly change the update step for $\mathbf{o}, \hat{X}$, while the other steps remain the same.

5.4 Experiments

In this section, we present the experimental results of applying the method to synthetic data. We simulate a sensor network with $m = 50$ receivers and $n = 50$ transmitters, the coordinates of which are drawn independently from a standard normal distribution, that is, $\mathcal{N}(0, 1)$. The delay

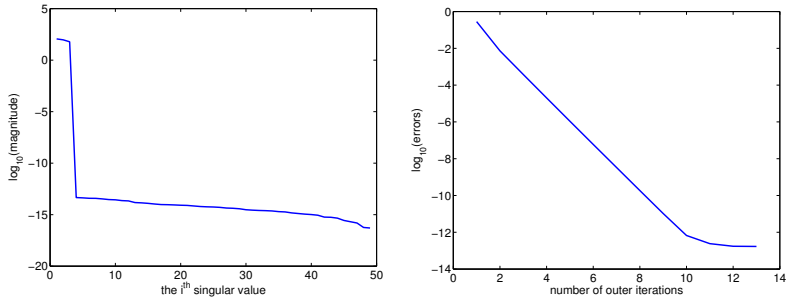


Figure 5.1: Synthetic TDOA measurements with no noise ($m = n = 50$). Left: Singular values of the matrix X after optimization; Right : Speed of convergence ($\|\mathbf{o} - \mathbf{o}_{gt}\|_F$) for the ADMM algorithm ($\mu = 10,\ \lambda = 1, \epsilon = 10^{-12}$).

distance for each transmitter is also sampled from a standard normal distribution, that is, $o_j \sim \mathcal{N}(0, 1)$. All the experiments were conducted using a MacBook Air with a 1.8 GHz Intel Core i5 CPU and 8 GB of memory.

We first examine the convergence of the algorithm in the noise-free case. Figure 5.1 (left) shows the distribution of the singular values in $\log_{10}$ scale (after sorting) for the recovered measurement matrix, $\tilde{F}$, defined in (5.12). All the singular values except the largest three are around the order of 10^{-15}, which means that a rank-3 solution is achieved. From Figure 5.1 (right), we can see that the algorithm converges within fewer than 20 iterations. The running time in this case is 2 s.

To better understand the performance of the method, we run the algorithm on noise-free data with varying numbers of m transmitters and n receivers. For this experiment, we run our method using 100 randomly generated synthetic data as described above and plot the results in Figure 5.2. For parameters, we set $\epsilon = 10^{-12}$ and the maximum number of iterations $N_{max} = 5000$. From Figure 5.2 (left), one can see that for a small number of receivers, for example, $m = 5$, the method does not converge to a reliable solution. For a larger value of m, that is, $m \geq 6$, the relative error of the delay distance, that is, $\|\mathbf{o} - \mathbf{o}_{gt}\|_F / \|\mathbf{o}_{gt}\|_F$, decreases as the number of transmitters increases, suggesting the benefit of using more measurements.

We also test the method at different noise levels for varying numbers of transmitters and receivers. From Figure 5.2 (right), we can see that the algo-

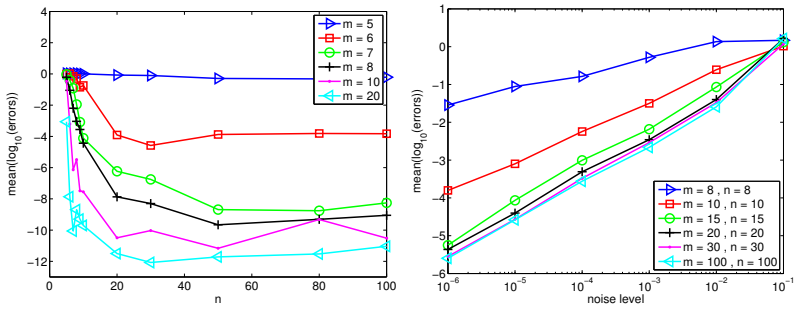


Figure 5.2: Synthetic experiments - relative errors $\|\mathbf{o} - \mathbf{o}_{gt}\|_F / \|\mathbf{o}_{gt}\|_F$ in average on 100 random synthetic TDOA measurements. Left: (Noise-free data) varied n for different m. Right: (Noisy data) - varied m and n for different levels of noise

rithm generally achieves better results with more transmitters and receivers. However, the performance improves little when the sensor network exceeds a certain size, for example, $m \geq 30$ and $n \geq 30$. The figure also shows that at a high noise level, for example, $\sigma = 10^{-1}$, the method performs poorly no matter how many transmitters and receivers are used in the network.

While the above experiments assume that measurements are complete, it is interesting to see how the algorithm behaves in the case of missing data. In this experiment, we run the algorithm on synthetic TDOA data from which a certain number of observations are removed as missing data. The noise-free case is illustrated in Figure 5.3 (left). One can see that for relatively large values of m and n, for example, $m = n = 30$, the algorithm only breaks down when more than 50% of the measurements are missing. For smaller values of m and n, for example, $m = n = 10$, the algorithm displays rather low tolerance of missing data, that is, no more than 20% of the data can be missing. For noisy measurements where $\sigma = 10^{-2}$ (see Figure 5.3, right), the algorithm works well until more than 20% of the data are missing in the case $m = n = 20$. With more transmitters and receivers, for example, $m = n = 30$, the algorithm performs much better up to 50% missing data.

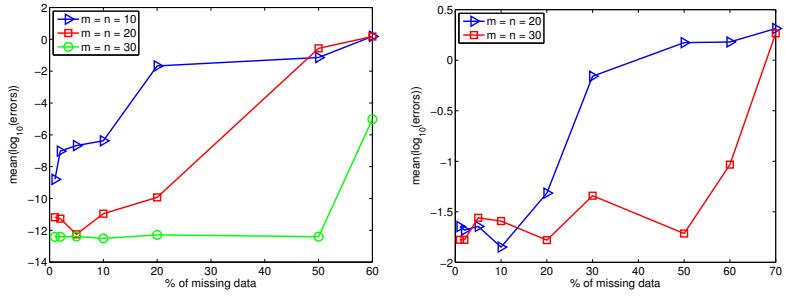


Figure 5.3: Synthetic TDOA measurements with varying percentage of missing data. Left : noisy-free data for different m and n; Right : noisy data with Gaussian noise of standard deviation 10^{-2}.

5.5 Conclusions

In this chapter, we have studied the problem of estimating the unknown delay distances in TDOA self-calibration. We derived a new rank constraint on the measurement matrix that is linear with respect to the delay distances and showed how to find such a low-rank matrix using ADMM optimization of the truncated nuclear norm. Synthetic experiments show that the proposed method gives good estimate of the time delays when noise or missing data are present. The experiments are for $\mathbb{R}^3$ but the algorithm can be generalized to any $\mathbb{R}^k$.

Chapter 6

Relative Pose with Radial Distortion

Relative pose is one of the fundamental problems in geometric computer vision. Although it has been widely studied, there are still some minimal problems that have not yet been solved. In this chapter, we will investigate the unsolved minimal problem when two cameras have the same but unknown radial distortion and focal length. Incorporating both radial distortion and focal length in the formulation will yield a complicated polynomial system in terms of degrees and number of variables. However, with recent progress in Gröbner basis and the action matrix methods, it is possible to derive an efficient minimal solver, as we will illustrate later.

6.1 Introduction

Given two images, we are aiming to estimate the relative motion between two cameras from feature matching. Solving the relative pose problem is one essential step in building a large-scale 3D reconstruction system, for example Snavely et al. (2006). Radial distortion, if present, is a non-negligible factor when estimating the relative pose. If radial distortion is not modeled appropriately, the epipolar constraints have to be set loosely enough to find a sufficiently number of inlier feature matches, which will possibly include outliers and cause significant skewness in a Structure-from-Motion (SfM) pipeline, see Fitzgibbon (2001).

In this chapter, we study the minimal problem of estimating the essential matrix between two cameras with the same but unknown focal length and radial distortion. This is a general and useful setting when images are captured by a single camera with distortion.

6.1.1 Related Works

Minimal problems arise when computing geometric models from image data. One aims to use a minimal set of feature matches to estimate epipolar geometry. Both the five-point algorithm proposed in Nistér (2004) for estimating the essential matrix E and the seven-point algorithm for solving the fundamental matrix F are minimal solvers. However, solving a minimal problem usually involves some higher-order algebraic equations, for example, the determinant constraint on F or the trace constraint on E. Instead of incorporating the algebraic equations, the eight-point algorithm proposed in Longuet-Higgins (1981) solves a simpler linear system from a non-minimal set of measurements in order to estimate the fundamental matrix. Due to its sensitivity to noise, the normalized eight-point algorithm presented in Hartley (1997) is usually a better choice.

When radial distortion is considered, early methods also ignore the higher-order constraints and instead solve a simpler non-minimal problem. For example, in Fitzgibbon (2001), a non-minimal algorithm is proposed to simultaneously estimate F and the single radial distortion parameter. The division model proposed in Fitzgibbon (2001) for radial distortion is extensively used in later works. In Li and Hartley (2005), a hidden-variable method is used to solve the same problem as in Fitzgibbon (2001). A 15-point non-minimal solver is proposed in Barreto and Daniilidis (2005). It estimates the so-called 4×4 radial fundamental matrix, which models the bilinear relationship between the lifted image point in one view and the corresponding epipolar curve in the other view.

More recent works focus on solving minimal problems. In Kukelova and Pajdla (2007a), an eight-point minimal solver is given to estimate F and the parameter λ of radial distortion in the uncalibrated case. In Kukelova and Pajdla (2007b), two minimal problems are proposed; one is estimating E and λ assuming that two cameras are partially calibrated with known focal length f, the other is estimating the fundamental matrix and radial distortion assuming that two cameras have different distortion parameters λ_1 and λ_2 in the uncalibrated case. The exact rational arithmetic solvers are given for both problems in Maple, which is slow and hardly practical. In K. Et al. (2010), based on an efficient implementation of the Gröbner basis method in floating point arithmetic, two solvers are proposed for the same two problems as in Kukelova and Pajdla (2007b). In Kuang et al.

(2014), the one-sided minimal problems are studied, assuming that one of the radial distortion parameters, λ_1 and λ_2, is known. Three minimal problems are proposed and solved: the uncalibrated case using eight points, the calibrated case with unknown focal length f using seven points and the calibrated case with known focal length f using six points.

6.2 Problem Formulation

6.2.1 Camera Model and Radial Distortion

In our formulation, we use the pin-hole camera model, see Section 2.1. For the calibration matrix $\mathtt{K}$ defined in (2.7), we assume that the skew s is zero and that the aspect ratio γ is unity. In addition, the principal point given by (p_x, p_y) is assumed to lie in the centre of the image, that is $(0, 0)$. Based on these assumptions, we could parameterize $\mathtt{K}$ using a single parameter w as

$$\mathtt{K} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & w \end{bmatrix} \tag{6.1}$$

where $w = 1/f$. Note this can be done since $\mathtt{K}$ is only defined up to an unknown scale.

For radial distortion, we use the one-parameter division model proposed in Fitzgibbon (2001). If we denote an undistorted image point as $\mathbf{x}_u = (x_u, y_u)^{\mathrm{T}}$ and the corresponding radially distorted image point as $\mathbf{x}_d = (x_d, y_d)^{\mathrm{T}}$, both using inhomogeneous coordinates, then the relationship between $\mathbf{x}_d$ and $\mathbf{x}_u$ is given by the division model as

$$\mathbf{x}_u = \frac{1}{1 + \lambda \|\mathbf{x}_d\|^2} \mathbf{x}_d \tag{6.2}$$

where λ is the distortion coefficient and $\|\mathbf{x}_d\|$ is the distance from $\mathbf{x}_d$ to the centre of distortion, which is assumed to be at the image center, that is $(0, 0)$.

If we use homogeneous coordinates notated as $\mathbf{p}_u = (x_u, y_u, 1)^{\mathrm{T}}$, $\mathbf{p}_d = (x_d, y_d, 1)^{\mathrm{T}}$ and $r_d = \|\mathbf{x_d}\|$, then (6.2) can be written as

$$\begin{bmatrix} x_u \\ y_u \\ 1 \end{bmatrix} \sim \begin{bmatrix} x_d \\ y_d \\ 1 + \lambda r_d^2 \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} + \lambda \begin{bmatrix} 0 \\ 0 \\ r_d^2 \end{bmatrix}, \tag{6.3}$$

that is

$$\mathbf{p}_u \sim \mathbf{p}_d + \lambda \mathbf{z}, \tag{6.4}$$

where $\mathbf{z} = (0, 0, r_d^2)^{\mathrm{T}}$ is known since $\mathbf{x}_d$ is given. We also need the normalized image point $\mathbf{p}_n$, which can be represented as

$$\mathbf{p}_n \sim \mathtt{K}^{-1}\mathbf{p}_u. \tag{6.5}$$

It is well known that for two calibrated cameras, the essential matrix $\mathtt{E}$ has five degrees of freedom, see Hartley and Zisserman (2004). For two cameras with the same but unknown focal length f and radial distortion λ, the problem has a total of seven degrees of freedom. For the minimal problem, we therefore need seven feature correspondences to solve for $\mathtt{E}$, f and λ.

6.2.2 Parameterization and Formulation

Given undistorted feature correspondences $\{(\mathbf{p}_{u_i}, \mathbf{p}'_{u_i})\}$ for $i = 1, \cdots, n$, the epipolar constraints on the fundamental matrix $\mathtt{F}$ are given as

$$\mathbf{p}_{u_i}^{\mathrm{T}}(\lambda)\ \mathtt{F}\ \mathbf{p}'_{u_i}(\lambda) = 0, \tag{6.6}$$

where $\mathbf{p}_u(\lambda)$ and $\mathbf{p}'_u(\lambda)$ are parametrized using (6.4). Using the normalized image points $\mathbf{p}_{n_i} \sim \mathtt{K}^{-1}\mathbf{p}_{u_i}$, the epipolar constraint can be equivalently expressed using the essential matrix $\mathtt{E}$ as

$$\mathbf{p}_{n_i}^{\mathrm{T}}(\lambda, w)\ \mathtt{E}\ \mathbf{p}'_{n_i}(\lambda, w) = 0. \tag{6.7}$$

Instead of directly parameterizing the essential matrix $\mathtt{E}$, we parameterize $\mathtt{F}$ and solve for $\mathtt{E}$ implicitly. The motivation is to eliminate parameter w of the focal length using the constraint (6.6) instead of (6.7). This leads to a polynomial system that can be simplified using a linear elimination strategy. To parameterize $\mathtt{F}$, we have

$$\mathtt{F} = \begin{bmatrix} f_1 & f_4 & f_7 \\ f_2 & f_5 & f_8 \\ f_3 & f_6 & 1 \end{bmatrix}, \tag{6.8}$$

where the last element is set to one to fix the scale. From the relationship $\mathtt{E} = \mathtt{K}^{\mathrm{T}}\mathtt{F}\mathtt{K}'$ where $\mathtt{K}' = \mathtt{K}$ in our case, the essential matrix $\mathtt{E}$ is parameterized using $\mathtt{F}$ and $\mathtt{K}$ defined in (6.1) as

$$\mathtt{E} = \mathtt{K}^{T}\mathtt{F}\mathtt{K}' = \begin{bmatrix} f_1 & f_4 & wf_7 \\ f_2 & f_5 & wf_8 \\ wf_3 & wf_6 & w^2 \end{bmatrix}. \tag{6.9}$$

The essential matrix $\mathtt{E}$ has zero determinant, which gives

$$\det(\mathtt{E}) = 0. \tag{6.10}$$

We also hold that two non-zero singular values of $\mathtt{E}$ are equal, which is equivalent to the following trace constraint,

$$2(\mathtt{E}\mathtt{E}^{T})\mathtt{E} - \mathrm{trace}(\mathtt{E}\mathtt{E}^{T})\mathtt{E} = 0. \tag{6.11}$$

Inserting (6.9) into (6.10) and (6.11), together with the epipolar constraints in (6.6), we formulate the problem as solving a polynomial equation system that contains 17 equations (seven from (6.6), that is, one from (6.10) and nine from (6.11)) in ten unknowns, namely, $\{f_1, f_2, ..., f_8, \lambda, w\}$.

6.2.3 Eliminating Variables

Let us look at the equations given by the epipolar constraints in (6.6), or equivalently

$$\begin{bmatrix} x_d \\ y_d \\ 1 + \lambda r_d^2 \end{bmatrix}^{T} \begin{bmatrix} f_1 & f_4 & f_7 \\ f_2 & f_5 & f_8 \\ f_3 & f_6 & 1 \end{bmatrix} \begin{bmatrix} x'_d \\ y'_d \\ 1 + \lambda r'^2_d \end{bmatrix} = 0. \tag{6.12}$$

By expanding the above multiplication and stacking all seven equations, one can rewrite it as the following form

$$A\mathbf{x} = 0, \tag{6.13}$$

where $\mathbf{x} = (\lambda^2, \lambda f_3, \lambda f_6, \lambda f_7, \lambda f_8, f_1, \ldots, f_8, \lambda, 1)$ is a vector of monomials and A is the coefficient matrix. After applying Gaussian elimination to (6.13), one can linearly eliminate seven monomials by expressing them in

terms of the remaining eight monomials (including the constant 1). Since f_1, f_2, f_4, f_5 appear only as linear terms in (6.12), it is natural to eliminate them. In addition, we choose to eliminate $f_3, \lambda f_3$ and λ^2. This choice will simplify the system as we will demonstrate. The eliminating monomials $\{f_1, f_2, f_3, f_4, f_5, \lambda f_3, \lambda^2\}$ can now be represented as a linear combination of the remaining monomials, namely, $\{\lambda f_6, \lambda f_7, \lambda f_8, f_6, f_7, f_8, \lambda\}$, or the equivalent, each of which can be expressed as a quadratic function of the unknown variables $\{f_6, f_7, f_8, \lambda\}$. For $f_1, f_2, \ldots, f_5$, we have

$$f_i = h_i(f_6, f_7, f_8, \lambda), \quad i = 1, 2, 3, 4, 5 \tag{6.14}$$

For λf_3 and λ^2, we have

$$\lambda f_3 = h_6(f_6, f_7, f_8, \lambda), \tag{6.15}$$

and

$$\lambda^2 = h_7(f_6, f_7, f_8, \lambda), \tag{6.16}$$

One can further eliminate f_3 from (6.15) by replacing it with $h_3(f_6, f_7, f_8, \lambda)$, that is,

$$\lambda h_3(f_6, f_7, f_8, \lambda) - h_6(f_6, f_7, f_8, \lambda) = 0. \tag{6.17}$$

Therefore, one can now substitute f_i into (6.9) with $h_i(f_6, f_7, f_8, \lambda)$ for $i = 1, 2, \ldots, 5$ and insert (6.9) into (6.10) and (6.11). Together with (6.16) and (6.17), we obtain a well-defined system of 12 polynomial equations with five unknowns, that is $\{f_6, f_7, f_8, \lambda, w\}$. The equations are at most of the ninth degree.

Solving such a polynomial system is certainly a non-trivial task. It is a more complicated system than those presented in previous works. The formulation in Kukelova and Pajdla (2007a) for the uncalibrated case gives three equations with three unknowns, the degree of which is five. The polynomial system in Kuang et al. (2014) for the calibrated case, assuming that the radial distortion and focal length of one view are already known, contains 11 equations of degree six with four unknowns if the other focal length is unknown. With recent progress in the Gröbner basis method, we will demonstrate in the next section that it is possible to find an efficient and stable solver for our minimal problem.

6.3 A Polynomial Solver

We first study and explore the structure of the polynomial system generated from (6.10), (6.11), (6.16) and (6.17). One observation is that of the system's twelve equations, four have w as a common multiplier; to avoid the trivial and false solution $w = 0$, we divide all four equations by w. After that, we find that the equations that still contain w now only contain the even-degree terms of w, for example, w^2, w^4 or w^6. Therefore, we could simply replace w^2 with a new variable z, that is $z = w^2$. This would reduce the degrees of the polynomial system from nine to seven.

We then verify the number of solutions using Macaulay2, software (see Grayson and Stillman (1993-2002)). By generating a polynomial system with coefficients in $\mathbb{Z}_p$, we find that there are in general 68 solutions for this problem. Although the number of solutions is relatively large, we will later demonstrate in the experiments that most solutions to the polynomial system are complex and thus can be ignored without any further validation.

To solve the system, we follow the method based on Gröbner basis, see Byröd et al. (2009). A redundant set of higher order polynomials, called the *eliminating template*, is systematically generated from our initial polynomial equations by multiplying them by a set of monomials. By doing this, one generates a sufficiently large set of monomials, denoted $\mathcal{M}$, from which one can find the set of *basis* monomials, denoted $\mathcal{B}$. All the monomials that are not in $\mathcal{B}$ can be represented as a linear combination of basis monomials. The *permissible* set, denoted $\mathcal{P}$ comprises the monomials that remain in $\mathcal{M}$ after multiplying by a certain variable, x_k, called the action variable. Empirically, we find that using the following rules to generate the elimination template yields a numerically stable solver.

- The highest degrees of the multiplication monomials are $\{4, 4, 4, 2, 4\}$ for $\{f_6, f_7, f_8, w, \lambda\}$ respectively.
- The highest degree of the equations is nine after multiplication.

After this multiplication step, the resulting elimination template contains 886 equations with 1011 monomials. Further attempting to reduce the size of the elimination template, for example, by decreasing the highest degrees of the multiplication monomials (as we have tried), will affect the numerical stability.

We use a column-pivoting scheme, see Byröd et al. (2009) for basis selection to improve numerical stability. The permissible set contains 120 monomials from which 68 are selected as a basis. An action matrix of size 68×68 is constructed. By performing eigen-decomposition on the action matrix, we can extract the solutions to our system from the resulting eigenvectors. Once we obtain the values of $\{f_6, f_7, f_8, w, \lambda\}$, the other unknowns of the fundamental matrix `F` can be found using (6.14). The essential matrix `E` is then solved also using (6.9).

6.4 Experiments

All the experiments are performed on a Macbook Air with 8GB of memory and a 1.8 GHz Intel Core i5 CPU. The average running time of our minimal solver is around 400ms. We implement the minimal solvers in MATLAB. However, one could further reduce the running time by implementing the solver in C or C++. Using a different optimization strategy, for example Kuang and Åström (2012); Naroditsky and Daniilidis (2011) is also a possibility.

6.4.1 Synthetic Data

We generate random 3D points within a cube of width 1000 units centered at the origin. Two cameras are placed around 1000 units away from the origin and 300 units away from each other. The viewing directions of two cameras point toward the origin. Both cameras have the same focal length, which is randomly generated to be around 1000 units. The 3D points are projected onto the image planes of two cameras, both 1000×1000 units in size. Radial distortion with a coefficient of λ uniformly sampled from $[-0.5, 0]$ is applied to the normalized image points, which are roughly in the range of $[-1, 1]$.

The first experiment is to test the numerical stability and validate the number of real solutions. For this experiment, we use noise-free image points, and 1000 synthetic scenes are randomly generated as described above. For each scene, we run our seven-point minimal solver and evaluate the relative errors of the distortion coefficient λ and the focal length f. The

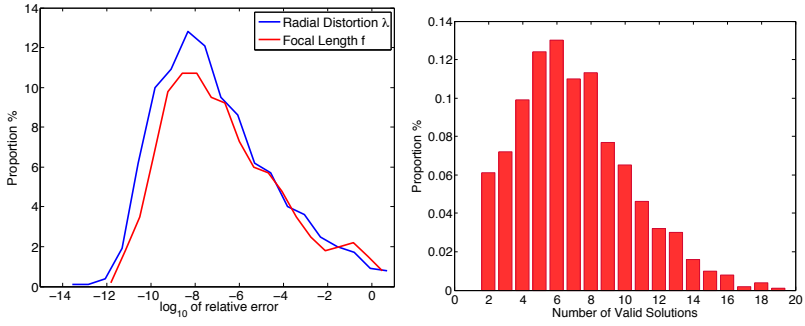


Figure 6.1: Experiments on synthetic data without noise. Left: The distribution of log_{10} relative error of radial distortion λ and the focal length f. Right: The distribution of number of real-valued solutions

relative error of λ is defined as

$$e(\lambda_{est}) = \frac{|\lambda_{est} - \lambda_{gt}|}{|\lambda_{gt}|}, \tag{6.18}$$

where λ_{est} and λ_{gt} are the estimate and ground truth of λ respectively. The relative error of f is defined in a similar way.

We plot the distribution of the $\log_{10}$ relative error of λ and f in Figure 6.1(left). One can see that the solver is generally very stable and accurately estimates both λ and f. The medians of the $\log_{10}$ relative errors are -7.49 for λ and -7.16 for f. The statistic of the number of valid solutions is also plotted in Figure 6.1(right). By valid solutions, we mean the real roots of the polynomial system. Although the system has a total of 68 solutions, most of them are complex and can be simply ignored without further validation. In most cases, there are two to twelve valid solutions.

The performance of the solver in the presence of noise is also tested. The image points are perturbed by Gaussian noise of various standard deviations, that is $\sigma \in \{0, 0.01, 0.1, 0.5, 1, 2\}$ respectively. We tried different radial distortion parameters of $\lambda \in \{-0.01, -0.1, -0.2, -0.5\}$ as well. In Figure 6.2, we illustrate the $\log_{10}$ relative error of λ in a box plot. With noise-free data, the solver in general gives accurate estimates of the radial distortion of λ. We also note that the solver achieves more accurate results when λ has a large absolute value, that is the image has significant

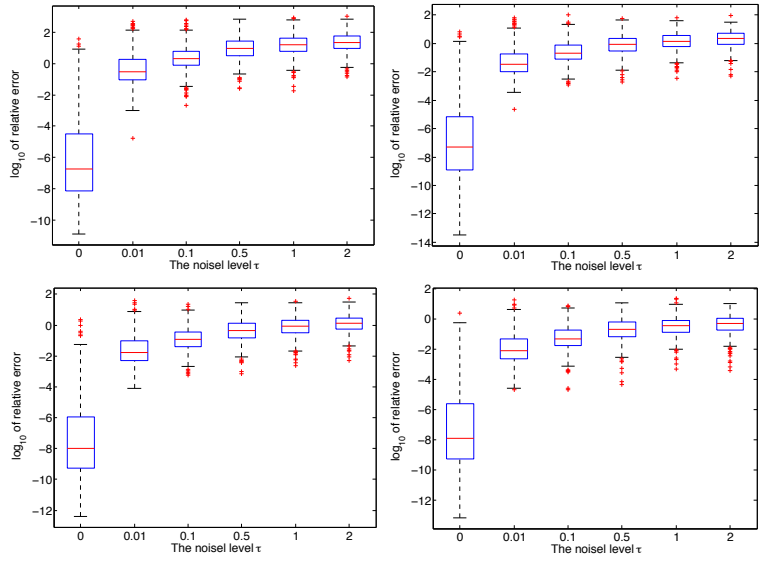


Figure 6.2: Experiments on synthetic data with different noise levels. The log_{10} of relative error of λ for different radial distortion, where the ground truth are $\lambda = -0.01$ (Top left), $\lambda = -0.1$ (Top right), $\lambda = -0.2$ (Bottom left) and $\lambda = -0.5$ (Bottom right). See the text for detailed description.

radial distortion. When the noise level increases, the $\log_{10}$ relative errors also increase greatly, to around 10^{-1} or 10^{0}. This suggests that one should use the solver on repeatedly drawn minimal samples and combine it with either the kernel voting scheme or RANSAC when the data contain noise or outliers. We also test the noise sensitivity of estimating the focal length f and obtain similar results.

6.4.2 Real Data

We further evaluate the proposed solver on real images. A GoPro Hero3 camera is used to capture images with significant radial distortion. The camera is calibrated with a fixed focal length, which serves as ground truth for evaluation. A set of 36 paired images is used to test the solver. SIFT features are extracted from all the images and matched between each pair as in Lowe (2004). This produces a set of tentative matches that also contains outliers.

We first use RANSAC with our minimal solver to estimate the essential matrix $\mathtt{E}$, the focal length f and the radial distortion λ. In each iteration of RANSAC, we estimate $\mathtt{E}$, f and λ and obtain an inlier set by including the correspondences the algebraic distances of which are below a threshold. The solution with the largest inlier set gives the final estimates of $\mathtt{E}$, f and λ. For comparison, we use RANSAC with the standard seven-point algorithm, which does not consider radial distortion. For both methods, the number of RANSAC iterations is set to 1000 and the inlier threshold is set to three pixels. The detected inliers for both methods are illustrated in Figure 6.3. One can see that our minimal solver obtains a substantially larger inlier set due to the explicit modeling of radial distortion. Quantitatively, our minimal solver gains a 49.37% increase in the number of inliers, on average, compared with the standard seven-point solver, when applied to the image set.

We also use a kernel voting scheme to estimate $\mathtt{E}$, λ and f. To avoid computing the inlier set in each iteration of RANSAC, the kernel voting scheme fits a Gaussian kernel to the distribution of all the roots obtained from different random samples. The motivation is that all the solved real roots are around the genuine solution regardless of noise. In other words, it reveals a peak in the distribution of all the real roots obtained from different measurements.

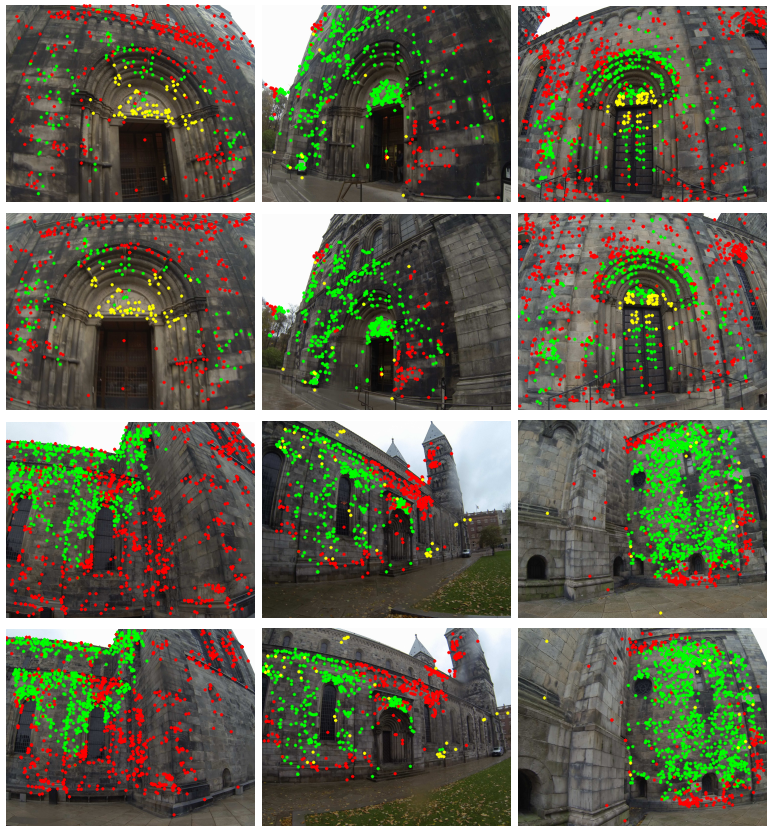


Figure 6.3: Experiments on read data using RANSAC. The inlier set using different methods are marked. The green marks are inliers found by both methods. The red marks are the inliers only found using our solver, The yellow starts are the inliers only found by standard 7 point solver. Note that the images are paired for the top two rows and bottom two rows.

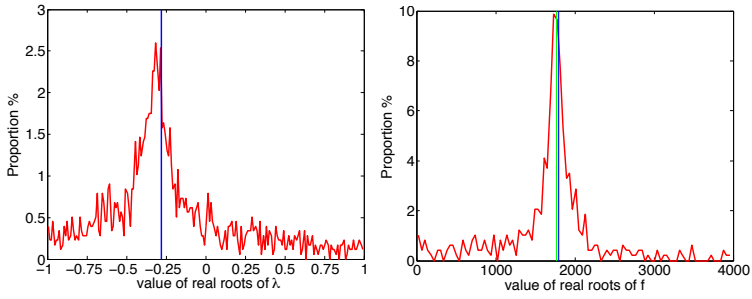


Figure 6.4: Experiments on read data using kernel voting. The distribution of real roots (red curve) for radial distortion coefficient λ and focal length f are shown in the left and right figure respectively. The estimate of λ and f (blue line) as well as the ground truth of f (green line in the right figure) is also marked.

In this experiment, we randomly draw 1000 minimal sets, solve them and keep all the real roots. Gaussian kernels with a bandwidth of $w = 0.02$ for the parameter of radial distortion λ and $w = 200$ for the focal length f are fitted to the distribution of all the estimates of λ and f respectively. The peaks of the Gaussian kernels are picked out as the final estimate of λ and f, respectively. From the distribution of real roots of λ and f, see Figure 6.4, one could easily see the peak shape. The kernel voting gives an estimates of -0.28 for λ and of 1793.4 for f, which is very close to the group truth value of 1765.6. Note that we only plot the results for one of the image pairs, as we have tested, all the other image pairs give consistent estimates of λ and f.

6.5 Conclusions

This chapter described an efficient and numerically stable solver for the minimal problem of relative pose with unknown focal length and radial distortion. More specifically, we derived a parametrization and formulated the problem as solving a polynomial system. The system was simplified using linear elimination. The solver was constructed based on a Gröbner basis method and evaluated both on synthetic data and real images. This

proved the solver to be numerically stable and, when used in RANSAC, it finds more inlier feature correspondences than a standard seven-point minimal solver, while giving an accurate estimate of radial distortion and focal length.

Chapter 7

A Brute-Force Algorithm for Relative Pose Estimation

In this chapter, we continue with the relative pose problem, but from a different perspective. Unlike RANSAC, we are aiming to find a globally optimal solution to the relative pose problem. Based on a non-standard epipolar parameterization, we propose a brute-force algorithm that conducts an exhaustive search in a discretized low-dimensional parameter space. The algorithm is robust and can handle several cost functions, for example, maximizing the consensus set or some robust norm such as the truncated L_2-norm. The computations are easily parallelizable, leading to an efficient algorithm. It can also be adapted to certain restricted motions, such as planar motion and does not suffer from algorithmic degeneracy, unlike minimal solvers.

7.1 Introduction

Already in 1981, Longuet-Higgins suggested a simple and yet elegant solution to the problem of finding the relative pose of two viewpoints in Longuet-Higgins (1981). The algorithm, known as the eight point algorithm, still plays a major role in computer vision, see Hartley and Zisserman (2004). However, the algorithm suffers from many degeneracies, for example, if the scene is planar then the algorithm fails. Perhaps even more serious is that the algorithm assumes that the correspondence problem is already solved. Therefore, more robust approaches have been developed to cope with outliers. Here RANSAC methods using minimal solvers are considered to be state-of-the-art, see Nistér (2004); Chum and Matas (2000). Still, the problem of algorithmic degeneracies remains for minimal

solvers.

Another problem that has been recognized by several researchers is the importance of optimizing a suitable cost function, where costs based on reprojection errors are preferable to algebraic errors, see Hartley and Zisserman (2004). Bundle adjustment does optimize the statistically correct criterion (given that measurement errors are independent and normally distributed), but the method is sensitive to initialization. Therefore global optimization algorithms have been developed in Kahl and Hartley (2008); Hartley and Kahl (2009); Enqvist and Kahl (2009); Chiuso et al. (2000) which are not susceptible to local minima.

Looking back over more than 30 years of algorithmic development since the eight point algorithm proposed in Longuet-Higgins (1981), a set of criteria has emerged that we believe a good relative pose algorithm should possess

(i) Robustness to outliers,

(ii) No algorithmic degeneracies,

(iii) Cost function based on reprojection errors,

(iv) Not dependent on a good initialization,

(v) Practical.

For example, RANSAC is designed to fulfill *(i)*, requires no initialization *(iv)* and has been successfully applied in many real systems *(v)*, but the method does not meet objectives *(ii)* and *(iii)*. Similarly, recent global relative pose methods in Hartley and Kahl (2009); Enqvist and Kahl (2009) do meet criteria *(ii)-(iv)*, but cannot be considered practical *(v)* since the running times are in some cases extreme (in the order of minutes). In practice, a heuristic combination of different algorithms is often used to overcome the difficulties in fulfilling these objectives. For example, homographies are often used to detect if the scene is planar or if the motion is a pure rotation. Another example of this phenomenon is that particular motions have been examined separately in Vedaldi et al. (2007).

In this chapter, an algorithm using exhaustive search is developed that fulfills criteria *(i)-(iv)* by design. For example, provided that the discretization of the parameter space is fine enough, the method is guaranteed to find the globally optimal solution. The key idea in order to make it practical is that the expensive computations are done in lower dimensions, and only

very simple calculations are required in the high-dimensional search. The ultimate proof is of course by showing that the algorithm works in real experiments - this is done in the experimental section. In particular, when applied to 3D motion segmentation, our approach significantly outperforms state-of-the-art methods on 104 video sequences in the Hopkins 155 database, see Tron and Vidal (2007). Note that the database contains a variety of relative motions and scenes that are considered degenerate for several of the above standard algorithms.

7.2 Preliminaries

Consider two views of a scene. Let $\mathbf{x}$ denote and image point - represented by a unit vector - in the first view and $\mathbf{x}'$ the corresponding image point in the second view. The assumption is that these are both the projection of some 3D point X. We can always choose a global coordinate system such that the first camera lies at the origin and the second camera at $\mathtt{t} = (0, 0, 1)^{\mathrm{T}}$, which gives

$$\lambda \mathtt{R}\mathbf{x} = X, \quad \lambda' \mathtt{R}'\mathbf{x}' = X - \mathtt{t}, \tag{7.1}$$

where $\mathtt{R}$ and $\mathtt{R}'$ are 3×3 rotation matrices and λ and λ' are positive real numbers. To simplify the derivation, we will assume that there are no points at infinity, although the method works as well with points at infinity. We illustrate an example in Figure 7.1(a) where the first camera is located at the origin and the second camera is located at $(0, 0, t)^{\mathrm{T}}$. A 3D point X is projected onto the spherical image plane at $\mathbf{x}$ and $\mathbf{x}'$ respectively.

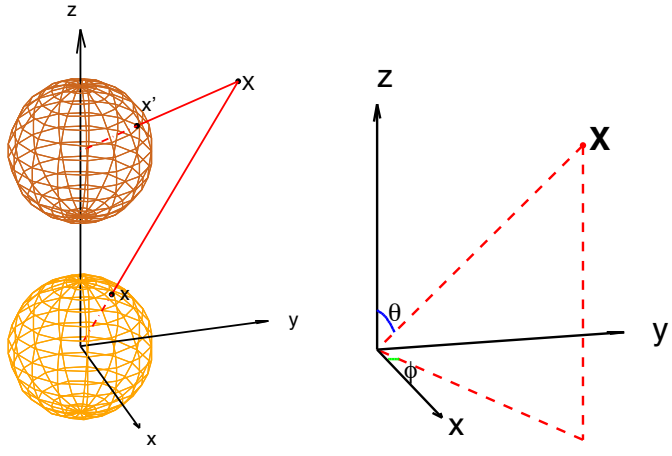
Note that the standard parametrization on relative pose uses a rotation matrix and a unit translation vector, whereas we use two rotation matrices as in (7.1). This is obviously an over-parametrization, but we will cover that later. We first look at the constraint on the relative pose induced by a pair of corresponding image points.

Theorem 7.1. *Let* $\mathtt{R}$ *and* $\mathtt{R}'$ *be rotation matrices with row vectors* $\mathbf{r}_1^T$, $\mathbf{r}_2^T$, $\mathbf{r}_3^T$ *and* $\mathbf{r'}_1^T$, $\mathbf{r'}_2^T$, $\mathbf{r'}_3^T$ *respectively and* $\mathbf{x}$ *and* $\mathbf{x}'$ *corresponding points. Then*

$$(\mathbf{r}_1^T\mathbf{x}, \mathbf{r}_2^T\mathbf{x}) = k(\mathbf{r'}_1^T\mathbf{x}', \mathbf{r'}_2^T\mathbf{x}') \text{ with } k > 0, \tag{7.2a}$$

$$\mathbf{r}_3^T\mathbf{x} > \mathbf{r'}_3^T\mathbf{x}', \tag{7.2b}$$

if and only if there exists a 3D point $\mathbf{X}$ *satisfying* (7.1)



(a) (b)

Figure 7.1: (a) A 3D point X projects to two camera at $(0,0,0)^{\mathrm{T}}$ and $(0,0,t)^{\mathrm{T}}$. (b) spherical coordinate.

Proof. We first proof the *only if* part that (7.2) $\Rightarrow$ (7.1). Clearly

$$k = \frac{\|(\mathbf{r}_1^{\mathrm{T}}\mathbf{x}, \mathbf{r}_2^{\mathrm{T}}\mathbf{x})\|}{\|(\mathbf{r}'^{\mathrm{T}}_1\mathbf{x}', \mathbf{r}'^{\mathrm{T}}_2\mathbf{x}')\|} = \sqrt{\frac{1-(\mathbf{r}_3^{\mathrm{T}}\mathbf{x})^2}{1-(\mathbf{r}'^{\mathrm{T}}_3\mathbf{x}')^2}} < 1. \tag{7.3}$$

Let λ be the solution to

$$\lambda \mathbf{r}_3^{\mathrm{T}}\mathbf{x} - 1 = \lambda k \mathbf{r}'^{\mathrm{T}}_3\mathbf{x}', \tag{7.4}$$

and put $\lambda' = \lambda k$. From $k < 1$ and $\mathbf{r}_3^{\mathrm{T}}\mathbf{x} > \mathbf{r}'^{\mathrm{T}}_3\mathbf{x}'$ it is straightforward to show that $\lambda > 0$ and hence $\lambda' > 0$. Now let $X = \lambda \mathtt{R}\mathbf{x}$. To see that (7.1) is satisfied, consider

$$X - \mathtt{t} = \lambda \mathtt{R}\mathbf{x} - \mathtt{t} = \begin{pmatrix} \lambda \mathbf{r}_1^{\mathrm{T}}\mathbf{x} \\ \lambda \mathbf{r}_2^{\mathrm{T}}\mathbf{x} \\ \lambda \mathbf{r}_3^{\mathrm{T}}\mathbf{x} - 1 \end{pmatrix}, \tag{7.5}$$

but by (7.2a) and (7.4), this is equal to

$$\begin{pmatrix} \lambda k \mathbf{r'}_1^{\mathrm{T}} \mathbf{x'} \\ \lambda k \mathbf{r'}_2^{\mathrm{T}} \mathbf{x'} \\ \lambda k \mathbf{r'}_3^{\mathrm{T}} \mathbf{x'} \end{pmatrix} = \lambda' \mathtt{R}' \mathbf{x}'. \tag{7.6}$$

This proves that *only if* part. The *if* part that (7.2) $\Leftarrow$ (7.1) can be easily seen by rewriting (7.1) as

$$\lambda \begin{pmatrix} \mathbf{r}_1^{\mathrm{T}} \mathbf{x} \\ \mathbf{r}_2^{\mathrm{T}} \mathbf{x} \\ \mathbf{r}_3^{\mathrm{T}} \mathbf{x} \end{pmatrix} - \lambda' \begin{pmatrix} \mathbf{r'}_1^{\mathrm{T}} \mathbf{x'} \\ \mathbf{r'}_2^{\mathrm{T}} \mathbf{x'} \\ \mathbf{r'}_3^{\mathrm{T}} \mathbf{x'} \end{pmatrix} = \mathtt{t} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{7.7}$$

□

The description gets even simpler if we switch to spherical coordinates,

$$\mathtt{R}\mathbf{x} = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix}, \quad \mathtt{R}'\mathbf{x}' = \begin{pmatrix} \sin\theta'\cos\phi' \\ \sin\theta'\sin\phi' \\ \cos\theta' \end{pmatrix}. \tag{7.8}$$

where θ, θ' denote the polar angles and ϕ, ϕ' denote the azimuthal angles, see Figure 7.1(b).

Now the necessary and sufficient constraints are

$$\phi = \phi' \text{ and } \theta < \theta'. \tag{7.9}$$

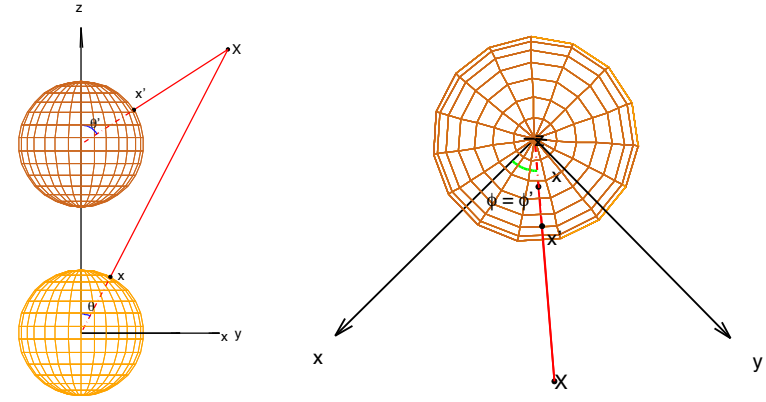
To help intuitively understand these constraint, we illustrate an example in Figure 7.2.

Remark 7.2. *In this work angles are considered equal if they are equal modulo 2π but to simplify the presentation this is not always written explicitly. For example $\xi \in [\alpha, \beta]$ if $\xi + 2\pi k$ does for some $k \in \mathbb{Z}$.*

The next step is to allow measurement errors. Given an error tolerance ϵ, we say that corresponding points x and x' are consistent with a relative pose defined by $\mathtt{R}$ and $\mathtt{R}'$ if there exists a 3D point X such that the reprojection errors are less than ϵ, that is

$$\angle(\mathtt{R}\mathbf{x}, X) < \epsilon \quad \angle(\mathtt{R}'\mathbf{x}', X) < \epsilon. \tag{7.10}$$

The following theorem gives the necessary and sufficient constraints that incorporate measurement errors.



(a) (b)

Figure 7.2: The illustration of Theorem 7.1 in spherical coordinates. (a) The front view of Figure 7.1(b), which shows that $\theta < \theta'$. (b) The top view of Figure 7.1(b), which shows that $\phi = \phi'$

Theorem 7.3. *Consider rotation matrices* $\mathtt{R}$ *and* $\mathtt{R}'$ *and spherical coordinates defined in* (7.8). *Further define* w *in the following way. For* $\theta < \theta'$,

$$w = \arcsin(\sin\epsilon / \sin\theta) + \arccos(\sin\epsilon / \sin\theta'), \tag{7.11}$$

if this is defined and otherwise $w = \pi$. *For* $\theta' < \theta < \theta' + 2\epsilon$

$$w = \arccos\left(\frac{\cos 2\epsilon - \cos\theta\cos\theta'}{\sin\theta\sin\theta'}\right), \tag{7.12}$$

if this is defined and otherwise π. *Then,*

$$\theta < \theta' + 2\epsilon, \tag{7.13}$$

$$\phi \in [\phi' - w, \phi' + w], \tag{7.14}$$

if and only if the angular reprojection errors are less than ϵ.

Proof. Since it is always possible to change coordinates, we can assume that $\mathtt{R} = \mathtt{R}' = I$. Furthermore, we note that if we can find points $\bar{x}$ and $\bar{x}'$ that satisfy the constraints in Theorem 7.3 as well as

$$\angle(\bar{\mathbf{x}}, \mathbf{x}) < \epsilon \text{ and } \angle(\bar{\mathbf{x}}', \mathbf{x}') < \epsilon \tag{7.15}$$

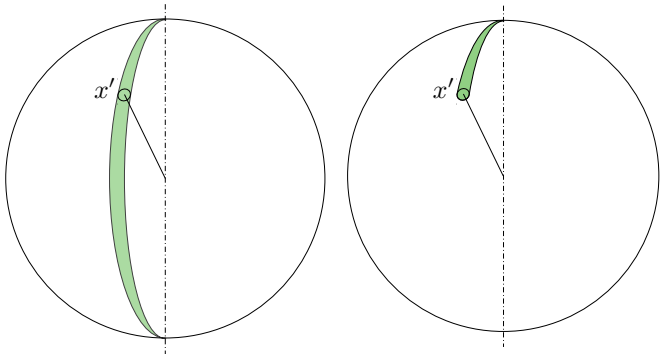


Figure 7.3: The constraints imposed on $\bar{x}$ being the reprojection of the 3D point in first camera. Equation (7.17) constrains $\bar{x}$ to the spherical wedge (left) and (7.16) to the upper part of that wedge (right).

then (by Theorem 7.1) we have also found our point X. This will prove useful. Let $(\bar{\theta}, \bar{\phi})$ and $(\bar{\theta}', \bar{\phi}')$ denote the spherical coordinates of these points as defined in (7.8). We assume that $\bar{x}'$ is fixed and examine what constraints we get on $\bar{x}$. Recall the constraints from Theorem 7.1,

$$\bar{\theta} < \bar{\theta}' \tag{7.16}$$

$$\bar{\phi} = \bar{\phi}' \tag{7.17}$$

From (7.15) we have that $\bar{\mathbf{x}}'$ must lie in a small circle around $\mathbf{x}'$. Consequently, (7.17) means that $\bar{x}$ must lie in the spherical wedge shown on the left in Figure 7.3 and (7.16) constrains it to the upper part of that wedge, as shown on the right in Figure 7.3.

But we also want $\angle(\bar{\mathbf{x}}, \mathbf{x}) < \epsilon$, which constraint $\bar{\mathbf{x}}$ to a small circle around $\mathbf{x}$. This means that we must require the wedge from above to intersect this small circle. To complete the proof we need to translate this constraint to a constraint in the spherical coordinates. We get three cases.

Case 1, $\theta < \theta'$: Figure 7.4 shows the critical case. If the difference between ϕ and ϕ' is larger than this, then the two sets have empty intersection. The limit can be computed by considering two right-angled triangles, see Figure 7.4. Let v denote the blue angle in that figure and v' the yellow one.

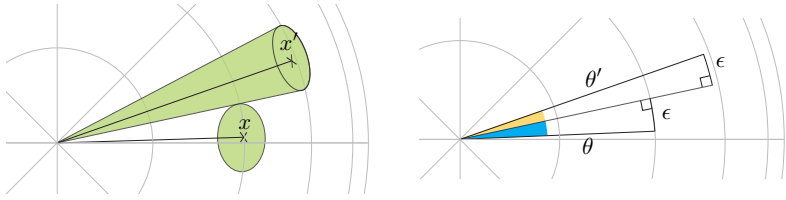


Figure 7.4: Case 1. Here the sphere from Figure 7.3 are viewed from above, i.e. the z-axis is pointing out of the paper. The green areas show the constraints on $\bar{x}$. For the two constraints to intersect they must not be further apart than this. The left image shows the setup for computing this limit angle.

The spherical law of sines yield,

$$\sin v = \frac{\sin \epsilon}{\sin \theta} \text{ and } \sin v' = \frac{\sin \epsilon}{\sin \theta'}. \tag{7.18}$$

and if we define $w = v + v'$, we can write the constraint $|\phi - \phi'| < w$. Note that, if either $\sin \theta$ or $\sin \theta'$ is smaller than $\sin \epsilon$ then w is not defined. In these cases one of the triangles is degenerated and the intersection is non-empty regardless of the ϕ's. One way to describe this is to set $w = \pi$.

Case 2, $\theta' < \theta < \theta' + 2\epsilon$: Figure 7.5 illustrates the critical position. Using the spherical law of cosines, we can compute w,

$$\cos\theta\cos\theta' + \sin\theta\sin\theta'\cos w = \cos 2\epsilon. \tag{7.19}$$

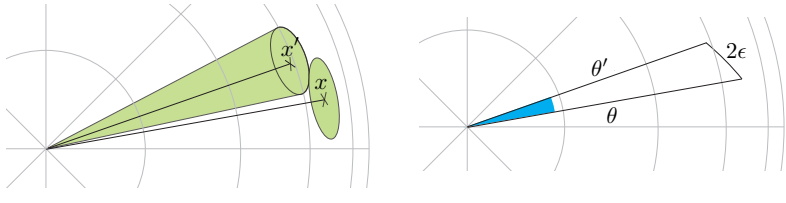


Figure 7.5: Case 2, see caption of Figure 7.4.

Case 3, $\theta \geq \theta' + 2\epsilon$: In this case the intersection is empty, regardless of ϕ and ϕ'.

□

Theorem 7.3 provides a way to handle the constraints induced by one point correspondences. When we have many hypothetical point correspondences, we could formulate the relative pose problem as searching the maximum consensus set which is described below.

Problem 7.4. *Given two sets of image points* $\{\mathbf{x}_i\}$ *and* $\{\mathbf{x'}_j\}$ *with hypothetical correspondences* $(\mathbf{x}_k, \mathbf{x'}_k)$*,* $k = 1, \ldots, N$ *and a prescribed error threshold* ϵ*, compute the relative pose of the cameras which is consistent with as many correspondences as possible.*

7.3 A Search Algorithm

Before proposing our algorithm to Problem 7.4, we first simplify the parametrization of relative pose. Naturally using two rotation matrices to represent a relative pose is an overparametrization. In fact if $\mathtt{S}$ is a rotation about the z-axis, then $(\mathtt{R}, \mathtt{R}')$ and $(\mathtt{SR}, \mathtt{SR}')$ describe the same relative pose. To see this we assume that

$$\lambda \mathtt{R}\mathbf{x} = X, \quad \lambda' \mathtt{R}'\mathbf{x}' = X - \mathtt{t} \tag{7.20}$$

where $t = (0, 0, 1)$ as prescribed. Now

$$\lambda \mathtt{SR}\mathbf{x} = \mathtt{S}X, \quad \lambda' \mathtt{SR}'\mathbf{x}' = \mathtt{S}X - \mathtt{St} = \mathtt{S}X - \mathtt{t}, \tag{7.21}$$

shows that a rotation around z-axis does not change the relative pose but only the global coordinate system.

To simplify the notation, let Γ be a function that maps a given unit vector $\mathbf{r}$ to a rotation matrix Γ_r, having $\mathbf{r}$ as its third row. Then any relative pose can be written as

$$\mathtt{R} = \mathtt{S}_\alpha \Gamma_r \text{ and } \mathtt{R}' = \Gamma_{r'} \tag{7.22}$$

where $\mathtt{S}_\alpha$ is a rotation by α about the z-axis. Hence the set of parameters consists of two unit vectors $\mathbf{r}$ and $\mathbf{r}'$ and an angle α. It is worth to note that $\mathbf{r}$ and $-\mathbf{r}'$ are the epipoles of the two cameras.

Now consider the spherical coordinates in (7.8), only ϕ depends on α. Let $\phi(r)$ denote the value if $\alpha = 0$. This changes the last constraint (7.14) of Theorem 7.3 to

$$\phi(\mathbf{r}) + \alpha \in [\phi'(\mathbf{r}') - w, \phi'(\mathbf{r}') + w], \tag{7.23}$$

which is easily translated to a constraint $\alpha \in [\alpha_{lo}, \alpha_{up}]$, where

$$\begin{aligned} \alpha_{lo} &= \phi'(\mathbf{r}') - \phi(\mathbf{r}) - w \\ \alpha_{up} &= \phi'(\mathbf{r}') - \phi(\mathbf{r}) + w \end{aligned} \tag{7.24}$$

Each pair of corresponding points yields such an interval and by sorting these lower and upper bounds for all correspondences, one can find the interval having the maximal number of inliers; see Algorithm 6.

Algorithm 5 Brute-Force Search

For a given level of discretization and error threshold ϵ, a relative pose having the maximal number of inliers $n_{\max}$ is computed.

Compute a discretization, $\mathcal{D}$ of S^2.
For each $\mathbf{r} \in \mathcal{D}$
  For each $\mathbf{x}$
    Compute $\phi(\mathbf{r})$, $\theta(\mathbf{r})$ and $v(\mathbf{r})$.
  For each $\mathbf{x}'$
    Compute $\phi'(\mathbf{r})$, $\theta'(\mathbf{r})$ and $v'(\mathbf{r})$.
Put $n_{\max} = 0$
For each pair $(\mathbf{r}, \mathbf{r}') \in \mathcal{D} \times \mathcal{D}$
  For each correspondence $(\mathbf{x}, \mathbf{x}')$
    If $\theta'(\mathbf{r}') + \epsilon > \theta(\mathbf{r}) - \epsilon$
      Compute $w = v(\mathbf{r}) + v'(\mathbf{r}')$.
      A lower bound $\alpha_{lo} = \phi'(\mathbf{r}') - \phi(\mathbf{r}) - w$.
      An upper bound $\alpha_{up} = \phi'(\mathbf{r}') - \phi(\mathbf{r}) + w$.
  Find the max intersection n, using Algorithm 6.
  If $n > n_{\max}$
    Store the current parameters.
    Set $n_{\max} = n$.

Now we are ready to look at the complete algorithm; see Algorithm 5. The idea is to perform a search for vectors $\mathbf{r}$ and $\mathbf{r}'$. Since both vectors have unit length they lie in S^2, being the unit sphere in $\mathbb{R}^3$, and the search space is a discretized version of $S^2 \times S^2$. For each pair $(\mathbf{r}, \mathbf{r}')$ we compute the lower and upper bounds on α which are given in (7.24). We sort these bounds and compute the maximal number of inliers. Hence the complexity of the algorithm is $O(k^2 m \log(m))$ where k is the number of points in the discretization of S^2 and m is the number of correspondences.

Remark 7.5. *As it matters only rarely and complicates the description, we ignore case 2 of Theorem 7.3 in the computation of w. Thus w can always be divided into*

$$w = v + v' \tag{7.25}$$

where v does not depend on $\mathbf{x}$ and v' does not depend on $\mathbf{x}'$.

Algorithm 6 Maximal Intersection

Given lower bounds $\mathcal{L}$ and upper bounds $\mathcal{U}$, a point is found that lies in as many intervals as possible. Outputs the number of intersecting intervals, n and the point.

Sort $\mathcal{L}$ and $\mathcal{U}$.
Initialize $j = 1$ and $n = 0$.
For $i \in \{1, \ldots, |\mathcal{L}|\}$
  While $\mathcal{U}_j < \mathcal{L}_i$
    Increase $j = j + 1$.
  If $i - j > n$
    Store $\mathcal{L}_i$.
    Set $n = i - j$.

7.4 Other Cost Functions

So far we have simply counted the number of inliers to assess the quality of a relative pose. Inliers are correspondences with the reprojection errors less than some prescribed threshold, ϵ. This approach is simple and generally yields good results, but it does have its limitations; see Hartley and Zisserman (2004). One problem is that the method might be sensitive to the choice of ϵ, but also that the distribution of the inlier errors are not considered. In Blake and Zisserman (1987) a more refined cost function is proposed. The assumption is that correct matchings have a clock-shaped error distribution similar to the Gaussian distribution, whereas incorrect matchings have approximately uniformly distributed errors. These assumptions lead to the cost function

$$C(d) = -\log\left(c + \exp\left(-d^2\right)\right) \tag{7.26}$$

where d is the reprojection error; see Figure 7.6. In the same book it is noted that a good approximation of this cost function can be obtained by truncating the ordinary squared error. A cost function of this kind cannot

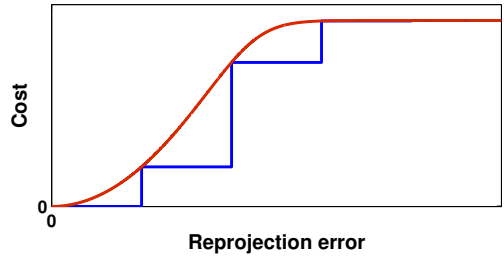


Figure 7.6: The robust cost function (red) suggested in Blake and Zisserman (1987), and a piecewise constant approximation of it (blue) which can be optimized using the proposed framework.

be handled directly by the proposed method, but one can approximate the function to arbitrary precision. An example of such an approximation is shown Figure 7.6. As the reprojection error increases it changes value three times. This means that when computing w in Algorithm 5, we should do so for three thresholds $\epsilon_1, \epsilon_2, \epsilon_3$. Consequently each correspondence will yield three intervals $I_1 \subset I_2 \subset I_3$ - one for each time the value of the cost function changes. The different types of intervals also get a weight indicating how much the cost function changes when entering this interval.

In Algorithm 6 we get three lists of lower bounds $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ and similarly for the upper bounds. The different lists are sorted separately and then gone through like before. Passing a lower bound from $\mathcal{L}_i$, weight w_i is subtracted from the current cost, and passing an upper bound from $\mathcal{U}_i$ the same weight is added. The computational cost will be approximately linear in the number of steps of the cost function.

7.5 Restricted Motions

One advantage of the suggested approach to relative pose estimation, is that restricted motions can be handled easily. In this section we present a few standard restrictions and discuss how they can be enforced.

Planar motion If the rotation axis is known and perpendicular to the translation, this can be used in the following way. Let $\mathbf{f}$ be the rotation axis. We get the following constraints,

$$\mathbf{r}_3^{\mathrm{T}}\mathbf{f} = 0, \quad \mathbf{r'}_3^{\mathrm{T}}\mathbf{f} = 0 \quad \text{and} \ \alpha = 0 \tag{7.27}$$

The first two constraints can easily be enforced in the discretization step. Only epipoles in these planes are generated. The third constraint reduces the set of angles that has to be considered in Algorithm 6.

Small motion In tracking applications, the motion between consecutive frames is generally small. This can easily be enforced by adding constraints

$$\angle(\mathbf{r}_3, \mathbf{r}'_3) < \gamma_{max} \quad \text{and} \quad \alpha < \alpha_{max}. \tag{7.28}$$

These constraints reduce the number of pairs that have to be considered in Algorithm 5.

7.6 Motion Segmentation

To examine how well the brute-force algorithm works in practice, it was tried in a simple system for motion segmentation. Given a sequence of images of multiple moving objects, the aim of motion segmentation is to estimate all these motions as well as the motion of the camera. Moreover, each detected feature point should be classified as belonging to one motion.

Like much of the work in this field, we assume that the number of motions is known. For the discussion let us assume that this number is three. A seemingly straightforward approach to segmentation would be to keep track of the top three motions in our brute-force search, but this turns out be difficult in practice. The peaks in the relative pose space are rather flat and it is hard to distinguish different motions.

Therefore, we proceed in a sequential manner using Algorithm 7. The first step is to estimate N hypothetical motions. This is done in a sequential manner, using Algorithm 5. Typically, N is chosen significantly larger than the true number of motions not to miss any motion. The next step is to choose three of these N motions to perform the motion segmentation. We do this by going through all possible choices of three motions and choosing the ones that yield the lowest number of outliers. Just as in Algorithm 7

Algorithm 7 Multiple Motions

Given two views A and B with multiple moving objects and point tracks $\mathcal{T}$, N hypothetical motions are estimated. An extra view C is used to validate motions.

Repeat N times
  Set $\mathcal{H} = \mathcal{T}$.
  For the view pairs (A, B), (A, C), (B, C)
    Estimate relative pose using $\mathcal{H}$ and threshold ϵ_1.
    Remove tracks with error larger than ϵ_2 from $\mathcal{H}$.
  Reestimate a relative pose between A and B using $\mathcal{H}$ and ϵ_1.
  Store this solution.
  Set $\mathcal{T} = \mathcal{T} \setminus \mathcal{H}$.

outliers are tracks having an error larger than ϵ_2. Having decided on three motions we match each point track to that motion which yields the smallest errors.

The classification obtained in this manner can be refined by standard bundle adjustment. Details are given in the experimental section.

7.6.1 Adding a Spatial Prior

To further improve the motion segmentation results, we tried using a spatial prior assuming that close points probably belong to the same motion. We formulate the spatial prior in an energy minimization framework with a data term and a smoothness term,

$$C(\mathbf{x}) = \sum_{p \in \mathcal{V}} C_p(x_p) + \lambda \sum_{(p,q) \in \mathcal{E}} C_{pq}(x_p, x_q). \tag{7.29}$$

Here $G = (\mathcal{V}, \mathcal{E})$ is an undirected graph. The set of nodes $\mathcal{V}$ corresponds to the point tracks and x_p denotes the label of node p. The set of edges $\mathcal{E}$ describes the neighborhood relationship. We use the reprojection error of point p as data term $C_p(x_p)$ and define the smoothness term as,

$$C_{pq}(x_p, x_q) = \begin{cases} 0 & \text{if } x_p = x_q \\ \frac{d_{max} - d(p,q)}{d_{max}} & \text{if } x_p \neq x_q \end{cases} \tag{7.30}$$

where $d(p, q)$ denotes the Euclidean distance of point p, q and d_{max} is a threshold to define the size of the neighborhood. If $d(p, q) < d_{max}$, then $(p, q) \in E$. This smoothness term will penalize the case when two points lie close to each other but belong to different motions. The constant λ determines the balance between the data and smoothness term. Energy minimization was performed using α-expansions; see Boykov et al. (2001).

7.7 Parallel Implementation

Normally, the weakness of a brute-force algorithm is its computational performance. However, studying Algorithm 5 we note that the computations for different pairs $(\mathbf{r}, \mathbf{r}')$ are independent, so we can easily parallelize the algorithm using a MapReduce model. In the Map step, the lower and upper bounds are sorted simultaneously and then intersections are computed simultaneously for all pairs $(\mathbf{r}, \mathbf{r}')$. In the Reduce step, the pair $(\mathbf{r}, \mathbf{r}')$ that yields most inliers is picked by reduction operations.

Nvidia's parallel computing architecture, CUDA, was used for parallel implementation. Algorithm 6, was implemented in a 2-dimensional grid of k by k blocks, where k is again the number of points in the discretization. Each block executes the computation for one pair of epipoles, $(\mathbf{r}, \mathbf{r}')$. Inside each block, a parallel bitonic sorting algorithm with $O(n \log(n)^2)$ complexity in serial and $O(\log(n)^2)$ complexity in parallel is implemented since it is well-suited for sorting within a block using shared memory. To find the maximal intersection, each thread goes through the upper bound list to find the maximal intersection for the current lower bound. This is done using binary search.

In the end, the parallel implementation is up to 30 times faster than the serial implementation, making the performance of our algorithm quite practical. To make sure global memory access coalescing, we pad the lower and upper bounds with dummy values. Constant memory is used to store the epipoles during the computation of spherical coordinates. This works to reduce global memory latency.

7.8 Experiments

For testing we primarily used the GPU implementation. Timings are for 3GHz Core2 Duo with 8GB Memory with an NVidia Tesla 2050 with

3GB global memory.

To get some data on the execution times, synthetic data was generated. First 100 random 3D points were generated in a cube centered at the origin, having side 300 units. The cameras were placed randomly at distance of approximately 1000 units. Gaussian noise with standard deviation 0.0002 was added to the image points. Figure 7.7 shows angular errors in rotation and translation when compared to the ground truth. The threshold $\epsilon = 0.005$ was used with different degrees of discretization.

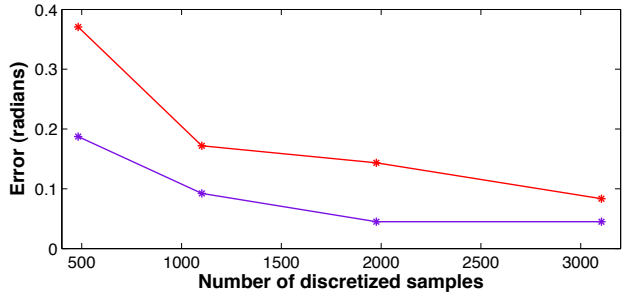


Figure 7.7: The plot shows errors for different discretizations. The error in rotation is shown in red and the error in translation is shown in blue.

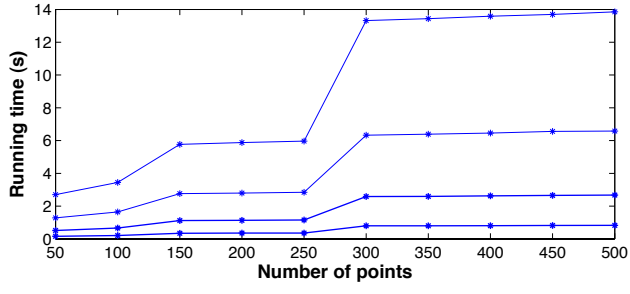


Figure 7.8: Execution times for different discretizations. Starting from below the curves were generated using 700, 1258, 1976 and 2862 points in the discretization of the unit sphere, S^2.

7.8.1 Other Cost Functions

To verify the possibility of using other cost functions we tried it on some random data generated as described above. Using the approximated truncated L_2 norm in the way described in Section 7.4 the rotational error decreased from the average 0.17 radians to an average of 0.11 radians. This was using 1100 points in the discretization. The threshold for the standard method was $\epsilon = 0.005$ and the thresholds for the approximate truncated L_2 was set to $\epsilon/2$, ϵ, $3\epsilon/2$ and 2ϵ.

7.8.2 Outliers

To test the proposed algorithm on data with a lot of outliers, synthetic data was generated in the following way. First 50 random 3D point were generated in a cube centered at the origin, having side 100 units. The cameras were placed randomly at distance of approximately 1000 units. Then 450 outliers were added to each image. They were generated in the same way as the inliers but separately for the two images. Gaussian noise with standard deviation 0.0002 was added to the image points. Figure 7.9 shows how many of the 50 inliers were found by the proposed algorithm. The threshold $\epsilon = 0.0005$ was used in the algorithm and the average computation time for the parallel implementation was 6s.

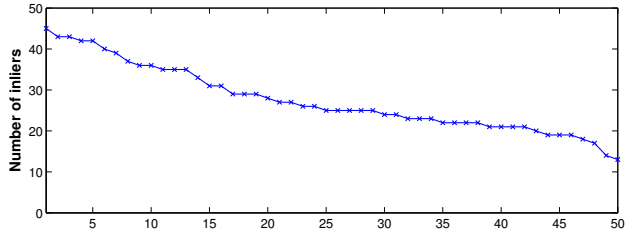


Figure 7.9: The number of inliers for the 50 outlier experiments. The list was sorted for better visualization. In each example there were 50 inliers and 450 outliers. The error threshold was $\epsilon = 0.0005$ and 1976 points were used in the discretization of the sphere.

The outlier rate in these experiments was 90%. This means that using standard RANSAC and a five-point solver, the expected number of itera-

tions before picking just one single set with 5 inliers is 100 000 and using reprojection errors that also means performing 50 million triangulations.

7.8.3 Planar Motion

The performance on planar scenes was tested on 64 image pairs from eniro.se. These are street-view images taken from a car so the motion is approximately planar. Since the images are given with direction information we could compute the deviation between the estimated rotation matrix and the ground truth. This deviation in radians is given in Figure 7.10. The results were produced using 100 points to discretize the unit circle and a threshold of 0.0005. The average execution time was 0.47 s for a sequential java implementation.

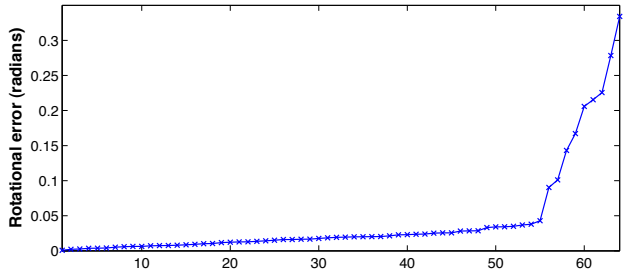


Figure 7.10: Angular error when comparing with the ground truth rotation.

7.8.4 Motion Segmentation

We will now look at the performance of this 3D motion segmentation algorithm for rigid scenes from the Hopkins 155 database, see Tron and Vidal (2007). The state-of-the-art results are reported in Elhamifar and Vidal. (2009) and all the top performers are included in the comparison below. In each sequence, there are typically 20-30 frames and a few hundred 2D feature tracks given. The number of motions in each sequence is also specified.

Some of the sequences contain articulated motions to which the presented framework does not apply. Therefore we focus on the subset of

checkerboard sequences, 26 sequences with 3 motions and 78 sequences with 2 motions, hence 104 out of the 155 sequences are considered. Based on Elhamifar and Vidal. (2009), one can conclude that the checkerboard sequences are the most difficult ones as the classification errors are significantly lower for the remaining ones.

All of the top performing algorithms are based on the affine camera model. Hence they are not dependent on the internal calibration of the cameras, whereas we assume calibrated cameras. To resolve this, the principal point is set to the middle of the image and the focal length to 700 pixels for images of size 480×640. This is the size for all sequences, but the last one, which has frame size 240×320 and consequently we halve the focal length for this case. Note that the true focal length is unknown, so the chosen is only empirically motivated[1].

The thresholds $\epsilon_1 = 0.0003$ and $\epsilon_2 = 0.0015$ are the same for all sequences. Parameters for spatial regularization: $\lambda = 1.66 \times 10^{-4}$ and $d_{max} = 0.04$. These have been found empirically and fixed for all sequences.

We compare with the following algorithms: Generalized Principal Component Analysis (GPCA) in Vidal and Hartley (2004), Local Subspace Affinity (LSA) in Yan and Pollefeys (2006), RANSAC in Fischler and Bolles (1981), Multi-Stage Learning (MSL) in Sugaya and Kanatani (2004), Agglomerative Lossy Compression (ALC) in Ma et al. (2007) and two variants of Sparse Subspace Clustering (SSC) in Elhamifar and Vidal. (2009). There are two versions of our brute-force algorithm. The first one (BF) is implemented according to the description in Section 7.6 and the second one is with the addition of a spatial prior (BF-S) as described in Section 7.6.1.

In Tables 7.1 and 7.2, the misclassification rates are presented. Our brute-force algorithm achieves very low error rates, both in terms of mean and median error rates. Note that even though we are only using three frames (the first, the middle and the last) in each sequence, we are able to obtain state-of-the-art results. Since we are actually recovering the 3D motion, it is very simple to add spatial regularization to the results. Still, even without such regularization, our approach outperforms the competitors, and with regularization, the error rates are significantly lower.

[1] In the dataset, a 3×3 calibration matrix is provided, but this calibration is clearly incorrect since it has an aspect ratio of 0.75.

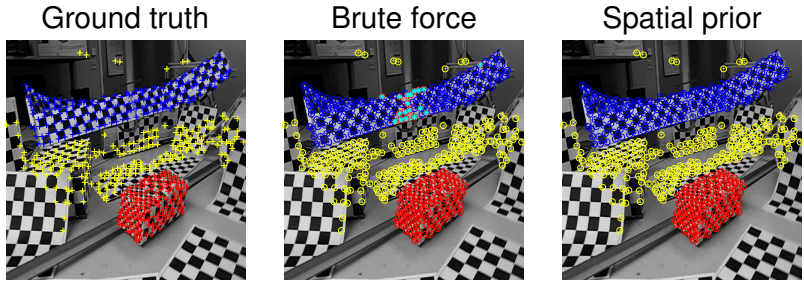


Figure 7.11: Example frame from one sequence with ground truth (left), brute-force (middle) and brute-force with spatial prior (right). The colors of the feature points indicate which motion class (blue, yellow, red). Feature points that are misclassified have been colored cyan (see middle figure). Note that the spatial prior is able to correct for all the errors.

Method	GPCA	LSA	RANSAC	MSL	ALC	SSC-B	SSC-N	BF	BF-S
Mean	31.95	5.80	25.78	10.38	5.20	4.49	2.97	2.11	**0.99**
Median	32.93	1.77	26.00	4.61	0.67	0.54	0.27	0.81	0.00

Table 7.1: Classification errors (%) for the 26 checkerboard sequences with 3 motions.

7.9 Conclusions

Using a brute-force algorithm for computing the relative pose of two projections may seem like a step back considering the many sophisticated algorithms that have been developed over the years. But why is it that none of the best performing algorithms for 3D motion segmentation does not use a pinhole camera model? This paper shows that a pinhole model is the correct choice and the lack of perspective methods that perform well on the Hopkins 155 benchmark is likely due to algorithmic failure modes, for example, the incapability of handling planar scenes.

The reported running times of the algorithm are well within the limits of being a suitable choice for many vision applications. Of course, the full search space cannot be used for a real-time system, but restricting the parameter space to small motions, the brute force approach becomes a viable

Method	GPCA	LSA	RANSAC	MSL	ALC	SSC-B	SSC-N	BF	BF-S
Mean	6.09	2.57	6.52	4.46	1.55	0.83	1.12	0.85	**0.43**
Median	1.03	0.27	1.75	0.00	0.29	0.00	0.00	0.00	0.00

Table 7.2: Classification errors (%) for the 78 checkerboard sequences with 2 motions.

and robust alternative for real-time visual odometry. Such an investigation is left as an avenue of further research.

Chapter 8

Joint Object Detection and Pose

This last chapter contributes to object detection and pose estimation. Object recognition and detection leads machine vision one step toward the human-level cognition. For humans 3D geometry is an important cue to recognizes a object. This chapter aims to improve automated object detection by incorporating 3D geometry. More specifically, we extend the previous work on deformable part-based models, see Felzenszwalb et al. (2010c), by using accurate geometric models both in the training phase and at detection. Richer annotations with 3D geometry are manually generated to reduce perspective distortion before training the part-based models. Training is performed on rectified images, which leads to more specific models and reduces the risk of false positives. A set of representative object poses are learned from the training set and used to rectify test images without annotations. The method is evaluated on the *bus* category of the Pascal VOC dataset with promising results.

8.1 Introduction

An early work that is frequently cited in object detection is that of Viola and Jones (2001). They introduced *integral images* to object detection, allowing for very efficient feature computations. They also propose a method for combining successively more complex classifiers in a cascade structure which dramatically reduces the running time of detection by focusing attention on promising regions of the image. Another influential work is Dalal and Triggs (2005), focused on pedestrian detection. Linear SVM is adopted together with the proposed Histogram of Oriented Gradients (HOG) descriptors,

which are shown to outperform the existing features in this specific task.

Apparently using one single model to capture the global appearance of an object is not enough. The idea of using several parts to represent an object is first proposed in Fischler and Elschlager (1973), where the model is referred to as *pictorial structures*. The appearance of each part is modeled separately while the connections between pairs of parts are represented using *spring-like* structures. The idea was revived by Felzenszwalb and Huttenlocher (2005) who proposed an efficient algorithm for classical pictorial structure energy minimization of a cycle-free structure. A learning algorithm is also proposed to extend the original method.

The idea of part-based models is taken even further in Felzenszwalb et al. (2010c) where the global appearance is modeled by a root filter, together with several part filters to describe the local appearance. Each part is associated with an anchor position, and deviations from this are penalized with a deformation cost in the energy function. The anchor positions are treated as latent variables and a latent SVM is used to discriminatively train the model with partially labeled data. This deformable part-based model (DPM) achieves state-of-the-art results in Pascal VOC Challenge.

To address the change of appearance due to perspective distortion, the training examples are first clustered based on the aspect ratio of the bounding boxes. One model is trained for each cluster and at detection the parameter vectors of component models are concatenated from which high scoring hypothesis is generated independently for each component. However, limited training data prevent one from training a faithful model effectively with more mixture components. A workaround proposed in Ott and Everingham (2011) is to allow part sharing between the mixture components and the object classes. Several works explicitly treat multi-view detection. In Thomas et al. (2006), the proposed approach can detect objects from arbitrary viewpoints. It is achieved by combining the implicit shape model with the multi-view specific object recognition system by Ferrari et al. (2004). In Savarese and Li (2007), a compact model is built by linking together parts of the object from different viewing points. The parts - defined as large and discriminative regions comprising many local invariant features - are connected through their mutual homographic transformation. In Liebelt and Schmid (2010), a few synthetic 3D models are used to learn a geometrical representation of an object category, which are incorporated in the discriminatively trained part-based model. In Xiang and Savarese

(2012), an object is modeled by several planar aspects, using some 3D CAD model as prior knowledge. A conditional random field (CRF) model is constructed with maximal margin parameter estimation. In Branson et al. (2011) the possibility is explored of reducing manual annotations, where an interactive labeling tool is used to refine the automatic annotation with human intervention.

Our method also uses rich annotations of the training dataset. Similarly to Xiang and Savarese (2012), we model an object using a number of planar aspects. By annotating each aspect (side) of an object, one can estimate its pose. With the estimated pose, each aspect of the object is rectified such that one can train a viewpoint-independent model of each aspect of the object category. For detection, we hypothesize different object poses from a learned set of *typical poses*, transforming the image according to the hypothetical pose and running the detector. By doing this, we obtain not only an accurate bounding box but also a rough pose of the object. This is appealing, as it extracts more information from a 2D image and enables a richer understanding of the object in its context. In the experiments, we evaluate our method on the *bus* category of Pascal VOC Challenge.

8.2 Modeling Appearance and Geometry

Our object model is defined as a number of planar aspect models together with a set of typical poses. We will assume that we have annotated training images and have a method to estimate the object pose from annotations. Hence we train each aspect model from rectified image patches using a deformable part-based model.

The fact that aspect models are learned from rectified image patches, introduces a problem at detection, when the ground-truth object pose is unknown. To resolve this we learn a small set of *typical object poses* from the annotated training set. At detection we transform the image according to each of the learned typical poses. If the training set is large enough it will contain the common object poses.

Detection scores for aspects are generated by running each aspect detector on each of the transformed images. Detections from different aspect models are combined and thresholded to produce the final full detection. More specifically, once we run the aspect models on the transformed images, multi-scale score pyramids are generated, one for each aspect of the object.

Any location in a score pyramid defines a bounding box and a score indicating the confidence of the aspect occurring at that location. We merge the score pyramids by combining, for example, the frontal detection and the side detection which have consistent size and proper relative locations.

8.2.1 Pose Estimation

Pose estimation is a crucial building block in training aspect models and learning representative poses. It is not possible to infer the object pose from training images labeled only with rectangular bounding boxes. Extra information is necessary either from some pre-built CAD models or from manual annotations. Here we manually annotate each visible aspect of objects in training images, as shown in Figure 8.1.

Figure 8.1: Annotation of each visible aspect of the object in training images

.

Each annotation gives us a set of known points on the object. The next task is to estimate the camera pose relative to the object. We will assume that all intrinsic camera parameters have standard values, except the focal length which we estimate. More precisely, we assume that the principal point lies at the center of the image, that the aspect ratio is 1 and that skew is 0.

If the 3D model of an object is known, it is sufficient to estimate camera pose and focal length using four points, see Triggs (1999). However, for block-shaped objects, for example buses we use a more specialized approach, which does not require an explicit 3D model. The goal is to estimate the camera matrix $P = [\mathtt{R} \mid \mathtt{t}]$ and the focal length f.

We assume that the upper and lower edges of a block-shaped object are parallel. Hence the corresponding lines in the image intersect at a vanishing point, being the projection of a point at infinity, see Figure 8.2. Let $\mathbf{x}$ be a vanishing point in the image and $(X^{\mathrm{T}}, 0)^{\mathrm{T}}$ be the homogeneous coordinates of the corresponding point at infinity. Then we have

$$\lambda \begin{pmatrix} \mathbf{x} \\ f \end{pmatrix} = [\mathtt{R} \,|\, \mathtt{t}] \begin{pmatrix} X \\ 0 \end{pmatrix} = \mathtt{R}X. \tag{8.1}$$

The same computation for the other side of the object yields

$$\lambda \begin{pmatrix} \mathbf{y} \\ f \end{pmatrix} = \mathtt{R}Y. \tag{8.2}$$

We note that X and Y represent the front and sideways directions of the object. This means that they must be perpendicular, that is

$$x^{\mathrm{T}}y + f^2 \propto X^{\mathrm{T}}R^{\mathrm{T}}RY = X^{\mathrm{T}}Y = 0, \tag{8.3}$$

from which one can estimate the focal length. Knowing the focal length, (8.1) and (8.2) allow us to estimate the camera rotation relative to the orientation of the object. Finally, we place the origin in the front left corner of the object and compute the camera translation from its corresponding image projection.

8.2.2 Training Aspect Models

Rectifying the training images Detectors trained on the objects under varying viewpoints will tend to be less specific and can lead to high false positive rates. As described in Section 8.2.1, we can estimate the pose of an object and use this to transform the training images such that each visible aspect is rectified. However, at detection we cannot estimate the object pose very accurately, so a model trained on perfectly rectified image patches might not be flexible enough. Hence we add a small perturbation to the exact pose when rectifying the training images.

Assume the pose we estimated for an object is P, where $P = [\mathtt{R} \,|\, \mathtt{t}]$. We generate a small perturbation by picking a random rotation angle θ from a uniform distribution on $[0, 5^\circ]$. Now let $\mathtt{R}_x(\theta)$ be a rotation about the x-axis, with rotation angle θ and let $\mathtt{R}_r$ be a random rotation picked from

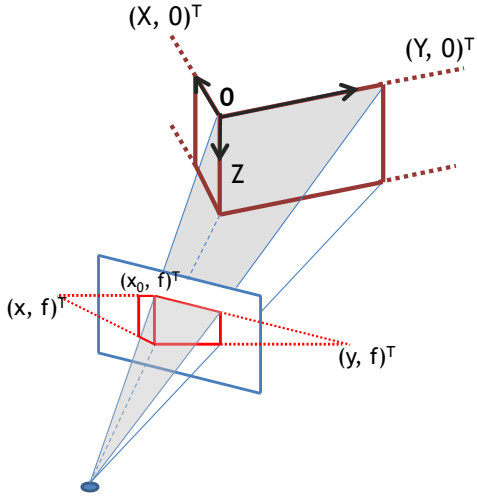


Figure 8.2: Estimate the pose from annotations of a block-shaped object

the uniform distribution over all rotations. Then the desired perturbed rotation is $\mathtt{R}_p = \mathtt{R}_r^T \mathtt{R}_x(\theta) \mathtt{R}_r \mathtt{R}$. We also add a small perturbation to the ground truth translation, $\mathtt{t}_p = \mathtt{t} + \mathbf{n}$, where $\mathbf{n}$ is sampled from $\mathcal{N}(0, \sigma^2)$ for $\sigma = 0.005$.

The next step is to transform the image such that each planar aspect is rectified. Let us say that we want to transform a bus image such that the side is rectified. Let $\mathbf{v}_1$ and $\mathbf{v}_2$ be an orthonormal basis for the subspace parallel to the plane and let $\mathbf{q}$ be a point in the plane. Any point X in the plane can be written as $x_1\mathbf{v}_1 + x_2\mathbf{v}_2 + \mathbf{q}$ and its projection

$$\lambda \begin{pmatrix} x_1' \\ x_2' \\ 1 \end{pmatrix} = P \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{q} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} = H \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}, \quad (8.4)$$

where $(x_1', x_2', 1)^{\mathrm{T}}$ is a point on the image plane, $(x_1, x_2, 1)^{\mathrm{T}}$ is the corresponding point on the actual plane of the object's side. For example, if the coordinate system is used as in Figure 8.2 and consider a point X on the side of the object, that is the YZ-plane, then we have $\mathbf{q} = (0, 0, 0)^{\mathrm{T}}$, $\mathbf{v}_1 = (0, 1, 0)^{\mathrm{T}}$ and $\mathbf{v}_2 = (0, 0, 1)^{\mathrm{T}}$, the homograph

$H = (\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4)$ where $\mathbf{p}_i$ is the i-th column of P. Similarly one can show that the homographs to rectify the front, that is XZ-plane in Figure 8.2 is $H = (\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4)$.

Thus we have found the homography H, relating points in the aspect plane with points in the image. By transforming the image according to H^{-1}, we get a rectified image of the aspect, having axis-parallel edges.

Building the models We use latent-SVM (LSVM) to train deformable part-based models. Here we give only a brief review of training. More details are referred to Felzenszwalb et al. (2010c). Since the part locations are unknown, they are treated as latent variables in training. For a given example $\mathbf{x}$, One seeks for the best configuration of parts which minimize the following cost function with respect to the latent variable $\mathbf{z}$ as

$$f_{\boldsymbol{\beta}}(\mathbf{x}) = \max_{\mathbf{z} \in Z(\mathbf{x})} \boldsymbol{\beta} \cdot \phi(\mathbf{x}, \mathbf{z}) \tag{8.5}$$

where $\boldsymbol{\beta}$ is a vector of model parameters describing the root filter, part filters, anchor positions of parts and deformation coefficients, $\mathbf{z}$ is the latent variable specifying the location of root filter and part filters and $\phi(\mathbf{x}, \mathbf{z})$ yields the feature vector for a specific configuration.

The goal is to learn the model parameters $\boldsymbol{\beta}$ from the labeled examples $\langle \mathbf{x}_1, y_1 \rangle, \cdots, \langle \mathbf{x}_n, y_n \rangle$, where $y_i = 1$ for positive examples and $y_i = -1$ for negative ones. This is achieved by minimizing

$$L(\boldsymbol{\beta}) = \mu ||\boldsymbol{\beta}||^2 + \sum_{i=1}^{n} \max\{0, 1 - y_i f_{\boldsymbol{\beta}}(\mathbf{x}_i)\}, \tag{8.6}$$

where $\max\{0, 1 - y_i f_{\boldsymbol{\beta}}(\mathbf{x}_i)\}$ is the standard hinge loss and μ determines the relative weight of the regularization term. In Figure 8.3 we illustrate the aspect models trained on the bus category of Pascal VOC 2011 training set. One can clearly make out some parts of the bus, for example, the wheels.

8.2.3 Finding the Representative Poses

In the last section we have shown how to estimate highly specific models by compensating for varying viewpoint. This introduces a problem since at detection the viewpoint is unknown and cannot be compensated for. To handle this we learn a small set of typical object poses. At detection an

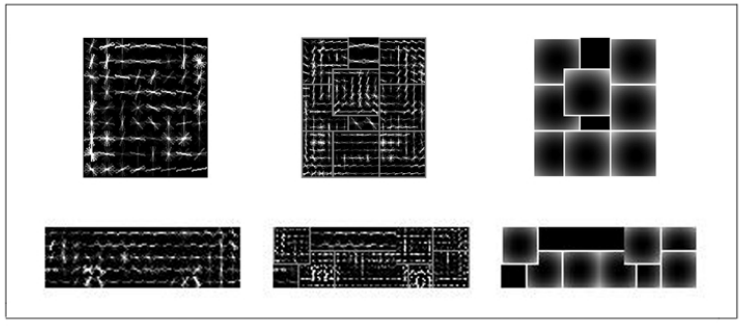


Figure 8.3: Two aspect models trained from the bus category of Pascal VOC 2011. The top row shows the frontal model and the bottom row shows the side model. From left to right are illustrations of a root filter, part filters and deformation cost.

image is transformed according to each of these poses and the detector is run on each of the generated views.

To find this set of representative object poses we look again to the annotated training set. Each annotated object yields an object pose that we can use in the learning. Let P_i be the pose of a certain object i and let $\mathcal{S}_i$ be the set of annotated objects which have similar poses to P_i. We will define in the following if two poses are similar or not. To find k poses which are representative for the training we seek k sets $\mathcal{S}_{i_1}, \ldots, \mathcal{S}_{i_k}$ such that $\left|\bigcup_j \mathcal{S}_{i_j}\right|$ is maximized. This is a maximum k-cover problem and for unbounded k it is NP-hard, but in our case, we could pre-determine k and solve with software such as CPLEX.

It remains to define the notion of similar poses. To determine if the pose of an annotated object is similar to a given pose P, we transform the annotated points according to P. Ideally, both the front and the side of the object will be transformed to rectangles. Hence as measure of similarity, we measure how much the upper/lower edge of the transformed aspect deviates from being horizontal, as illustrated in Figure 8.4. A certain training example is similar to a pose P if the average angular deviations for the front

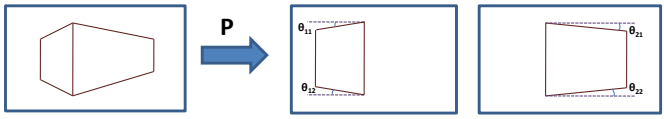


Figure 8.4: Measure of angular deviation for pose similarity.

and side are both below a predefined threshold ϵ as

$$\frac{1}{2}(\theta_{11}+\theta_{12})<\epsilon \quad \text{and} \quad \frac{1}{2}(\theta_{21}+\theta_{22})<\epsilon. \tag{8.7}$$

8.3 Detection

Training the aspect models and learning a set of representative poses gives us a more specific model as well as increased flexibility at detection. For a given test image, we use the set of typical poses to transform it. The aspect detectors are run on each of these images individually. Given that the model consists of M aspects and N typical poses, each pose will define M homographies to rectify the corresponding aspects. Thus a test image is transformed into $M \cdot N$ candidate images. Still, with small M and using a fast cascade implementation of the detector, see Felzenszwalb et al. (2010a), the computational load is no big issue.

8.3.1 Generating the Score Pyramids

With the deformable part-based model in Felzenszwalb et al. (2010c), the given image is first resized to multiple scales. The trained detector runs on each scaled image, creating a pyramid of score maps of different scales. Any location (x_i, y_i, l_i) in the score pyramid defines a score to indicate the confidence that an object with a fixed size occurs at (x_i, y_i) in the scaled image. The scale is defined by the level l_i of the score pyramid.

In our model, each typical pose P_i defines two homographies H_i^f, H_i^s for frontal and side respectively. We transform the image using $\{H_i^f\}$ and $\{H_i^s\}$ for $i = 1, \cdots, N$ and run the frontal and side detector respectively. This will yield the score pyramids, denoted by $\mathcal{P}_i^f$ and $\mathcal{P}_i^s$ respectively for $i = 1, \cdots, N$. Now each location (x_k, y_k, l_k) in the score pyramid $\mathcal{P}_i^f$ or

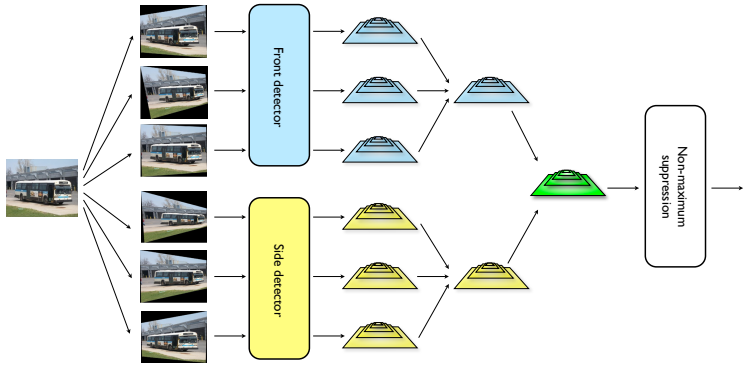


Figure 8.5: Overview of the detection pipeline. In the first step the input image is transformed according to each of representative poses. This produces a multiple images that are individually run through the aspect detectors creating a set of score pyramids containing the detector scores at different scales. These are merged into one pyramid per aspect, in the original image coordinate system. Finally, the front and side scores are combined and non-maximum suppression is performed.

$\mathcal{P}_i^s$ defines a frontal or side detection with a rectangular bounding box for the scaled image with scale defined by l_k.

However, a location (x_k, y_k, l_k) in the score pyramid of the transformed image does not directly map to the location in the original image due to transformation. To merge score pyramids, we need to transform score maps back to the original image coordinate system using $\{(H_i^f)^{-1}\}$ and $\{(H_i^s)^{-1}\}$. This yields new score pyramids $\hat{\mathcal{P}}_i^f$, $\hat{\mathcal{P}}_i^s$ where each location defines a skewed bounding box and directly maps to the original image coordinate system. See Figure 8.6.

8.3.2 Combining the Score Pyramids

To combine the score pyramids $\hat{\mathcal{P}}_i^f$ and $\hat{\mathcal{P}}_i^s$ from different poses P_i for $i = 1, \cdots, N$, we need to match a front detection at (x_j, y_j, l_j) in $\hat{\mathcal{P}}_i^f$ with the corresponding side detection at (x_k, y_k, l_k) in $\hat{\mathcal{P}}_i^s$.

In the original DPM, bounding boxes defined at any locations (x_j, y_j) in the same level of a score pyramid correspond to hypothetical detections of

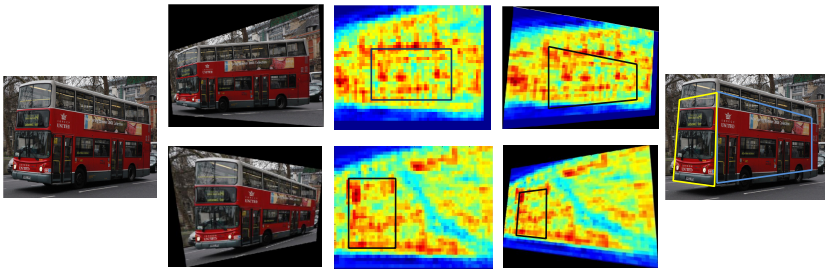


Figure 8.6: Transform the score map and combine the detection. From left to right: The original image. The transformed image using H_i^f and H_i^s. The score map and bounding box. The transformed score map using $(H_i^f)^{-1}$ and $(H_i^s)^{-1}$ and skewed bounding box. The final combined detection.

the same size in the original image. However, in our model, as we transform score maps using $(H_i^f)^{-1}$ and $(H_i^s)^{-1}$, the skewed bounding boxes in the same level of the transformed score pyramids can have varied size depends on (x_i, y_i). For a front detection at (x_j, y_j, l_j) in the new score pyramid $\hat{\mathcal{P}}_i^f$, this means we need to search not only in the corresponding level but possibly other levels of score pyramid $\hat{\mathcal{P}}_i^s$ where a side detection with compatible size might exist.

To make the search more efficient, score pyramids $\bar{\mathcal{P}}^f$ and $\bar{\mathcal{P}}^s$ are created with different levels defined by the length of shared edges. More specifically, we divide the height range of all bounding boxes into L intervals $l_1, ..., l_L$ and create L levels in the score pyramid $\bar{\mathcal{P}}^f$ and $\bar{\mathcal{P}}^s$, each corresponding to a height interval. For each skewed bounding box in the transformed score pyramids $\hat{\mathcal{P}}_i^f$ and $\hat{\mathcal{P}}_i^s$, we use the length of the shared edge to determine which level to put it in the new score pyramid. To further simplify the search, we use either the top-left or the top-right corner to represent a front bounding box, depending on if a left side or a right side is to be paired. It avoids the extra search since the matched front and side detection should coincide at the same point (x_i, y_i). By doing this, we merge the score pyramids $\hat{\mathcal{P}}_i^f$ and $\hat{\mathcal{P}}_i^s$ from different poses P_i for $i = 1, \cdots, N$ into $\bar{\mathcal{P}}^f$ and $\bar{\mathcal{P}}^s$ respectively. Note for each location in $\bar{\mathcal{P}}^f$

and $\bar{\mathcal{P}}^s$, we only keep the detection with maximum score from $\hat{\mathcal{P}}_i^f$ and $\hat{\mathcal{P}}_i^s$.

Now we only need to merge these two score pyramids into one, denoted by $\mathcal{P}$ to yield final detections. Due to the way we create $\bar{\mathcal{P}}^f$ and $\bar{\mathcal{P}}^s$, the merge can be done in a very efficient way as we only need to sum the score at the same location in $\bar{\mathcal{P}}^f$ and $\bar{\mathcal{P}}^s$. To handle the case when only the front or side is visible, we introduce a threshold s_{thr}. If the matched front or side detection has a score lower than s_{thr} we assume that it is not visible. In that case the total score is computed as the sum of s_{thr} and the score of the visible aspect. Hence, the score of a combined detection is

$$\mathcal{P}(x,y,l) = \begin{cases} \bar{\mathcal{P}}^f(x,y,l) + \bar{\mathcal{P}}^s(x,y,l) & \text{if } \bar{\mathcal{P}}^f(x,y,l) \geq s_{thr} \\ & \text{and } \bar{\mathcal{P}}^s(x,y,l) \geq s_{thr} \\ \bar{\mathcal{P}}^f(x,y,l) + s_{thr} & \text{if } \bar{\mathcal{P}}^s(x,y,l) < s_{thr} \\ \bar{\mathcal{P}}^s(x,y,l) + s_{thr} & \text{if } \bar{\mathcal{P}}^f(x,y,l) < s_{thr} \end{cases} \tag{8.8}$$

where $\bar{\mathcal{P}}^f(x,y,l)$ and $\bar{\mathcal{P}}^s(x,y,l)$ denote the score at location (x,y,l) in the score pyramids $\bar{\mathcal{P}}^f$ and $\bar{\mathcal{P}}^s$ respectively.

For each location in the final score pyramid $\mathcal{P}$, two skewed bounding boxes are implicitly defined respectively for front and side, from which one can estimate the object pose. Finally, non-maximum suppression is applied to greedily remove detections which have significant overlap; see Felzenszwalb et al. (2010c) for details.

8.4 Experiments

We evaluate our method on the bus category of Pascal VOC 2011 training and validation dataset. The training set has 5717 images of 20 categories among which there are 213 images containing buses. The validation set has 5823 images with 208 containing buses. Since the ground truth annotation is not provided for Pascal VOC 2011 test set, we train the model on the training set and tested on the validation set.

For training, we manually annotate every visible aspect for each positive training example. Pose estimation on annotated training images gives the pose and the actual dimensions of the object up to a scale factor, from which we rectify each aspect of the object. We use both the left and right side to train a side detector. Considering the rear and frontal of a bus are quite

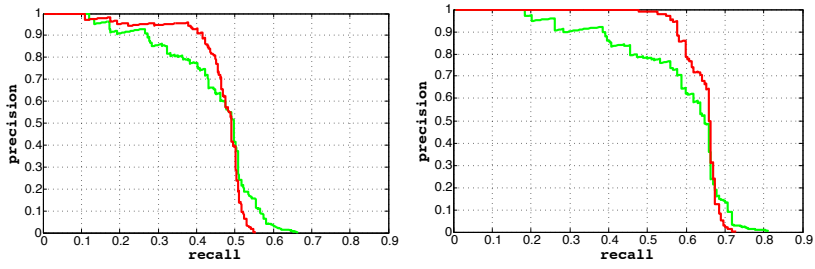


Figure 8.7: Precision-recall curves for bus category of Pascal VOC 2011 validation set. The left plot shows the result for the entire set and the right plot shows the result when examples with a lot of occlusion are removed. The green curves are the result using the original part-based model with the average precision AP = 0.450 and 0.587 respectively. The red curves are obtained with our method with AP = 0.468 and 0.644 respectively.

similar, we train a frontal detector using both frontal and rear patches. We use the max-cover algorithm to select 10 typical poses. When determining the similarity of two poses, we set the threshold ϵ in (8.7) to be 5 degrees.

For detection, we use 10 different poses to transform the input image. The frontal and side detectors are run on these transformed images but also on the original image. To combine the front and side score pyramids, we set the threshold in (8.8) to $s_{thr} = -1.0$. The experiments are done on a 3.6 GHz Intel Core i7 PC with 64 GB memory, the training takes around 4 hours to train a single aspect model. Detection takes on average 50 seconds per image, but one could speed up significantly by using the cascade detector from Felzenszwalb et al. (2010a).

The precision-recall curve for the bus category is obtained by thresholding all the detection scores at different values, as shown in Figure 8.7. Average precision is calculated by measuring the area under the precision-recall curve using numerical integration. In terms of average precision the improvement over the original part-based model from Felzenszwalb et al. (2010c) is limited - from 0.450 to 0.468, but at high precision rates the difference is more significant. We note that Pascal VOC dataset is regarded as a very difficult dataset for detection, occluded and truncated objects usually exist in the dataset, which we do not really expect to handle.

Figure 8.8: Example results. Detected bounding boxes are shown in green and their layout in red. The first three rows shows correctly detected objects with roughly correct pose. The method was able to automatically handle cases when only one side is visible. The last row shows the buses of which the pose estimation is less accurate.

To examine this more closely, we remove a third of the bus images that contained largely occluded buses or very distant ones with small size. On the remaining examples we got an average precision of 0.644 compared to 0.587 for Felzenszwalb et al. (2010c). We illustrate some detection results as well as the layout estimation in Figure 8.8.

8.5 Conclusions

We described an approach to joint object detection and pose estimation for objects that can be represented by a number of roughly planar aspects. By training several aspect models and learning a small set of representative poses,

we obtained a model with high specificity and flexible choices for detecting objects from various viewpoints. The cost is some extra annotation work as well as increased detection complexity. We should also mention a limitation of our method, it requires that the object be rigid with discriminative aspects. The pose estimation in our method is especially well-suited to block-structured objects such as buses.

For the bus category of the Pascal VOC 2011 dataset, our method achieved better detection results than did the original deformable part-based model while retaining a specific and compact model representation. Beyond that, the proposed method also produces geometric information pertaining to the detected object, for example, pose/layout estimation.

Bibliography

Aanaes, H., R. Fisker, K. Åström, and J.M. Carstensen (2002). "Robust Factorization." In: *Trans. Pattern Analysis and Machine Intelligence*. Cited on page 22.

Alldrin, N., T. Zickler, and D. Kriegman (2008). "Photometric Stereo With Non-Parametric and Spatially-Varying Reflectance." In: *Conf. Computer Vision and Pattern Recognition*. Cited on pages 44 and 47.

Barreto, J. and K. Daniilidis (2005). "Fundamental Matrix for Cameras with Radial Distortion." In: *IEEE International Conference on Computer Vision*. Beijing, China. Cited on page 84.

Bennett, J. and S. Lanning (2007). "The Netflix Prize." In: *In KDD Cup and Workshop in conjunction with KDD*. Cited on page 21.

Blake, A. and A. Zisserman (1987). *Visual Reconstruction*. MIT Press, Cambridge, USA. Cited on pages 107 and 108.

Boykov, Y., O. Veksler, and R. Zabih (2001). "Fast Approximate Energy Minimization via Graph Cuts." In: *Trans. Pattern Analysis and Machine Intelligence*. Cited on page 111.

Branson, S., P. Perona, and S. Belongie (2011). "Strong Supervision From Weak Annotation: Interactive Training of Deformable Part Models." In: *Int. Conf. Computer Vision*. Cited on page 121.

Brooks, J.P. and J.H. Dulá (2013). "The L1-norm best-fit hyperplane problem." In: *Applied Mathematics Letters*. Cited on page 26.

Buchanan, A. M. and A. W. Fitzgibbon (2005). "Damped Newton Algorithms for Matrix Factorization with Missing Data." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 22.

Buchberger, B. (2006). "An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal." English. PhD thesis. RISC (Research Institute for Symbolic Computation). Cited on page 15.

Bue, Alessio Del, João M. F. Xavier, Lourdes de Agapito, and Marco Paladini (2012). "Bilinear Modeling via Augmented Lagrange Multipliers (BALM)." In: *IEEE Trans. Pattern Anal. Mach. Intell.* Cited on page 22.

Byröd, Martin, Klas Josephson, and Kalle Åström (2009). "Fast and Stable Polynomial Equation Solving and Its Application to Computer Vision." In: *Int. Journal Computer Vision*. Cited on pages 13, 17, 18, 19, 89, and 90.

Cai, J.F., E.J. Candès, and Z. Shen (2010). "A Singular Value Thresholding Algorithm for Matrix Completion." In: *SIAM Journal of Optimization*. Cited on page 78.

Candès, E. J. and B. Recht (2008). "Exact Matrix Completion via Convex Optimization." In: *CoRR*. Cited on page 23.

Candès, E. J., X. Li, Y. Ma, and J. Wright (2009). "Robust Principal Component Analysis?" In: *Journal of ACM*. Cited on page 23.

Chandraker, M. K. and D. J. Kriegman (2008). "Globally optimal bilinear programming for computer vision applications." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 23.

Cheng, B., G. Liu, J. Wang, Z. Huang, and S. Yan (2011). "Multi-task low-rank affinity pursuit for image segmentation." In: *Int. Conf. Computer Vision*. Cited on page 23.

Chiuso, A., R. Brockett, and S. Soatto (2000). "Optimal structure from motion: local ambiguities and global estimates." In: *Int. Journal Computer Vision*. Cited on page 98.

Chum, O. and J. Matas (2000). "Optimal randomized RANSAC." In: *Trans. Pattern Analysis and Machine Intelligence*. Cited on page 97.

Chum, O. and J. Matas (2005). "Matching with PROSAC - Progressive Sample Consensus." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 37.

Cox, D., J. Little, and D. O'Shea (2005). *Using Algebraic Geometry*. English. 2nd. Springer. URL: http://www.cs.amherst.edu/~dac/uag.html. Cited on pages 14 and 15.

Dalal, N. and B. Triggs (2005). "Histograms of oriented gradients for human detection." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 119.

Elhamifar, E. and R. Vidal. (2009). "Sparse Subspace Clustering." In: *Conf. Computer Vision and Pattern Recognition*. Cited on pages 114 and 115.

Enqvist, O. and F. Kahl (2009). "Two View Geometry Estimation with Outliers." In: *British Machine Vision Conf.* London, UK. Cited on page 98.

Enqvist, O., F. Jiang, and F. Kahl (2011). "A brute-force algorithm for reconstructing a scene from two projections." In: *Conf. Computer Vision and Pattern Recognition*. (Not cited.)

Eriksson, A. and A. Hengel (2012). "Efficient Computation of Robust Weighted Low-Rank Matrix Approximations Using the L_1 Norm." In: *Trans. Pattern Analysis and Machine Intelligence*. Cited on pages 22, 31, 40, 42, 45, and 64.

Felzenszwalb, P., R. Girshick, and D. McAllester (2010a). "Cascade object detection with deformable part models." In: *Conf. Computer Vision and Pattern Recognition*. Cited on pages 127 and 131.

Felzenszwalb, P. F., R. B. Girshick, D. A. McAllester, and D. Ramanan (2010b). "Object Detection with Discriminatively Trained Part-Based Models." In: *IEEE Trans. Pattern Anal. Mach. Intell.* Cited on page 5.

Felzenszwalb, Pedro, Ross Girshick, David McAllester, and Deva Ramanan (2010c). "Object Detection with Discriminatively Trained Part-Based Models." In: *Trans. Pattern Analysis and Machine Intelligence*. Cited on pages 119, 120, 125, 127, 130, 131, and 132.

Felzenszwalb, Pedro F. and Daniel P. Huttenlocher (2005). "Pictorial Structures for Object Recognition." In: *Int. Journal Computer Vision*. Cited on page 120.

Ferrari, V., T. Tuytelaars, and L. Van Gool (2004). "Integrating multiple model views for object recognition." In: *Conf. Computer Vision and Pattern Recognition*. IEEE. Cited on page 120.

Fischler, M. and R. Elschlager (1973). "The representation and matching of pictorial structures." In: *Transactons on Computers*. Cited on page 120.

Fischler, M. A. and R. C. Bolles (1981). "Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography." In: *Commun. Assoc. Comp. Mach.* Cited on pages 33 and 115.

Fitzgibbon, Andrew W (2001). "Simultaneous linear estimation of multiple view geometry and lens distortion." In: *Conf. Computer Vision and Pattern Recognition*. IEEE. Cited on pages 10, 11, 83, 84, and 85.

Grayson, D. and M. Stillman (1993-2002). *Macaulay 2*. Available at http://www.math.uiuc.edu/Macaulay2/. An open source computer algebra software. URL: http://www.math.uiuc.edu/Macaulay2/. Cited on pages 17 and 89.

Hartley, R. and F. Kahl (2009). "Global Optimization through rotation space search." In: *Int. Journal Computer Vision*. Cited on page 98.

Hartley, R. I. (1997). "In Defense of the Eight-Point Algorithm." In: *IEEE Trans. Pattern Anal. Mach. Intell.* Cited on pages 12 and 84.

Hartley, R. I. and A. Zisserman (2004). *Multiple View Geometry in Computer Vision*. Second Edition. Cambridge University Press. Cited on pages 11, 40, 86, 97, 98, and 107.

Hu, Y., D. Zhang, J. Ye, X. Li, and X. He (2013). "Fast and Accurate Matrix Completion via Truncated Nuclear Norm Regularization." In: *IEEE Trans. Pattern Anal. Mach. Intell.* Cited on pages 63, 75, and 76.

Jacobs, D. (2001). "Linear Fitting with Missing Data for Structure-from-Motion." In: *Computer Vision and Image Understanding*. Cited on page 22.

Jiang, F., Y. Kuang, J. E. Solem, and K. Åström. "A Minimal Solution to Relative Pose with Unknown Focal Length and Radial Distortion." In: (Not cited.)

Jiang, F., O. Enqvist, F. Kahl, and K. Åström (2013a). "Improved Object Detection and Pose Using Part-Based Models." In: *Scandinavian Conf. on Image Analysis*. (Not cited.)

Jiang, F., Y. Kuang, and K. Åström (2013b). "Time delay estimation for TDOA self-calibration using truncated nuclear norm regularization." In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. (Not cited.)

Jiang, F., O. Enqvist, and F. Kahl (2015). "A Combinatorial Approach to L_1-Matrix Factorization." In: *Journal of Mathematical Imaging and Vision*. (Not cited.)

K., Zuzana, M. Byröd, K. Josephson, T. Pajdla, and K. Åström (2010). "Fast and robust numerical solutions to minimal problems for cameras with radial distortion." In: *Computer Vision and Image Understanding*. Cited on page 84.

Kahl, F. and R. Hartley (2008). "Multiple View Geometry Under the L_∞-Norm." In: *Trans. Pattern Analysis and Machine Intelligence*. Cited on page 98.

Kahl, F. and A. Heyden (1999). "Affine Structure and Motion from Points, Lines and Conics." In: *Int. Journal Computer Vision*. Cited on page 22.

Ke, Q. and T. Kanade (2005). "Robust L_1 Norm Factorization in the Presence of Outliers and Missing Data by Alternative Convex Programming." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 22.

Keshavan, R. H., S. Oh, and A. Montanari (2009). "Matrix Completion from a Few Entries." In: *CoRR*. URL: http://arxiv.org/abs/0901.3150. Cited on page 63.

Koren, Y., R. Bell, and C. Volinsky (2009). "Matrix Factorization Techniques for Recommender Systems." In: *IEEE Computer*. Cited on page 21.

Kuang, Y. and K. Åström (2012). "Numerically Stable Optimization of Polynomial Solvers for Minimal Problems." In: *European Conf. Computer Vision*. Cited on page 90.

Kuang, Y., J. E. Solem, F. Kahl, and K. Åström (2014). "Minimal Solvers for Relative Pose with a Single Unknown Radial Distortion." In: *Proc. Conf. Computer Vision and Pattern Recognition*. Cited on pages 84 and 88.

Kukelova, Z. and T. Pajdla (2007a). "A minimal solution to the autocalibration of radial distortion." In: *In Proc. Conf. Computer Vision and Pattern Recognition*. Cited on pages 84 and 88.

Kukelova, Z. and T. Pajdla (2007b). "Two Minimal Problems for Cameras with Radial Distortion." In: *OMNIVIS*. Cited on page 84.

L., Gerard (1970). "On graphs and rigidity of plane skeletal structures." In: *Journal of Engineering mathematics*. Cited on page 51.

Larsson, V., C. Olsson, E. Bylow, and F. Kahl (2014). "Rank Minimization with Structured Data Patterns." In: *European Conf. Computer Vision*. Cited on pages 61, 62, 65, and 66.

Lebrecht, H. (1911). *Die graphische Statik der starren Systeme*. BG Teubner. Cited on pages 51, 52, and 54.

Liebelt, J. and C. Schmid (2010). "Multi-view object class detection with a 3D geometric model." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 120.

Lin, Z., M. Chen, L. Wu, and Y. Ma (2009). "The Augmented Lagrange Multiplier Method for Exact Recovery of a Corrupted Low-rank Matrices." In: *CoRR*. Cited on page 23.

Longuet-Higgins (1981). "A computer algorithm for reconstructing a scene from two projections." In: *Nature*. Cited on pages 12, 84, 97, and 98.

Lowe, D. G. (2004). "Distinctive image features from scale-invariant keypoints." In: *Int. Journal Computer Vision*. Cited on page 93.

Lucas, B. D. and T. Kanade (1981). "An Iterative Image Registration Technique with an Application to Stereo Vision." In: *International Joint Conference on Artificial Intelligence*. Cited on page 65.

Ma, Y., H. Derksen, W. Hong, and J. Wright (2007). "Segmentation of Multivariate Mixed Data via Lossy Data Coding and Compression." In: *Trans. Pattern Analysis and Machine Intelligence*. Cited on page 115.

Mangasarian, O. L. (1997). "Arbitrary-Norm Separating Plane." In: *Operations Research Letters*. Cited on pages 26 and 27.

Medved, D. et al. (2014). "Combining Text Semantics and Image Geometry to Improve Scene Interpretation." In: *Intl. Conference on Pattern Recognition Applications and Methods*. (Not cited.)

Naroditsky, O. and K. Daniilidis (2011). "Optimizing polynomial solvers for minimal geometry problems." In: *Int. Conf. Computer Vision*. Cited on page 90.

Nistér, D. (2004). "An Efficient Solution to the Five-Point Relative Pose Problem." In: *IEEE Trans. Pattern Anal. Mach. Intell.* Cited on pages 4, 13, 84, and 97.

Okatani, T., T. Yoshida, and K. Deguchi (2011). "Efficient algorithm for low-rank matrix factorization with missing components and performance comparison of latest algorithms." In: *Int. Conf. Computer Vision*. Cited on pages 22 and 23.

Olsen, Søren I. and Adrien Bartoli (2008). "Implicit Non-Rigid Structure-from-Motion with Priors." In: *Journal of Mathematical Imaging and Vision*. Cited on page 62.

Olsson, C. and M. Oskarsson (2011). "A Convex Approach to Low Rank Matrix Approximation with Missing Data." In: *Scandinavian Conf. on Image Analysis*. Cited on page 23.

Ott, P. and M. Everingham (2011). "Shared Parts for Deformable Part-Based Models." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 120.

Pollefeys, M. and D. Nister (2008). "Direct Computation of Sound and Microphone Locations from Time-Difference-of-Arrival Data." In: *International Conference on Acoustics, Speech and Signal Processing*. Cited on page 71.

Savarese, S. and F. Li (2007). "3D Generic Object Categorization, Localization and Pose Estimation." In: *Int. Conf. Computer Vision*. Cited on page 120.

Snavely, N., T. M. Seitz, and R. Szeliski (2006). "Photo tourism: exploring photo collections in 3D." In: *ACM SIGGRAPH*. Boston, Massachusetts, ACM. Cited on page 83.

Stewénius, H., F. Kahl, D. Nistér, and F. Schaffalitzky (2005). "A Minimal Solution for Relative Pose with Unknown Focal Length." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 4.

Strelow, D. (2012). "General and Nested Wiberg Minimization." In: *Conf. Computer Vision and Pattern Recognition*. Cited on pages 22, 42, and 45.

Sugaya, Y. and K. Kanatani (2004). "Geometric structure of degeneracy for multi-body motion segmentation." In: *In Workshop on Statistical Methods in Video Processing*. Cited on page 115.

Tardif, J.-P., A. Bartoli, M. Trudeau, N. Guilbert, and S. Roy (2007). "Algorithms for Batch Matrix Factorization with Application to Structure-from-Motion." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 22.

Tegen, A. et al. (2014). "Image segmentation and labeling using free-form semantic annotation." In: *Proceedings of the International Conference on Pattern Recognition*. (Not cited.)

Thomas, A. et al. (2006). "Towards Multi-View Object Class Detection." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 120.

Tomasi, C. and T. Kanade (1992). "Shape and Motion from Image Streams under Orthography: a Factorization Method." In: *Int. Journal Computer Vision*. Cited on page 41.

Tordoff, B. and D. W. Murray (2002). "Guided sampling and consensus for motion estimation." In: *European Conf. Computer Vision*. Cited on page 37.

Torre, F. De la and M. J. Black (2003). "A Framework for Robust Subspace Learning." In: *International Journal of Computer Vision*. Cited on page 22.

Triggs, B. (1999). "Camera Pose and Calibration from 4 or 5 Known 3D Points." In: *Int. Conf. Computer Vision*. Cited on page 122.

Tron, R. and R. Vidal (2007). "A Benchmark for the Comparison of 3-D Motion Segmentation Algorithms." In: *Conf. Computer Vision and Pattern Recognition*. Cited on pages 99 and 114.

Vedaldi, A., G. Guidi, and S. Soatto (2007). "Moving Forward in Structure From Motion." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 98.

Vidal, R. and R. Hartley (2004). "Motion Segmentation with Missing Data using PowerFactorization and GPCA." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 115.

Viola, P. and M. Jones (2001). "Rapid object detection using a boosted cascade of simple features." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 119.

Wiberg, T. (1976). "Computation of Principal Components When Data Are Missing." In: *Proc. Second Symp. Computational Statistics*. Cited on page 22.

Xiang, Y. and S. Savarese (2012). "Estimating the Aspect Layout of Object Categories." In: *Conf. Computer Vision and Pattern Recognition*. Cited on pages 120 and 121.

Xiong, F., O. I. Camps, and M. Sznaier (2012). "Dynamic Context for Tracking behind Occlusions." In: *European Conf. Computer Vision*. Cited on page 23.

Yan, J. and M. Pollefeys (2006). "A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and nondegenerate." In: *European Conf. Computer Vision*. Cited on page 115.

Yuille, A. and D. Snow (1997). "Shape and albedo from multiple images using integrability." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 44.

Zheng, Y., S. Sugimoto, S. Yan, and M. Okutomi (2012a). "Generalizing Wiberg algorithm for rigid and nonrigid factorizations with missing components and metric constraints." In: *Conf. Computer Vision and Pattern Recognition*. Cited on page 23.

Zheng, Y., G. Liu, S. Sugimoto, S. Yan, and M. Okutomi (2012b). "Practical Low-Rank Matrix Approximation under Robust L_1-Norm." In: *Conf. Computer Vision and Pattern Recognition*. Cited on pages 23, 42, 45, and 64.