Maximum Consensus with Mixed Integer Linear Programming

by

Yang Heng KEE

A dissertation submitted to the School of Computer Science,
The University of Adelaide for the degree of Master of Philosophy

August, 2016

Contents

List of Figures

List of Tables

Abstract

Maximum consensus is fundamentally important in computer vision as a criterion for robust estimation, where the goal is to estimate the parameters of a model of best fit. It is computationally demanding to solve such problems exactly. Instead, conventional methods employ randomised sample-and-test techniques to approximate a solution, which fail to guarantee the optimality of the result. This thesis makes several contributions towards solving the maximum consensus problem exactly in the context of Mixed Integer Linear Programming. In particular, two preprocessing techniques aimed at improving the speed and efficiency of exact methods are proposed.

Declaration

I, Yang Heng KEE, certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I give consent to this copy of my thesis when deposited in the University Library, being made available for loan and photocopying, subject to the provisions of the Copyright Act 1968.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Signature Date

Preface

During my studies at the University of Adelaide from 2014 to 2016, I have contributed to the production of one conference paper. Chapter 4 of this thesis is based on the content presented in the aforementioned paper, with details of the paper listed below:

Conference Publication

- T.-J. Chin, Y.H. Kee, A. Eriksson, and F. Neumann. Guaranteed outlier removal with mixed integer linear programs. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Acknowledgements

This study was an interesting experience for me, and the completion of this research would not have been possible without the support of many people.

First and foremost, I would like to express my deepest gratitude to my supervisor Dr. Tat-Jun Chin for his patience and his continued guidance and support. He was always at hand with his expertise and advice, and this journey would not have been possible without him.

It is also a pleasure to work with my co-supervisor Prof. David Suter, who held monthly research group meetings with students under his supervision to discuss and to throw around ideas regarding the current state of our research. I have benefitted greatly from those monthly sessions and enjoyed the group discussions with my peers.

Finally, I would like to thank my family and friends for providing their support and encouragement throughout my research for the past two years.

Chapter 1

Introduction

One of the earliest known attempts to enable computers to understand and interpret the world as we see it is known as The Summer Vision Project. Drafted by Seymour Papert in the July of 1966, participants were asked to "construct a system of programs to divide a picture into regions such as likely objects, likely background areas and chaos" [39]. Only a single summer was allocated for its completion, which turned out to be a gross underestimation of the vast scope and difficulty of the problem. Half a century later, there remain multiple research groups in universities and similar institutions from all over the world still working hard to understand, solve, and improve upon the task of computer-based vision.

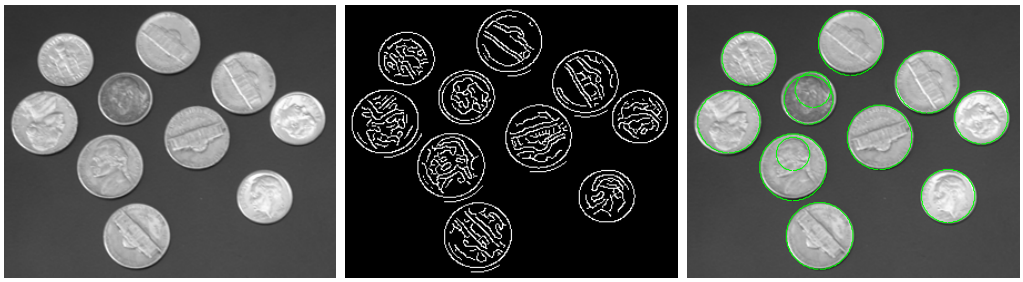
Human brains excel at processing the visual signals our eyes receive. Since birth, we began learning how to recognise, navigate and interact with the environment using mainly our sight. What we see as brightness, hues, and colours, are recorded by computers as a series of ones and zeroes of digitised measurements. From the perceived visual signals it is trivial for humans to identify and recognise shapes, objects scenes, and faces. However, giving computers the same level of ability is still a topic for ongoing research.

In the field of security, computer vision is used in biometrics authentication through iris and facial recognition. In medicine, computer vision is used to perform classification and registration of medical images to help identify potential lesions or tumours, and 3D human organ reconstruction is used to simulate a view of the internal structure of the patient's body for better diagnosis and visualisation in preparation for surgical procedures. Vision aided servos are also widely used in the industrial manufacturing process to provide guidance for parts assembly, control of timing and measurements, automated

sorting and classification, manipulation and interaction with various objects, and it is even used for quality control by automated imaging-based inspection for defect detection. In transportation we have autonomous self-driving vehicles equipped with cameras used for pedestrian detection and avoidance.

The applications previously mentioned are examples of high-level computer vision, which attempts to emulate human cognitive functions. In order to perform such high-level tasks, images have to be put through a pipeline of low to mid-level operations. Low-level vision is about extracting anything of interest from the image at the most basic level - things such as edges, ridges and localised points of interest such as corners and texture patches. In computer vision, this step is known as feature detection or feature extraction. Mid-level vision is where the observed features are processed, typically by selecting, matching, or segmenting relevant points or regions of interest. The information may undergo further processing, estimation, and refinement in the mid-level pipeline to produce models used in high-level tasks.

An example of low to mid-level vision processing is to identify circles in a photograph, such as the grayscale image in Fig. 1.1 (a). Low-level processing is illustrated in Fig. 1.1 (b), where an edge detection algorithm is employed to extract basic structural information from the image. The information is then passed on to the mid-level step, where a parameter estimation algorithm is used to identify potential circles. The circles identified are then superimposed upon the original image, as illustrated in Fig. 1.1 (c).



(a) An image of coins. (b) Low-level edge detection. (c) Mid-level circle fitting.

Figure 1.1: An example of low to mid-level vision[1].

The focus of this thesis will be on the mid-level pipeline, where data from low-level operations are processed.

[1] All 3 images were provided by and generated using MATLAB.

1.1 Parameter Estimation

A large number of tasks performed in computer vision involve using parametric models to describe information of interest, such as the calibration of cameras, the parameters of geometric shapes including straight lines and ellipses, and the different types of transformations an image had undergone. The majority of computer vision applications rely on estimating the parameters of such geometric models to perform high level vision tasks.

In the general sense, parameter estimation involves estimating the parameters of a model that best fits the observed data, making it an instance of the problem of optimisation, where the aim is to find or approximate the best possible solution from all feasible solutions. Over the years researchers from a wide range of fields including computer vision, signal processing, statistics and econometrics have developed different techniques to solve such problems, some of which will be reviewed in section 1.2.

To illustrate the challenges faced in parameter estimation, we begin by looking at the classic line fitting problem: given a set of 2D points, estimate a straight line that best fits the given points.

Figure 1.2: Points for line fitting.

The line fitting problem appears trivial at first glance. While there may be slight deviations in each given point, they all seem to conform to a single, straight line. It is common knowledge that a straight line can be drawn by connecting two unique points. To find a line of best fit, we can select any two points from the given set and draw a line through the two. However, it can be seen from Fig. 1.3 that the line obtained does not produce a good fit.

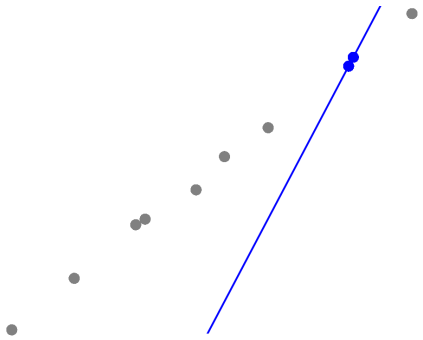


Figure 1.3: A line drawn using two points can result in a bad fit.

To explain this, recall the equation of a straight line

$$y = mx + c, \tag{1.1}$$

which contains two unknowns, the first unknown m being the slope of the line, and the second unknown c being the y-intercept. The two unknowns can be solved using two equations of the form (1.1) corresponding to two points. However, the problem becomes a little more complex when the number of points we have is greater than two.

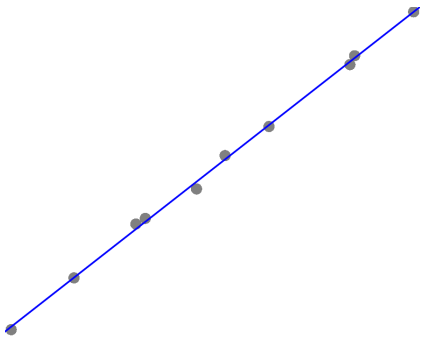


Figure 1.4: Line of best fit estimated using least squares.

When the number of data points or measurements exceeds the minimum required to form a model estimate, the system of equations is said to be overdetermined. Almost all problems encountered in computer vision consists of overdetermined systems, and the measurements we attain are often subject to the presence of noise, meaning that there are generally no single, exact solution that perfectly fits all the given measurements, as can be seen in the

case of Fig. 1.4. Rather than demanding for a perfect solution which does not necessarily exist, we instead seek a suitable approximation, and one way to do so would be to find a model that minimises some suitable cost function. The most common way to deal with linear problems similar to (1.1) is to find an estimate using an approach known as linear regression, which is commonly based on the method of linear least squares.

Unfortunately, regression based on linear least squares is well known for its lack of robustness [42], where a single outlier in the measurements can have a disastrous effect on its estimate, as shown in Fig. 1.5. Its intolerance towards outliers means that the least squares method is ill-suited for use in computer vision, where measurement errors are inevitable and may even outnumber the inlying data.

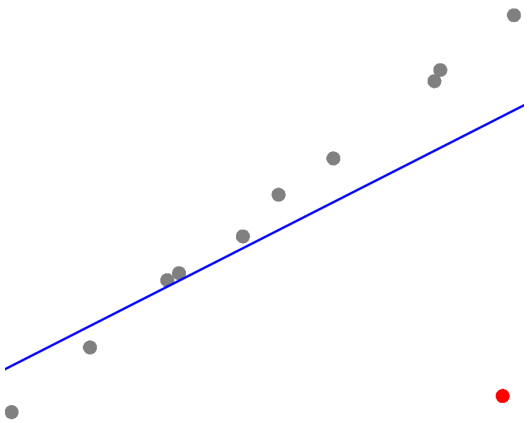


Figure 1.5: Least squares estimate skewed by a single (red) outlying point.

The challenge of parameter estimation in computer vision applications is to find suitable model parameters that best describes the measured observations despite the presence of *noise* and *outliers*. However, this is generally a computationally challenging task. In this thesis we shall focus on researching more efficient robust geometric optimisation algorithms for parameter fitting.

In basic geometry, parameter estimation is used in the fitting of lines, ellipses, planes, and hyperplanes. In computer vision parameter estimation is frequently used to solve problems such as optical flow correspondences for rigid motion segmentation (Fig. 1.6), the estimation of a homography matrix for image stitching to create panoramas (Fig. 1.7), and multiview triangulation for scene reconstruction (Fig. 1.8).

Figure 1.6: Motion segmentation from video sequences[1].

(a) Solving the correspondence problem [57]. (b) Stitched panorama.

Figure 1.7: Image stitching using homography estimation[2].

Figure 1.8: A reconstruction of the Notre Dame cathedral based on photos taken from http://www.flickr.com[3].

[1]Image from http://www.bristol.ac.uk/vi-lab/projects/casblip/.
[2]Photos from http://hugin.sourceforge.net/tutorials/two-photos/en.shtml.
[3]Image from http://phototour.cs.washington.edu.

1.2 Literature Review

1.2.1 Linear Regression

There are many ways to solve the problem of parameter estimation, and one of them is by means of regression. Over the years many different techniques and methods have been developed for regression, and some of the more common regression approaches are listed in this section.

1.2.1.1 Linear Least Squares

One way to approach parameter estimation is regression based on linear least squares [42]. As the name suggests, the least squares method finds an estimate to a given set of data by minimising the sum of squared errors, also known as residuals, between the estimate and the entire dataset. If the residual values are arranged in a vector, we will find that linear least squares regression is analogous to the minimisation of the L_2-norm or the Euclidean norm, which is basically the sum of all squared elements in a vector. Unfortunately, the method of least squares is sensitive to the presence of outliers, making it unsuitable for use in computer vision.

1.2.1.2 Smallest Maximum Error

A different method of approximation that has been gaining attention in the vision community is the minimisation of the L_∞-norm [24, 25, 27, 28, 30, 36, 46, 59]. The L_∞-norm of a vector is basically the entry with the largest absolute value, and thus minimising the L_∞-norm produces an estimate that minimises the value of the largest (maximum) error. Since the estimate is essentially being fitted to the worst outlying points, it is also easily biased by outliers. This property allows for the detection and removal of outliers during the optimisation process, and [46] provided a scheme for recursive outlier removal during geometric multi-view reconstruction using a sequence of images. The work of [30, 36, 59] proposed more efficient methods for dealing with outliers. The greatest advantage of the L_∞ framework is that many computer vision problems relating to multiple view geometry possesses a single, and hence global minimum on a convex parameter domain when formulated using the L_∞-norm [24]. [27, 28] expanded upon this concept of a global minimum and showed that a variety of structure and motion problems

can be cast as a quasiconvex optimisation problem within the L_∞ framework. A list of geometric vision problems solvable using L_∞ can be found in [37, 43].

1.2.1.3 Least Absolute Residual

Another metric that can be used for robut regression works by minimising the L_1 error norm, which is the sum of absolute values of all elements in a vector. The L_1-norm has been receiving significant attention [7, 8, 17, 18] for its uses in recovering sparse representations of a set of data, which is basically a parametric model with as little non-zero elements as possible. Sparse solutions are desired in many practical applications because it provides important benefits such as decreasing the use of antenna elements, reducing the number of measurements needed, and minimising data storage [17]. This makes it suitable for applications in geophysics, data compression, image processing, and sensor networks [58]. Based on this, [16] advocates the effectiveness of the L_1-norm in dealing with certain quasiconvex problems in multiple-view geometry. L_1-norm minimisation can also be used directly as a preprocessor to identify and weed out potential outliers [44]. Unfortunately, the L_1-norm approach is unable to cope well with gross outliers in the leverage [42, 50] (see Section 2.1 for definitions on "leverage" and "response").

1.2.1.4 M-Estimation

One of the earliest robust method is the M-estimator, which is a generalisation of maximum likelihood estimators and least squares [30, 50]. M-estimation works by replacing the traditional squared residuals cost function with another function of the residuals with certain robust properties [42, 50, 52, 60]. This can be solved using an iteration scheme based on reweighted least squares [42, 50, 52], which is unfortunately highly reliant on a good initialisation to avoid local minima. While it is indeed more robust, the M-estimator shares similar weaknesses with L_1 regression in that it is unable to cope with gross amounts of outliers in the leverage [42, 50].

1.2.1.5 Least Median of Squares

Another regression approach is the Least Median of Squares regression (LMS) [21], which is one of the first few approaches that is truly robust. LMS works by replacing the sum of squared residuals in the classical least squares regression with the median of all squared residuals, thereby making it highly

resistant to outliers in the data [42, 50]. Unfortunately, since the median function is not differentiable, an exhaustive search in the space of all possible estimates generated from subsets of the data is required to solve LMS exactly [34, 60], which is expensive to compute. Consequently, approximation methods are usually the only viable option for most practical applications, with random sampling being the most commonly used approach [34, 50].

1.2.2 Maximum Consensus

Maximum consensus is one of the most popular robust parameter estimation criterions in computer vision. The concept of "consensus" in parameter fitting was first proposed by [22], which introduces the use of a *threshold* or *error tolerance* value to separate the data into inliers and outliers. By consensus, we refer to the set of data points that "agrees" with the estimated model, whose error to the estimate falls within the predetermined threshold. This is known as the inlier set. The rest of the data whose error lies without the threshold is called the outlier set. Consequently, consensus maximisation aims to find an estimate that maximises the cardinality of the inlier set.

1.2.2.1 Random Sample Consensus

The prevailing paradigm for robust estimation in computer vision is an approximation algorithm known as RANdom SAmple Consensus (RANSAC) [22]. Based on an iterative resampling technique, RANSAC generates a new estimate each iteration using a minimal number of samples randomly drawn from the data. The generated estimates are then evaluated against all available data and scored by the size of its consensus set. The estimate with the highest score is returned as the best solution when the algorithm terminates. LMS and RANSAC share a common similarity in their methods - the solution of LMS can be approximated using random sampling, whereas RANSAC is developed based entirely on the idea of random sampling. This unfortunately means that RANSAC shares the same problem with many randomised approximation methods, meaning that it is non-deterministic and offers no guarantees to the optimality of its solution.

1.2.2.2 Mixed Integer Linear Programming

Linear programming (LP) is a method used to achieve the best outcome in a linear model, whose requirements are constrained by a system of linear in-

equalities. Mixed Integer Linear Programming (MILP) is an extension of LP problems where some of its variables are further constrained to be integers, and it can be solved using a search algorithm known as Branch-and-Bound (BnB) [29, 41, 61]. On the other hand, the maximum consensus problem is an instance of the Maximum Feasible Subsystem (MaxFS) problem [11, Chap. 7], which can be formulated as a MILP by introducing additional binary variables and a sufficiently large constant known as "slack". Unfortunately, the MILP formulation is not appealing due to potential numerical issues due to overly large slack. Moreover, currently known approaches to solving MILPs are slow in nature and incurs a large computational expense, thereby making it practical only for relatively small problem sizes, low dimensionality, and a low outlier rate.

1.3 Thesis Contributions

In this thesis, we investigated ways to improve the efficiency of exact methods, with a primary focus on solving the maximum consensus problem using commercial off-the-shelf MILP solvers. Contributions of this thesis include devising a method to improve computational speed by minimising appropriate slack values, and also the development of a a more efficient preprocessing scheme to speed up maximum consensus.

1.4 Thesis Organisation

This thesis is organised into five chapters.

Chapter 1 We give an introduction to parameter estimation, and reviewed some of the more common approaches used to solve the problem.

Chapter 2 We provide a theoretical foundation on how some of the approaches reviewed in chapter 1 are applied.

Chapter 3 We devise a method to alleviate issues with computation speed in MILP.

Chapter 4 We present an efficient preprocessing scheme to to improve the efficiency of outlier removal using commercial MILP solvers.

Chapter 5 We give a summary of the thesis.

Chapter 2

Parameter Estimation Methods

Previously, we discussed why there is a need for parameter estimation or model fitting in computer vision. We have also discussed the challenges involved with estimating the parameters of geometric models.

The goal of this chapter is to provide a theoretical foundation on how parameter estimation techniques are applied in the context of computer vision by delving deeper into some of the methods surveyed in the previous chapter.

2.1 Linear Regression

Before we look into solving linear regression, we need a concrete definition of the problem we are working with. In the context of computer vision, say we have $\mathcal{X}$ as a set of data containing N measurements represented using

$$\mathcal{X} = \{\mathbf{a}_i, b_i\}_{i=1}^{N} \ , \tag{2.1}$$

where each $\mathbf{a}_i$ is a vector of n predictor variables (the "*leverage*"), and each b_i is a measured scalar value (the "*response*"). We can consider $\mathcal{X}$ to be an overdetermined set of *inhomogeneous* linear equations represented using

$$A\boldsymbol{\theta} \approx \mathbf{b} \ , \tag{2.2}$$

where A is a $N \times n$ matrix with $N > n$, $\boldsymbol{\theta}$ is a column vector parameterised by $\boldsymbol{\theta} \in \mathbb{R}^n$, and $\mathbf{b}$ is a column vector with N entries. The i-th row of matrix A corresponds to $\mathbf{a}_i$ of the i-th datum in $\mathcal{X}$, whereas the i-th entry of vector $\mathbf{b}$ corresponds to b_i.

The goal of linear regression is to estimate the parameters of a model $\boldsymbol{\theta}$ that best represents the data given in $\mathcal{X}$. The error between the i-th datum and a model $\boldsymbol{\theta}$ is called the residual, denoted by $r_i(\boldsymbol{\theta})$. In linear regression, the residual of datum i is calculated as

$$r_i(\boldsymbol{\theta}) = |\mathbf{a}_i\boldsymbol{\theta} - b_i| \ . \tag{2.3}$$

We represent the residual of the entire dataset with a vector:

$$\mathbf{r}(\boldsymbol{\theta}) \quad = \quad \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{pmatrix} \ , \tag{2.4}$$

where the i-th entry of $\mathbf{r}(\boldsymbol{\theta})$ corresponds to the residual of the i-th datum. An example can be given using line fitting, where the x and y-coordinates of the given points corresponds to the set of $\mathcal{X}$, and the model $\boldsymbol{\theta}$ corresponds to the two unknown variables that define a straight line. We achieve this by manipulating the equation of a straight line (1.1):

$$\begin{bmatrix} x & 1 \end{bmatrix} \begin{pmatrix} m \\ c \end{pmatrix} = y \ . \tag{2.5}$$

Using (2.5) we formulate $\mathcal{X}$ in the form of (2.1), where $\mathbf{a}_i = (x_i, 1)$ contains the x-coordinates of the i-th point, $b_i = y_i$ contains the y-coordinates of the i-th point, and $\boldsymbol{\theta} = (m, c)^T$ contains the parameters for the line of best fit. Geometrically, this means that the linear regression residual corresponds to the vertical distance between each point and the estimated $\boldsymbol{\theta}$. Since we are dealing with an overdetermined system of linear equations under the influence of outliers and noise, it is almost always true that $\mathbf{r}(\boldsymbol{\theta}) \neq \mathbf{0}$ for any $\boldsymbol{\theta}$. Different regression methods differ by how the error values in $r(\boldsymbol{\theta})$ are aggregated and minimised.

2.1.1 Linear Least Squares

The objective function of this regression technique minimises the L_2-norm, or the sum of squared residuals between $\mathcal{X}$ and $\boldsymbol{\theta}$:

$$\min_{\boldsymbol{\theta}} \sum_i r_i^2(\boldsymbol{\theta}) \ . \tag{2.6}$$

The solution $\boldsymbol{\theta}$ to (2.6) can be found in closed form using a matrix decomposition technique known as singular value decomposition (SVD). We will not discuss the details of SVD, and interested readers can instead refer to [23, 26] for more information.

2.1.2 Smallest Maximum Error

The L_∞-norm of a vector is basically the vector entry with the largest absolute value, and thus L_∞-norm minimisation is also known as the minmax problem, in which the desired estimate minimises the maximum error

$$\min_{\boldsymbol{\theta}} \max_{i} r_i(\boldsymbol{\theta}) \ . \tag{2.7}$$

Readers are referred to [9, Chap. 2] to learn more about solving linear L_∞ minimisation problems using an efficient vertex-to-vertex descent algorithm. We can also formulate and solve L_∞ minimisation as a linear program (LP)

$$\begin{aligned} \min_{\boldsymbol{\theta},z} \quad & z \\ \text{subject to} \quad & r_i(\boldsymbol{\theta}) \leq z \quad \forall i \in \{1, 2, ..., N\} \ , \\ & z \geq 0 \ , \end{aligned} \tag{2.8}$$

which will be discussed in section 2.2.2. Since the estimate in L_∞-norm minimisation is obtained by minimising the largest residual values, it is essentially fitting the estimate to the worst outlying points, making it non-robust.

2.1.3 Least Absolute Error

The L_1-norm is also known as the Manhattan norm, which is the sum of absolute values of all elements in a vector. Hence the objective function of L_1-norm regression minimises the sum of absolute errors between $\mathcal{X}$ and $\boldsymbol{\theta}$:

$$\min_{\boldsymbol{\theta}} \sum_{i} r_i(\boldsymbol{\theta}) \ . \tag{2.9}$$

This objective function places equal weight on all given measurements, making the L_1-norm more robust against outliers in the data, unlike the sum of squared residuals which gives greater emphasis to measurements with larger

errors. We can formulate and solve L_1 minimisation as a linear program (LP)

$$\begin{aligned} \min_{\boldsymbol{\theta},\mathbf{z}} \quad & \sum_i z_i \\ \text{subject to} \quad & r_i(\boldsymbol{\theta}) \leq z_i \quad \forall i \in \{1, 2, ..., N\} \ , \\ & z_i \geq 0 \ , \end{aligned} \tag{2.10}$$

which will be further discussed in section 2.2.2. The L_1-norm approach is unfortunately unable to cope well with gross outliers in the leverage [42, 50], meaning that more robust methods are required.

2.1.4 M-Estimation

Another robust approximation method widely used in computer vision and statistics literature is the M-estimatior, which is a generalisation of maximum likelihood estimators (MLE) and least squares [30, 50]. Similar to the L_1-norm, M-estimation is proposed as an alternative to the L_2-norm to reduce the effect of outliers [60]. M-estimation achieves this by replacing the squared residuals cost function r_i^2 in (2.6) with another function of the residuals $\rho(r_i)$, resulting in the new objective:

$$\min_{\boldsymbol{\theta}} \sum_i \rho(r_i) \ . \tag{2.11}$$

The replacement cost function $\rho(u)$ has to satisfy the following mathematical properties: it has to be symmetric $\rho(u) = \rho(-u)$, has a unique minimum at zero $\rho(0) = 0$, nonnegative or positive-definite $\rho(u) \geq 0 \ \forall u$, and monotonically non-decreasing with increasing $|u|$ [42, 50, 52, 60]. If $\rho(u) = u^2$, (2.11) becomes equivalent to the least squares objective in (2.6).

The solution to (2.11) can be calculated using an iteration scheme known as iteratively reweighted least squares (IRLS) [42, 50, 52, 60]

$$\min_{\boldsymbol{\theta}} \sum_i w(r_i^{[k-1]})(r_i)^2 \ , \tag{2.12}$$

where the superscript $[k-1]$ indicates a previous iteration. IRLS is achieved by alternating between calculating the weights $w(r_i)$ using the latest estimate $\boldsymbol{\theta}$, and then fixing the weights to find the next estimate $\boldsymbol{\theta}$ using

$$\sum_i w(r_i) r_i \frac{\partial r_i}{\partial \boldsymbol{\theta}} = 0 \ . \tag{2.13}$$

The starting estimate $\boldsymbol{\theta}^{[0]}$ for initialising the weights $w(r_i^{[0]})$ in the initial iteration can be obtained using various ways such as linear regression or other more robust methods. The *weight function* is calculated using

$$w(u) = \frac{\psi(u)}{u}, \tag{2.14}$$

in which the *influence function*

$$\psi(u) = \frac{\partial \rho(u)}{\partial u} \tag{2.15}$$

is the derivative of $\rho(u)$. The influence function provides a measure of the influence of a single datum on the estimate, such as for the case of least squares where $\rho(u) = u^2$, $\psi(u) = 2u$. This indicates that the influence of each measurement in $\mathcal{X}$ increases linearly with the size of its error, illustrating that the least squares approach is indeed non-robust. In order to reduce the influence of outliers, the replacement cost function $\rho(u)$ has to increase subquadratically, meaning that it has to increase slower than the quadratic square u^2. Different M-estimators have been proposed based on different $\rho(\cdot)$ functions, and interested readers can refer to [5, 50, 60] and the references therein for a discussion on the more commonly used estimators.

The downside to IRLS is that it is highly reliant on a good initialisation to avoid converging towards a local minima.

2.1.5 Least Median of Squares

Another widely used estimator is the Least Median of Squares (LMS) [21]. LMS works by replacing the sum of squared residuals in classical least squares regression with the median of squared residuals.

$$\min_{\boldsymbol{\theta}} \operatorname*{median}_{i} r_i^2(\boldsymbol{\theta}) \tag{2.16}$$

This objective function makes LMS highly resistant (robust) against outliers, allowing it to tolerate up to 50% contamination in the data [42, 50].

Unfortunately, since the median function is not differentiable, an exhaustive search in the space of all possible estimates generated from the data is required to solve the problem exactly [34, 60]. This is impractical even for problems of moderate sizes. According to [50], there exists computationally efficient algorithms to solve LMS exactly for special cases such as the fitting

of regression lines [19, 49]. However, approximation methods are usually the preferred option for more general settings, with Monte Carlo (random sampling) methods being the most commonly used approach [34, 50]. This results in an approach rather similar to RANSAC. More on RANSAC and the specifics of random sampling will be discussed in section 2.2.1, with the main difference between the two algorithms being their objective functions.

2.2 Maximum Consensus

Maximum consensus is popular as a parameter estimation criterion because it is very robust, and has a clear cut definition of what consists as "inliers" and "outliers". Given a set of measurements $\mathcal{X} = \{\mathbf{a}_i, b_i\}_{i=1}^{N}$, maximum consensus seeks to find an estimate $\boldsymbol{\theta}$ that maximises the number of measurements whose residuals $r_i(\boldsymbol{\theta})$ are within a predetermined threshold ϵ.

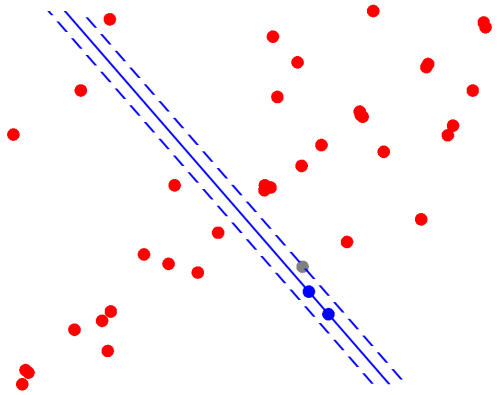
$$\begin{aligned} \max_{\boldsymbol{\theta},\, \mathcal{I} \subseteq \mathcal{X}} \quad & |\mathcal{I}| \\ \text{subject to} \quad & r_i(\boldsymbol{\theta}) \leq \epsilon \quad \forall \{\mathbf{a}_i, b_i\} \in \mathcal{I} \end{aligned} \tag{2.17}$$

The solution to (2.17) gives us the maximum consensus set, denoted as $\mathcal{I}^*$, with a consensus size of $|\mathcal{I}^*|$. We shall refer to the data contained within consensus set $\mathcal{I}$ as *inliers*, and the data within the set of $\mathcal{X} \setminus \mathcal{I}$ as *outliers*. The challenge of maximum consensus is to find the optimum estimate $\boldsymbol{\theta}^*$ that maximises the cardinality of $\mathcal{I}^*$ for problems that are often overdertermined and subject to the presence of noise and outliers.

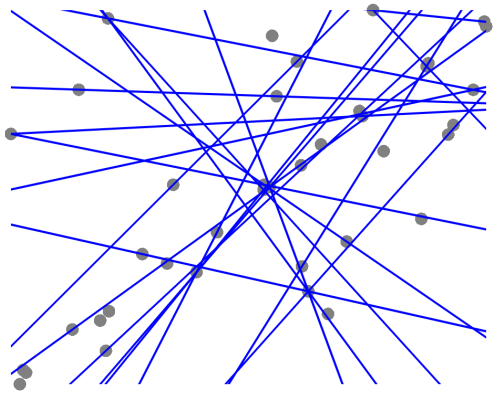
2.2.1 RANSAC

We mentioned previously that RANdom SAmple Consensus (RANSAC) is one of the more popular robust optimisation technique for consensus maximisation in computer vision. RANSAC is developed based on random sampling, where each candidate solution or hypothesis $\boldsymbol{\theta}$ is generated by fitting a model to a minimal subset of points randomly sampled from the dataset. The parameters of each hypothesis $\boldsymbol{\theta}$ can be calculated using standard regression techniques such as linear least squares. Each candidate $\boldsymbol{\theta}$ is then evaluated against the entire dataset and scored based on the size of their consensus set. Once the RANSAC algorithm terminates, we can further improve the solution by performing a final regression over the largest known consensus

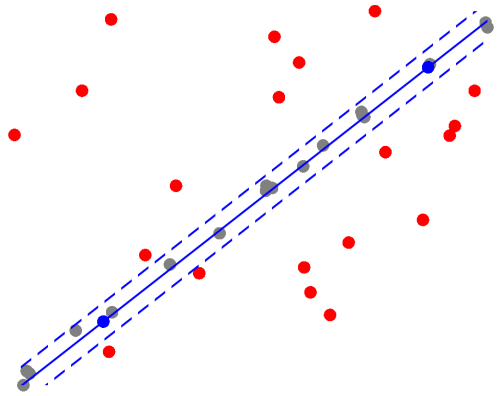
set to calculate the model of best fit. The workings of RANSAC can again be illustrated using the line fitting problem.



(a) The first RANSAC iteration.



(b) 20 consecutive RANSAC iterations.



(c) Best solution after 1000 iterations.

Figure 2.1: RANSAC in action.

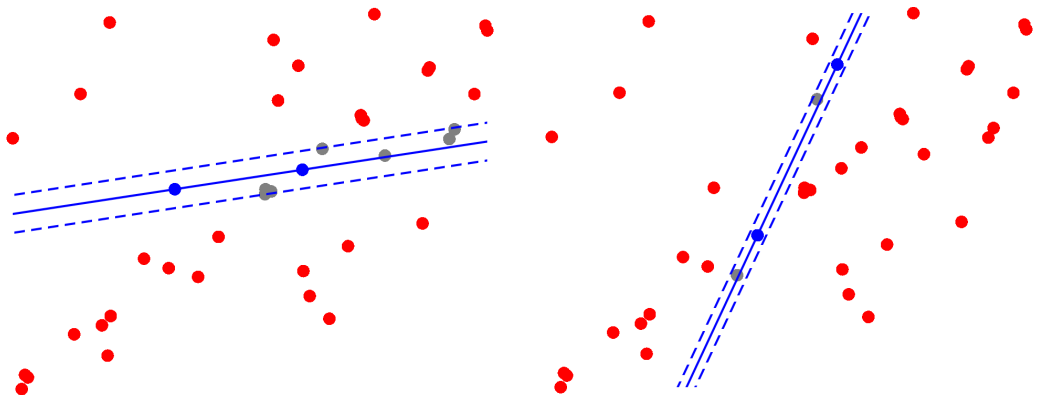
Fig. 2.1 (a) illustrates the first RANSAC hypothesis generated using two points (a minimum of two points is required to define a line) sampled randomly from the given measurements. The points highlighted in blue shows the minimum subset used to generate the candidate estimate, whereas the points highlighted in red represent measurements that "disagrees" with the candidate, meaning that their errors to the hypothesised line is greater than the predetermined threshold. Fig. 2.1 (b) illustrates the next 20 RANSAC iterations, where multiple candidates are generated and evaluated. Finally, Fig. 2.1 (c) illustrates the best estimate returned by RANSAC when it ter-

minates after 1000 iterations. The intuition behind RANSAC is that if the randomly selected minimal subset is free of outliers, then there is a very good chance for the generated estimate to produce a good overall fit of the inlying data. The more times you iterate, the better your chances will be of finding a good estimate with a larger consensus size.

Maximum consensus requires the definition of a *threshold*, which we shall designate as ϵ, with $\epsilon \geq 0$. The threshold can be visualised as a "strip" that encloses the estimated model, with the width of the strip determined by the value of ϵ. Points that fall within the strip are known as *inliers*, and their corresponding residuals are less than or equal to the threshold. Conversely, points that lie beyond the strip are known as *outliers*, and their corresponding residuals are greater than the threshold.

$$\begin{aligned} r_{inliers}\left(\boldsymbol{\theta}\right) &\leq \epsilon \\ r_{outliers}(\boldsymbol{\theta}) &> \epsilon \end{aligned} \tag{2.18}$$

The value of ϵ is usually defined as the maximum tolerable deviation from the estimated model under the effects of noise. RANSAC is highly sensitive to the choice of a noise threshold - if ϵ is too large, RANSAC tends to judge all candidates to be equally good, which would likely result in a poor estimate [53]. If ϵ is too small, RANSAC tends to become unstable and results would differ significantly across different RANSAC runs over the same dataset.



(a) A RANSAC candidate. (b) Another RANSAC candidate.

Figure 2.2: An illustration of the threshold "strip".

Fig. 2.2 illustrates how each randomly generated candidate is evaluated in a single RANSAC run. Note how the width of the strip for the two candi-

dates seem to be different even though they are both generated in the same RANSAC run using the same threshold value ϵ. This artefact is an interesting consequence of adopting the linear regression residual in equation (2.5), which causes the error of each measurement to correspond to the vertical distance between the point and the estimated line

$$r_i(\boldsymbol{\theta}) = | \begin{bmatrix} x_i & 1 \end{bmatrix} \boldsymbol{\theta} - y_i | \ .$$

This results in a strip that is infinitely thin when the slope of the line tends to infinity (a vertical line), whereas the width of the strip is maximised when the slope of the line is 0 (a horizontal line). In this thesis we choose to use the linear regression residual for sake of convenience and ease of implementation, rather than a residual based on orthogonal distance [40, 52] which is more complex to implement and solve (in the context of line fitting, orthogonal distance is the shortest perpendicular distance from point to line).

Another aspect of concern is the termination criterion: how many randomly selected minimal subsets should RANSAC evaluate before terminating? Ideally, we want to evaluate all possible subsamples of the data, but this is computationally infeasible even with problems of moderate sizes. Instead, we aim to sample enough subsets to lend *statistical significance* to the best RANSAC hypothesis [53]. Let k be the number of RANSAC iterations needed to make the result statistically significant, and $\mathcal{P}$ be the probability of successfully selecting a random minimal subset that contains only inliers, which will likely give us a good estimate. We calculate k using

$$k = \frac{\log(1 - \mathcal{P})}{\log(1 - w^n)} \ , \tag{2.19}$$

where n is the minimum number of measurements required to generate a candidate estimate, and w is the probability of choosing an inlier each time a single measurement is selected, which makes it equivalent to the inlier ratio

$$w = \frac{\text{number of inliers in the data}}{\text{total number of measurements in the data}} \ . \tag{2.20}$$

While it is usually non-trivial to know w beforehand, we can get better and better estimates of the true inlier ratio as better candidates are found in consecutive iterations. We derive (2.19) by assuming that the inlier ratio w is known, and hence the probability of randomly selecting a minimal subset containing n inliers is w^n, whereas the probability of selecting a random

minimal subset containing *at least* one outlier is $1 - w^n$. Out of k randomly selected minimum subsets, the probability of failing to sample an all-inlier subset is thus $(1-w^n)^k$, which can be seen as the probability of failure, where RANSAC fails to generate a good estimate. This is in contrast to $\mathcal{P}$, which is defined as the probability of success. Thus

$$1 - \mathcal{P} = (1 - w^n)^k \ . \tag{2.21}$$

Equation (2.19) is derived by taking the logarithm on both sides of (2.21). Generally a high probability of successfully selecting an uncontaminated minimal subset is preferred, usually with $\mathcal{P} > 0.95$, where $0 \leq \mathcal{P} \leq 1$.

RANSAC is inherently nondeterministic due to the Monte Carlo nature of its algorithm, and running RANSAC twice on the same dataset $\mathcal{X}$ with the same parameters can yield different results. While RANSAC is capable of finding structures formed by substantially fewer than half of the data, it might also return random structures if the inlier ratio is very low [50]. Another lesser known issue is that the optimal solution $\boldsymbol{\theta}^*$ to $\mathcal{X}$ might not exist within the set of possible estimates generated using a minimal subset [10]. This highlights one of the bigger concerns with RANSAC and other similar randomised approaches in that it offers no guarantees to the optimality of its solution, and there is no way to efficiently tell whether its solution is even a satisfactory estimate of the true optimal.

Several variants of RANSAC have been developed over the years in an attempt to overcome the deficiencies of the algorithm, and to increase its speed and reliability. An example is the PROgressive SAmple Consensus (PROSAC) algorithm [12], which exploits a priori knowledge of the input data to guide the sampling procedure, increasing the chances of examining uncontaminated minimal subsets early on. This improves the overall efficiency and accuracy of PROSAC over the standard RANSAC algorithm, allowing it to function better with data containing high outlier rates. Another variant is known as Maximum Likelihood Estimation SAmple Consensus (MLESAC), which abandons the idea of maximum consensus, opting instead to evaluate each of its generated hypothesis based on maximum likelihood [53]. Other variants proposed as improvements to the original RANSAC algorithm include R-RANSAC [13], LO-RANSAC [14], and Preemptive RANSAC [33], just to name a few. Unfortunately, since the RANSAC variants all follow the same hypothesise and test framework based on random sampling, none of them are guaranteed to find an exact (i.e. the optimal) solution.

2.2.2 Mixed Integer Linear Programming

A linear program (LP) is a method widely used in various industries for the purpose of resource optimisation, with the goal of achieving the best outcome (such as maximising profits or minimising cost). As can be inferred from its name, LP is only suited for solving problems whose constraints and objective function is linear (or can be linearised), in which case the optimal solution lies within a convex feasible region. Canonically, LP can be expressed as

$$\textit{objective function:} \quad \min \ \mathbf{c}^T\boldsymbol{\theta} \ , \tag{2.22a}$$

$$\textit{subject to:} \quad A\boldsymbol{\theta} \leq \mathbf{b} \ , \tag{2.22b}$$

$$\boldsymbol{\theta} \geq \mathbf{0} \ , \tag{2.22c}$$

where $\boldsymbol{\theta}$ is a vector of variables to be determined by solving the LP, while vectors $\mathbf{b}$ and $\mathbf{c}$ and matrix A consist of known coefficients. The objective of LP is to either minimise or maximise the objective value derived from the linear objective function (2.22a) subject to restrictions set by the various linear inequality constraints (2.22b) and variable bounds (2.22c), which specify the convex polytope in which the objective function is to be optimised. LP can be solved using conventional simplex or interior-point (barrier) methods

A Mixed Integer Linear Program (MILP) is a generalisation of LP where *at least one* of the unknown variables in $\boldsymbol{\theta}$ is further constrained to be integer-valued. There is an exact MILP formulation for maximum consensus, the specifics of which will be discussed in Chapter 3. MILP can be solved using search methods such as Branch-and-Bound (BnB) and Cutting-Plane. State-of-the-art MILP solvers usually work by relaxing the integer constraints to become continuous values, and then solving for the resultant LP. The aforementioned search methods are then applied to yield multiple new (and further constrained) versions of the original problem, which can again be solved as a LP. This solve-and-search process is repeated until an optimal solution to the original MILP problem is found. There exists many commercial off-the-shelf optimisation software such as CPLEX, MOSEK, and Gurobi which can be used as a "black box" for solving MILP. Interested readers are referred to [15, 29, 32, 41, 61] for further details on how LP and MILP are solved.

The main attraction of exact methods such as MILP is in finding the optimal solution. The drawback, however, lies in its speed and efficiency. MILP is very slow compared to the likes of RANSAC for problems of high dimensions and a large number of constraints. In the following chapters of this thesis we investigate ways to alleviate this issue.

Chapter 3

Big-M and M-bisection

In the previous chapter (section 2.2), we talked about consensus maximisation as a robust parameter estimation criterion. Then in section 2.2.1, we showed how maximum consensus can be solved using RANSAC. However, RANSAC is non-deterministic and suboptimal, and global optimisation techniques are preferred in some applications for its capability to find an exact solution. In section 2.2.2, we mentioned that maximum consensus can be formulated and solved exactly as a Mixed Integer Linear Programming (MILP) problem, and its weaknesses in terms of speed and efficiency.

In this chapter, we will discuss the details of the MILP formulation, and suggest a method to improve the speed of its computation.

3.1 Feasibility and Maximum Consensus

Before we begin talking about the MILP formulation, we need to introduce the concept of *feasibility*. In Fig. 3.1 we illustrate a system of linear inequalities in 2D, with each blue line representing a unique inequality constraint. Every linear inequality constraint defines a closed half-space (illustrated using gray arrows) in the search region in which the constraint is satisfied. The shaded region represents a mutual intersection of all closed half-spaces, and it is called the *feasible region*, in which *all* constraints are simultaneously satisfied. Within the feasible region lies the optimal solution, which in this case is a single 2D point, and it can be computed using linear programming algorithms. If no feasible region exists, then there exists no solution where all constraints are satisfied, and the problem is said to be infeasible.

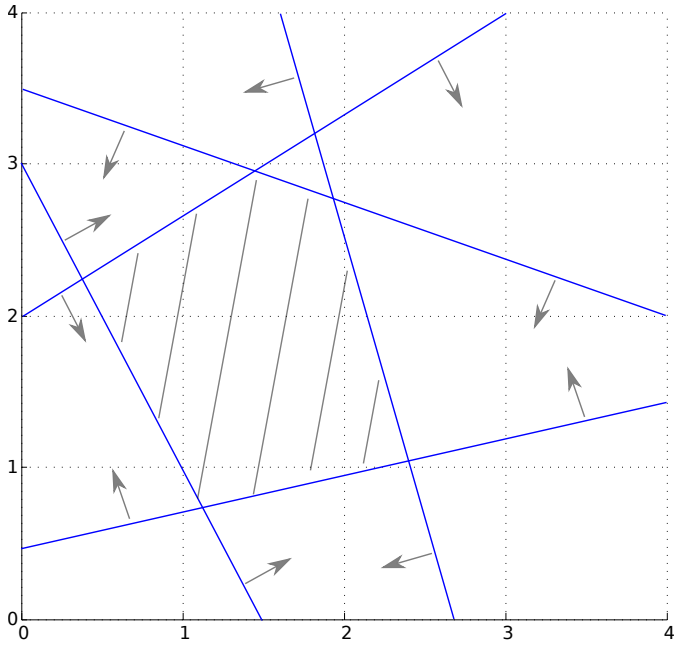


Figure 3.1: A graphed system of linear inequalities.

How can we express maximum consensus as a system of linear inequalities? We illustrate this again using an example based on the classic line fitting problem. Note that while this example is based on a low-dimensional problem in 2D, the concepts discussed can be generalised to linear problems in higher dimensions. Given a set of data $\mathcal{X} = \{\mathbf{a}_i, b_i\}_{i=1}^{N}$ containing N measurements, recall that the error of each measurement to the estimated model $\boldsymbol{\theta}$ can be calculated using the linear regression residual (2.3) as

$$r_i(\boldsymbol{\theta}) = |\mathbf{a}_i\boldsymbol{\theta} - b_i| \ .$$

Then recall (2.5) from Section 2.1, which shows how the data for line fitting (consisting of N 2D points, or N pairs of x and y-coordinates) can be reformulated in the form of $\mathcal{X}$ such that $\mathbf{a}_i = (x_i, 1)$ contains the x-coordinates of the i-th point, $b_i = y_i$ contains the y-coordinates of the i-th point, and $\boldsymbol{\theta} = (m, c)^T$ contains the parameters for the line of best fit, where m is the slope of the line and c is the y-intercept. Hence the residual becomes

$$r_i(\boldsymbol{\theta}) = |mx_i + c - y_i| \ .$$

Now recall from Section 2.2 that maximum consensus aims to maximise the cardinality of the inlier set, where the residual value of inlying measurements

fall within a predetermined threshold ϵ

$$r_i(\boldsymbol{\theta}) \leq \epsilon \ . \tag{3.1}$$

This gives us a set of linear inequalities constraints. Following the rule of absolute values, each derived residual results in a pair of inequality constraints

$$\mathbf{a}_i\boldsymbol{\theta} - b_i \leq \epsilon \ , \tag{3.2a}$$
$$-\mathbf{a}_i\boldsymbol{\theta} + b_i \leq \epsilon \ . \tag{3.2b}$$

Each pair can be seen as a strip of constant width enclosing the region that satisfies both constraints, in which the width of the strip is defined by the value of the threshold ϵ. With the line fitting problem, we can illustrate this using a graph plotted against the parameters $\boldsymbol{\theta} = (m, c)^T$.

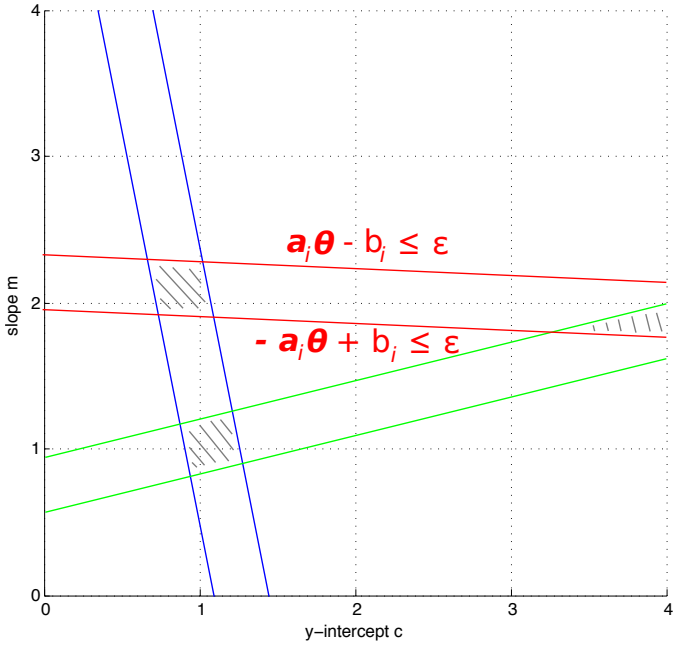


Figure 3.2: Linear inequality constraints for line fitting.

The graph in Fig. 3.2 is drawn based on a dataset consisting of three unique 2D points ($N = 3$). Each pair of linear inequality constraints derived are distinguished using the colours red, green, and blue, and the regions where the strips overlap are shaded in gray. Note how each shaded region

satisfies only two pairs of constraints - there exists no region where all strips simultaneously intersect. The line fitting problem in Fig. 3.2 is considered infeasible and unsolvable in LP, because there exists no single point that lies simultaneously within all three strips. This is a common occurrence in overdetermined systems containing outliers, which will usually result in a set of contradicting constraints. Unfortunately, as mentioned back in Chapter 1, almost all problems encountered in computer vision consists of overdetermined systems, and it would be quite troublesome if we are unable to find solutions to a majority of those problems. Intuitively, we know that maximum consensus requires us to find the point in space that falls within as many strips as possible, but how exactly can this be achieved?

3.2 The Big-M Method

It has been mentioned at the start of the chapter that there exists an exact MILP formulation for the maximum consensus problem. Specifically,

$$\min_{\boldsymbol{\theta},\mathbf{z}} \quad \sum_i z_i \tag{3.3a}$$

$$\text{subject to} \quad |\mathbf{a}_i\boldsymbol{\theta} - b_i| \leq \epsilon + z_i M \ , \tag{3.3b}$$

$$z_i \in \{0, 1\} \ . \tag{3.3c}$$

The above formulation (3.3) is also known as the Big-M method [11], where M is a large positive constant known as *slack*, whereas z_i are known as *indicator variables*.

Recall from Section 2.2.2 that MILP is a generalisation of LP, where some of the variables are required to take on integer values, rather than being continuous. This additional requirement implies that MILP can not be solved using linear programming algorithms. The unknown variables to be determined in (3.3) are the continuous variables in $\boldsymbol{\theta}$ and the binary indicator variables in $\mathbf{z} = (z_1, z_2, \ldots, z_i)^T$. Indicator variables can be viewed as dummy binary values used to "activate" the corresponding slack M in order to "satisfy" an inequality constraint. Activating a slack to artificially compensate for a constraint labels the corresponding datum as an outlier, because this indicates that the residual of the corresponding datum is in effect greater than the allowable threshold $r_i(\boldsymbol{\theta}) > \epsilon$, and M has to be activated to satisfy the aforementioned constraint. Thus the objective of minimising

the sum of z_i is to find a solution $\boldsymbol{\theta}^*$ that minimises the number of active indicators (the outliers). The use of a large constant to "ignore" constraints is a common practice in optimisation [11, 38].

From (3.2) we know that if $\mathcal{X}$ has N measurements, (3.3b) will comprise of $2N$ inequality constraints. We can see from the illustrations in Section 3.1 that each constraint can be considered individually independent of each other. Hence for the sake of brevity, from this point onwards we shall represent the constraints in (3.3b) using a more general expression

$$\mathbf{a}_i\boldsymbol{\theta} - b_i \leq \epsilon + z_i M \ . \tag{3.4}$$

The geometrical effect of adding slack M is shown in Fig. 3.3, which essentially expands the closed half-space defined by the corresponding constraint.

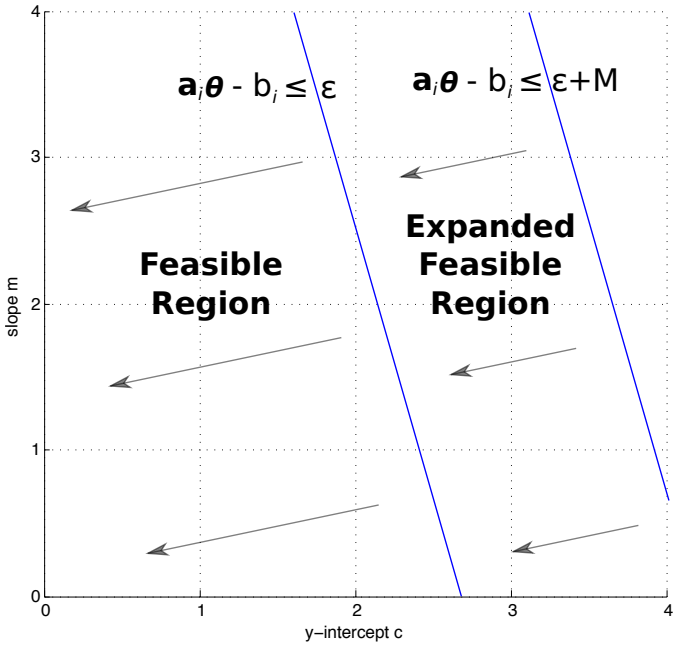


Figure 3.3: Expanding the feasible region of a single constraint.

Let us revisit the line fitting problem discussed in Section 3.1. Fig. 3.4 shows what we can potentially attain if we artificially expand the feasible region of a *single* residual thresholding constraint derived from (3.1) by activating M. The shaded region now represents an overlap of all constraints - a feasible region where all constraints are simultaneously satisfied.

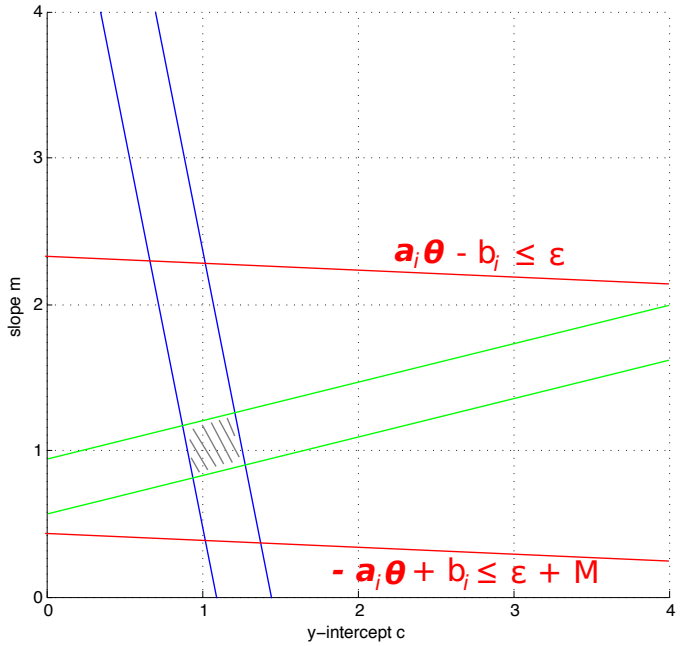


Figure 3.4: Artificially expanding a constraint to create a feasible region.

Hence with the addition of M, the problem becomes one of deciding to which constraint it should be applied. From Fig. 3.2 we can see that M can potentially be applied to three of the six linear inequality constraints to create a feasible region. Can M be applied to all three? Or could we apply M to all existing constraints? What would happen if the number of constraints increases? Applying M to all constraints is equivalent to increasing the threshold value ϵ. This results in a different problem altogether, where the solution obtained is not robust at all towards outliers. Whether M should be applied to a certain constraint really depends on where the optimal solution resides. This is actually a chicken-and-egg problem: in order to find which constraints to apply M, we need to know the optimal solution. However, in order to find the optimal solution, we need to know which constraints to apply M. Fortunately, the task of determining which constraint to apply M to can be resolved simply by solving the MILP in (3.3).

Another point of concern is the selection of a proper value for M. Firstly, the value of M has to be at least large enough to create a feasible region, otherwise the problem will simply remain infeasible. Furthermore, M also has to be large enough such that the resulting feasible region actually contains the optimal solution to the original maximum consensus problem.

Theoretically, any slack value that is big enough will suffice. However, the Big-M method is not favoured as its outcome depends on the selection of a value for M [29, 61]. Apart from the fact that the corresponding problem can remain infeasible or unsolvable with insufficient slack, giving too much slack also proves to be problematic because it can result in numerical issues. This is because an overly large M value dominates the calculations, e.g. it is possible to "partially" activate an overly large slack by setting the corresponding indicator variable to a minimal value that is very close to zero ($z_i \leq 1.0 \times 10^{-6}$) without violating the integrality constraints.

Furthermore, the value of M can potentially affect computation speed. Recall from Section 2.2.2 that solutions to MILP can computed by solving its LP relaxation, which in the case of (3.3) this is achieved by replacing $z_i \in \{0, 1\}$ in (3.3c) with $z_i \in [0, 1]$. The resulting LP is used as a bounding function for search algorithms such as branch-and-bound (BnB), which in turn yield new and further constrained LPs to solve. This solve-and-search process is repeated until an optimal solution is found - further information on this and how MILP can be solved are referenced in Section 2.2.2. Since the objective is to minimise (3.3a), we can observe that if M is very large, the objective value of the corresponding LP will be small. This gives a loose bounding function, which slows down search algorithms such as BnB. The problem tackled in this chapter is how to best optimise M.

3.3 M-bisection

There exists various suggestions and guidelines on how to select M [2, 11, 38, 61]. Say we have determined an initial large value for M, and we wish to reduce the slack of each constraint. This necessitates the use of individual slack values for each constraint in (3.3b), resulting in

$$\min_{\boldsymbol{\theta}, \mathbf{z}} \quad \sum_i z_i \tag{3.5a}$$

$$\text{subject to} \quad \mathbf{a}_i \boldsymbol{\theta} - b_i \leq \epsilon + z_i M_i \ , \tag{3.5b}$$

$$z_i \in \{0, 1\} \ , \tag{3.5c}$$

where the initial value of M_i is equivalent to M. Ideally, we want M to be big enough to ensure feasibility and correct operation of MILP, yet small enough to avoid giving too much slack. To that end, we propose a novel method called M-bisection to optimise M.

Suppose now we wish to reduce the slack of single, specific constraint k. We can reduce M_k by applying a multiplier $\alpha_k \in [0, 1]$, giving us

$$\mathbf{a}_k\boldsymbol{\theta} - b_k \leq \epsilon + \alpha_k M_k \ . \tag{3.6}$$

In this case, the value of the corresponding binary indicator variable z_k is set to 1. By applying α_k, we are essentially cutting off an interval from the expanded region, one that is constrained between $\mathbf{a}_k\boldsymbol{\theta} - b_k \geq \epsilon + \alpha_k M_k$ and $\mathbf{a}_k\boldsymbol{\theta} - b_k \leq \epsilon + M_k$. We illustrate this in Fig. 3.5, where the shaded area is the interval removed by applying α_k. Hence the problem becomes one of determining the value of α_k to safely reduce M_k - such that the resulting MILP (3.5) will still yield the same solution as the original MILP (3.3).

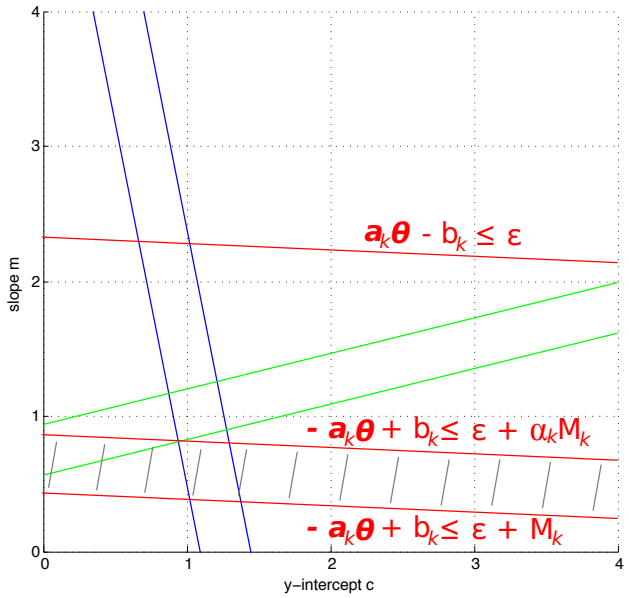


Figure 3.5: Cutting off an interval by applying $\alpha_k \in [0, 1]$.

The MILP is formulated by applying the concept in (3.6) to (3.5):

$$\min_{\boldsymbol{\theta},\mathbf{z}} \quad \sum_i z_i \tag{3.7a}$$

$$\text{subject to} \quad \mathbf{a}_i\boldsymbol{\theta} - b_i \leq \epsilon + z_i M_i \ , \tag{3.7b}$$

$$z_i \in \{0, 1\} \ , \tag{3.7c}$$

$$\mathbf{a}_k\boldsymbol{\theta} - b_k \geq \epsilon + \alpha_k M_k \ . \tag{3.7d}$$

The initial slack M_i allocated for all constraints is equivalent to M. Basically, (3.7) seeks to find the optimal solution to the original maximum consensus problem within the interval constrained between $\mathbf{a}_k\boldsymbol{\theta} - b_k \geq \epsilon + \alpha_k M_k$ and $\mathbf{a}_k\boldsymbol{\theta} - b_k \leq \epsilon + M_k$, the width of which is determined by M_k and α_k.

Assuming that the initial M is large enough, computing (3.7) will result in one of two possible outcomes. The first being infeasibility, meaning that there exists no solution in the aforementioned interval. From this we can conclude that constraint k has excess slack, and the interval resulting from the excess slack can be eliminated via reduction of the value of α_k. The second outcome is feasibility, in which the computation will produce a solution and an objective value to (3.7a). Using a bisection scheme on α_k in conjunction with (3.7) allows us to minimise the slack M_k allocated for constraint k. This can be done by utilising the previous objective value to adjust the bisection on α_k, either reducing or increasing the allocated slack until the objective value is minimised. Once an optimal value for α_k has been determined, the slack for constraint k is updated such that the updated slack M_k holds the value of α_k multiplied by the old M_k.

We can optimise the slack values for each constraint (and hence minimising the regions defined by each slack) by iterating through $k = \{1, 2, \ldots, 2N\}$ repeatedly until $\alpha_k = 1 \; \forall k = 1, 2, \ldots, 2N$, meaning that the regions defined by each slack can no longer be further reduced. However, this equates to solving the original maximum consensus problem in (3.3). Moreover, the computational overhead involved in iterating through all constraints repeatedly will cause this approach to run much slower compared to a single MILP computation of the original maximum consensus problem.

We remedy this by relaxing the integer constraints in (3.7) such that it becomes the following LP:

$$\begin{aligned}
\min_{\boldsymbol{\theta},\mathbf{z}} \quad & \sum_i z_i && \text{(3.8a)}\\
\text{subject to} \quad & \mathbf{a}_i\boldsymbol{\theta} - b_i \leq \epsilon + z_i M_i \;, && \text{(3.8b)}\\
& z_i \in [0, 1] \;, && \text{(3.8c)}\\
& \mathbf{a}_k\boldsymbol{\theta} - b_k \geq \epsilon + \alpha_k M_k \;. && \text{(3.8d)}
\end{aligned}$$

Let us define $\underline{l}_k$ as the optimal objective value to the LP in (3.8).

Lemma 1 *$\underline{l}_k$ serves as a lower bound to the outlier count of the original maximum consensus problem subject to the interval constrained by α_k.*

Proof The optimal objective value of (3.3) returns a minimal value to the outlier count to the original maximum consensus problem. (3.7) is a further constrained version of (3.3), whose optimal objective value returns a minimal to the outlier count subject to the interval constrained by α_k. It is a standard approach to calculate a lower bound to a MILP formulation via relaxation of its integer constraints, which turns the MILP into a LP [15]. Intuitively, the optimal objective value of the LP relaxation cannot exceed the optimal objective value of the original MILP formulation by the simple argument that the relaxation $z_i \in [0, 1]$ (a continuous range between 0 and 1, inclusive) is a superset of the binary constraints $z_i \in \{0, 1\}$. Since (3.8) is a LP relaxation of (3.7), we can conclude that its optimal objective value $\underline{l}_k$ serves as a lower bound to (3.7).

We can obtain an upper bound $\hat{u}$ on the outlier count of the original maximum consensus problem in (3.3). For this thesis, we designate RANSAC as our preferred approximation method, and thus

$$\hat{u} = |\mathcal{X}| - |\hat{\mathcal{I}}| \ , \tag{3.9}$$

where $\hat{\mathcal{I}}$ is a suboptimal consensus set attained from RANSAC. We will use $\hat{u}$ as the upper limit to the largest possible objective value we are willing to accept. Using (3.8), the value of α_k can be approximated by comparing the corresponding outlier lower bound $\underline{l}_k$ to a predetermined outlier upper bound $\hat{u}$ via repeated bisection on the interval between $\alpha_k \in [0, 1]$.

Lemma 2 *Given that M is big enough, M can be reduced if $\underline{l}_k > \hat{u}$, or if α_k causes (3.8) to be infeasible.*

Proof Having $\underline{l}_k > \hat{u}$ indicates that the region constrained between $\mathbf{a}_k\boldsymbol{\theta} - b_k \geq \epsilon + \alpha_k M_k$ and $\mathbf{a}_k\boldsymbol{\theta} - b_k \leq \epsilon + M_k$ will not contain a better solution than $\hat{u}$, which allows us to discard the region by reducing the value of the corresponding slack to $\alpha_k M_k$. Likewise, if α_k causes the resulting LP in (3.8) to be infeasible, we know that the corresponding region does not contain a feasible solution and hence can also be discarded.

We can quickly determine the value of each α_k by applying Lemma 2 with a bisection scheme, allowing us to reduce the region defined by the slack M_k. A single M-bisection run attempts to minimise all regions defined by the slack values of each constraint by iterating through all constraints

$k = \{1, 2, \ldots, 2N\}$. Note that a single M-bisection run is not guaranteed to reduce the slack values for all constraints, and hence it is possible for a slack M_i to hold the same value both before and after a M-bisection run. This can be remedied by running M-bisection repeatedly until none of the slack values can be further reduced.

3.4 Main Algorithm

Algorithm 1 M-bisection using LP

Require: Data $\mathcal{X}$ with N datums, inlier threshold ϵ, slack value M.

Execute RANSAC on $\mathcal{X}$ to obtain an upper bound $\hat{u}$.
Initialise individual slacks $M_i = M$ for all constraints derived from $\mathcal{X}$.
for $k = 1, 2, \ldots, 2N$ **do**
    Initialise variables
    $\alpha_l = 0$
    $\alpha_u = 1$
    $\alpha_{\text{prev}} = \alpha_u$
    $\alpha = 0.5 \times (\alpha_l + \alpha_u)$
    while $|\alpha - \alpha_{\text{prev}}| > 1 \times 10^{-6}$ **do**
        Solve LP in (3.8) with $\alpha_k \leftarrow \alpha$ to obtain a lower bound $\underline{l}_k$.
        if infeasible **or** $\underline{l}_k > \hat{u}$ **then**
            $\alpha_u \leftarrow \alpha$
        else
            $\alpha_l \leftarrow \alpha$
        end if
        $\alpha_{\text{prev}} \leftarrow \alpha$
        $\alpha \leftarrow 0.5 \times (\alpha_l + \alpha_u)$
    end while
    $M_k \leftarrow \alpha_u * M_k$
end for
return Adjusted slack values M_k for each constraint.

Algorithm 1 summarises our proposed method for a single M-bisection run. Since LPs are solvable in polynomial time, M-bisection is usually relatively efficient and will terminate much faster than running MILP on the original maximum consensus problem (3.3). Once it terminates, we can sub-

stitute M with the individually adjusted slack values M_i, and then perform a MILP computation to find the solution to the maximum consensus problem.

3.5 Applications in Computer Vision

Previously in section 2.1, we have discussed the linear regression residual (2.3), which produces equations *linear* to the unknown variables that can be formulated as a system of inhomogeneous linear equations $A\boldsymbol{\theta} \approx \mathbf{b}$. We have also discussed an alternate and more robust means of estimation using maximum consensus. In this section we will show how certain computer vision problems can be solved using the MILP formulation of maximum consensus.

Figure 3.6: Local feature points detected in a photograph.

We mentioned in chapter 1 that applications in computer vision consists of a pipeline of low to high-level tasks. Low level operations begin by identifying from an image the most basic points of interest, such as edges with sharp changes in image gradient, or corner regions with a high contrast. For example, to reliably match a sequence of images (such as frame sequences from a video, or two different photographs of the same scene), we need to find unique features local to each image that are invariant under various transformations. This includes both geometric invariance (translation, rotation, scaling) and photometric invariance (such as brightness, colour, and exposure). Those features or *keypoints* can be identified using an algorithm proposed by [31] known as the Scale Invariant Feature Transform (SIFT).

The SIFT keypoints in Fig. 3.6 are computed using the open source VLFeat library[1], which packs a feature detector implemented based on SIFT.

[1] http://www.vlfeat.org

The feature detector takes in a single precision grayscale image and returns a *descriptor* for each keypoint detected in the image, which is basically a succinct description of the corresponding keypoint. Local descriptors from one image can be matched to descriptors from other images using a feature matching algorithm (also available as an implementation of SIFT in the VLFeat library) to find similar regions. To achieve high matching accuracies, the SIFT feature descriptor has to be invariant to scaling, orientation (changes in viewpoint), distortion, and significant differences in illumination. SIFT achieves this by capturing and summarising the information taken from a region around the detected feature point.

Fig. 3.7 illustrates the set of matched descriptors between two photographs. The feature detection and matching algorithm is not perfect, as we can observe matches that are obviously wrong. The most apparent false positives are the ones originating from the far left of the left image and the ones ending in the far right of the right image. Such outliers can be removed using robust estimation methods based on maximum consensus, where $\mathcal{X}$ usually consists of monomials calculated from all pairs of putative correspondences, whereas $\boldsymbol{\theta}$ models the geometric relationship between the matched pairs.

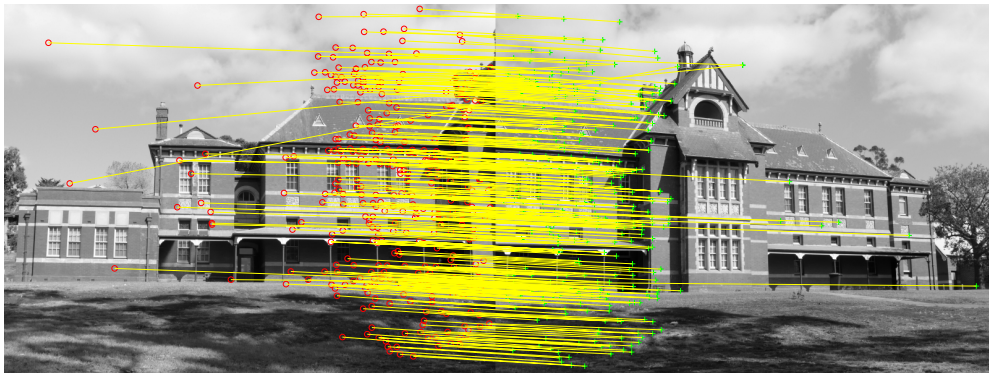


Figure 3.7: Putative pairwise correspondences between two photographs[1].

3.5.1 Epipolar Geometry

Say we have a set of keypoints matched to another set of keypoints $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, where $\mathbf{x_i} = (x_i, y_i)$ is the Euclidean coordinates of the i-th keypoint in the first image, and $\mathbf{x}'_i = (x'_i, y'_i)$ is the Euclidean coordinates of the corresponding keypoint match from the second image. Two photos of the same scene taken

[1]The correspondences can be used for image stitching, as illustrated in Fig. 1.7.

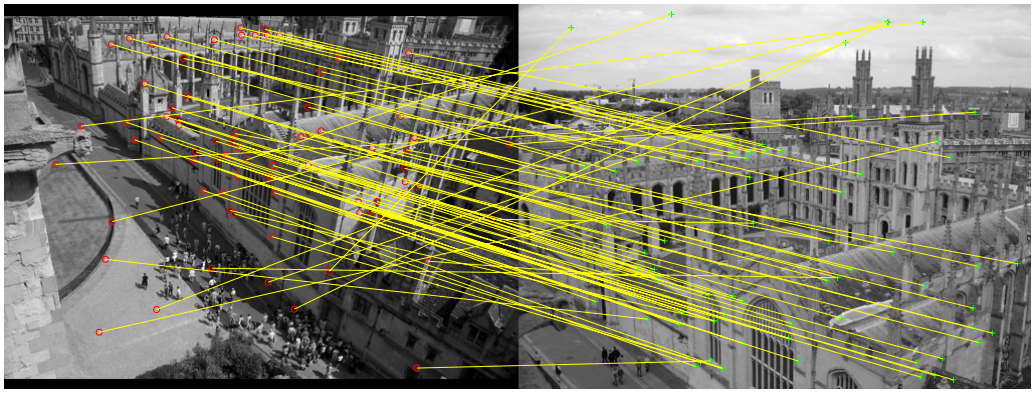


Figure 3.8: Pairwise correspondences for affine epipolar geometry[1].

from two distinct positions are related by a geometric relationship known as epipolar geometry. In the case where the cameras are affine, we can model the relationship using an *affine fundamental matrix* F_A

$$\tilde{\mathbf{x}}_i' F_A \tilde{\mathbf{x}}_i^T = \mathbf{0} \ , \tag{3.10}$$

where $\tilde{\mathbf{x}}_i$ is $\mathbf{x}_i$ in homogeneous coordinates. F_A has the form

$$F_A = \begin{bmatrix} 0 & 0 & \theta_1 \\ 0 & 0 & \theta_2 \\ \theta_3 & \theta_4 & \theta_5 \end{bmatrix} \ , \tag{3.11}$$

where θ_i indicates non-zero entries. F_A is a homogeneous matrix with five non-zero entries, giving it 4 degrees of freedom, and with a sufficient number of point matches $\mathbf{x}_i \leftrightarrow \mathbf{x}_i'$ we can derive the value of the non-zero entries in F_A. The matrix dimensions are consistent in equation (3.10) because $\mathbf{x}_i$ and $\mathbf{x}_i'$ are represented using vectors in the projective plane, where $\tilde{\mathbf{x}}_i = (x_i, y_i, 1)$. The coordinate of a point (x, y) in the $\mathbb{R}^2$ Euclidean plane can be represented in the $\mathbb{P}^2$ projective plane simply by appending a third coordinate with the value of 1 to the end to get $(x, y, 1)$. Such coordinates are called *homogeneous coordinates* because scaling is unimportant in projective geometry, and the coordinates $(x, y, 1)$ are equivalent to the coordinates $(\alpha x, \alpha y, \alpha)$ for any $\alpha \neq 0$. More on projective geometry and why it is crucial in computer vision can be found in [26, Chap. 2] and [4] (and the references therein). The derivation of F_A in equation (3.10) can be found in [26, Chap. 14].

[1]Grayscale photographs of All Souls College in Oxford, United Kingdom. Original images provided by http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/.

From (3.10) each point match in $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ gives rise to one linear equation

$$x'_i\theta_1 + y'_i\theta_2 + x_i\theta_3 + y_i\theta_4 + \theta_5 = 0 \ . \tag{3.12}$$

The set of all point matches for $i = 1, 2, ..., N$ results in an overdetermined system of *homogeneous* linear equations of the form

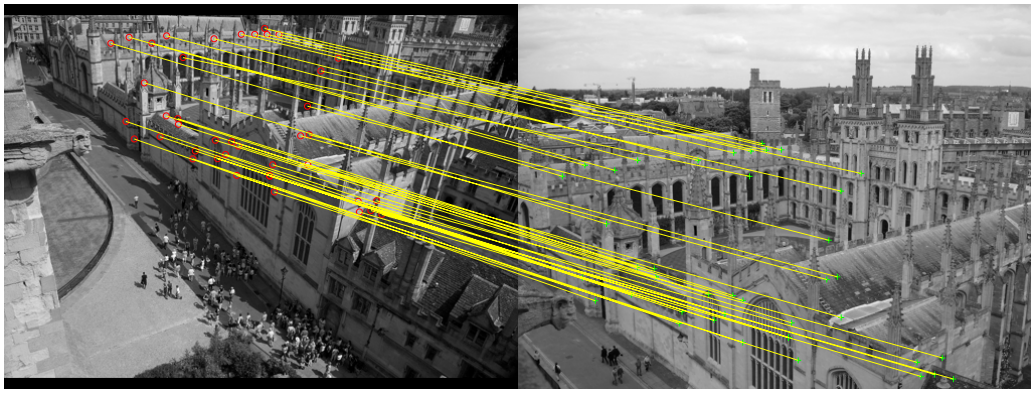
$$A\boldsymbol{\theta} = \mathbf{0} \ . \tag{3.13}$$

This is a special case of (2.2), where the constant terms are all zeroes, and the residual is the algebraic error $\mathbf{r}(\boldsymbol{\theta}) = A\boldsymbol{\theta}$. Observe that there always exists a trivial solution to (3.13), which is the zero solution $\boldsymbol{\theta} = \mathbf{0}$. However, a model consisting of all zeroes provides no useful information, and further constraints should be imposed to avoid a zero solution. Generally the restriction is imposed on the norm of $\boldsymbol{\theta}$, e.g. requiring its value to be $\|\boldsymbol{\theta}\| = 1$. Since $\boldsymbol{\theta}$ can only be determined up to a non-zero scale factor, the value chosen for $\|\boldsymbol{\theta}\|$ matters little. We note again that for an overdetermined system of homogeneous linear equations, there will be no exact solutions apart from the (usually undesired) zero solution. Instead, as mentioned in section 1.1, we seek a solution that minimises some suitable cost function. In the same vein, given that there is no exact solution to (3.13), we elect to find a solution that minimises $\|A\boldsymbol{\theta}\|$ subject to $\|\boldsymbol{\theta}\| = 1$. With linear least squares regression the solution is the column of V corresponding to the smallest singular value of A in a SVD decomposition, where $A = UDV^T$ [26, Sec. 4.1.1]. This approach is known as the Direct Linear Transformation (DLT) algorithm.

To achieve consensus maximisation, the problem can also be converted into inhomogeneous equations (2.2) and solved using MILP by linearising and dehomogenising equation (3.13). This is achieved by imposing a condition on the j-th entry of $\boldsymbol{\theta}$ such that $\theta_j = 1$, which is justified because $\boldsymbol{\theta}$ can only be determined up to scale; and the scale can be chosen such that $\theta_j = 1$ [26, Sec. 4.1.2]. E.g. if we chose $\theta_4 = 1$, then equation (3.12) becomes

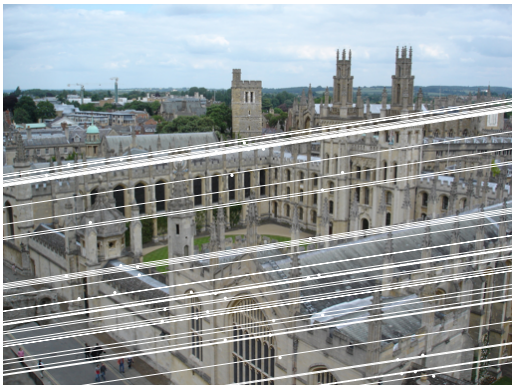
$$x'_i\theta_1 + y'_i\theta_2 + x_i\theta_3 + \theta_5 = -y_i \ ,$$

which gives a system of equations of the form (2.2), where $\mathbf{a}_i = (x'_i, y'_i, x_i, 1)$, $b_i = -y_i$, the parameters we wish to solve for are $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3, \theta_5)^T$ with $\boldsymbol{\theta} \in \mathbb{R}^4$, and the error becomes the linear regression residual in (2.3). Hence $\mathcal{X} = \{\mathbf{a}_i, b_i\}_{i=1}^N$ contains the coordinates of the keypoint matches $\mathbf{x} \leftrightarrow \mathbf{x}'$, and solving for $\boldsymbol{\theta}$ gives us the non-zero entries of F_A, with $\theta_4 = 1$.



(a) The inlying correspondences to the estimated F_A.

(b) Epipolar lines in the first view.



(c) Epipolar lines in the second view.

Figure 3.9: Affine epipolar geometry estimated using MILP.

The results of estimating F_A using MILP is shown in Fig. 3.9. (a) Illustrates inliers whose errors fall within a threshold of 2 pixels ($\epsilon = 2$). (b) and (c) illustrates the epipolar lines derived using F_A and the inliers from (a). We can see from the illustration that all affine epipolar lines are parallel. More information on affine epipolar geometry and its uses including image retrieval and motion recovery can be found in [26, Chap. 14] and [3, 45].

3.6 Experimental Results

We test our conjecture on M-bisection using linear data that is synthetically generated. In particular, we compared the results of exactly solving (3.3) on input data $\mathcal{X}$ (we call this EXACT), against the results of running

M-bisection (Algorithm 1) repeatedly to reduce the individual slack values M_i, then solving (3.3) exactly after replacing M for each constraint with the adjusted slack values M_i (we call this M-Bisection+EXACT). The industry grade Gurobi Optimiser is applied to exactly solve both the MILP formulation for maximum consensus (3.3) and the LP formulation for M-bisection (3.8).

As mentioned previously in section 3.2, if the slack value M is too large, it will cause numerical issues with Big-M (3.3). For the purpose of our experiment, we expect M-bisection+EXACT to conform to the integrality constraints and produce an integer result. If there are no numerical issues, both EXACT and M-bisection+EXACT are expected to return the same objective value. Theoretically, we would also expect the runtime of EXACT following M-bisection to be shorter than the runtime of EXACT alone, since M-bisection essentially reduces the search space and thereby the amount of computation required for EXACT.

The experiments were applied on synthetically generated data and affine epipolar geometry estimation discussed in section 3.5.1. The experiments were carried out on a standard 1.40 GHz machine with 8 GB of RAM. A slack value of $M = 1000$ was used in all MILP instances.

3.6.1 Synthetic Data

Synthetic data in the form $\mathcal{X} = \{\mathbf{a}_i, b_i\}_{i=1}^N$ (2.1) was produced for testing GORE. The ground truth $\boldsymbol{\theta}$ was generated uniformly in $[-1, 1]^3$. Each $\mathbf{a}_i$ was drawn uniformly from the range $[-50, 50]^{1\times 3}$, and b_i was obtained as $b_i = \mathbf{a}_i\boldsymbol{\theta}$, which essentially produces a linear plane fitting problem in 3D. This justifies the linear regression residual (2.3)

$$r_i(\boldsymbol{\theta}) = |\mathbf{a}_i\boldsymbol{\theta} - b_i| \ , \tag{3.14}$$

which is applicable to the MILP formulation of maximum consensus. To simulate outliers, around 55% of the *dependent measurements* $\{b_i\}_{i=1}^N$ (i.e. the "*response*", see section 2.1) were perturbed with independent and identically distributed (i.i.d.) uniform noise in the range of $[-50, 50]$, while the rest were perturbed with i.i.d. normal inlier noise with $\sigma = 0.01$. Using a threshold of $\epsilon = 0.02$, this produces datasets with an outlier rate between $40\% \sim 60\%$. For this experiment, all sets of synthetic data are generated using $N = 100$.

Results are shown in table 3.1. For all instances tested, the results of EXACT and M-bisection+EXACT remain consistent with each other. The

	Dataset 1		Dataset 2		Dataset 3	
Methods	$\lvert\mathcal{I}\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	time(s)
EXACT	50	270.94	50	330.95	45	220.39
M-bisection		63.76		82.36		97.81
M-bisection+EXACT	50	166.59	50	254.49	45	214.87
	Dataset 4		Dataset 5		Dataset 6	
Methods	$\lvert\mathcal{I}\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	time(s)
EXACT	55	987.34	50	164.37	47	174.16
M-bisection		48.09		47.50		66.22
M-bisection+EXACT	55	979.02	50	182.54	47	294.66

Table 3.1: Results from 6 different sets of synthetic data. $|\mathcal{I}|$ = size of optimised consensus set.

computational gain achieved by M-Bisection+EXACT on input data $\mathcal{X}$ can be expressed as the ratio

$$1 - \frac{\text{runtime of M-bisection+EXACT}}{\text{runtime of EXACT}} \ . \tag{3.15}$$

From the results we can see that datasets 1 and 2 achieved a positive gain of 0.39 and 0.23. On the other hand, datasets 3 and 4 experience almost no improvement in runtime with gains of 0.03 and 0.01, whereas datasets 5 and 6 experience a slowdown with negative gains of -0.11 and -0.69. We can see that only half of the instances tested achived a reduction in runtime.

3.6.2 Epipolar Geometry Estimation

We tested GORE on images used in previous works for affine epipolar geometry estimation [3], including Kapel and Valbonne Church from the multi-view reconstruction dataset[1], Notre Dame from the Paris dataset[2], and All Souls from the Oxford Buildings dataset[3]. All images mentioned are provided by the Oxford Visual Geometry Group.

The results are displayed in table 3.2. For all instances tested, the results of EXACT and M-bisection+EXACT remain consistent. Notre Dame and All Souls has the respective positive gains of 0.10 and 0.53, whereas Kapel and Valbonne has the respective negative gains of -0.39 and -0.99. The rest

[1] http://www.robots.ox.ac.uk/~vgg/data/data-mview.html
[2] http://www.robots.ox.ac.uk/~vgg/data/parisbuildings/
[3] http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/

	Kapel $N = 47$, $\epsilon = 2$		Notre Dame $N = 84$, $\epsilon = 2$	
Methods	$\lvert\mathcal{I}\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	time(s)
EXACT	21	78.03	50	350.92
M-Bisection		5.75		13.62
M-Bisection+EXACT	21	108.46	50	319.01
	All Souls $N = 70$, $\epsilon = 2$		Valbonne $N = 100$, $\epsilon = 2$	
Methods	$\lvert\mathcal{I}\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	time(s)
EXACT	38	186.14	63	96.35
M-Bisection		10.29		17.62
M-Bisection+EXACT	38	87.19	63	191.37

Table 3.2: Results of affine epipolar geometry estimation. $|\mathcal{I}|$ = size of optimised consensus set.

of the instances experience a significant slowdown in runtime for the subsequent EXACT computation after preprocessing with M-bisection. Again, only about half of the instances tested achieved a reduction in runtime.

3.7 Summary

We proposed M-bisection as a way to reduce M to deal with the issues inherent in the Big-M method, which makes use of a bisection scheme to reduce M. Theoretically, a reduction of M should result in an overall speedup in the computation of MILP.

Experiments however showed that the improvements gained from M-bisection are unstable. Preprocessing with M-bisection might result in a speedup, but it may as well result in an overall slowdown of the computation. This is due to the fact that the value of M is not the only deciding factor when it comes to computation efficiency. Changing the values of M will likely affect the various decisions made by the internal heuristics of the MILP solver, which varies between the implementation of different solvers, and hence affecting the speed.

The stability and usefulness of M-bisection leaves much to be desired. Nevertheless, the idea of minimising slack by means of bisection holds merit, and a more in-depth investigation into the implementations of a MILP solver may allow for further improvements to M-bisection.

Chapter 4

Guaranteed Outlier Removal

In section 2.2, we discussed the problem of consensus maximisation, how it is useful in computer vision and how it can be achieved. However, it is computationally demanding to solve the problem exactly, especially since the effort required increases rapidly with the problem size. In this chapter, we present a Guaranteed Outlier REmoval (GORE) technique based on MILP as a way to reduce the runtime of exact algorithms. GORE can also be employed as a means to improve the success rate of randomised methods by reducing the ratio of outlying measurements.

Specifically, before performing global optimisation, we conduct GORE as a preprocessing technique for removing data that are provably outliers. This results in a dataset with a lower outlier rate, which can potentially speed up subsequent globally optimal algorithms while preserving the same globally optimal result. A potential speedup of 80% can be achieved on common computer vision problems such as the ones discussed in section 3.5.

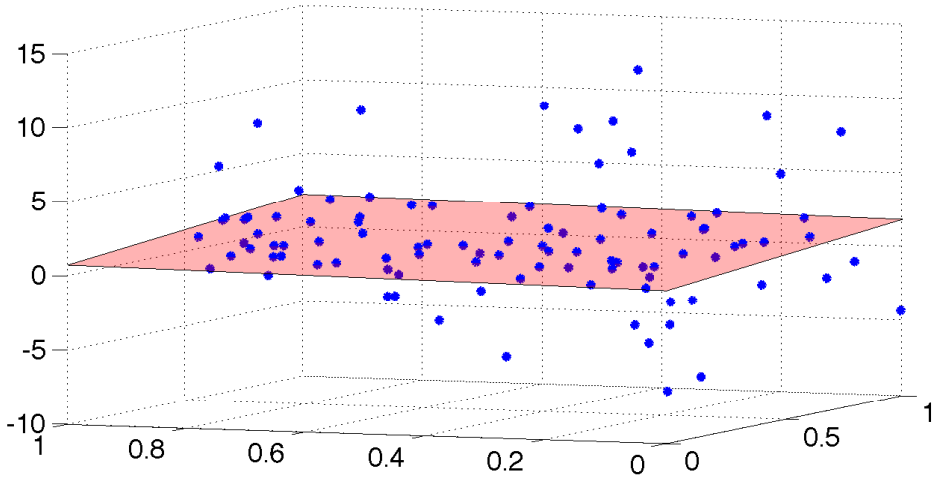
4.1 Overview of GORE

Given a dataset $\mathcal{X} = \{\mathbf{a}_i, b_i\}_{i=1}^N$ containing N measurements, GORE reduces $\mathcal{X}$ to a subset $\mathcal{X}'$ under the condition

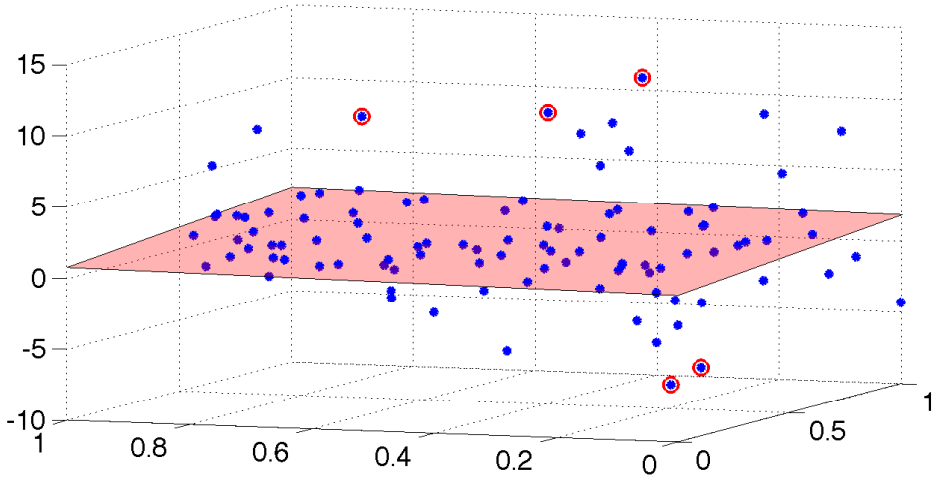
$$\mathcal{I}^* \subseteq \mathcal{X}' \subseteq \mathcal{X} \ , \tag{4.1}$$

where $\mathcal{I}^*$ is the maximum consensus set as defined in section 2.2. As its name suggests, any data removed by GORE are guaranteed to be true outliers. While the resulting $\mathcal{X}'$ may not be completely outlier-free, solving the

maximum consensus problem (2.17) on $\mathcal{X}'$ will generally take less time while preserving the same result as the original dataset $\mathcal{X}$.



(a) $|\mathcal{X}| = 100$, time to solution $= 423.07s$ (global optimization)



(b) $|\mathcal{X}'| = 95$, time to sol. $= 50s$ (GORE) $+ 32.5s$ (glob. opt.) $= 82.5s$

Figure 4.1: (a) Solving (2.17) exactly on $\mathcal{X}$ with $N = 100$ to robustly fit an affine plane ($d = 3$) using the Gurobi solver took $423.07s$. (b) Removing 5 true outliers (points circled in red) using the proposed GORE algorithm ($50s$) and subjecting the remaining data $\mathcal{X}'$ to Gurobi returns the same maximum consensus set in $32.5s$, representing a reduction of 80% in the total runtime.

Of course, the duration spent conducting GORE should be included in the overall runtime. Intuitively, the effort required to identify and remove *all* outliers is unlikely to be less than that of solving the original maximum consensus problem. Therefore, the problem becomes *how many outliers should be removed for the outlier reduction scheme to pay off?* Experiments show that it is relatively cheap to remove a small portion of the most egregious outliers using GORE (i.e. usually the ones with the largest residuals), and the removal of those outliers are sufficient to produce a substantial speedup of subsequent exact methods. Fig. 4.1 demonstrates that removing a mere 5% of data from $\mathcal{X}$ using GORE decreases the overall runtime required to find the same globally optimal result by 80%.

Our version of GORE is an extension of two recent techniques [6, 51], respectively specialised for estimating 3D rotations and 2D rigid transforms, both of which are problems with 3 degrees of freedom (DOF). Both works exploit the underlying geometry of their respective models to quickly prune away outlying feature matches, which unfortunately means that their methods cannot be applied to models other than their respective target problems. The GORE method proposed in this chapter is generalised to a larger range of problems, which will be formulated based on the linear regression residual. However, the formulation can easily be extended to geometric (quasiconvex) residuals as discussed in section 4.4.

4.2 MILP Formulation for GORE

Similar to M-bisection (see Chap. 3), the flexibility of our proposed GORE method is founded on the basis of a MILP formulation, allowing us to "abstract away" the calculation of upper and lower bounds required for determining outliers. The MILP formulation allows us to utilise efficient off-the-shelf linear optimisation solvers (e.g. Gurobi) to handle up to 6DOF models.

With the linear regression residual, the MILP formulation of the maximum consensus problem (3.3) can be written as

$$\min_{\boldsymbol{\theta},\mathbf{z}} \quad \sum_i z_i \tag{4.2a}$$

$$\text{subject to} \quad |\mathbf{a}_i\boldsymbol{\theta} - b_i| \leq \epsilon + z_i M \ , \tag{4.2b}$$

$$z_i \in \{0,1\} \ , \tag{4.2c}$$

where z_i are indicator variables, and M is a slack value that is large enough. Intuitively, $z_i = 1$ amounts to identifying $\{\mathbf{a}_i, b_i\}$ as an outlier. Hence given the solution $\mathbf{z}^*$ to (2.17), the maximum consensus set is defined as

$$\mathcal{I}^* = \{\mathbf{a}_i, b_i \mid z_i^* = 0\} \ . \tag{4.3}$$

GORE begins by rewriting (4.2) as the following "nested" problem

$$\underset{k=1,\ldots,N}{\text{minimise}} \ \beta_k, \tag{4.4}$$

where β_k is defined as the optimal objective value of the following subproblem

$$\begin{aligned}
\min_{\boldsymbol{\theta},\mathbf{z}} \quad & \sum_{i \neq k} z_i && \text{(4.5a)} \\
\text{subject to} \quad & |\mathbf{a}_i\boldsymbol{\theta} - b_i| \leq \epsilon + z_i M \ , && \text{(4.5b)} \\
& z_i \in \{0, 1\} \ , && \text{(4.5c)} \\
& |\mathbf{a}_k\boldsymbol{\theta} - b_k| \leq \epsilon \ . && \text{(4.5d)}
\end{aligned}$$

The formulation in (4.5) seeks to remove as little data as possible to achieve a feasible subset, under the strict condition that a datum $\{\mathbf{a}_k, b_k\}$ is included (fixed) as an inlier in said subset. Note that the formulation in (4.5) remains a MILP and not a LP, and that the problem in (4.4) is no easier to solve than the original maximum consensus problem in (2.17). Instead, the utility of (4.4) derives from how a bound on (4.5) allows us to identify datums as outliers to the maximum consensus set $\mathcal{I}^*$.

Let $\hat{\boldsymbol{\theta}}$ and its corresponding indicator variables $\hat{\mathbf{z}}$ indicate a suboptimal approximation to (4.2), and let

$$\hat{u} = \|\hat{\mathbf{z}}\|_1 \geq \|\mathbf{z}^*\|_1 \tag{4.6}$$

be its corresponding objective value. Let $\underline{\alpha}_k$ be a lower bound to (4.5), i.e.

$$\underline{\alpha}_k \leq \beta_k \ . \tag{4.7}$$

Given $\hat{u}_k$ and $\underline{\alpha}_k$, a test can be performed according to the following lemma to decide whether $\{\mathbf{a}_k, b_k\}$ is an outlier to (4.2).

Lemma 3 *If $\underline{\alpha}_k > \hat{u}$, then $\{\mathbf{a}_k, b_k\}$ is an outlier to $\mathcal{I}^*$.*

Proof The lemma can be established via contradiction. The k-th datum $\{\mathbf{a}_k, b_k\}$ is an inlier if and only if

$$\beta_k = \|\mathbf{z}^*\|_1 \leq \hat{u} \ . \tag{4.8}$$

In other words, if $\{\mathbf{a}_k, b_k\}$ is an inlier, insisting it to be an inlier in (4.5) does not change the fact that removing $\|\mathbf{z}^*\|_1$ datums is sufficient to achieve consensus. However, if we are given that $\underline{\alpha}_k > \hat{u}$, then from (4.7)

$$\hat{u} < \underline{\alpha}_k \leq \beta_k \ , \tag{4.9}$$

meaning that the necessary inlier condition in (4.8) cannot hold. Thus $\{\mathbf{a}_k, b_k\}$ *must* be an outlier to $\mathcal{I}^*$.

The following lemma shows that there are cases where for a datum $\{\mathbf{a}_k, b_k\}$, the test above will always fail to affirm it as an outlier.

Lemma 4 *If $\hat{z}_k = 0$, then $\underline{\alpha}_k \leq \hat{u}$.*

Proof If $\hat{z}_k = 0$ (i.e. an inlier to the suboptimal approximation $\hat{\boldsymbol{\theta}}$), then $(\hat{\boldsymbol{\theta}}, \hat{\mathbf{z}})$ is also suboptimal in regards to β_k in (4.5). Thus if $\hat{z}_k = 0$,

$$\hat{u} \geq \beta_k \geq \underline{\alpha}_k \ , \tag{4.10}$$

and the condition in Lemma 3 will never be met.

GORE applies Lemma 3 iteratively to all datums to achieve outlier removal. Critical to the operation of GORE is the calculation of the bounds $\hat{u}$ and α_k, in which the former can be obtained from the various approximation techniques. For this thesis, we designate RANSAC as our preferred approximation method, and thus

$$\hat{u} = |\mathcal{X}| - |\hat{\mathcal{I}}| \ , \tag{4.11}$$

where $\hat{\mathcal{I}}$ is a suboptimal consensus set attained from RANSAC. The main challenge of GORE lies in computing a tight lower bound $\underline{\alpha}_k$ for each datum, and we will describe our method for calculating $\underline{\alpha}_k$ in the following section.

Lemma 4 establishes that there are certain datums in which the test in Lemma 3 are ineffective. Hence our main GORE algorithm will prioritise tests for data with the largest errors with respect to the suboptimal solution $\hat{\boldsymbol{\theta}}$, i.e. the largest residual values corresponding to those with $\hat{z}_i = 1$.

4.2.1 Lower Bound Calculation

As mentioned in Lemma 1, the standard approach for lower bounding MILPs is via a LP relaxation [15]. In the context of the MILP formulation for GORE in (4.5), $\underline{\alpha}_k$ is obtained as the optimal value of the following LP

$$\min_{\boldsymbol{\theta},\mathbf{z}} \quad \sum_{i\neq k} z_i \tag{4.12a}$$

$$\text{subject to} \quad |\mathbf{a}_i\boldsymbol{\theta} - b_i| \leq \epsilon + z_i M \ , \tag{4.12b}$$

$$z_i \in [0,1] \ , \quad \text{/*continuous*/} \tag{4.12c}$$

$$|\mathbf{a}_k\boldsymbol{\theta} - b_k| \leq \epsilon \ , \tag{4.12d}$$

where the binary constraints in (4.5c) are relaxed to become continuous. $\underline{\alpha}_k$ cannot exceed β_k by the simple argument that the range $[0,1]^{N-1}$ is a superset of $\{0,1\}^{N-1}$.

However, the lower bound obtained solely via (4.12) tends to be loose. Observe that since the value of the slack M is very large, each continuous z_i in (4.12) needs only a small value to provide sufficient slack to satisfy its corresponding constraint. As a result the optimised $\mathbf{z}$ tends to be small and fractional, leading to a large difference in value between $\underline{\alpha}_k$ and β_k.

To obtain a more useful lower bound, we leverage on existing algorithms based on Branch-and-Bound (BnB) for solving MILPs [15]. In the context of solving (4.5), BnB maintains a pair of lower and upper bound values $\underline{\alpha}_k$ and $\hat{\gamma}_k$ over time, where

$$\underline{\alpha}_k \leq \beta_k \leq \hat{\gamma}_k \ . \tag{4.13}$$

The method of BnB progressively raises the value of the lower bound $\underline{\alpha}_k$ by solving the corresponding LP relaxation (4.12) on recursive subdivisions of the parameter space. If an exact solution to the original MILP (4.5) is desired, BnB is executed until $\underline{\alpha}_k = \hat{\gamma}_k$, which determines the exact value of β_k. For the purpose of GORE, however, BnB can simply be executed until one of the following two conditions is satisfied

$$\underline{\alpha}_k > \hat{u} \text{ (Condition 1)} \quad \text{or} \quad \hat{\gamma}_k \leq \hat{u} \text{ (Condition 2)} \ , \tag{4.14}$$

or until a pre-determined time budget c is exhausted. Achieving Condition 1 implies that $\{\mathbf{a}_k, b_k\}$ is an outlier, whereas achieving Condition 2 indicates that Condition 1 will never be met. Satisfying Condition 2 also indicates that a better solution has been identified (one that involves identifying $\{\mathbf{a}_k, b_k\}$ as an inlier) has been discovered, and thus $\hat{u}$ should be updated accordingly.

4.3 Main Algorithm

Algorithm 2 MILP-based GORE

Require: Data $\mathcal{X} = \{\mathbf{a}_i, b_i\}_{i=1}^N$, inlier threshold ϵ, number of rejection tests T, maximum duration per test c.

1: Execute RANSAC to obtain an upper bound $\hat{u}$ (4.11).
2: Order $\mathcal{X}$ increasingly based on RANSAC residuals.
3: **for** $k = N, N-1, \ldots, N-T+1$ **do**
4: Run BnB to solve (4.5) on $\mathcal{X}$ until one of the following is satisfied:

- $\underline{\alpha}_k > \hat{u}$ (Condition 1);
- $\hat{\gamma}_k \leq \hat{u}$ (Condition 2);
- c seconds have elapsed.

5: **if** Condition 1 was satisfied **then**
6: $\mathcal{X} \leftarrow \mathcal{X} \setminus \{\mathbf{a}_k, b_k\}$
7: **end if**
8: **if** Condition 2 was satisfied **then**
9: $\hat{u} \leftarrow \hat{\gamma}_k$
10: **end if**
11: **end for**
12: **return** Reduced data $\mathcal{X}$.

Algorithm 2 summarises our method for GORE. Again, RANSAC is used to obtain the upper bound $\hat{u}$, and to re-order $\mathcal{X}$ such that datums more likely to be outliers are first tested for removal. In our experiments, we apply the Gurobi Optimiser[1] to derive the lower bound $\underline{\alpha}_k$. The parameters crucial to the success of GORE are T and c, where the former is the number of datums we attempt to reject, and the latter is the maximum allowable duration devoted to rejecting a particular datum. The total runtime of GORE is thus $T \times c$. As we will show in a later section, for many real-life scenarios setting T and c to small values (e.g. $T \approx 0.1N$, $c \in [5\text{s}, 15\text{s}]$) is generally sufficient to reduce $\mathcal{X}$ to a subset $\mathcal{X}'$ that significantly speeds up global optimisations.

[1] http://www.gurobi.com/

4.4 GORE with Quasiconvex Residuals

Thus far, the problems we have discussed are all linear in form and can be handled using the linear regression residual (2.3) which is strictly convex. However, many applications in computer vision result in geometric residuals, meaning that those problems can not be solved directly using (3.3). In this section, we will show how the MILP formulation of maximum consensus can be generalised to solve a particular class of geometric residuals that are quasiconvex, which is involved in many common computer vision applications including triangulation, homography estimation and camera resectioning [27].

Quasiconvex residuals are error functions of the form

$$r_i(\boldsymbol{\theta}) = \frac{\|A_i\boldsymbol{\theta} + \mathbf{b}_i\|_p}{\mathbf{c}_i^T\boldsymbol{\theta} + d_i} \quad \text{with} \quad \mathbf{c}_i^T\boldsymbol{\theta} + d_i \geq 0 \ , \tag{4.15}$$

where $\boldsymbol{\theta} \in \mathbb{R}^n$ is a column vector consisting of the unknown variables, $A_i \in \mathbb{R}^{2\times n}$ is a $2 \times n$ matrix, $\mathbf{b}_i \in \mathbb{R}^2$ and $\mathbf{c}_i \in \mathbb{R}^n$ are column vectors, d_i is a scalar constant, and the dataset $\mathcal{X} = \{A_i, \mathbf{b}_i, \mathbf{c}_i, d_i\}_{i=1}^N$. In this case, $\mathcal{X}$ would contain additional information in tandem to the coordinates of keypoint matches, e.g. camera matrices. Residual (4.15) is quasiconvex for all L_p-norms with $p \geq 1$, where the general form of the L_p-norm is

$$\|\mathbf{u}\|_p = (|u_1|^p + |u_2|^p + \ldots + |u_n|^p)^{\frac{1}{p}} \quad \text{for } \mathbf{u} \in \mathbb{R}^n \ . \tag{4.16}$$

In the context of maximum consensus, applying a threshold ϵ to (4.15) yields

$$\|A_i\boldsymbol{\theta} + \mathbf{b}_i\|_p \leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) \ , \tag{4.17}$$

and thus the formulation of maximum consensus in (3.3) becomes

$$\min_{\boldsymbol{\theta},\mathbf{z}} \quad \sum_i z_i \tag{4.18a}$$

$$\text{subject to} \quad \|A_i\boldsymbol{\theta} + \mathbf{b}_i\|_p \leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) + z_i M \ , \tag{4.18b}$$

$$z_i \in \{0, 1\} \ . \tag{4.18c}$$

Historically, quasiconvex residuals have been investigated using the L_2-norm, where $p = 2$ [24, 27]. The condition $\mathbf{c}_i^T\boldsymbol{\theta} + d_i > 0$ can thus be dropped since it is implied by (4.17) for any $\epsilon \geq 0$ [27, Sec. 4.2]. This is because $\|A_i\boldsymbol{\theta} + \mathbf{b}_i\|_2$

will always be ≥ 0, and inequality (4.17) will not hold if $\mathbf{c}_i^T\boldsymbol{\theta} + d_i < 0$. However, the usage of $p = 2$ turns (4.18) into a problem of the form

$$\begin{aligned} \min_{\mathbf{u}} \quad & f(\mathbf{u}) \\ \text{subject to} \quad & \|A_i\mathbf{u} + \mathbf{b}_i\|_2 \leq \mathbf{c}_i^T\mathbf{u} + d_i \ , \end{aligned} \tag{4.19}$$

where $f(\mathbf{u})$ is a linear objective function. Convex optimisation problems of the form (4.19) is called Second-Order Cone Programming (SOCP), and constraint (4.18b) becomes a nonlinear second-order cone constraint under the L_2-norm, which is not applicable to MILP.

Of course, the L_2-norm is not the only possible norm we can use. Other commonly used norms include the L_1-norm and the L_∞-norm, which will both yield linear constraints that can be applied in MILP. Let

$$A_i = \begin{bmatrix} \mathbf{a}_{i,1} \\ \mathbf{a}_{i,2} \end{bmatrix} \quad \text{and} \quad \mathbf{b}_i = \begin{bmatrix} b_{i,1} \\ b_{i,2} \end{bmatrix} \ ,$$

where $\mathbf{a}_{i,j}$ is a row vector corresponding to the j-th row of A_i, and $b_{i,j}$ is a scalar value corresponding to the j-th entry of $\mathbf{b}_i$. The L_1-norm is the sum of all absolute values in a vector, and hence applying $p = 1$ to (4.18b) gives

$$|\mathbf{a}_{i,1}\boldsymbol{\theta} + b_{i,1}| + |\mathbf{a}_{i,2}\boldsymbol{\theta} + b_{i,2}| \leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) + z_i M \ .$$

By recursively applying the rule of absolute values, we find that the above is equivalent to the four linear constraints

$$\begin{aligned} (\mathbf{a}_{i,1} + \mathbf{a}_{i,2})\boldsymbol{\theta} + b_{i,1} + b_{i,2} &\leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) + z_i M \ , \\ (\mathbf{a}_{i,1} - \mathbf{a}_{i,2})\boldsymbol{\theta} + b_{i,1} - b_{i,2} &\leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) + z_i M \ , \\ -(\mathbf{a}_{i,1} - \mathbf{a}_{i,2})\boldsymbol{\theta} - b_{i,1} + b_{i,2} &\leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) + z_i M \ , \\ -(\mathbf{a}_{i,1} + \mathbf{a}_{i,2})\boldsymbol{\theta} - b_{i,1} - b_{i,2} &\leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) + z_i M \ . \end{aligned}$$

Note how a single indicator variable z_i "chains" the four constraints together to the same datum $\{A_i, \mathbf{b}_i, \mathbf{c}_i, d_i\}$.

Likewise, the L_∞-norm is the vector entry with the largest absolute value, and hence applying $p = \infty$ to (4.18b) gives

$$\max\{|\mathbf{a}_{i,1}\boldsymbol{\theta} + b_{i,1}|, \ |\mathbf{a}_{i,2}\boldsymbol{\theta} + b_{i,2}|\} \leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) + z_i M \ ,$$

which is equivalent to imposing the following two constraints simultaneously

$$|\mathbf{a}_{i,1}\boldsymbol{\theta} + b_{i,1}| \leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) + z_i M \ ,$$
$$|\mathbf{a}_{i,2}\boldsymbol{\theta} + b_{i,2}| \leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) + z_i M \ .$$

We again apply the rule of absolute values to yield the following four linear constraints

$$\begin{aligned}
\mathbf{a}_{i,1}\boldsymbol{\theta} + b_{i,1} &\leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) + z_i M \ , \\
-\mathbf{a}_{i,1}\boldsymbol{\theta} - b_{i,1} &\leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) + z_i M \ , \\
\mathbf{a}_{i,2}\boldsymbol{\theta} + b_{i,2} &\leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) + z_i M \ , \\
-\mathbf{a}_{i,2}\boldsymbol{\theta} - b_{i,2} &\leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) + z_i M \ .
\end{aligned}$$

Note again how the variable z_i connects all four constraints to the i-th datum. And hence we have a method of solving the quasiconvex residual (4.15) as a MILP formulation of maximum consensus (3.3) with four linear constraints instead of two using the linear regression residual.

4.4.1 MILP Formulation

For completeness, the subproblem analogous to (4.5) is

$$\begin{aligned}
\min_{\boldsymbol{\theta},\mathbf{z}} \quad & \sum_{i \neq k} z_i && \text{(4.20a)} \\
\text{subject to} \quad & \|A_i\boldsymbol{\theta} + \mathbf{b}_i\|_p \leq \epsilon(\mathbf{c}_i^T\boldsymbol{\theta} + d_i) + z_i M \ , && \text{(4.20b)} \\
& z_i \in \{0, 1\} \ , && \text{(4.20c)} \\
& \|A_k\boldsymbol{\theta} + \mathbf{b}_k\|_p \leq \epsilon(\mathbf{c}_k^T\boldsymbol{\theta} + d_k) \ . && \text{(4.20d)}
\end{aligned}$$

By applying either the L_1-norm or the L_∞-norm in (4.20b) and (4.20d) and following the derivations discussed in section 4.4 to convert them to linear inequalities, a lower bound $\underline{\alpha}_k$ for (4.20) can similarly be obtained using MILP solvers. Algorithm 2 can thusly be applied as a preprocessor to conduct GORE for quasiconvex problems.

4.5 Applications in Computer Vision

4.5.1 Triangulation

The process of determining a point in 3D given its projection onto two or more 2D images is known as triangulation. Say we have set of N images of similar scenes (Fig. 4.2) photographed in different angles and positions using cameras of known calibration and pose.

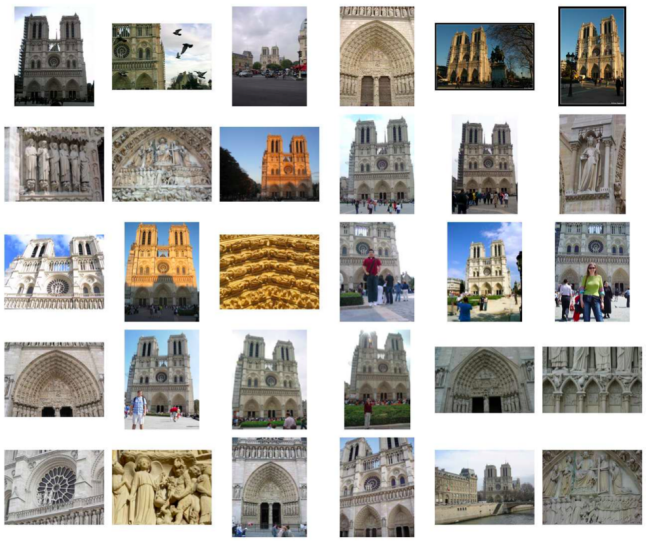


Figure 4.2: Photos of the Notre Dame cathedral taken and uploaded by various users to the website http://www.flickr.com[1].

Using feature detection and matching methods (e.g. SIFT - section 3.5), we can find and match keypoints between pairs of photographs. Matching keypoints between each pair are then organised into tracks, where a track is a set of connected images containing corresponding keypoint matches. Ideally, matched feature points in the same track should all converge to the same object point in real world 3D coordinates (Fig. 4.3). Realistically, there will always exist mismatches and false positives due to automated processes, and rays back-projected from the 2D image points are not guaranteed to converge. With maximum consensus, we can rebuild the 3D structure of

[1]Photo collage taken from http://phototour.cs.washington.edu.

the scene captured in those tracks by estimating the coordinates of each 3D object point projected onto the track (Fig. 1.8).

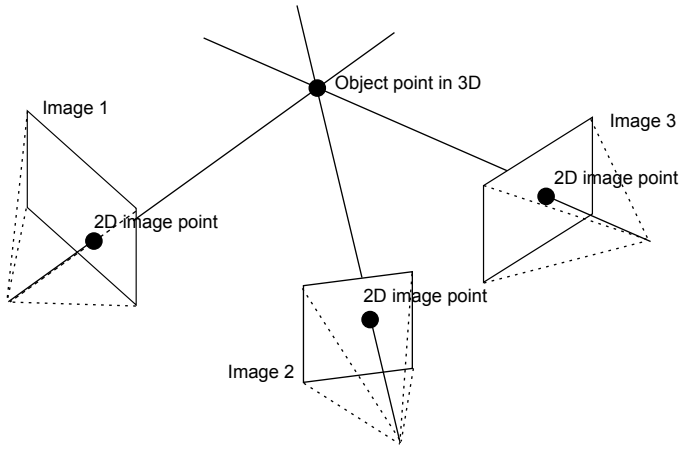


Figure 4.3: Ideal track with back-projected rays converging at a single point.

In this context, the model we wish to estimate is the coordinates of a single 3D object point $\boldsymbol{\theta} \in \mathbb{R}^3$ corresponding to a single matched keypoint in a track of N images. The data available to us is $\mathcal{X} = \{\mathbf{x}_i, P_i\}_{i=1}^{N}$, where $\mathbf{x}_i^T = (x_i, y_i)$ refers to the 2D coordinates of the matched keypoint in the i-th image, while $P_i \in \mathbb{R}^{3\times4}$ are matrices containing information on the cameras used to capture each image in the track. Readers are referred to [27, Chap. 6 & 7] and [47] for more on camera matrices. A camera matrix P_i models the intrinsics and extrinsics of each shot, such as the relative location, orientation, and the focal length of the camera. It is used to map (reproject) the estimated 3D point onto the 2D image plane

$$\tilde{\mathbf{x}}_i = P_i \tilde{\boldsymbol{\theta}} \ , \tag{4.21}$$

where $\tilde{\mathbf{x}}_i \in \mathbb{P}^2$ and $\tilde{\boldsymbol{\theta}} \in \mathbb{P}^3$ are column vectors in homogeneous coordinates, with $\tilde{\boldsymbol{\theta}} = (\boldsymbol{\theta}^T, 1)^T$. We can model the rows of camera matrix P_i using vectors

$$P_i = \begin{bmatrix} \mathbf{a}_i \\ \mathbf{b}_i \\ \mathbf{c}_i \end{bmatrix} \ ,$$

where $\mathbf{a}_i$, $\mathbf{b}_i$, and $\mathbf{c}_i$ are all row vectors in $\mathbb{R}^4$. Thus from (4.21) we get

$$\tilde{\mathbf{x}}_i = (\mathbf{a}_i\tilde{\boldsymbol{\theta}} \ , \ \mathbf{b}_i\tilde{\boldsymbol{\theta}} \ , \ \mathbf{c}_i\tilde{\boldsymbol{\theta}})^T \ .$$

Converting $\tilde{\mathbf{x}}_i$ into Euclidean coordinates $\hat{\mathbf{x}}_i \in \mathbb{R}^2$ gives

$$\hat{\mathbf{x}}_i^T = \left(\frac{\mathbf{a}_i \tilde{\boldsymbol{\theta}}}{\mathbf{c}_i \tilde{\boldsymbol{\theta}}} \quad , \quad \frac{\mathbf{b}_i \tilde{\boldsymbol{\theta}}}{\mathbf{c}_i \tilde{\boldsymbol{\theta}}} \right) \ . \tag{4.22}$$

The triangulation residual thus involves the error between $\hat{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_i$

$$r_i(\boldsymbol{\theta}) \ = \ \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_p \ = \ \frac{\| \ (\ \mathbf{a}_i \tilde{\boldsymbol{\theta}} - x_i \cdot \mathbf{c}_i \tilde{\boldsymbol{\theta}} \ , \ \mathbf{b}_i \tilde{\boldsymbol{\theta}} - y_i \cdot \mathbf{c}_i \tilde{\boldsymbol{\theta}} \ )^T \|_p}{\mathbf{c}_i \tilde{\boldsymbol{\theta}}} \ , \tag{4.23}$$

where $\mathbf{c}_i \tilde{\boldsymbol{\theta}}$ is constrained to be strictly positive to ensure that the estimated 3D object point $\boldsymbol{\theta}$ lies in front of the camera. This is known as the *cheirality constraint*. Observe that (4.23) is a special case of the quasiconvex residual in (4.15). The triangulation residual (4.23) is known as the *reprojection error*, which is the normed difference between the measured image point $\mathbf{x}_i$ and the reprojected point $\hat{\mathbf{x}}_i$ from the estimated $\boldsymbol{\theta}$ onto the camera image plane.

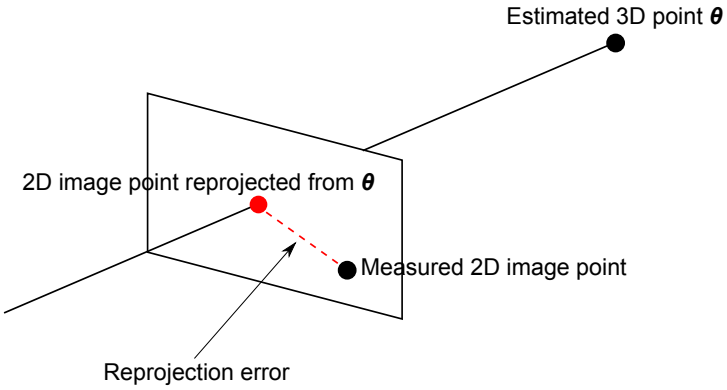


Figure 4.4: Reprojection error using the L_2-norm.

As illustrated in Fig. 4.4, the L_2-norm takes the shortest straight-line (Euclidean) distance between $\hat{\mathbf{x}}_i$ and $\mathbf{x}_i$ as the reprojection error. However, the L_2-norm results in a SOCP constraint, which can not be solved using MILP. Viable alternatives include the L_1-norm, which takes as the error the sum of differences in the vertical and horizontal components between $\hat{\mathbf{x}}_i$ and $\mathbf{x}_i$ (i.e. the Manhattan distance); and the L_∞-norm, which takes the largest of differences in either the horizontal or the vertical component. Fig. 4.5 illustrates the differences in unit circles between the three vector norms.

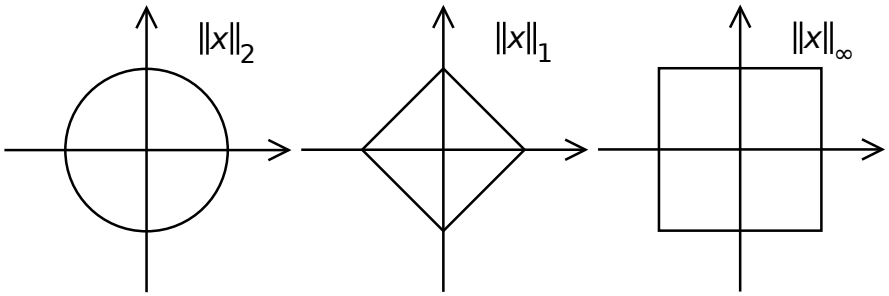


Figure 4.5: Unit circles of the three most commonly used vector norms[1].

The successful triangulation of 3D object points from 2D images is contingent on knowing the camera matrices for each photograph. However, in the case where the only thing we have are a batch of photographs of similar scenes, it is still possible to recover information on the camera poses by solving the Structure from Motion problem (SfM). Interested readers can refer to [1], which addresses in further detail the challenges of 3D reconstruction.

Figure 4.6: A sparse 3D reconstriction of the Notre Dame Cathedral using photos sourced from photo sharing sites[2].

[1] Adapted from http://en.wikipedia.org/wiki/Lp_space.
[2] Image taken from http://phototour.cs.washington.edu.

4.5.2 Image Matching

Given two images or photographs of the same scene taken from different distances, angles, and orientation, we perform feature detection and matching (e.g. SIFT - section 3.5) to obtain a set of keypoint matches $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, where $\mathbf{x_i} = (x_i, y_i)^T$ is the Euclidean coordinates of the i-th keypoint in the first image, and $\mathbf{x}'_i = (x'_i, y'_i)^T$ is the Euclidean coordinates of the corresponding keypoint match from the second image. Apart from epipolar geometry, the two images are also related by an affine transformation $\mathbf{T} \in \mathbb{R}^{2\times 3}$, which is what we aim to estimate. $\mathbf{T}$ aligns the two images

$$\mathbf{x}'_i = \mathbf{T}\tilde{\mathbf{x}}_i \ , \tag{4.24}$$

where the vector $\tilde{\mathbf{x}}_i^T = (x_i, y_i, 1) \in \mathbb{P}^2$ is $\mathbf{x}_i$ in homogeneous coordinates, and $\mathbf{T}$ is a 2×3 matrix with 6 degrees of freedom

$$\mathbf{T} = \begin{bmatrix} \boldsymbol{\theta}_1 & \boldsymbol{\theta}_2 & \boldsymbol{\theta}_3 \\ \boldsymbol{\theta}_4 & \boldsymbol{\theta}_5 & \boldsymbol{\theta}_6 \end{bmatrix} \ .$$

The matching error for the i-th correspondence is thus

$$r_i(\mathbf{T}) = \|\mathbf{T}\tilde{\mathbf{x}}_i - \mathbf{x}'_i\|_p \ . \tag{4.25}$$

Let the vector $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \boldsymbol{\theta}_4, \boldsymbol{\theta}_5, \boldsymbol{\theta}_6)^T \in \mathbb{R}^6$ be the vectorised form of $\mathbf{T}$. Thus the error in (4.25) can be rewritten as

$$r_i(\boldsymbol{\theta}) = \| \ (\ \mathbf{a}_i\boldsymbol{\theta} - x'_i \ , \ \mathbf{b}_i\boldsymbol{\theta} - y'_i \ )^T \|_p \tag{4.26}$$

for row vectors $\mathbf{a}_i = (\tilde{\mathbf{x}}_i^T, \mathbf{0}_{1\times 3}) \in \mathbb{R}^6$ and $\mathbf{b}_i = (\mathbf{0}_{1\times 3}, \tilde{\mathbf{x}}_i^T) \in \mathbb{R}^6$. Clearly, (4.26) is yet another special case of (4.15), and it can again be converted into four MILP constraints using either the L_1-norm or the L_∞-norm.

Fig. 4.7 illustrates the correspondences that form the dataset $\mathcal{X} = \{\mathbf{x}_i, \mathbf{x}'_i\}_{i=1}^N$, which is detected and matched using SIFT. The results of estimating the affine transformation matrix $\mathbf{T}$ using MILP is shown in Fig. 4.8, where only the inlying correspondences are displayed. An alternate way of illustrating the inliers is shown in Fig. 4.9, which is a composite image created by overlaying the two photographs in red-cyan, and then superimposing the inlying correspondences on top of the composite. The "flow" of the inlying points seen in Fig. 4.9 can be interpreted as a combination of the rotational, shear, and translational components of the affine transformation.

More on the uses of affine transformations such as in medical image registration and feature detection and matching can be found in [54, 56].

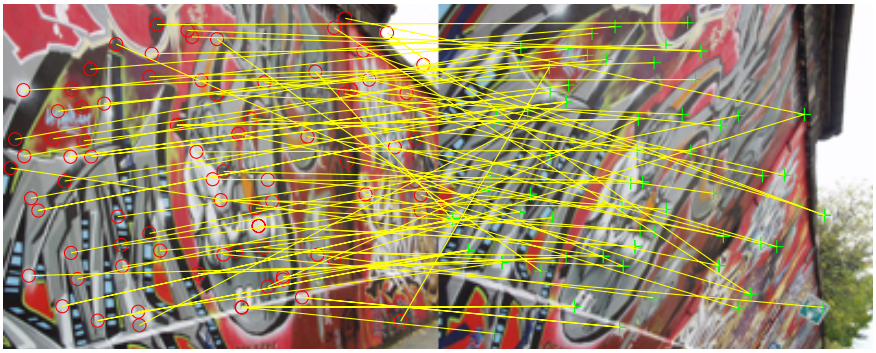


Figure 4.7: Putative pairwise correspondences for image matching[1].

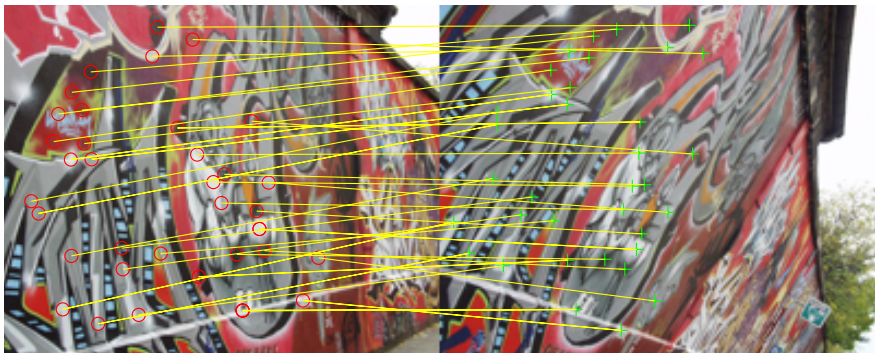


Figure 4.8: Inliers to the affine transformation $\mathbf{T}$ estimated using MILP.

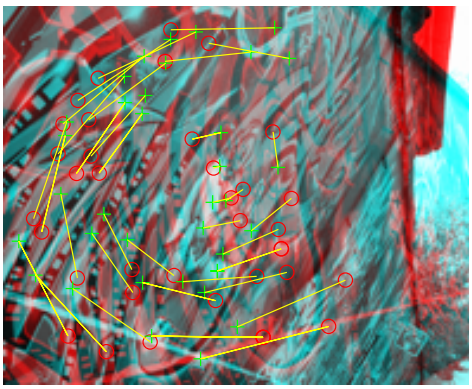


Figure 4.9: A red-cyan overlay with superimposed inliers.

[1]Photos taken from http://www.robots.ox.ac.uk/~vgg/research/affine/.

4.6 Experimental Results

Several experiments were performed to examine the efficacy of GORE in speeding up maximum consensus. In particular, we compared the runtime of exactly solving (2.17) on input data $\mathcal{X}$ (we call this EXACT), with the total runtime of running GORE to reduce $\mathcal{X}$ to $\mathcal{X}'$ and then exactly solving (2.17) on $\mathcal{X}'$ (we call this GORE+EXACT). To solve (2.17) exactly, the industry grade Gurobi Optimiser is applied to exactly solve the MILP formulation (4.2). Of course, the GORE preprocessor is also implemented using Gurobi; a combination that enables a cogent test of the benefits of GORE.

We emphasise that GORE is a *preprocessing* routine that is independent of any exact algorithm for maximum consensus [10, 20, 29, 35, 61], and therefore should not be treated as "competitors" of these algorithms. While GORE is expected to significantly speed up the computation of the aforementioned methods, the lack of mature implementations for each of those algorithms hamper accurate evaluations. More relevant competitors are [6, 51]. However these methods are highly specialised (for 2D rigid transforms and 3D rotations), where the associated maximum consensus problems do not have generalised MILP formulations, making direct comparisons with GORE infeasible.

To evaluate the practicality of GORE+EXACT, we compared it against the following approximate methods:

- RANSAC [22];
- L_∞ outlier removal [46]; and
- L_1 outlier removal [36].

Note that RANSAC is also used in the GORE (Algorithm 2) for estimating an upper bound $\hat{u}$.

The experiments were applied on several problems common in computer vision. The experiments were carried out on a standard 2.70 GHz machine with 128 GB of RAM. A slack value of $M = 1000$ was used in all the MILP instances for solving computer vision problems, and a slack value of $M = 10000$ was used to solve synthetic data.

4.6.1 Synthetic Data

Synthetic data in the form $\mathcal{X} = \{A_i, \mathbf{b}_i, \mathbf{c}_i, d_i\}_{i=1}^N$ (4.15) was produced for testing GORE. For the sake of simplicity, we chose to set $\mathbf{c}_i = \mathbf{0}$ and $d_i = 1$

for all i. The ground truth was generated uniformly in $[-1, 1]^n$. Each A_i was drawn uniformly from the range $[-50, 50]^{2\times n}$, and $\mathbf{b}_i$ was obtained as $\mathbf{b}_i = -A_i\boldsymbol{\theta}$. This justifies the residual

$$r_i(\boldsymbol{\theta}) = \|A_i\boldsymbol{\theta} + \mathbf{b}_i\|_p \ , \tag{4.27}$$

which is a special case of (4.15). Note that this generation method is different from the one that produced the dataset illustrated in Fig. 4.1. To simulate outliers, 55% of the *dependent measurements* $\{\mathbf{b}_i\}_{i=1}^N$ (i.e. the "*response*", see section 2.1) were perturbed with independent and identically distributed (i.i.d.) uniform noise in the range of $[-50, 50]$, while the rest were perturbed with i.i.d. normal inlier noise with $\sigma = 1$. This produces datasets with an outlier rate of $50\% \sim 60\%$.

Combinations of (n, N) tested were $(3, 140)$, $(4, 120)$, $(5, 100)$, $(6, 80)$, $(7, 70)$, and $(8, 50)$. N was reduced correspondingly for higher dimensions of n to avoid excessively long runtimes. This does not invalidate our assessment of GORE since we are primarily interested in the speed-up ratio. For each (n, N) pair, we created 20 data instances $\mathcal{X}$. On each $\mathcal{X}$, we executed EXACT and GORE+EXACT with a threshold of $\epsilon = 2$. For GORE, the number of rejection tests T was varied from 1 to $\lceil 0.15N \rceil$, while the maximum duration per test was fixed to a maximum of 15s.

The computational gain achieved by GORE+EXACT on input data $\mathcal{X}$ can be expressed as the ratio

$$1 - \frac{\text{runtime of GORE+EXACT}}{\text{runtime of EXACT}} \ . \tag{4.28}$$

The median gain across varying T (expressed as a ratio of N) are shown in Fig. 4.10. The gain is negative for $n = 3$, since $EXACT$ was very fast on such a low dimension and preprocessing with GORE inflated the runtime needlessly. Again, note that the data used in this experiment is different from the one shown in Fig. 4.1. For $n = 4$ to 7, the gain increases with T, implying that as more true outliers were removed, the runtime of EXACT was reduced quickly when $\mathcal{X}$ is preprocessed with GORE. We note that the gain increases very slowly beyond $T \approx 0.1N$. For $n = 8$, however, the gain is negative; this was because GORE was unable to reject sufficient number of outliers within the time limit of $c = 15$s for the preprocessing procedure to pay off. This suggests that there exists a lower and upper limit on n where GORE can be useful.

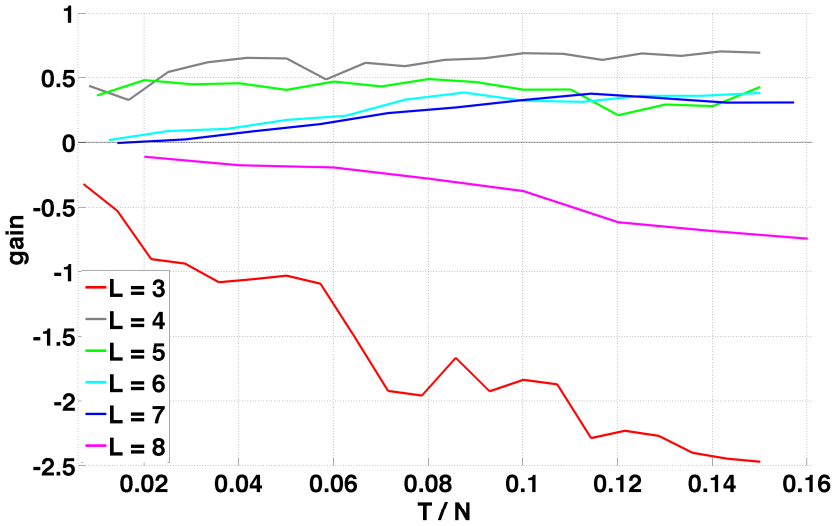


Figure 4.10: Computational gain of GORE on synthetic data for dimensions $n = 3, \cdots, 8$ and increasing number of rejection tests T as a ratio of problem size N. Time per test c is fixed at a maximum of 15s.

4.6.2 Image Matching

We tested GORE on images that have been previously used for affine image matching: Wall, Graffiti, Boat, and Bark from the affine covariant features dataset[6], and Tissue and Dental from a medical image registration dataset [55, 56]. The images were resized before SIFT features were detected and matched using VLFeat[2] to yield ≈ 100 point matches per image pair.

For GORE (Algorithm 2), the upper bound $\hat{u}$ was obtained by running RANSAC for 10,000 iterations, T was set to 10, and c was set to 15s. In this experiment, we used $p = \infty$ (i.e. the L_∞-norm) on the matching error (4.25) for outlier rejection using GORE.

Table 4.1 shows the results, where for each method or pipeline, we recorded the obtained consensus size and total runtime; the size of the reduced input $\mathcal{X}'$ by GORE is also shown. As expected, although the approximate methods were fast, they did not return the globally optimal result. GORE was able to reduce $\mathcal{X}$ by 5 to 10 outliers with the given runtime. However, this was sufficient to significantly reduce the runtime of solving (2.17) such that

[6] http://www.robots.ox.ac.uk/~vgg/research/affine/index.html
[2] http://www.vlfeat.org

	Wall $N = 85$, $\epsilon = 1$			Graffiti $N = 92$, $\epsilon = 1$			Boat $N = 98$, $\epsilon = 1$		
Methods	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)
l_∞ method [46]	9		0.04	25		0.04	8		0.05
l_1 method [36]	14		0.02	20		0.02	3		0.02
RANSAC [22]	27		0.88	42		0.89	34		0.91
EXACT	31		1028.07	47		472.90	34		603.55
GORE		75	45.25		82	70.08		88	129.39
GORE+EXACT	31		226.45	47		100.41	34		262.58
	Bark $N = 121$, $\epsilon = 1$			Tissue $N = 110$, $\epsilon = 2$			Dental $N = 101$, $\epsilon = 2$		
Methods	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)
l_∞ method [46]	11		0.04	2		0.05	21		0.04
l_1 method [36]	5		0.02	0		0.02	16		0.02
RANSAC [22]	44		0.92	35		0.89	48		0.89
EXACT	46		1077.89	37		284.07	49		258.34
GORE		115	138.79		100	90.82		91	64.41
GORE+EXACT	46		508.66	37		185.59	49		96.33

Table 4.1: Results of affine image matching. N = size of input data $\mathcal{X}$, ϵ = inlier threshold (in pixels) for maximum consensus, $|\mathcal{I}|$ = size of optimised consensus set, $|\mathcal{X}'|$ = size of reduced data by GORE.

GORE+EXACT was much faster than EXACT on all of the data instances shown (respectively 78%, 79%, 57%, 53%, 35% and 63% gain).

4.6.3 Epipolar Geometry Estimation

Please refer to a previous discussion in section 3.5.1 on the affine fundamental matrix for affine epipolar geometry estimation and how it can be achieved using maximum consensus using the linear regression residual. We tested GORE with datasets from similar groups that was used in a previous experiment on M-bisection in section 3.6.2. The images were previously used in other works for affine epipolar geometry estimation [3], such as Dinosaur (frames 7 and 8), Kapel, and Valbonne Church from the multi-view reconstruction dataset[3], Notre Dame from the Paris dataset[4], and All Souls from

[3] http://www.robots.ox.ac.uk/~vgg/data/data-mview.html
[4] http://www.robots.ox.ac.uk/~vgg/data/parisbuildings/

the Oxford Buildings dataset[1]. All images mentioned were provided by the Oxford Visual Geometry Group.

The results are shown in Table 4.2. Despite having a lower degree of freedom than an affine transform, solving maximum consensus (2.17) exactly on the affine fundamental matrix generally takes a longer time. Unlike in image matching using affine transforms, GORE was less prolific in removing outliers; on average GORE is only able to remove 2 outliers out of $T = 10$ attempts, an indication that GORE is not suited for preprocessing problems in higher dimensions with the linear regression residual, e.g. $n \geq 4$. interestingly, however, we can observe that simply rejecting 2 outliers from $\mathcal{X}$ allows for a significant speed-up in the overall runtime of GORE+EXACT for at least 3 of the chosen image pairs.

	Dinosaur $N = 115$, $\epsilon = 1$			Kapel $N = 107$, $\epsilon = 2$			Notre Dame $N = 69$, $\epsilon = 2$		
Methods	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)
l_∞ method [46]	30		0.03	42		0.03	14		0.03
l_1 method [36]	38		0.01	43		0.01	7		0.01
RANSAC [22]	68		1.04	62		0.99	28		1.00
EXACT	71		3924.57	65		5538.66	31		515.08
GORE		113	135.95		104	148.27		68	150.92
GORE+EXACT	71		1237.90	65		1263.25	31		634.98

	All Souls $N = 76$, $\epsilon = 2$			Valbonne $N = 63$, $\epsilon = 2$		
Methods	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)
l_∞ method [46]	31		0.02	13		0.03
l_1 method [36]	15		0.01	20		0.01
RANSAC [22]	39		0.99	29		0.98
EXACT	42		2795.35	32		593.34
GORE		72	75.25		62	59.58
GORE+EXACT	42		858.06	32		470.91

Table 4.2: Results of affine epipolar geometry estimation. N = size of input data $\mathcal{X}$, ϵ = inlier threshold (in pixels) for maximum consensus, $|\mathcal{I}|$ = size of optimised consensus set, $|\mathcal{X}'|$ = size of reduced data by GORE.

[1] http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/

4.6.4 Triangulation

For the purpose of testing GORE we made use of the Photo Tourism dataset from [48] containing a set of image observations and camera matrices. We specifically chose 6 of the given 3D points with over 100 views (i.e. $N > 100$) to triangulate; the index of these points are listed in Table 4.3. With a threshold of $\epsilon = 1$ (i.e. one pixel), these problem instances will have an outlier ratio between $40\% \sim 60\%$. Note that our triangulation model given in (4.23) ignores the radial distortion error, and thus the actual outlier rate may be much lower when it is taken into account. Nonetheless, it is sufficient for the purpose of testing the potential speed-ups we achievable by preprocessing with GORE. In this experiment we used $p = \infty$ (i.e. the L_∞-norm) for the residual given in (4.23). For GORE, T was set to 10 and c was set to 15s.

The results are shown in Table 4.3. For 5 of the 6 points tested, preprocessing with GORE provided a considerable reduction in total runtime, especially for cases with a high outlier rate.

	Point 1 $N = 167$, $\epsilon = 1$			Point 2 $N = 105$, $\epsilon = 1$			Point 36 $N = 175$, $\epsilon = 1$		
Methods	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)
l_∞ method [46]	96		1.93	23		1.83	63		2.17
l_1 method [36]	111		0.03	33		0.03	73		0.04
RANSAC [22]	114		1.16	36		1.12	84		1.15
EXACT	115		17.80	38		112.83	90		956.35
GORE		157	14.65		95	11.02		165	20.09
GORE+EXACT	115		27.80	38		75.67	90		771.32

	Point 594 $N = 159$, $\epsilon = 1$			Point 682 $N = 153$, $\epsilon = 1$			Point 961 $N = 129$, $\epsilon = 1$		
Methods	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{X}'\rvert$	time(s)
l_∞ method [46]	53		1.42	79		1.78	58		1.81
l_1 method [36]	67		0.02	87		0.03	61		0.03
RANSAC [22]	85		1.12	96		1.13	69		1.12
EXACT	87		812.81	97		56.44	70		110.90
GORE		149	17.72		143	16.12		119	8.55
GORE+EXACT	87		56.35	97		35.40	70		25.86

Table 4.3: Results of triangulation. N = size of input data $\mathcal{X}$, ϵ = inlier threshold (in pixels) for maximum consensus, $|\mathcal{I}|$ = size of optimised consensus set, $|\mathcal{X}'|$ = size of reduced data by GORE.

4.7 Summary

We proposed GORE as a way to speed up the computation of exact solutions to the maximum consensus problem. GORE achieves this by preprocessing the data to identify and remove instances that are guaranteed to be outliers to the globally optimal solution. The proposed methods works by inspecting data instances that are most likely outliers to the globally optimal solution, and prove using contradiction that it is indeed an outlier. The rejection test is based on comparing upper and lower bound values derived using MILP. We also showed how GORE can be extended to deal with geometric residuals that are quasiconvex. The fact that GORE is formulated as a MILP allows it to be easily implemented using any off-the-shelf optimisation software.

GORE is experimentally evaluated on both synthetically generated and real data, based on common computer vision applications. While there are limitations to GORE in terms of the range of model dimensions it can handle, we demonstrate that it is still very useful and effective on a wide range of practical computer vision applications.

Chapter 5

Conclusion

In this work we are focused on the problem of parameter estimation and how it can be applied to various problems in computer vision. The first two chapters are devoted to discussions on how parameter estimation can be achieved, in particular

- A brief introduction is given on the problem parameter estimation and how it is applicable in the field of computer vision. We discussed the challenges in discovering suitable model parameters in the presence of noise and outliers in the data. Various minimisation criterias are then presented with a focus on consensus maximisation, which is one of the more popular robust estimation criterion in the field of computer vision.
- The more common techniques for parameter estimation are reviewed, devoting particular attention to the robustness of various techniques and where each of the methods fall short. Techniques that approximates the optimal solution are usually efficient but are non-deterministic and offers no guarantees to the optimality of its solution; whereas techniques that compute the exact solution to any particular problem will generally incur a high computational cost. Current applications favour the use of highly efficient approximation methods, and the present work is focused on improving the efficiency of exact methods in an attempt of provide a more deterministic alternative.

In the following chapters, two preprocessing methods are proposed for improving the efficiency and reliability of a particular class of exact parameter estimation techniques based on linear programming known as the Big-M method. In particular:

- A M-bisection method is proposed to optimise M by means of reduction. While it can potentially speed up the computation of an exact solution, the method itself is rather unstable and can also potentially result in an overall slowdown instead of a speedup. However, the idea of decreasing M by means of bisection holds merit and it may be possible to improve the viability of this approach with further investigation and refinement.

- A Guaranteed Outlier Removal scheme (GORE) is proposed to speed up the computation of Big-M by finding and removing data points that are guaranteed to be outliers. Experiments show that removing a mere 5% of the data can yield an overall decrease in runtime by up to 80%. The GORE method has been verified to work with general quasiconvex problems in the range of 4 to 7 dimensions, which makes it applicable to a good range of applications. Unfortunately, the preprocessing procedure itself is also rather expensive to compute, and current approximation methods render this class of approaches unattractive for general use. However, GORE represents a step forward in the area of exact robust estimation, and brings us one step closer to that goal.

We have demonstrated in this thesis that it is indeed possible to reduce the computational cost of Big-M by preprocessing the data using either M-bisection or GORE. However, there remain many opportunities for extending the findings of this thesis, with one possibility being the use of both M-bisection and GORE in conjunction with each other. With further investigation and analysis, it may be possible to develop a new preprocessing algorithm as a combination of the core concepts of the two aforementioned methods. Another possible extension for future research involves investigating the possibilities of using a specialised, binary based MILP solver for improving the efficacy of GORE. We hope that the ideas brought forth can provide new insight into the problem of parameter estimation and act as a spark towards further discussions in the community.

Bibliography

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011. 56

[2] E. Amaldi, M. Bruglieri, and G. Casale. A two-phase relaxation-based heuristic for the maximum feasible subsystem problem. *Computers and Operations Research*, 35(5):1465–1482, 2008. 29

[3] R. Arandjelović and A. Zisserman. Efficient image retrieval for 3D structures. In *Proceedings of the British Machine Vision Conference*, 2010. 38, 40, 62

[4] Stan Birchfield. An introduction to projective geometry (for computer vision). 36

[5] Michael J Black and Anand Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International Journal of Computer Vision*, 19(1):57–91, 1996. 15

[6] A. Parra Bustos and T.-J. Chin. Guaranteed outlier removal for rotation search. In *ICCV*, 2015. 45, 59

[7] Emmanuel J Candes, Paige Randall, et al. Highly robust error correction by convex programming. *Information Theory, IEEE Transactions on*, 54(7):2829–2840, 2008. 8

[8] Emmanuel J Candes and Terence Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215, 2005. 8

[9] E. W. Cheney. *Inroduction to approximation theory*. McGraw-Hill, 1966. 13

[10] T.-J. Chin, P. Purkait, A. Eriksson, and D. Suter. Efficient globally optimal consensus maximisation with tree search. In *CVPR*, 2015. 20, 59

[11] J. W. Chinneck. *Feasibility and infeasibility in optimization: algorithms and computational methods*. Springer, 2008. 10, 26, 27, 29

[12] Ondřej Chum and Jiří Matas. Matching with prosac-progressive sample consensus. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 220–226. IEEE, 2005. 20

[13] Ondřej Chum and Jiří Matas. Optimal randomized RANSAC. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(8):1472–1482, 2008. 20

[14] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Pattern recognition*, pages 236–243. Springer, 2003. 20

[15] M. Conforti, G. Cornuéjols, and G. Zambelli. *Integer programming*. Springer, 2014. 21, 32, 48

[16] Arnak Dalalyan and Renaud Keriven. L_1-Penalized Robust Estimation for a Class of Inverse Problems Arising in Multiview Geometry. In *Advances in Neural Information Processing Systems*, pages 441–449, 2009. 8

[17] David L Donoho. For most large underdetermined systems of linear equations the minimal l_1-norm solution is also the sparsest solution. *Communications on pure and applied mathematics*, 59(6):797–829, 2006. 8

[18] David L Donoho, Michael Elad, and Vladimir N Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *Information Theory, IEEE Transactions on*, 52(1):6–18, 2006. 8

[19] Herbert Edelsbrunner and Diane L Souvaine. Computing least median of squares regression lines and guided topological sweep. *Journal of the American Statistical Association*, 85(409):115–119, 1990. 16

[20] O. Enqvist, E. Ask, F. Kahl, and K. Åström. Robust fitting for multiple view geometry. In *ECCV*, 2012. 59

[21] Jeff Erickson, Sariel Har-Peled, and David M Mount. On the least median square problem. *Discrete & Computational Geometry*, 36(4):593–607, 2006. 8, 15

[22] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24(6):381–395, 1981. 9, 59, 62, 63, 64

[23] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420, 1970. 13

[24] R. I. Hartley and F. Schaffalitzky. L_∞ minimization in geometric reconstruction problems. In *CVPR*, 2004. 7, 50

[25] Richard Hartley and Fredrik Kahl. Optimal algorithms in multiview geometry. In *Computer Vision–ACCV 2007*, pages 13–34. Springer, 2007. 7

[26] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 13, 36, 37, 38

[27] F. Kahl and R. Hartley. Multiple-view geometry under the L_∞ norm. *IEEE TPAMI*, 30(9):1603–1617, 2008. 7, 50, 54

[28] Qifa Ke and Takeo Kanade. Quasiconvex optimization for robust geometric reconstruction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(10):1834–1847, 2007. 7

[29] H. Li. Consensus set maximization with guaranteed global optimality for robust geometry estimation. In *ICCV*, 2009. 10, 21, 29, 59

[30] Hongdong Li. A practical algorithm for L_∞ triangulation with outliers. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. 7, 8, 14

[31] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999. 34

[32] Jiří Matoušek, Micha Sharir, and Emo Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4-5):498–516, 1996. 21

[33] David Nistér. Preemptive ransac for live structure and motion estimation. *Machine Vision and Applications*, 16(5):321–329, 2005. 20

[34] Clark F Olson. An approximation algorithm for least median of squares regression. *Information Processing Letters*, 63(5):237–241, 1997. 9, 15, 16

[35] C. Olsson, O. Enqvist, and F. Kahl. A polynomial-time bound for matching and registration with outliers. In *CVPR*, 2008. 59

[36] C. Olsson, A. Eriksson, and R. Hartley. Outlier removal using duality. In *CVPR*, 2010. 7, 59, 62, 63, 64

[37] C. Olsson, A. Eriksson, and F. Kahl. Efficient optimization for L_∞-problems using pseudoconvexity. In *ICCV*, 2007. 8

[38] M. Padberg. *Linear optimization and extensions*. Springer, 1999. 27, 29

[39] S. Papert. The Summer Vision Project. http://hdl.handle.net/1721.1/6125. 1

[40] Karl Pearson. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. 19

[41] M. Pfetsch. Branch-and-cut for the maximum feasible subsystem problem. *SIAM Journal on Optimization*, 19(1):21–38, 2008. 10, 21

[42] Peter J. Rousseeuw. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880, 1984. 5, 7, 8, 9, 14, 15

[43] Yongduek Seo and Richard Hartley. A fast method to minimize L_∞ error norm for geometric vision problems. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007. 8

[44] Yongduek Seo, Hyunjung Lee, and Sang Wook Lee. Outlier Removal by Convex Optimization for L-Infinity Approaches. In *Advances in Image and Video Technology*, pages 203–214. Springer, 2009. 8

[45] Larry S Shapiro, Andrew Zisserman, and Michael Brady. 3d motion recovery via affine epipolar geometry. *International Journal of Computer Vision*, 16(2):147–182, 1995. 38

[46] K. Sim and R. Hartley. Removing outliers using the L_∞ norm. In *CVPR*, 2006. 7, 59, 62, 63, 64

[47] Kyle Simek. Dissecting the Camera Matrix. http://ksimek.github.io/2012/08/14/decompose/, 2012. 54

[48] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. http://phototour.cs.washington.edu/, 2006. 64

[49] Diane L Souvaine and J Michael Steele. Time-and space-efficient algorithms for least median of squares regression. *Journal of the American Statistical Association*, 82(399):794–801, 1987. 16

[50] Charles V Stewart. Robust parameter estimation in computer vision. *SIAM review*, 41(3):513–537, 1999. 8, 9, 14, 15, 16, 20

[51] L. Svärm, O. Enqvist, M. Oskarsson, and F. Kahl. Accurate localization and pose estimation for large 3d models. In *CVPR*, 2014. 45, 59

[52] Philip HS Torr and David W Murray. The development and comparison of robust methods for estimating the fundamental matrix. *International journal of computer vision*, 24(3):271–300, 1997. 8, 14, 19

[53] Philip HS Torr and Andrew Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000. 18, 19, 20

[54] Tinne Tuytelaars and Luc Van Gool. Matching widely separated views based on affine invariant regions. *International journal of computer vision*, 59(1):61–85, 2004. 58

[55] C.-W. Wang. Improved image registration for biological and medical data. http://www-o.ntust.edu.tw/~cweiwang/ImprovedImageRegistration/. 61

[56] C.-W. Wang and H.-C. Chen. Improved image alignment method in application to x-ray images and biological images. *Bioinformatics*, 29(15):1879–1887, 2013. 58, 61

[57] Wikipedia. Correspondence problem. http://en.wikipedia.org/wiki/Correspondence_problem. 6

[58] Allen Y Yang, Zihan Zhou, Arvind Ganesh Balasubramanian, S Shankar Sastry, and Yi Ma. A review of fast l_1-minimization algorithms for robust face recognition. *Image Processing, IEEE Transactions on*, 22(8):3234–3246, 2013. 8

[59] J. Yu, A. Eriksson, T.-J. Chin, and D. Suter. An adversarial optimization approach to efficient outlier removal. In *ICCV*, 2011. 7

[60] Zhengyou Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and vision Computing*, 15(1):59–76, 1997. 8, 9, 14, 15

[61] Y. Zheng, S. Sugimoto, and M. Okutomi. Deterministically maximizing feasible subsystems for robust model fitting with unit norm constraints. In *CVPR*, 2011. 10, 21, 29, 59