LUND UNIVERSITY

Robust Algorithms for Multiple View Geometry: Outliers and Optimality

Enqvist, Olof

2011

Robust Algorithms for Multiple View Geometry

Outliers and Optimality

Olof Enqvist

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

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

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

Doctoral Theses in Mathematical Sciences 2011:1
ISSN 1404-0034

ISBN 978-91-7473-102-6
LUTFMA-1042-2011

Printed in Sweden by MediaTryck, Lund 2011

Preface

This thesis is based on the following papers.

Main papers

- O. Enqvist, K. Josephson and F. Kahl, Optimal Correspondences, Geometry and the Vertex Cover Problem, submitted to *International Journal of Computer Vision*.
- O. Enqvist and F. Kahl, Two View Geometry with Outliers, in *Proc. British Machine Vision Conference*, 2009.
- O. Enqvist, F. Jiang and F. Kahl, A Brute-Force Algorithm for Estimating a Scene from Two Projections, to appear in *Proc. Conference on Computer Vision and Pattern Recognition*, 2011.
- C. Olsson, O. Enqvist and F. Kahl, A Polynomial-Time Bound for Matching and Registration with Outliers, in *Proc. Conference on Computer Vision and Pattern Recognition*, 2008.
- O. Enqvist, F. Kahl and C. Olsson, Non-Sequential Structure from Motion, submitted to *International Conference on Computer Vision*, 2011.
- O. Enqvist, F. Kahl, C. Olsson and K. Åström, Global Optimization in One-Dimensional Vision, in *SIAM Journal on Imaging Sciences*, 2010.

Subsidiary papers

- K. Åström, O. Enqvist, R. Hartley, F. Kahl and C. Olsson, An L_∞ Approach to Structure and Motion Problems in 1D Vision, in *Proc. International Conference on Computer Vision*, 2007.
- O. Enqvist and F. Kahl, Robust Optimal Pose Estimation, in *Proc. European Conference on Computer Vision*, 2008.

– O. Enqvist, K. Josephson and F. Kahl, Optimal Correspondences from Pairwise Constraints, in *Proc. International Conference on Computer Vision*, 2009.

– H. Källén, H. Ardö and O. Enqvist, Reconstruction of Vehicles for Accurate Position Estimation, in *Proc. Workshop on Applications in Computer Vision*, 2011.

– C. Olsson and O. Enqvist, Stable Structure from Motion for Unordered Image Collections, to appear in *Proc. Scandinavian Conference on Image Analysis*, 2011.

Acknowledgements

This thesis would be a lot worse were it not for the people around me. For that reason and for other reasons, I would like to express my gratitude to my supervisor Fredrik Kahl and to my co-authors Carl Olsson, Kalle Åström, Richard Hartley, Klas Josephson, Fangyuan Jiang, Hanna Källén and Håkan Ardö and to everyone else at the department but also to my family and to my friends and to Nina.

Contents

Chapter 1
Introduction

The aim of the whole field of computer vision is to make computers see. In the early years of computers, leading researchers believed this to be suitable task for a summer project; see [69]. Perhaps they looked at the ease with which a child can solve complicated vision tasks and failed to realize the complexity of human vision and the multitude of different techniques that the human eyes and brain use to interpret an image.

This thesis is concerned with the geometrical part of vision, or more specifically with the three-dimensional aspects. The ability of humans to perceive the three-dimensional world from the two-dimensional projections on the retinae is fascinating. This ability is partly dependent on perspective effects, that is, the fact that three-dimensional objects look different from different view points. In human vision our two eyes have slightly different perspectives, but perhaps even more important are perspective changes due to movement. It should also be noted that occlusion, size and other effects are very important for our own three-dimensional perception.

1.1 The Camera

If light falls through a small hole into an otherwise dark room it will create an image on the wall. This dark room is the camera obscura, the veiled chamber, from which the modern camera has borrowed its name. The first mention of this principle belongs to the Chinese philosopher Mozi around 400 BC. Through history, it fascinated many scientists, e.g. the Greek philosopher Aristotle, but it was the Arab mathematician Ibn al-Haytham who started to understand the geometry of the camera.

Figure 1.1 illustrates the principle. Light from different points in space passes through a small hole at C and projects an inverted image on the image plane π. For obvious reasons cameras of this type are called pinhole cameras. The point C is the so called focal point or camera centre and the distance between C and the image plane is the focal length, f.

In the beginning of the 19th century, techniques to chemically capture the projected image were developed. Although the pinhole was soon replaced with a lens, allowing for more light to enter the camera, the geometrical properties changed only slightly, and in computer vision, the pinhole camera model is still widely used.

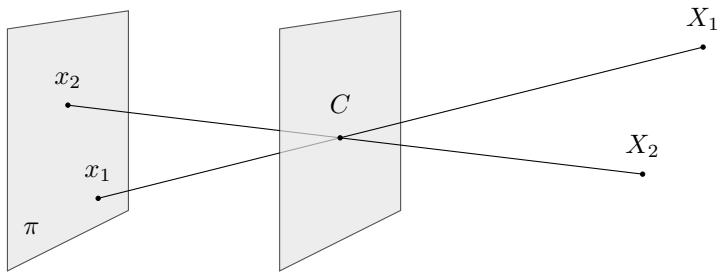

Figure 1.1: Geometry of a pinhole camera. Each X_i is a point in three-dimensional space and x_i is the corresponding image point. C is the focal point, also referred to as the camera centre and π is the image plane. The distance from the focal point to the image plane is the focal length.

As in the camera obscura, the projection in a modern digital camera is inverted. Before storing the image however, it is inverted again to fit our perception of the world. Geometrically, we get the same effect by placing the image plane in front of the focal point, being the standard model of a pinhole camera.

We will now derive the equations of the pinhole camera model. Consider a coordinate system with origin at the focal point and z-axis perpendicular to the image plane. This axis is the focal axis of the camera. Let X be the position of a 3D point given in this coordinate system. We want to determine the projected image point. The line of projection is

$$v = k\,X\,. \tag{1.1}$$

To find the coordinates of the image point, we compute the intersection of this line with the image plane, $v_z = f$. The resulting relation between image coordinates x and 3D coordinates X is often written

$$\lambda\,x = X\,, \tag{1.2}$$

where λ is chosen such that $x_z = f$.

If the 3D points are not described in the camera coordinate system, with origin at the camera centre, we need to transform the coordinates to this coordinate system before performing the projection. If R is the rotation of the camera and C is the position of the camera centre this can be written

$$\lambda\,x = R\,(X - C)\,. \tag{1.3}$$

Note that any x satisfying this equation with $\lambda > 0$, could be used to represent the projected image point. Often we will use unit vectors. Geometrically, this corresponds to projections on an image sphere rather than an image plane.

1.1.1 Camera Calibration

If image coordinates and 3D coordinates satisfy (1.3) and the focal length, $f = 1$, we are dealing with a *calibrated camera*. In practice however, image coordinates are normally measured in pixels, the origin $(0, 0)$ lies in a corner of the image rather than on the focal axis and the focal length is unknown. Conversion between calibrated coordinates and pixel coordinates is a linear transformation, $x \to Kx$, where K is called the calibration matrix. In the simplest case K scales and translates the coordinates,

$$K = \begin{pmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{1.4}$$

Including the calibration matrix in (1.3) yields $\lambda\, x = KR\,(X - C)$. For some cameras it is useful to introduce more parameters into the calibration matrix. The aspect ratio, γ, is used to model non-square pixels, and the skew, s is useful if the pixel coordinate system is not orthogonal. If all these parameters are unknown, we are dealing with an *uncalibrated camera* with calibration matrix,

$$K = \begin{pmatrix} f & s & p_x \\ 0 & \gamma f & p_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{1.5}$$

To accurately model a camera the linear transformation of a calibration matrix is not fully sufficient. The photographic lens also gives rise to a radial distortion depending on the distance to the focal axis. For the work in this thesis it is mainly sufficient to note that if the radial distortion and the calibration matrix are both known, the image can be transformed to fit the calibrated camera model of (1.3).

1.2 Geometry from Images

A central problem in computer vision, and in this thesis, is how to estimate the geometry of a scene from a set of images. The scene can be a face or a building or anything else that we want to model. Let us first define what we mean by the geometry of a scene. This thesis deals mainly with discrete point sets. Thus the geometry of a scene is a set of points having position and appearance. This might seem very restricted. Not only is the visible world built of continuous shapes of varying texture and color, but more so these shapes move and deform with time. Still, because of their simplicity, point models can be used as tools or building blocks for almost any type of geometric models. For one thing, points provide a simple and robust way to estimate camera positions, which are required to estimate more complex geometric models. Moreover, knowing a large number of points on a surface is often a sufficient model of the surface and in other cases it gives means to estimate a continuous representation.

Fundamental in this context is the notion of point-to-point correspondences. Points in different views are corresponding points if they are projections of the same point in three dimensions. Similarly, a point in an image corresponds to a three-dimensional point if it is the projection of this point.

We have two basic tools to determine correct correspondences, appearance and geometry. To establish correct correspondences between two images it is generally important to consider both the local appearance of points and their position relative to the other points and the camera. Often a large set of hypothetical correspondences are generated based on appearance. Then robust methods are used to remove correspondences which are not geometrically consistent.

A method for appearance-based matching typically consists of three steps. The first step determines a set of interest points in each image. Ideally the same points should be singled out in both views. A classic approach to this is to detect corner points in both images [38]. An alternative is to detect blobs in the images by looking for scale-space extrema in the response from a difference of Gaussians filter [58] or looking for particularly stable regions [63].

Having detected a suitable set of interest points the next step uses a feature descriptor to capture the appearance of these points. A feature descriptor that received a lot of attention is SIFT [58]. It considers a small neighbourhood of the interest point, describing it using histograms of image gradients. Similar ideas are implemented in SURF [5], but the design is modified to get a faster algorithm.

The last step of the appearance-based matching is to generate the actual correspondences. For each interest point the nearest neighbour with respect to the feature vectors is computed. To reduce the amount of incorrect matchings, called outliers, some way to discard spurious correspondences is required. The following is a simple and remarkably efficient way to do this. For a given interest point let d_k be the distance to the kth closest match in the other image. If d_1/d_2 is small then the match has high probability of being correct.

For the following discussion, let us assume that we have a set hypothetical correspondences and that we want to determine the geometry. Estimating point geometry from a set of images is called *structure from motion* estimation or sometimes simultaneous localization and mapping, SLAM. The most common way to approach structure from motion estimation is by solving a series of basic subproblems. A few of the most important ones are listed below.

- Given corresponding points in two images, estimation of the orientation and position of one camera relative to the other. This will be referred to as *relative orientation estimation.*

- If positions and orientations of set of cameras is known and some point is visible in at least two of these cameras, then *triangulation* is the process of estimating the 3D coordinates of that point.

- Given 3D points X_i and corresponding image points x_i, estimation of the camera position and orientation is called *camera pose estimation.*

If all correspondences are correct and exact, then these problems can be tackled by setting up and solving a system of polynomial equations. In practice though, there will be noise as well as outliers so proper handling of these uncertainties is crucial to obtain accurate estimates.

To illustrate we look at the estimation of a relative orientation. The noise depends mainly on the uncertainty in detecting feature points, so it is reasonable to expect similar errors, measured in pixels, for the two cameras and for different points. Thus, it is widely accepted that one should minimize the deviation of the reprojected 3D points from the measured image coordinates. These deviations are the so called reprojection errors. Take, for example, the reprojection error of a 3D point X and the corresponding image point x, given a rotation R and a camera centre C,

$$e = \left\| \frac{\big(r_1(X - C_1), r_2(X - C_2)\big)}{r_3(X - C_3)} - x \right\|_2 \tag{1.6}$$

where r_k is the kth row of R. Sometimes we will use angular reprojection errors instead. If we let $\angle(x, y)$ denote the angle between vectors x and y we can write the angular reprojection error as

$$\varepsilon = \angle\left(x, R(X - C)\right). \tag{1.7}$$

For correct correspondences one often assumes that the reprojection errors are independent and follow a Gaussian distribution. If this is true we can compute maximum likelihood estimates by minimizing the L_2 norm of the reprojection errors. This can be achieved using local optimization, so called bundle adjustment [42]. However, a good initial guess is required and all outliers must be removed.

1.2.1 Estimation with Outliers

Geometric estimation in presence of outliers is a central problem in vision and it has been studied extensively. It should be safe to say that the field is still dominated by RANSAC-type methods. RANSAC, which stands for random sample consensus, was introduced in 1981 by Fischler and Bolles [30] to handle erroneous correspondences. A subset of the correspondences is randomly picked and used to estimate a transformation. Then all points are transformed and the reprojection errors are measured. Correspondences with a reprojection error below some predefined threshold are considered inliers to the candidate solution. This procedure is repeated a large number of times and the solution having the largest number of inliers is chosen.

During the thirty years since since Fischler and Bolles wrote their paper, many variants of RANSAC have been developed; see e.g., [83, 20]. From a combinatorial viewpoint,

the RANSAC-type algorithms are best used with a so called minimal solver. A minimal solver uses a minimal number of correspondences to estimate some desired geometry. For example, a minimal solver for computing the relative orientation requires five correspondences.

The standard method to solve geometric vision problems is to use the solution obtained by RANSAC as a starting guess for a local optimization minimizing the L_2 norm of the reprojection errors [42]. Statistically, the hidden assumption here is that those correspondences having a reprojection error less than our threshold have normal distribution with a common variance. This is not unreasonable. One could imagine that there are two types of correspondences, correct ones having small normally distributed reprojection errors and incorrect ones with large reprojection errors. The RANSAC step removes practically all incorrect correspondences such that the remaining correspondences satisfy the normal distribution assumption.

Still, there are problems with this approach and the first half this thesis is concerned with exploring alternatives. Of course, one drawback of RANSAC is that is does not guarantee an optimal choice of correspondences. Moreover algorithmic complexity increases rapidly with the amount of outliers. For the relative orientation it is inversely proportional to the rate of inliers to the power of five!

1.2.2 Estimation from Multiple Views

The previous section discussed briefly, how some of the basic subproblems of geometric vision can be solved, but the overall aim was to reconstruct a scene from several images. So how do we combine the basic solvers to handle multiple images? Of course there are many answers to this question, some of which will be discussed in Chapter 6.

One popular method starts from a single pair of views and then adds the others incrementally. Consequently, we will refer to it as sequential structure from motion. In the first stage of this estimation the relative orientation of the starting pair is estimated. Then all points visible in those two views are reconstructed using triangulation. Often this is followed by bundle adjustment [42] to improve the quality both of the relative orientation and the 3D points. Having done this the sequential part of the algorithm starts. A new camera having many correspondences among the estimated 3D points is added to the model by computing its camera pose. After this any new points being visible in two views are reconstructed and then a new camera is posed in. To stabilize the estimation bundle adjustment is normally performed a number of times, sometimes for every new camera that is added.

Sequential structure from motion has been made to work for very large data sets, producing impressive reconstructions. But as we will see in Chapter 6 there are severe stability issues. Estimation from multiple views is the subject of the second part of this thesis.

1.3 Some Useful Theory

This section presents some elementary theory that is important in the thesis. The first section discusses convexity and how it can be used to find optimal solutions to some problems from multiple view geometry. Subsequent sections go through some elementary theory of graphs, rotations, quaternions and spherical geometry. Readers who are familiar with these subjects can jump directly to Section 1.4.

1.3.1 Convexity and L_∞ Optimization

A subset S of a real vector space is convex if,

$$x, y \in S \quad \Rightarrow \quad x + t(y - x) \in S, \text{ for all } t \in [0, 1], \tag{1.8}$$

i.e., if the line segment between two points in the set is contained in the set. A function $f : S \to \mathbb{R}$ is a convex function if the epigraph of f,

$$\text{epi}f = \{(x, y) : x \in S, y \geq f(x)\} \tag{1.9}$$

is convex. Hence the minimum of a convex function can be formulated

$$\min_{(x,y)\in \text{epi}f} y. \tag{1.10}$$

This explains the large interest in minimizing linear functions over convex sets. In fact, these problems can be solved in polynomial time, meaning that the computation time is bounded by a polynomial in the size of the problem. A family of algorithms capable of this are called interior point methods or barrier methods [12].

One interesting property of convex functions is that their sublevel sets,

$$\{x : f(x) \leq \alpha\}, \tag{1.11}$$

are convex. A function with this property is called quasiconvex. Hence any convex function is quasiconvex, whereas a quasiconvex function is not necessarily convex.

A nice property of a quasiconvex function, f, is that for fixed α, checking whether there exists an x such that $f(x) \leq \alpha$ is a convex feasibility problem that can be solved using interior-point methods. As a consequence quasiconvex functions can be minimized using bisection: Suppose we have bounds m and M such that $m \leq \min_x f(x) \leq M$. Then the following bisection algorithm will converge to $\min_x f(x)$.

1. $\alpha = (m + M)/2$.
2. If there exists x fulfilling $f(x) \leq \alpha$, set $M = \alpha$, otherwise $m = \alpha$.

L_∞ Optimization. Many geometric problems in computer vision can be formulated as optimization problems. The deviation between measured coordinates and the *exact* coordinates, the measurement error, is often assumed to follow Gaussian distribution. If so, the statistically optimal solution is given by minimizing the L_2 norm of the reprojection errors. Unfortunately it is very hard to find this solution. Existing techniques rely on a good initialization followed by local optimization and cannot guarantee global optimality. To handle this problem L_∞ optimization was introduced in [41]. Using the L_∞ norm of the reprojection errors instead of the L_2 norm, makes it possible to find and verify the global optimum. The reason is that a number of L_∞ optimization problems in geometric vision are quasiconvex; see [49, 50]. More precisely, the maximum norm of the reprojection error is a quasiconvex function for a number of geometric problems.

This owes to the fact that quasiconvexity is preserved under the maximum operation. More precisely, if $f_1, \ldots, f_n$ are quasiconvex functions, then so is the function f defined by

$$f(x) = \max\{f_1(x), \ldots, f_n(x)\}. \tag{1.12}$$

The proof is short: A sublevel set for f is the intersection of n sublevel sets for f_1 to f_n, and the intersection of convex sets is convex.

Now let us return to formula for a reprojection error, e, that we saw in (1.6). Recall that X is a 3D point and x its projection in a camera with orientation R and position C. We can rewrite (1.6) as

$$e = \frac{||(r_1(X - C_1), r_2(X - C_2)) - x(r_3(X - C_3))||_2}{r_3(X - C_3)}. \tag{1.13}$$

To see when this is a quasiconvex function, we consider the sublevel sets. We set $e \leq \epsilon$ and multiply both sides with $r_3(X - C_3)$. This yields

$$||(r_1(X - C_1), r_2(X - C_2)) - x(r_3(X - C_3))||_2 \leq \epsilon(r_3(X - C_3)). \tag{1.14}$$

If the R, x and ϵ are known, then this is a convex set, more precisely a second-order cone. Consequently, the reprojection error is a quasiconvex function of X and C. Moreover, if we consider any number of 3D points and their projections in any number of views, then the maximum reprojection error is still a quasiconvex function of the variables. This problem is known as structure from motion with known rotations and will be important in Chapter 6.

To handle outliers in this framework [24] suggests using auxiliary variables, which results in

$$||(r_1(X - C_1), r_2(X - C_2)) - x(r_3(X - C_3))||_2 \leq \epsilon(r_3(X - C_3)) + s, \tag{1.15}$$

where there is one $s \geq 0$ for every image point. The extra variables will allow reprojection errors to be larger than the prescribed threshold ϵ. Ideally, one would like to minimize

the number of non-zero s_i but this is difficult so [24] use the L_1 relaxation and minimize $\sum_i s_i$ subject to $s_i \geq 0$ for all i. This allows us to handle outliers but note that there is no longer any guarantee of finding an optimal solution.

For other examples of quasiconvex vision problems, see [41, 50, 49]. Moreover, [40] shows that by combining quasiconvex optimization with a parameter search, optimal solutions for calibrated camera pose and relative orientation can also be obtained.

1.3.2 Graphs

Graphs have turned out to be a powerful tool for computer vision. This section presents some elementary theory that will be used later in the thesis.

A graph is an abstract object consisting of a set of vertices V and a set of edges E, where each edge is an unordered pair of vertices. Figure 1.2 shows how a graph can be depicted with a dot for each vertex and lines representing the edges. Sometimes E is taken to be a multiset rather than a set, meaning that there can be multiple edges connecting the same vertices. To distinguish these two kinds of graphs one talks of simple graphs and multigraphs. Another generalization is allowing edges that start and end in the same vertex. Such edges are called loops.

A *path* is a sequence of vertices such that for each vertex there is an edge to the next one. If there is a path between any pair of vertices, then the graph is *connected*. A *tree* is a connected graph that contains no cycles. Alternatively, a tree is a graph such that for any pair of vertices there is exactly one simple path connecting them.

A *vertex cover* for a graph is a subset S of the vertices such that every edge has at least one endpoint in S. In Chapter 2 we will use graphs to analyze correspondence problems. Each point-to-point correspondence is encoded with a vertex in the graph and edges are added between inconsistent correspondences. To get a consistent set of correspondences, we need to find a vertex cover for this graph and remove those correspondences.

If S is a vertex cover then $V \setminus S$ is an independent set, meaning that no edge has both endpoints in $V \setminus S$. Hence finding a minimum vertex cover is just as hard as finding a maximum independent set. There is a similar connection to the problem of finding a largest complete subgraph. To see the connection, we consider a complete graph such that each edge is either red or blue. Now, if S is an independent set for the red graph, then clearly it is a complete subgraph of the blue graph. Hence an efficient algorithm for one of these problems could also solve the other two.

Given a graph G and a natural number k, is there a vertex cover of size k? This is the decision version of the vertex cover problem and it belongs to a class of problems called NP-complete. NP is a term from computational complexity theory that stands for non-deterministic polynomial time. That a decision problem is NP can be defined as follows. An instance of the problem for which the answer is yes has a proof that can be verified in polynomial time. For the vertex cover problem this proof simply consists of a vertex cover of size k and the time it takes to verify that this is indeed a vertex cover is

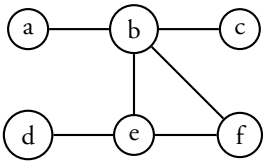

Figure 1.2: Example graph.

linear in the number of edges of the graph. Hence the decision variant of vertex cover is in NP. Moreover, the vertex cover problem is NP-hard. This means that any problem in NP can be formulated as a vertex cover problem. The term is motivated by the fact that an NP-hard problem must be at least as difficult as any problem in NP. A problem that is both NP and NP-hard is NP-complete.

So far no one has found a polynomial-time algorithm to solve NP-hard problems. Hence the term is often used to indicate that a problem is computationally heavy. Still it is possible to attack this type of problems. One way is to use approximation algorithms. In general these will not solve the original problem but can often produce a good approximation. For example, there is a simple linear-time approximation for the minimum vertex cover problem. It is obtained by repeatedly picking a random edge in the graph and then placing both its endpoints in the covering. Take for example the graph in Figure 1.2. We start by picking the edge between vertices a and b, add these vertices to our solution set and remove them from the graph. In the next step we pick the edge between d and e and remove these two vertices. We have found a vertex cover $\{a, b, d, e\}$ and it can be shown that the size of this vertex cover is at most twice as large as the minimum vertex cover. Hence in the example we have obtained a lower bound of 2 on the size of the minimum vertex cover. In this case the bound happens to be tight; see Algorithm 1.

Algorithm 1 Factor-2 approximation

Produces a vertex cover S at most twice as big as the minimum vertex cover.

Initialize $S = \emptyset$.
Repeat until no edges remain:
  Pick a random edge e from the graph.
  Add the two vertices connected to e to S.
  Remove the two vertices and all connecting edges from the graph.

Since the vertex cover will be useful later on we also consider an exact algorithm which is based on the following observation. Given a vertex v having at least one edge we can split the problem in the following way. Either v belongs to the minimum vertex cover or any vertex with an edge to v must lie in the minimum vertex cover.

Algorithm 2 uses this type of splitting iteratively until the solution is found. Each splitting creates new branches which can either be discarded using bounding techniques or are in turn splitted again. Figure 1.3 shows an example of branching.

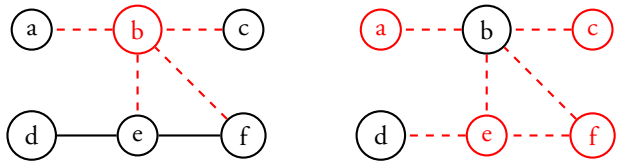

Figure 1.3: The two subproblems created by branching on the b node of the graph in Figure 1.2. Red nodes are those assumed to belong to the vertex cover and the red dashed lines are the covered edges.

Algorithm 2 Exact Algorithm

Let K be an upper bound for the size of the minimum vertex cover.
Initialize the queue with $S = \emptyset$.
Iterate until the queue is empty:
  Let S denote the vertex set currently at the top of the queue.
  If S is a vertex cover and $|S| < K$:
    Set $K = |S|$ and save $S_{best} = S$.
  Else:
    Find a lower bound for the size of the smallest vertex cover containing S.
    If this bound $< K$:
      Pick a vertex v with at least one edge which is not
      covered by S and add the following sets to the end of the queue.
      (i) $S \bigcup \{v\}$
      (ii) $S \bigcup \{$all vertices having an edge to v$\}$
  Remove S from the queue.

To compute the lower bounds required in Algorithm 2, we first remove all vertices in S and all edges to any of these vertices. This leaves a smaller graph and clearly, if V is a vertex cover for this reduced graph $S \bigcup V$ is a vertex cover for the original graph. Similarly, if B is a lower bound on the size of any vertex cover for the reduced graph, $|S| + B$ is the required lower bound for any vertex cover containing S. The factor-2 approximation of Algorithm 1 can be used to provide such a bound, B.

Weighted Graphs. We get a weighted graph from a graph by associating a real number with each edge. We will use this number to encode how reliable different links in the camera graph is.

A subgraph of a graph (V, E) is a *spanning tree* if it is a tree that contains all vertices in V. It is easy to prove that every connected graph has a spanning tree. A minimum spanning tree is a spanning tree such that the sum of edge weights minimized. Compared to the vertex cover problem, it is easy to find a minimum spanning tree. It can be done in polynomial time using a greedy algorithm; see [11].

Hypergraphs. The notion of hypergraphs generalizes graphs by allowing an edge to connect any number of vertices. Hence an edge in a hypergraph is an arbitrary subset of the vertices.

Many of the basic concepts defined for graphs are naturally generalizable to hypergraphs. Thus a vertex cover for a hypergraph $G = (V, E)$ is a subset $S \subset V$ such that every generalized edge has at least one vertex in S.

Again there is a number of ways to state essentially the same problem and it turns out that vertex cover for hypergraphs is equivalent to the more well-known set cover problem. Consider a set U and family F of subsets of U. A set cover is a subfamily $S \subset F$ such that its union is the equal to U.

1.3.3 Rotations and Quaternions

Later on some knowledge of the characteristics of the group of three-dimensional rotations, often referred to as $SO(3)$. The usual metric on $SO(3)$ can be defined by the following equation,

$$d(R, S) = \max_{x \in \mathbb{R}^3 \setminus \{0\}} \angle(x, S^T R x), \tag{1.16}$$

where $\angle(x, y)$ denotes the angle between vectors x and y taking values in $[0, \pi]$. From this definition it is easy to derive that if $\alpha \in [-\pi, \pi]$ and R is a rotation an angle α about some axis then $d(R, I) = |\alpha|$.

Of course, the standard representation of a rotation is as an orthogonal 3×3 matrix with determinant 1. Another useful representation is quaternions. Quaternions generalize complex numbers to four-dimensional numbers

$$q = a + bi + cj + dk, \tag{1.17}$$

where $(a, b, c, d) \in \mathbb{R}^4$ and

$$i^2 = j^2 = k^2 = ijk = -1. \tag{1.18}$$

Like with complex numbers we define the conjugate of a quaternion by switching sign for the complex parts

$$q' = a - bi - cj - dk. \tag{1.19}$$

Three-dimensional vectors can be described as quaternions with zero real part, often called pure imaginary quaternions. Let r and v be vectors written on this form. It is

straightforward to derive that quaternion multiplication of r and v yields

$$rv = -\langle r, v\rangle + r \times v, \tag{1.20}$$

where $\langle r, v\rangle$ is the ordinary scalar product and $r \times v$ is the ordinary cross product but represented with a quaternion. In fact, quaternion multiplication is the origin of scalar products and cross products.

The following theorem shows how quaternions can be used to represent rotations.

Theorem 1. *Let r be a unit length pure imaginary quaternion and*

$$q = \cos\frac{\alpha}{2} + r\sin\frac{\alpha}{2}. \tag{1.21}$$

If u is another pure imaginary quaternion representing a vector, then quq' is u rotated an angle α around the axis r.

Proof. We start by writing as $u = ar + bv$, where v is a unit vector that is perpendicular to r and $a, b \in \mathbb{R}$. Since

$$quq' = a(qrq') + b(qvq'), \tag{1.22}$$

it is sufficient to consider qrq' and qvq'. Using (1.20) we get

$$\begin{aligned} qrq' = (r\cos\frac{\alpha}{2} + \sin\frac{\alpha}{2})(\cos\frac{\alpha}{2} - r\sin\frac{\alpha}{2}) = \\ r\cos^2\frac{\alpha}{2} - \cos\frac{\alpha}{2}\sin\frac{\alpha}{2} + \sin\frac{\alpha}{2}\cos\frac{\alpha}{2} + r\sin^2\frac{\alpha}{2} = r \end{aligned} \tag{1.23}$$

so the part parallel to r will not be changed. Moreover,

$$\begin{aligned} qvq' = (v\cos\frac{\alpha}{2} + r\times v\sin\frac{\alpha}{2})q' = v\cos^2\frac{\alpha}{2} + r\times v\sin\frac{\alpha}{2}\cos\frac{\alpha}{2} \\ - v\times r\cos\frac{\alpha}{2}\sin\frac{\alpha}{2} - v\sin^2\frac{\alpha}{2} = v\cos\alpha + r\times v\sin\alpha, \end{aligned} \tag{1.24}$$

so the part perpendicular to r is indeed rotated an angle α. □

Remark. *One consequence of this theorem is that if q is a rotation quaternion then $-q$ represents the same rotation. We will se in Chapter 6 that this ambiguity can pose a significant problem.*

The quaternion representation also provides a convenient way to compute the standard metric. Let $\langle x, y\rangle$ denote scalar multiplication with quaternions seen as 4-vectors. If $\langle p, q\rangle \geq 0$ we have

$$d(R_p, R_q) = d(R_p R_q^T, I) = d(R_{pq'}, I), \tag{1.25}$$

but this is just the rotation angle of the $R_{pq'}$ and we know from (1.21) that the real part of pq' is the cosine of half the rotation angle. Hence

$$d(R_p, R_q) = 2 \arccos \mathrm{Re}(pq') \tag{1.26}$$

and by (1.20), $\mathrm{Re}(pq') = \langle p, q \rangle$. Thus,

$$d(R_p, R_q) = 2 \arccos \langle p, q \rangle. \tag{1.27}$$

1.3.4 Spherical Trigonometry

In Chapter 4, we will use some simple results from spherical geometry. Since this subject is not quite mainstream mathematics, this section presents the required theorems. At least here, spherical geometry refers to the geometry of S^2 being the set of unit 3-vectors or the set of points on the unit sphere. If two points are given by unit vectors u, v then the distance along the sphere between these points is equal to $\angle(u, v)$.

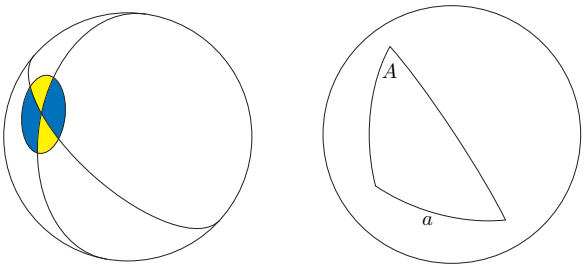

Figure 1.4: Left: Two great circles form two pairs of similar angles at their intersection. Right: We use lower case letters for side lengths and upper case letters for the opposing corner angle.

We define a great circle as the intersection of S^2 and a plane through the origin. When two great circles intersect they form two pairs angles as shown in Figure 1.4. These angles can be computed as the angle between the planes defining the great circles.

The results that will be used later on concern spherical triangles. A spherical triangle is formed by three great circles. We will refer to the side lengths of the triangle using lower case letters and the corner angles using upper case letters; see Figure 1.4 for an example.

Let u, v and w be pure imaginary quaternions representing the corners of a spherical triangle. The following formula will be useful,

$$\begin{aligned} v\bar{w} = v\bar{u}u\bar{w} &= -\overline{(uv)}(uw) = (\langle u, v \rangle + u \times v)(\langle u, w \rangle - u \times w) \\ &= \langle u, v \rangle \langle u, w \rangle + \langle u \times v, u \times w \rangle \\ &- \langle u, v \rangle u \times w + \langle u, w \rangle u \times v - (u \times v) \times (u \times w). \end{aligned} \tag{1.28}$$

Theorem 2. *Consider a spherical triangle on the unit sphere S^2 with corners in u, v and w. Let $a = \angle(u, v)$, $b = \angle(u, w)$, $c = \angle(v, w)$ be the sides of the triangle and A, B, C be the corner angles. Then,*

$$\cos c = \cos a \cos b + \sin a \sin b \cos C. \tag{1.29}$$

Proof. Let u, v and w be quaternions representing the three corners of the triangle. By (1.28) we have

$$\begin{aligned}\cos c = \langle v, w\rangle = Re(v\bar{w}) &= \langle u, v\rangle\langle u, w\rangle + \langle u \times v, u \times w\rangle \\ &= \cos a \cos b + \sin a \sin b \cos C.\end{aligned}$$

□

Theorem 3. *The angles in a spherical triangle satisfy,*

$$\frac{\sin A}{\sin a} = \frac{\sin B}{\sin b} = \frac{\sin C}{\sin c}. \tag{1.30}$$

Proof. Note that (1.20) implies,

$$Re(uvw) = -Re(uwv) = Re(wuv) = -Re(wvu) = Re(vwu). \tag{1.31}$$

But if we use (1.28) on vw we get

$$Re(uvw) = \langle u, (u \times v) \times (u \times w)\rangle = \sin a \sin b \sin C \tag{1.32}$$

and similarly for $Re(wuv)$ and $Re(vwu)$. Hence

$$\sin a \sin b \sin C = \sin a \sin B \sin c = \sin A \sin b \sin c. \tag{1.33}$$

□

1.4 This Thesis

This section gives a brief overview of this thesis. To comply with the rules for a doctoral thesis my own contributions to the different chapters are also described.

Chapter 2 shows how some geometric problems can be attacked using graph techniques. Constraints on pairs of correspondences are used to set up a vertex cover problem. By solving one or a sequence of such vertex cover problems it is possible to minimize the number of outliers. The idea of using vertex cover to solve geometric problems is not new to computer vision, but it has not been shown before how these techniques can be combined with parameter search to find optimal solutions, in terms of the number of outliers.

My own contributions were both the idea and the theorems concerning sufficiency and convergence.

Chapter 3 shows how the ideas from Chapter 2 can be used in practice. Results for registration as well as camera pose estimation are presented. The coding and experiments was joint work with Klas Josephson. The results show how hard bounds on the optimal solutions can be produced even when the residual functions are non-convex. For registration the proposed algorithm represents a significant speed-up compared to earlier optimal methods. In the case of calibrated camera pose estimation the proposed method is, to the best of my knowledge, the first published method that can produce guaranteed optimal solutions with respect to the number of outliers.

Chapter 4 is concerned with relative orientation estimation in presence of outliers. Two new methods are presented. The first one is based on parameter search and is guaranteed to minimize the number of outliers. As far as I know, this is the first published algorithm that is guaranteed to find the optimal relative orientation with respect to the number of outliers. The other method is a brute-force algorithm with complexity that is independent of the outlier rate. Both algorithms were originally my ideas and I also did the theoretic derivations and the basic coding. The motion segmentation algorithm was developed together with Fredrik Kahl and Fangyuan Jiang worked on the spatial regularization and implemented the brute-force algorithm for a graphics card.

Chapter 5 discusses outlier problems in cases where the residual functions are quasi-convex. There are mainly two contributions. We provide an algorithm with time complexity that is polynomial in the number of correspondences. Using the same principles, we also show how to verify that a given solution is the global optimum. or find a better solution. My contributions was the verification scheme and Theorem 21.

In *Chapter 6* a complete system for structure from motion estimation is proposed. Often this problem has been solved in a sequential manner, but recently there has been large interest in non-sequential structure from motion. The results in this thesis show the advantages of a non-sequential approach based on graph methods as well as convex optimization. In this case the main algorithm was developed together with Fredrik Kahl and Carl Olsson. The results concerning estimations with short baseline was joint work with Carl Olsson whereas the results on cycles and consistency was mainly my work.

Chapter 7 is also concerned with the geometry of multiple views, but for a special type of imaging devices called 1D cameras. It is shown how reconstruction problems in 1D vision can be solved optimally with respect to the L_∞ norm of the reprojection errors. I worked on some of the proofs and did about half of the experiments. Moreover, I worked out how to find initial constraints on the orientations.

Part I

Estimation with Outliers

Chapter 2

Outliers and Vertex Cover

The following two chapters deal with geometric estimation in presence of outliers. This problem is addressed as an optimization task, seeking a transformation that minimizes the number of outliers. In this chapter, it is shown how constraints on pairs of correspondences induces a graph problem, which is solved in order to find the optimal correspondences. The ideas are general enough for a wide range of application problem but the focus lies on calibrated camera pose estimation and 3D-3D registration. The second chapter shows how these techniques can be applied to 3D-3D registration and camera pose estimation.

2.1 Introduction

Establishing point-to-point correspondences between images or point sets is a common problem in both computer vision and photogrammetry, for example, in object recognition, 3D reconstructions, image-based localization or medical image alignment. Naturally, due to its importance, many different solutions have been proposed. Most methods build on some type of appearance-based feature matching; see Section 1.2. However matching local features based on appearance only is difficult and errors are frequent. Therefore it is generally necessary to use geometry to remove incorrect matches. On this point prior work is mainly based on heuristic methods such as RANSAC or EM-like algorithms to solve this task and hence there is a risk of getting trapped in a poor, local solution.

We will focus on two important problems from multiple view geometry being calibrated camera pose estimation and 3D-3D registration. The key idea is to consider point correspondences and check whether pairs of such correspondences are consistent or not. Seeking a large set of pairwise consistent correspondences leads us to the vertex cover problem of graph theory; cf. Section 1.3.2. The bad news is that this problem is known to be NP-hard. The good news is that we can still solve instances of the problem for quite large data sets using a combination of branch-and-bound and approximation algorithms.

The chapter is structured as follows. In the next section, we review related work and compare to our approach. In Section 2.3, a mathematical problem formulation is given together with an introduction to the use of pairwise constraints. We first apply the frame-

work to a simple example, being the estimation of planar rotations, in Section 2.4. In the following section, the slightly more advanced problem of estimating 3D rotations is analyzed. Finally, the framework is generalized to incorporate more general transformations, in particular camera pose estimation.

2.2 Related Work

Graph matching. The idea of using pairwise constraints in combination with vertex cover for finding mutually consistent correspondences is not new in the vision literature. The earliest work we have found is [10] where it is used for 2D part location. Pairwise constraints are also discussed in [36]. In [86], an association graph is built for 2D matching which results in a maximum clique problem. Similarly, the stereo correspondence problem was formulated as a maximum clique problem in [43]. This is an alternative formulation of the same graph problem as ours. In [53], the registration problem was formulated in a graph setting, but solved using a non-optimal spectral technique. Other graph matching formulations include [16, 84, 93] where the objective is to match correspondences such that pairwise distances within the two point sets are similar. This leads to a purely combinatorial problem equivalent to the quadratic assignment problem. A similar approach is pursued in [60]. One advantage (besides being purely combinatorial) is that such approaches can be used for solving both rigid and non-rigid registration problems. However, the underlying transformation mapping one point set to the other is ignored.

Camera pose estimation. Camera pose estimation is a well studied problem in both computer vision and photogrammetry [3, 42] and one of the earliest references on the topic dates back to 1841 [37]. There are several heuristic methods with no guarantee of optimality such as [25, 44, 67] where RANSAC is perhaps the most popular one [30]. In multiple view geometry, the camera pose problem is often solved with DLT [42], but the method optimizes an algebraic cost function and cannot handle outliers among the correspondences. Another class of methods is based on subdividing transformation space and aiming for global solutions such as [46] using an affine camera model. The problem is important on its own as a core problem within the field of multiple view geometry, and moreover, it appears as a subproblem for many other vision applications, like motion segmentation [67], object recognition [17, 44, 46], and more generally model matching and fitting problems, see [14, 17, 25, 44, 46, 47, 67]. Yet, previous approaches for solving the camera pose problem have not been able to solve the problem in the presence of outliers with a guarantee of global optimality.

The first globally optimal algorithm for this problem using a geometric error norm was presented in [68]. They also investigate the problem of local minima for the pose problem and show that this is indeed a real problem for small numbers of correspondences. For larger numbers of correspondence pairs or small noise levels, the risk of

getting trapped in a local minimum is small. This is our experience as well. In their work the L_2 norm of the reprojection errors is used, but the algorithm converges rather slowly. In [40], the authors choose to work with the L_∞ norm instead and present a more efficient algorithm that finds the optimum by searching in the space of rotations and solving a series of second order cone programs. The reported execution times for both these algorithms are in the order of several minutes, while the proposed algorithm performs the same task within a few seconds. More importantly, the new algorithm can handle outliers. If some correspondences are incorrect, then fitting a solution to all data will give very bad results.

3D-3D registration. Given correct correspondences, it is straightforward to estimate the transformation; see [42]. On the other hand, given an estimate of the transformation, it is easy to determine likely correspondences. So, a natural idea is to use an alternating minimization procedure, and there are many such algorithms in the literature (e.g., [33]), the most famous one being Iterative closest point,ICP, [8]. However, these approaches require a good initial estimate of the transformation and still there is no guarantee of getting a reasonable solution, especially when there are lots of outliers. Other non-optimal approaches include the Hough transform, geometric hashing and hypothesize-and-test algorithms like RANSAC [30]. Recently, in [56], the problem was solved globally using branch and bound in rotation space. The method requires that all points are matched and that the translation component is given, which are severe limitations so this is effectively not solving the complete problem. Even though, we have compared our algorithm to this method in the experimental section.

In [14], a branch-and-bound algorithm over rigid transformations in the 2D plane is proposed. This works well as the transformations have only three degrees of freedom, but the approach becomes computationally infeasible for rigid transformations in 3D space (which have six degress of freedom).

Outliers. A method for detecting outliers for L_∞ solutions was given in [74] but it only applies to quasiconvex problems. The uncalibrated pose problem is quasiconvex, but the calibrated pose problem is not. Further, the strategy in [74] for removing outliers is rather crude - all measurements that are in the support set are discarded. Hence, inlier correspondences may also be removed. Possible solutions to this problem are given in [54] and in Chapter 5, but they are restricted to quasiconvex problems. Another well-known approach for estimating camera pose in cases where it is hard to find correct correspondences is to apply RANSAC-type algorithms [30]. Such algorithms leave no guarantees of optimality.

Our work is also related to the rich body of literature on matching problems; see [14, 17, 22, 25, 44, 46, 47, 60, 61, 67]. Many of these algorithms are quite sophisticated and have been an inspiration to our work. However, some do not guarantee any kind of optimality [22, 25, 67], while others use simplified camera models like affine approxi-

mations [14, 44, 46, 47, 61]. Other drawbacks of these methods include simplified cost functions not based on reprojection errors.

Unlike many of the above mentioned methods, the proposed framework makes it possible to generate multiple correspondence hypotheses for a single point, but in the final solution, it is guaranteed that no conflicting hypotheses are included. Another advantage is that it is possible to use extra information from the feature extractor, for example, the orientation.

2.3 Preliminaries

2.3.1 Formulation

The different geometric problems that we consider in this chapter have much in common. Given two point sets, we are looking for a transformation mapping one set to the other. To find this transformation we somehow need to determine the correct point-to-point correspondences between the sets. A correspondence is an ordered pair (m, μ) of natural numbers connecting the mth point of the first point set with the μth point of the second. We begin with a set of hypothetical correspondences. Often, these have been generated by some appearance-based point matching algorithm, but we can also choose to consider all possible correspondences. Whatever the case, it is likely that a large portion of these correspondences are incorrect.

We say that a set of correspondences is one-to-one if the same point does not occur in more than one correspondence. The following definition will also be useful.

Definition 1. *Consider two point sets, $\{x_i\}$ and $\{y_j\}$, a set of correspondences S and a transformation T. If S is one-to-one and*

$$d(T(x_m), y_\mu) \leq \epsilon, \tag{2.1}$$

for all $(m, \mu) \in S$, we say that S is ϵ-consistent with T.

To be more concrete, for 3D-3D registration, T is a rigid transformation and d is the euclidean distance. For camera pose, T is a rigid transformation followed by a perspective mapping and d is the angular reprojection error, that is, the angular difference between the measured image point and the reprojected 3D point.

Having many hypothetical correspondences, pure chance can lead to small sets being consistent with some transformation, but large consistent sets are extremely unlikely to appear by chance, so they are somehow reflecting the geometry of the problem. Hence it is natural to seek large consistent sets or equivalently, try to minimize the number of outliers.

Problem 1. *Given two point sets and a set, H, of hypothetical correspondences between the point sets, find a subset $I \subset H$ minimizing $|H \setminus I|$ subject to I being ϵ-consistent with some transformation T of the specified class.*

This way of formulating the problem was chosen to fit with the *minimum* vertex cover problem. Correspondences in I will be referred to as inliers and correspondences in $H \setminus I$ as outliers.

Note that the popular RANSAC method [30] is one way of attacking Problem 1, but even with exhaustive testing of all minimal subsets, the method will not necessarily obtain the optimal solution.

Pairwise Constraints and Vertex Cover. To approach Problem 1 we first consider a pair of correspondences. Our aim is to determine whether these two correspondences can both belong to the solution of Problem 1. We know that this cannot be the case if they match the same point, but it can also be clear from the geometry. Consider for example 3D-3D registration with fixed scale. Assume that a pair of correspondences matches two close points to two distant ones. Clearly these two correspondences cannot be consistent with one rigid transformation: They are geometrically inconsistent. To find the inlier set I of Problem 1 we have to remove correspondences until no pairwise inconsistencies remain. For optimality, we want to remove as few correspondences as possible. This is equivalent to finding a minimum *vertex cover* of a graph having all hypothetical correspondences as vertices, and edges connecting inconsistent ones.

2.4 Example: Planar Rotations

To introduce the approach of pairwise constraints, we first look at the rather elementary problem of estimating a rotation between points on the unit circle. The problem is presented merely as an example, but it does have practical interest when working with one-dimensional cameras; see Chapter 7.

To avoid ambiguities, we use unit-length complex numbers to represent the points. This way, a rotation can be performed as multiplication with a unit-length complex number r.

Problem 2. *Assume that we are given two sets of unit complex numbers, $\{x_i\}$ and $\{y_i\}$, and a set of hypothetical correspondences H. Find a subset $I \subset H$ minimizing $|H \setminus I|$ subject to I being ϵ-consistent with some rotation r.*

Consistency here is with respect to distances along the unit circle. Using complex numbers this can be computed as

$$d(rx_m, y_\mu) = |\arg(\bar{r}\bar{x}_m y_\mu)|, \tag{2.2}$$

where the bar implies complex conjugation and the arg function takes values in $(-\pi, \pi]$.

Now assume that we have found a set of correspondences, that is consistent in the sense of Definition 1 and let (m, μ) and (n, ν) be two of these correspondences. According to the definition and (2.2) these correspondences satisfy

$$d(\bar{x}_m y_\mu, \bar{x}_n y_\nu) = |\arg(x_m \bar{y}_\mu \bar{x}_n y_\nu)| = |\arg(r x_m \bar{y}_\mu \bar{r} \bar{x}_n y_\nu)| \tag{2.3}$$

$$\leq |\arg(rx_m\bar{y}_\mu)| + |\arg(\bar{r}\bar{x}_n y_\nu)| = d(rx_m, y_\mu) + d(rx_n, y_\nu) < 2\epsilon. \tag{2.4}$$

This means that we have found a necessary constraint for a pair of correspondences. It turns out that this constraint is sufficient in the following sense.

Theorem 4. *If $0 < \epsilon < \pi/3$ and S be a set of correspondences such that*

$$d(\bar{x}_m y_\mu, \bar{x}_n y_\nu) \leq 2\epsilon \tag{2.5}$$

is satisfied for every pair of correspondences in S. Then there exists a complex number r such that

$$d(rx_m, y_\mu) \leq \epsilon \tag{2.6}$$

for any $(m, \mu) \in S$.

Proof. First note that (2.6) can be rewritten as

$$d(r, \bar{x}_m y_\mu) \leq \epsilon \tag{2.7}$$

and that it restricts r to an interval on the unit circle. Our first claim is that an arbitrary pair of such intervals have a non-empty intersection. To see this consider two intervals generated by correspondences (m, μ) and (n, ν). Let ρ be the midpoint between $\bar{x}_m y_\mu$ and $\bar{x}_n y_\nu$. By (2.5) this point clearly satisfies

$$d(\rho, \bar{x}_m y_\mu) = d(\rho, \bar{x}_n y_\nu) \leq 2\epsilon/2 = \epsilon, \tag{2.8}$$

so (2.7) is satisfied for these two correspondences. Thus ρ lies in both intervals. To complete the proof, we need the following lemma.

Lemma 5. *Consider a set of intervals I_k on the unit circle such that $|I_k| < 2\pi/3$ for all k. If the intersection $I_j \bigcap I_k$ is non-empty for any pair (j, k), then $\bigcap_k I_k$ is non-empty as well.*

Proof. Pick any interval I_k and let p be the midpoint of this interval. Note that the distance between I_k and the point $p + \pi$ is larger than $2\pi/3$ and thus $p + \pi$ lies in no interval. Thus we can cut the unit circle at $p + \pi$ and map it to the real line $\mathbb{R}$. The theorem now follows from Helly's theorem [26]. □

Since $\epsilon < \pi/3$, it follows that $|I_k| < 2\pi/3$. This completes the proof of the theorem. □

Theorem 4 implies that we can solve Problem 2 by finding the largest set of correspondences that is one-to-one and such that each pair satisfies (2.5). To do this we consider a graph having a vertex for every node and an edge between two vertices if

(i) The correspondences have the first or second index in common.

(ii) The correspondences does not satisfy (2.5).

Problem 2 is now equivalent to the minimum vertex cover problem for this graph. An exact algorithm for this problem was given in Section 1.3.2; see Algorithm 2. It is worth to point out that we should not compute the full graph and then find a minimum vertex cover. Instead edges are determined when they are required by the vertex cover algorithm. More tricks for speeding up Algorithm 2 will be discussed in Chapter 3.

2.5 3D Rotations

Things get more complicated when we want estimate a 3D rotation between points on the unit sphere in $\mathbb{R}^3$. This problem has its own application when stitching images together to panoramas, but it will also be important when we discuss camera pose estimation. Problem 3 gives a more precise description. Note that this time ϵ-consistency is with respect to distances along the sphere.

Problem 3. *Given two sets of unit 3-vectors, $\{x_i\}$ and $\{y_j\}$, and a set of hypothetical correspondences H, find a subset $I \subset H$ minimizing the number of outliers, $|H \setminus I|$, subject to I being ϵ-consistent with some orthogonal transformation R.*

An orthogonal tranformation on $\mathbb{R}^3$ is either a rotation, a reflection or a rotation-reflection; see [4]. Hence we will detect reflections as well as rotations. Since orthogonal transformations preserve angles any correspondences (m, μ) and (n, ν) in the inlier set I will satisfy

$$\left|\angle(x_m, x_n) - \angle(y_\mu, y_\nu)\right| \leq 2\epsilon. \tag{2.9}$$

These are the constraints that we will use to set up the vertex cover problem. Unfortunately, a set of correpondences can be consistent with respect to these pairwise constraints without being ϵ-consistent with any rotation. The next sections will be concerned with this problem and how to get around it, but first we look at these constraints in the exact case.

Theorem 6. *Consider two sets of unit 3-vectors ordered such that x_i corresponds to y_i. If for all (i, j) the scalar products satisfy $\langle x_i, x_j \rangle = \langle y_i, y_j \rangle$, then there exists an orthogonal transformation R such that*

$$Rx_i = y_i \text{ for all } i. \tag{2.10}$$

Proof. Assume that $\{x_1, x_2, x_3\}$ is a basis for $\mathbb{R}^3$. The special case when no such triplet exists can be handled similarly. Let R be a linear mapping such that $Rx_i = y_i$ for $i = 1, 2, 3$. Then for $j = 1, 2, 3$

$$x_i^T x_j = y_i^T y_j = (Rx_i)^T Rx_j = x_i^T (R^T R) x_j \quad \text{for } i = 1, 2, 3 \tag{2.11}$$

so $R^T Rx_j = x_j$. Since this holds for $j = 1, 2, 3$ and $\{x_1, x_2, x_3\}$ is a basis, we can conclude that $R^T R = I$ so R is an orthogonal mapping. Moreover

$$(Rx_i - y_i)^T y_k = (Rx_i)^T (Rx_k) - y_i^T y_k = x_i^T x_k - y_i^T y_k = 0 \tag{2.12}$$

for $k = 1, 2, 3$. Since $\{y_1, y_2, y_3\}$ is a basis this implies $Rx_i = y_i$ so (2.10) holds. □

2.5.1 Splitting Correspondences

As mentioned the constraints in (2.9) are generally not sufficient. This means that finding a minimum vertex cover does not guarantee finding a solution to the original problem. To remedy this, we can split correspondences to refine our search.

We note that a correspondence (m, μ) is in a way the hypothesis that for the sought rotation R the rotated point Rx_m lies in the set $B = \{p : \angle(p, y_\mu) < \epsilon\}$. To make this clearer, we write a correspondence as a triplet (m, μ, B). Splitting the set B into smaller sets $B = \bigcup B_k$, allows us to generate new correspondences (m, μ, B_k) that can be handled just like the original ones; see Figure 2.1. Algorithm 3 shows how this splitting technique can be used. For each repetition of this algorithm, we get an updated set of hypothetical correspondences.

Definition 2. *A set of correspondences S is consistent with a transformation T if S is one-to-one and for each $(m, \mu, B) \in S$ it holds that $Tx_m \in B$.*

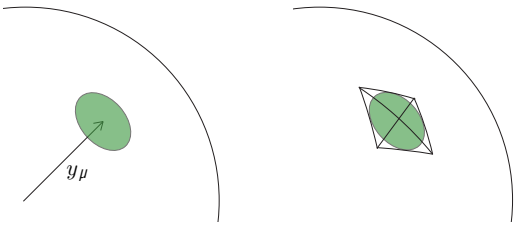

Figure 2.1: The images show a part of the unit sphere. Originally Rx_m is constrained to the green circle around y_μ shown on the left. In the refinement step this circle is divided as shown on the right, generating new, more precise hypotheses. Working with spherical triangles simplifies subsequent division steps.

Algorithm 3 Splitting correspondences

Given a set of hypothetical correspondences H_k.
Split all correspondences in H_k as described in the text.
Let H_{k+1} be the new correspondences obtained.
Set up a new consistency graph, G_{k+1}.
Compute a minimum vertex cover for this graph.

Most importantly this splitting technique allows us to sharpen the constraint in (2.9). The new constraint for correspondences (m, μ, M) and (n, ν, N) is

$$\min_{p \in M, q \in N} |\angle(p, q) - \angle(x_m, x_n)| = 0. \tag{2.13}$$

We define the correspondence uncertainty σ_c as

$$\sigma_c = \max_{p \in M, q \in N} |\angle(p, q) - \angle(x_m, x_n)|, \tag{2.14}$$

given that (2.13) holds. Clearly, this is bounded by

$$\sigma_c \leq \max_{p \in M, q \in N} \angle(p, q) - \min_{p \in M, q \in N} \angle(p, q) \leq \operatorname{diam}(M) + \operatorname{diam}(N). \tag{2.15}$$

In fact it will not be necessary to evaluate (2.13) exactly as long as the correspondence uncertainty $\sigma_c \to 0$ when $\operatorname{diam}(M) + \operatorname{diam}(N) \to 0$.

Let H_k denote the correspondences obtained after k splits and let G_k be the graph for these correspondences. Now consider a solution to Problem 3, that is a subset I of the original correspondences H such that I is ϵ-consistent with an orthogonal transformation R and $|H \setminus I|$ is minimal. For each correspondence $(m, \mu) \in I$, there is a correspondence in $(m, \mu, M) \in H_k$ such that $Rx_m \in M$. Let I_k denote the set of these correspondences. Clearly I_k is consistent according to Definition 2. Consequently, $H_k \setminus I_k$ is a vertex cover for G_k and thus a minimum vertex cover for G_k produces a lower bound on the number of outliers of in the original problem. We will soon see that after a finite number of splits this bound is tight.

Definition 3. *If a correspondence $a \in H_k$ was created by splitting of $b \in H$, then b is an ancestor of a and a is a descendant of b.*

Theorem 7. *After a finite number of repetitions of Algorithm 3, every minimum vertex cover corresponds to a solution to Problem 3, or more precisely: There exists a natural number K, such that if S is a vertex cover for G_K and $I \subset H$ is the set of ancestors to $H_K \setminus S$. Then I is an optimal solution to Problem 3.*

Proof. Let κ be the (unknown) size of a correspondence set solving Problem 3. Consider all subsets of H having size at least κ, which are not ϵ-consistent with any orthogonal transformation. Let $A_1, A_2, \ldots$ denote these sets.

Lemma 8. *Consider a set $A_i \subset H$ that is not ϵ-consistent with any orthogonal transformation R. Then there exists a natural number K_i such that after K_i repetitions of Algorithm 3, there is no set of $|A_i|$ descendants of A_i that is pairwise consistent.*

Proof. To simplify the notation we assume that y_i corresponds to x_i. Consider the function

$$h(p_1, p_2, \ldots) = \max_{i,j} |\angle(x_i, x_j) - \angle(p_i, p_j)| \tag{2.16}$$

on the set $\{(p_1, p_2, \ldots) : \angle(p_k, y_k) \leq \epsilon\,, \forall\, k\}$. This is a continuous function, defined on a compact set. Thus it takes a minimum value $\delta \geq 0$ in this set.

Assume that $\delta = 0$. Let $(p_1, p_2, \ldots)$ be the point where this minimum is obtained. Then $\angle(x_i, x_j) = \angle(p_i, p_j)$ for all pairs (i, j) and according to Theorem 2 there exists an orthogonal transformation such that $Rx_i = p_i$, but then $\angle(Rx_i, y_i) \leq \epsilon$, which contradicts the assumption, so $\delta > 0$.

Now, assume that the correspondences have been split a number of times such that for any correspondence (m, μ, M), $\mathrm{diam}(M) < \delta/2$. Consider an arbitrary set D_i of $|A_i|$ correspondences being descendants of A_i. Again we number the points so x_i corresponds to y_i. Thus $D_i = \{(i, i, M_i), i = 1, 2, \ldots\}$. For each correspondence we fix a point $\bar{y}_i \in M_i$ and note that

$$h(\bar{y}_1, \bar{y}_2, \ldots) \geq \delta. \tag{2.17}$$

Let (I, J) be the index pair for which the maximum in the definition of h is obtained. Thus

$$|\angle(x_I, x_J) - \angle(\bar{y}_I, \bar{y}_J)| \geq \delta, \tag{2.18}$$

but this is larger than the correspondence uncertainty,

$$\sigma_c \leq \mathrm{diam}(M_I) + \mathrm{diam}(M_J) < \delta, \tag{2.19}$$

so by definition, see (2.14),

$$\min_{p \in M_I, q \in M_J} |\angle(x_I, x_J) - \angle(p, q)| = 0. \tag{2.20}$$

cannot be satisfied. This proves that a set of $|A_i|$ descendants of A_i cannot be pairwise consistent. □

Now, let $K = \max K_i$ and consider a minimum vertex cover S for H_K. Consider the ancestors $I \subset H$ of $H_K \setminus S$. Clearly this I has size at least κ and clearly $H_K \setminus S$ is a set of $|I|$ descendants of I that is pairwise consistent. Hence we can deduce from Lemma 8 that I is ϵ-consistent with some orthogonal transformation R. □

2.5.2 Estimating Transformations

The previous section showed how to obtain better and better lower bounds on the solution to Problem 3 and showed that eventually these bounds will be tight. However, since the constant κ in Theorem 7 is unknown, we will not know that the lower bound is tight unless we can produce an equally good upper bound. This can be done by actually computing a transformation.

The basis for creating an upper bound will be a simple method to estimate a rotation from two correspondences. Given two pairs of orthogonal vectors (e_1, e_2) and (f_1, f_2)

there is a unique rotation mapping e_i to f_i, which can be found by simple matrix multiplication. Given two pairs of points on the sphere (x_1, y_1) and (x_2, y_2), simply set

$$e_1 = \frac{x_1 + x_2}{||x_1 + x_2||},\ e_2 = \frac{x_1 - x_2}{||x_1 - x_2||} \text{ and } f_1 = \frac{y_1 + y_2}{||y_1 + y_2||},\ f_2 = \frac{y_1 - y_2}{||y_1 - y_2||} \quad (2.21)$$

to get the orthogonal vectors. It is easy to see that this rotation is symmetric with regard to the two point pairs and minimizes the angular errors.

Algorithm 4 Finding an upper bound

Pick two indices μ and ν as described in the text below.
For each correspondence $(\cdot, \mu, M)$:
  For each correspondence $(\cdot, \nu, N)$:
    Pick points $\bar{y}_\mu \in M$ and $\bar{y}_\nu \in N$
    Compute a transformation[1] from $(x_m, \bar{y}_\mu)$ and $(x_n, \bar{y}_\nu)$.
    Apply the transformation and check the number of outliers.

To get guaranteed convergence, the indices in Algorithm 4 cannot be chosen arbitrarily. Given a minimum vertex cover for the current graph, we choose two correspondences outside this vertex cover, say (m, μ, M) and (n, ν, N). We choose these such that $\angle(x_m, x_n)$ is large - at least larger than 2ϵ. The first indices of these correspondences are used. Assuming that we have passed the limit in Theorem 7, every vertex cover corresponds to a solution to the original problem. Thus the conditions of Theorem 9 are satisfied.

Theorem 9. *Consider two sets of three-dimensional unit vectors ordered such that x_i corresponds to y_i. Assume that there exist a rotation R such that $\angle(Rx_i, y_i) < \epsilon$ for all i. After a finite number of repetitions of Algorithm 3, Algorithm 4 will find such an R.*

Proof. The conditions say that there exists an R such that $\angle(Rx_i, y_i) < \epsilon$ for all i. Let

$$e = \max \angle(Rx_i, y_i) < \epsilon \quad (2.22)$$

Let m and n be the indices selected by Algorithm 4 and consider $\tilde{y}_m = Rx_m$ and $\tilde{y}_n = Rx_n$. After each splitting step, there exist correspondences (m, m, M_k) and (n, n, N_k) such that $\tilde{y}_m \in M_k$ and $\tilde{y}_n \in N_k$. Let R_k be the rotation estimated from these correspondences using Algorithm 4. It is clear from the construction of R_k that $d(R_k, R) \to 0$ when $k \to \infty$ though we leave this without a formal proof. Thus eventually, $d(R_k, R) < \epsilon - e$ and then

$$\angle(R_k x_i, y_i) < \angle(Rx_i, y_i) + (\epsilon - e) < \epsilon, \quad (2.23)$$

which means that we have found an R. □

[1] In cases when the optimal transformation might be a reflection, we need to find both a rotation and a reflection.

Remark. If there exists an R such that $\angle(Rx_i, y_i) \leq \epsilon$ but not any R such that $\angle(Rx_i, y_i) < \epsilon$, Theorem 9 does not give any guarantees. Theoretically, the probability of this event is zero and in practice, due to discrete number representation it is not relevant. Still it indicates that if a correspondence has an optimal error very close to ϵ, it will be difficult to determine whether this in an inlier or an outlier.

2.6 Camera Pose Estimation

Calibrated camera pose estimation is the problem of estimating the position and orientation of a camera given an image of a scene. Since the camera is calibrated we can choose to represent the image as a sphere (rather than an image plane) and detected feature points in the image as points on the sphere or unit 3-vectors. We restate Problem 1 for this new case.

Problem 4. *Given a set of unit 3-vectors, $\{x_i\}$, a set of 3-vectors, $\{X_i\}$, and a set of hypothetical correspondences H, find a subset $I \subset H$ minimizing $|H \setminus I|$ subject to I being ϵ-consistent an orthogonal transformation R and a camera centre C.*

In this case, ϵ-consistency means that for any $(m, \mu) \in I$,

$$\angle\left(x_m\,,\, R(X_\mu - C)\right) < \epsilon. \tag{2.24}$$

To fit this problem into the framework, we need a way to determine pairwise inconsistency. We will combine the vertex cover techniques with a search algorithm over some of the parameters.

Assume for a moment that our camera is ideal and noise-free. Given two 3D points, X and Y, and corresponding image vectors x and y, it follows that

$$\angle\left(X - C, Y - C\right) = \angle\left(x, y\right) = \alpha. \tag{2.25}$$

Here X and Y are part of the 3D model and the angle α on the right hand side can be calculated from the measured image coordinates. Now, consider an optimal camera pose (R, C) in the sense of Problem 4. Assume that X and Y are points that satisfy (2.24). The triangle inequality for angles yields,

$$|\angle\left(X - C, Y - C\right) - \alpha| < 2\epsilon. \tag{2.26}$$

This means that again we can use pairwise angles to detect inconsistent correspondences. But it only works if we know an approximate camera position, and therefore, we combine this idea with a search over possible camera positions. Later on we will show how to evaluate pairwise angles for a whole set of camera positions. Algorithm 5 gives an overview of the approach.

For a given box in the parameter space, it is not necessary that we solve the complete vertex cover problem. Thus it is beneficial to modify the branch-and-bound approach described in Algorithm 2. Details will be given in Chapter 3.

Algorithm 5 Camera pose estimation

Let B be an upper bound on the size of the outlier set for the optimal solution.
Initialize the queue with a bounded set in $\mathbb{R}^3$.
Iterate until desired precision is reached:
  Pick a box from the queue.
  Use vertex cover to discard the box or find a small vertex cover.
  If the box cannot be discarded:
    Starting from a small vertex cover use Algorithm 4 to update B.
    Divide the box and update the queue.
  Remove the box from the queue.

2.6.1 Sufficiency

In order for the splitting strategy discussed in Section 2.5.1 to be applicable, we need our constraints to be sufficient as the correspondence uncertainty tends to zero.

Theorem 10. *Consider 3D points X_i and image unit vectors x_i ordered such that X_i corresponds to x_i. If for some camera centre C and all i, j*

$$\angle(X_i - C, X_j - C) = \angle(x_i, x_j), \tag{2.27}$$

then there exists an orthogonal transformation R such that

$$\angle(R(X_i - C), x_i) = 0 \tag{2.28}$$

for all i.

Proof. Let $y_i = (X_i - C)/|X_i - C|$. Then, (2.27) implies $\angle(y_i, y_j) = \angle(x_i, x_j)$ and since both x_i's and y_i's are unit vectors $\langle x_i, x_j \rangle = \langle y_i, y_j \rangle$. This means that Theorem 6 is applicable and that completes the proof. □

Remark. *If some 3D point $X = C$, then the angles in this proof are not well-defined. Though this should never be a problem in practice, it may complicate the theoretic discussion. We eliminate this complication by requiring inliers to have at least some distance d_{min} to the camera.*

For 3D rotations it was possible to prove convergence. The same is true for camera pose, although the fact that we are performing a search over camera positions complicates matters. Because we are considering multiple camera positions at once, we are not evaluating the left hand side of (2.27) exactly, but as we will show, the approximation errors tends to zero. Consider a set of camera positions

$$S = \{C : |C - S_c| < r\} \tag{2.29}$$

To get a bound on the angular uncertainty of $X - C$ given that $C \in S$, we consider a cone with apex at X circumscribing S. If $|X - S_c| > r$, the opening angle of this cone will be

$$\gamma = 2 \arctan\left(\frac{r}{|X - S_c|}\right). \tag{2.30}$$

We define the parameter uncertainty σ_p to be the uncertainty introduced in (2.27) due to the radius of S; cf. (2.14). Note that $\sigma_p < 2\gamma$. Thus, as the radius tends to zero, so does this uncertainty, except in the case $X = S_C$; see comment above.

Theorem 11. *There exists a real number $e > 0$, such that when the sum of the maximal correspondence uncertainty and the approximation error in* (2.30) *is less than e, then every minimum vertex cover S corresponds to a solution to Problem 4 in the following sense: Let I_0 be the set of ancestors of the correspondences in $H_K \setminus S$. Then I_0 is a solution to Problem 4.*

Proof. The proof is almost identical to that for Theorem 7. The biggest difference is the lemma:

Lemma 12. *Consider a set S of K hypothetical correspondences. Assume that there does not exist an orthogonal transformation R and a camera position C such that*

$$\angle(x_m, R(X_\mu - C)) \leq \epsilon \text{ for all } (m, \mu) \in S \tag{2.31}$$

There exists a real number $e > 0$, such that when the sum of the maximal correspondence uncertainty and the approximation error in (2.30) *is less than e, there will be no set of K descendants of correspondences in S that is pairwise consistent.*

Proof. We can follow closely, the proof of Lemma 3 but use the function

$$h(\tilde{x}_1, \ldots, \tilde{x}_K, C) = \max_{i,j} |\angle(\tilde{x}_i, \tilde{x}_j) - \angle(X_i - C, X_j - C)| \tag{2.32}$$

on the set $\angle(\tilde{x}_i, x_i) \leq \epsilon$ and $C \in S$. This is a continuous function, defined on a compact set. Thus it takes a minimum value $\delta \geq 0$ in this set. If $\delta = 0$ then there is a pair (R, C) satisfying (2.31). Otherwise, when the sum of the correspondence uncertainty σ_c and the parameter uncertainty σ_p is less than $\delta/2$, then all sets of K descendants will be inconsistent (see proof of Lemma 3 for more details). □

See proof of Theorem 7. □

2.6.2 Estimating a Transformation

As for the 3D rotations, we get lower bounds on the number of outliers by actually estimating a transformation. We can do this in essentially the same way as for the 3D rotations. First the camera centre is fixed to the centre of the current box in parameter space and then 3D points are projected on the image sphere. This leaves the problem of estimating a rotation, for which we use the method from Section 2.5.2.

2.6.3 Evaluating the Constraints

Consider two world points X and Y and the corresponding image points x and y. Given a bounded set, S, in parameter space, how can we determine if (2.26) holds for any camera position $C \in S$?

First consider a plane through X and Y. We know from elementary geometry that all points on a circular arc through X and Y form the same angle XCY. Thus the points C such that XCY equals α form two circular arcs in the plane. Moreover, if for another C the angle XCY is larger than α, then C lies in the set enclosed by the two arcs.

In space the points C for which $XCY = \alpha$ form a surface, obtained by rotating the circular arcs around the line through X and Y; see Figure 2.2. For angles smaller then $\pi/2$ this surface is pumpkin-shaped. Like in the planar case points inside this surface form larger angles and points outside form smaller.

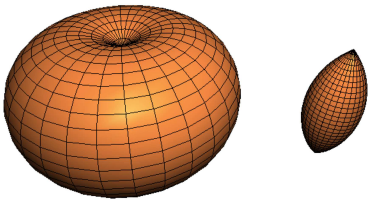

Figure 2.2: Sets $XCY = \alpha$ for $\alpha < \pi/2$ (*left*) and $\alpha > \pi/2$ (*right*).

We assume that set, S, in parameter space is a sphere. Other sets can always be handled by inscribing them in a sphere. First we note that if either X or Y lies in the S, then any angle XCY can be obtained, so (2.26) is satisfied. In the general case, we can use the following geometric observation.

Lemma 13. *The maximum and minimum for XCY are attained in the plane defined by X, Y and the sphere centre S_c.*

Proof. Let S_{2D} be the intersection of the mentioned plane and the sphere S. Now, it is clear that rotating C around the axis given by X and Y will not change the angle XCY. Thus any point not in the mentioned plane can be rotated into this plane without changing the problem. But any $C \in S$ will be rotated into the S_{2D}, which completes the proof. □

This means that we only need to consider the plane defined by X, Y and the sphere centre S_c. But in this plane the pumpkin constraint is simply two circular arcs as shown in Figure 2.2. Our problem is thus reduced to determining the intersection of the circle S_{2d} and these circular arcs.

2.7 Applications to Other Problems

The techniques discussed in this chapter can be extended to other problems in computer vision. We will look at one example here.

2.7.1 3D-3D Registration

In 3D-3D registration we are given two sets of 3D points that are related by a rigid transformation, i.e., a rotation and a translation. Like before we say that a set of correspondences is ϵ-consistent if it is one-to-one and errors are less than ϵ. In this case,

$$|x_i - R(y_j + t)| < \epsilon. \tag{2.33}$$

We want to solve the following problem.

Problem 5. *Given two sets of 3D points, $\{x_i\}$ and $\{y_i\}$ and a set of hypothetical correspondences H. Find a subset $I \subset H$ minimizing $|H \setminus I|$ subject to I being ϵ-consistent a rotation R and a translation t.*

This problem is very similar to estimating a 3D rotation and we can attack it in the same way. Pairwise consistency is determined by the point-to-point distances. Hence two correspondences, (m, μ) and (n, ν), are consistent if they satisfy

$$\big||x_m - x_n| - |y_\mu - y_\nu|\big| < 2\epsilon. \tag{2.34}$$

As in Section 2.5 a set of correspondences can be pairwise consistent even though they are not consistent according to Definition 1. So to get guaranteed convergence we need to use correspondence splitting and produce lower bounds by actually computing transformations. This can be done in a similar fashion as for 3D rotations, using a simple 2-point solver.

With Scaling. By a parameter search over an unknown scaling factor, we can also handle similarity transformations. In this case the consistency is defined by,

$$|x_i - \gamma R(y_j + t)| < \epsilon. \tag{2.35}$$

That this definition is not symmetrical hardly matters as the difference is only a scaling.

Chapter 3

Vertex Cover in Practice

The previous chapter provided a framework for computing optimal solutions to a variety of vision problems. In this chapter we examine the performance of these methods in experiments. The first section gives some experimental results for 3D-3D registration but also some modifications to speed up the algorithm. In the subsequent section the demand for guaranteed convergence is relaxed to get competitive methods for camera pose estimation. The relaxation means that we still get upper and lower bounds on the optimum, but there is no guarantee that these bounds will be tight. Still, the bounds tend to be very good.

3.1 3D-3D Registration

As noted at the end of the previous chapter, estimation of a rigid transformation can be handled similarly to the estimation of a rotation. In this case the aim was to handle datasets very difficult data sets in terms of the outlier rates. For example datasets where the correspondences are completely unknown. In such cases the number of hypothetical correspondences that has to be considered are N^2, where N is the number of points.

To be able to handle this type of problems efficiently, some new methods to handle the vertex cover problem was introduced. The first addition is a crude approximation algorithm. When the set of hypothetical correspondences is not one-to-one it can be very efficient. The new algorithm makes use of the way that the vertex cover graph was constructed. If there are multiple hypothetical correspondences matching the same point only one of them can be true. For example, if there are five correspondences matching the same point at least four of them must lie in the minimum vertex cover. Algorithm 6 shows how to use this fact to obtain a lower bound on the size of a vertex cover.

To eliminate correspondences, thereby reducing the complexity of the problem we need a way to prove that a certain correspondence v is an outlier to the optimal solution. To do so we use the same type of branching as in the exact algorithm in Chapter 1; see Algorithm 2. Either v is an outlier to the optimal solution, or all vertices having an edge to v are. If the latter hypothesis can be rejected, v must be an outlier; see Algorithm 7.

In the third step either Algorithm 1 or 6 or both can be used. The variant with Algorithm 6 is the fastest, especially if it is implemented in the following way. Assume

Algorithm 6 Fast approximation

Consider one of the point sets.
Let I be the set of point indices that we have considered.
Initialize $I = \emptyset$.
Initialize the lower bound $B = 0$.
Repeat for each vertex in the graph:
  Let (i, j) be the correspondence represented by this vertex.
  If $i \notin I$:
    Set $B = B + 1$.
    Set $I = I \bigcup \{i\}$.
B is now a lower bound for the size of the minimum vertex cover.

Algorithm 7 Proving that a correspondence, v, is an outlier.

Let K be an upper bound on the optimal solution.
Let S be the set of all vertices having an edge to v.
Compute a lower bound for any vertex cover containing S .
If this is $> K$, v is an outlier.

that v represents a correspondence (m, μ). Now compute the distance from point m to every other point and place it in a list ℓ_1. Similarly create a list ℓ_2 for distances to μ. Now by going through these lists once, we can single out those correspondences which are consistent with $v = (m, \mu)$.

The implementation for rigid transformation relies heavily on Algorithm 7 for eliminating the outliers one by one. First it is run once for every vertex with the sorting scheme and then if required again but with both Algorithm 6 and 1, in that order. Algorithm 2 is only used if some outliers remain after these steps.

Finally, to get an upper bound on the number of outliers we need to actually compute a transformation. First Algorithm 8 was used to find a small vertex cover. Then two correspondences that was not in this vertex cover was picked and inputted to the method in Section 2.5.2. To increase the probability of finding a good solution this was iterated a few times.

Experiments. The algorithm was evaluated on a 500-point 3D model of the Stanford bunny that was also used in [56]. To mimic the experiments from [56] a random rotation and translation was computed. Then uniform noise of magnitude 0.1 was added to the transformed points. The error threshold was set to 0.3, being approximately 1% of the object size a lower bound of 450 inliers.

Note that this experiment the total number of hypothetical correspondences was

Algorithm 8 Greedy vertex cover

Starting from a set S (possibly empty).
Remove all edges to vertices in S.
Repeat until no edges remain:
  Find the vertex v with most edges. Add v to S.
  Remove all edges connected to v.
When no edges remain S is a vertex cover for the graph.

250 000 and the rate of true correspondences 0.2%, since no correspondences were given and hence all points were matched against all points. This means that a basic RANSAC algorithm would have required on average 125 million iterations to converge. The proposed method had a mean execution time of 0.77 seconds. Essential to get this low average is that thanks to the sorting methods discussed above, we only have to consider a fraction of the complete vertex cover graph.

The algorithm in [56] obtains similar results on this data set as our algorithm, but they assume that the translation is known. They do not report any execution times for this problem but for a smaller problems with only 200 points their execution times is 1100 s, i.e., around 20 minutes. Furthermore, it is reported that the ICP and SoftAssign algorithms [8, 33] work only when the initial rotation is within an angle of less than 50 degrees of the correct solution for this data set.

3.2 Camera Pose Estimation

The algorithm that was implemented for camera pose was based on Algorithm 5 but to increase its efficiency, the following alterations were made.

- Splitting of hypotheses was not used.
- Algorithm 9 was used for vertex cover

Moreover, instead of using a fixed pair of points to estimate a transformation, as described in Section 2.5.2, a number of pairs was selected randomly, more precisely 10.

Initial bounds on the camera position are given as input parameters as well as some bound on the number of outliers. Bounds on the camera position were used to generate a first box for the parameter search. Each generated box was then attacked using the angular constraints from Section 2.6.3 and the simplified vertex cover algorithm described in Algorithm 9. To minimize computations, constraints are computed only when they are required.

Algorithm 9 uses the fast vertex cover techniques described in the previous section to compute lower bounds for the vertex cover problem. If this does not manage to discard the box we need to decide whether it is more beneficial to divide the box and continue

with the parameter search or to use branch and bound on the vertex cover problem. If there is a lot of outliers branch and bound might be necessary, but for lower rates of outliers this might slow down the execution. One way to handle this issue is to use branch and bound only if the rate of outliers is high. This method was chosen for the experiments using a threshold of 30%. To reduce the time spent on branch and bound, a maximum search depth was also set.

Algorithm 9 Vertex Cover for Camera Pose

Let B be the size of the best solution so far.
Let G be the graph for the current box.
Compute a lower bound on the size of the minimum vertex cover for G.
If this bound is $> B$:
  Discard the current box.
Else:
  Repeat as long as vertices are removed:
    For every vertex v:
      Try to prove that v lies in the minimal vertex cover using Algorithm 7.
      If this works:
        Remove v and update the graph.
  Get a new lower bound for the minimum vertex cover for G.
  If this bound is $> B$:
    Discard the current box
  Else:
      If the rate of outliers is greater than 0.3:
      Use Algorithm 2 to find the minimum vertex cover for G.
      If this has size $> B$:
        Discard the current box.

Shopping Street Experiments. The camera pose algorithm was tested on a data set consisting of 99 images of a shopping street covering approximately 100 meters. The experiments were performed in a leave-one-out fashion. One image was removed and a model consisting of 3D points and camera matrices were constructed from the remaining views using Bundler [75]. A 1000-iteration RANSAC loop with a minimal 3-point solver was used to find a plausible camera pose. To examine the quality of the RANSAC solution the proposed method was set to seek the optimal solution starting from a cube with side 5 meters around the RANSAC solution. Figure 3.1 shows the upper and lower bounds obtained in this fashion as well as the RANSAC solution.

To get some timing results, semisynthetic input data was generated. For each image twenty inliers were selected randomly and 180 outliers were generated. The outliers were

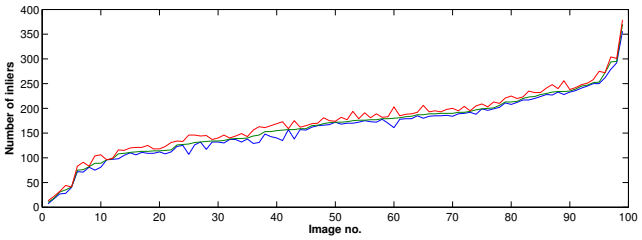

Figure 3.1: Results for the shopping street experiment. The blue line shows the number of inliers from RANSAC. The green line shows the lower bound from the proposed method and the red line is an upper bound for the optimal solution also obtained from the proposed solution. For visualization, the data has been sorted according to the green line.

placed randomly according to a uniform distribution bounded with the same bounds as the inliers. This way we got 99 examples with 90% outliers. The proposed method was started from the RANSAC solution as before. Figure 3.2 shows upper and lower bounds on the optimal solution as a function of time.

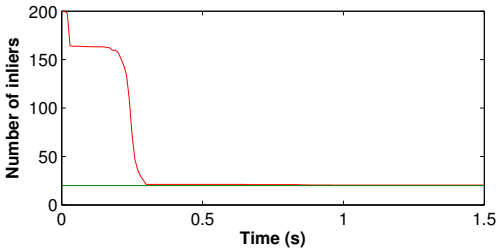

Figure 3.2: Average upper and lower bounds as a function of time. Note that most of the 180 outliers could were discarded quickly, but often one or two outliers are difficult to discard. The shown result is an average over all 99 images.

Restricted motions. One strength of the proposed approach is that it is easy to enforce certain type of restrictions on the camera position. One simply changes the search space of the algorithm. For example, we can constrain the camera to lie on any fixed surface that is easy to describe, or on some curve. One example is given in Figure 3.3 where the camera was constrained to a plane parallel with the ground plane. This figure also shows how the proposed algorithm can be used to evaluate the certainty of an estimated camera pose. The green points in Figure 3.3 are camera positions having as many inliers as the RANSAC solution.

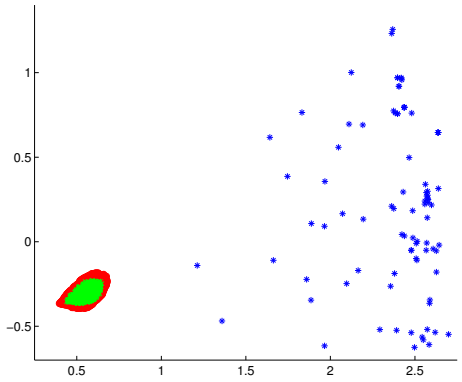

Figure 3.3: An example from the shopping street experiment. Blue points are 3D points. At the green points a solution having as many inliers as the RANSAC solution was detected. At the red points there might be a solution as good as the RANSAC solution.

3.3 Discussion

In these two chapters we have seen how graph methods can be used find optimal transformations to geometric problems in presence outliers. Even though vertex cover is an NP-hard problem we have seen that for 3D-3D registration, it is possible to achieve competitive results with use of approximation algorithms in combination with today's fast computers.

Combining the basic vertex cover approach with parameter search it is also possible to solve problems such as camera pose estimation. The experiments have shown how this algorithm can be used to evaluate the performance and uncertainty of standard methods such as RANSAC. One conclusion from these experiments must be that in most cases RANSAC works very well. This also explains why subsequent chapters are mainly concerned with more difficult problems.

Chapter 4

New Methods for Relative Orientation

In this chapter we look at the relative orientation problem for two calibrated cameras with outliers among the correspondences. Using an non-standard epipolar parameterization, two new methods for estimating the relative orientation are derived. The first method, which is based on branch and bound, is guaranteed to find the largest set of consistent correspondences. The other method performs a brute-force search over a discretization of the parameter space. By accepting a discretized parameter space, we get a robust method that is well-suited for parallelization and can handle several cost functions, for example maximizing the consensus set or robust norms like truncated least-squares. Furthermore, this method can easily be modified for restricted motions, such as planar motion. Experimental results are given for a variety of scenarios including scenes with very high rates of outliers. In addition, the algorithm is applied to 3D motion segmentation outperforming state-of-the-art on the well-known Hopkins-155 benchmark database.

4.1 Background

Already in 1981, Longuet-Higgins suggested a simple and yet elegant solution to the problem of finding the relative orientation of two viewpoints [57]. The algorithm, known as the eight-point algorithm, still plays a major role in computer vision [42]. In 1997, Hartley [39] modified the eight-point algorithm to include normalization. Although normalization made the algorithm more robust to measurement noise, there are still algorithmic degeneracies, e.g., if the scene is planar then the algorithm fails. Perhaps even more serious is that the algorithm assumes that the correspondence problem is already solved. Therefore, more robust approaches have been developed to cope with outliers. Here RANSAC methods using minimal solvers are considered to be state-of-the-art [66, 19]. Still, the problem of algorithmic degeneracies remains for minimal solvers.

Another problem that has been recognized by several researchers is the importance of optimizing a suitable cost function, where costs based on reprojection errors are preferable to algebraic errors [42]. Bundle adjustment does optimize the statistically correct

criterion (given that measurement errors are independent and normally distributed), but the method is sensitive to initialization. Therefore global optimization algorithms have been developed [49, 40, 18] which are not susceptible to local minima. However, all these methods assume that the correct correspondences are known.

Looking back at 30 years of algorithmic development since the eight point algorithm [57], a set of criteria has emerged that a good relative orientation algorithm should possess:

(i) Robustness to outliers,
(ii) No algorithmic degeneracies,
(iii) Cost function is based on reprojection errors,
(iv) Not dependent on a good initialization,
(v) Practical.

For example, RANSAC is designed to fulfil *(i)*, requires no initialization *(iv)* and has been successfully applied in many real systems *(v)*, but the method does not meet objectives *(ii)* and *(iii)*. Similarly, recent global relative orientation methods [40] do meet criteria *(ii)-(iv)*, but cannot be considered to be practical *(v)* since it cannot handle outliers. In practice, a heuristic combination of different algorithms is often used to overcome the difficulties in fulfilling these objectives. For example, homographies are often used to detect if the scene is planar or if the motion is a pure rotation. Another example of this phenomena is that particular motions have been examined separately [88].

This chapter will go through two methods for relative orientation. The first of them, extends the work on globally optimal methods by showing how to find an optimal solution in presence of outliers. To our knowledge, this algorithm is the first that solves this problem. However, execution times tend to be high, especially if the baseline is short, so although interesting from a theoretical viewpoint this method fails on point (v).

The second algorithm is based on exhaustive search and fulfils *(i)-(iv)* by design. For example, provided that the discretization of the parameter space is fine enough, the method is guaranteed to find the optimal solution. The key idea in order to make it practical is that the expensive computations are done in lower dimensions, and only very simple calculations are required in the high-dimensional search. The ultimate proof is of course by showing that it works on real experiments - this is done in the experimental section. In particular, when applied to 3D motion segmentation, our approach significantly outperforms state-of-the-art methods on 104 video sequences in the Hopkins 155 database [85]. Note that the database contains a variety of relative motions and scenes which are degenerate for several of the standard algorithms mentioned above.

4.2 Preliminaries

Consider two views of a scene. Let x be a unit vector representing an image point in the first view, and x' the corresponding image point in the second view. Our assumption is

that these are both the projection of some 3D point X. If we choose a global coordinate system such that the first camera lies at the origin and the second camera at $t = (0, 0, 1)^T$, we get

$$\lambda Rx = X, \quad \lambda' R'x' = X - t. \tag{4.1}$$

where R and R' are 3×3 rotation matrices and λ and λ' are positive real numbers. To simplify the derivations, we assume that there are no points at infinity, but the method works just as well with points at infinity.

Whereas the standard parameterization for a relative orientation consists of a rotation and a unit vector, (4.1) uses two rotations. Clearly this is an overparameterizaton, but this will be dealt with later. First we look at the constraint on the relative orientation induced by a pair of corresponding points.

Theorem 14. *Let R and R' be rotation matrices with row vectors r_1, r_2, r_3 and r'_1, r'_2, r'_3, respectively and let x and x' be corresponding points. Then,*

$$(r_1 x, r_2 x) = k(r'_1 x', r_2 x') \text{ with } k > 0, \tag{4.2}$$

$$r_3 x > r'_3 x', \tag{4.3}$$

if and only if there exists a 3D point X satisfying (4.1).

Proof. Clearly

$$k = \frac{||(r_1 x, r_2 x)||}{||(r'_1 x', r'_2 x')||} = \sqrt{\frac{1 - (r_3 x)^2}{1 - (r'_3 x')^2}} < 1. \tag{4.4}$$

Let λ be the solution to

$$\lambda r_3 x - 1 = \lambda k r'_3 x', \tag{4.5}$$

and put $\lambda' = \lambda k$. From $k < 1$ and $r_3 x > r'_3 x'$ it is straightforward to show that $\lambda > 0$ and hence $\lambda' > 0$. Now let $X = \lambda Rx$. To see that (4.1) is satisfied, consider

$$X - t = \lambda Rx - t = \begin{pmatrix} \lambda r_1 x \\ \lambda r_2 x \\ \lambda r_3 x - 1 \end{pmatrix}. \tag{4.6}$$

By (4.2) and (4.5) this is equal to

$$\begin{pmatrix} \lambda k r'_1 x' & \lambda k r'_2 x' & \lambda k r'_3 x' \end{pmatrix}^T = \lambda' R' x'. \tag{4.7}$$

This proves the *if* part, and the *only if* follows easily from (4.1). □

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

$$Rx = \begin{pmatrix} \sin\theta\cos\varphi \\ \sin\theta\sin\varphi \\ \cos\theta \end{pmatrix}, \; R'x' = \begin{pmatrix} \sin\theta'\cos\varphi' \\ \sin\theta'\sin\varphi' \\ \cos\theta' \end{pmatrix}. \tag{4.8}$$

Now the necessary and sufficient constraints are

$$\varphi = \varphi' \text{ and } \theta < \theta'. \tag{4.9}$$

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

The next step is to allow measurement errors. We say that corresponding points x, x' are consistent with a relative orientation if the angular reprojection errors are less than some prescribed threshold, ϵ.

Definition 4. *Given an error tolerance ϵ, a correspondence (x, x') is said to be* consistent *with a relative orientation given by R and R', if there exists a 3D point X such that*

$$\angle(Rx, X) < \epsilon \quad \textit{and} \quad \angle(R'x', X - t) < \epsilon. \tag{4.10}$$

Just as in Chapter 2 we seek the largest set of consistent correspondences. This task is formulated in Problem 6.

Problem 6. *Given two sets of image points $\{x_i\}$ and $\{x'_j\}$ with hypothetical correspondences (x_k, x'_k), $k = 1, \ldots, N$ and a prescribed error threshold ε, compute the relative orientation of the cameras which is consistent with as many correspondences as possible.*

Theorem 15. *Consider rotation matrices R and R' and spherical coordinates as defined in* (4.8)*. Further define w in the following way. For $\theta < \theta'$,*

$$w = \arcsin(\sin\epsilon / \sin\theta) + \arcsin(\sin\epsilon / \sin\theta'), \tag{4.11}$$

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

$$w = \arccos\left(\frac{\cos 2\epsilon - \cos\theta\cos\theta'}{\sin\theta\sin\theta'}\right), \tag{4.12}$$

if this is defined and otherwise π. Then,

$$\begin{aligned} &\theta < \theta' + 2\epsilon \\ &\varphi \in [\varphi' - w, \varphi' + w], \end{aligned} \tag{4.13}$$

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

Proof. Since it is always possible to change coordinates, we can assume that $R = R' = I$. Furthermore, we note that if we can find points $\bar{x}$ and $\bar{x}'$ that satisfy the constraints in Theorem 14 as well as

$$\angle(\bar{x}, x) < \epsilon \text{ and } \angle(\bar{x}', x') < \epsilon, \tag{4.14}$$

then (by Theorem 14) we have also found our point X. This will prove useful. Let $\bar{\theta}$, $\bar{\theta}'$, etc denote the spherical coordinates of these points as defined in (4.8). We assume that $\bar{x}'$ is fixed and examine what constraints we get on $\bar{x}$. Recall the constraints from Theorem 14,

$$\bar{\theta} < \bar{\theta}' \tag{4.15}$$

$$\bar{\varphi} = \bar{\varphi}'. \tag{4.16}$$

From (4.14) we have that $\bar{x}'$ must lie in a small circle around x'. Consequently, (4.16) means that $\bar{x}$ must lie in the spherical wedge shown on the left in Figure 4.1 and (4.15) constrains it to the upper part of that wedge, as shown on the right in Figure 4.1.

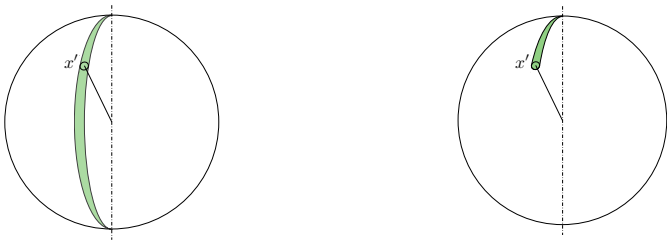

Figure 4.1: The constraints imposed on $\bar{x}$ being the reprojection of the 3D point in first camera. Equation (4.16) constrains $\bar{x}$ to the spherical wedge (left) and (4.15) to the upper part of that wedge (right).

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

Case 1: $\theta < \theta'$ Figure 4.2 shows the critical case. If the difference between φ and φ' is larger than this, then the two sets have empty intersection. The limit can be computed by considering two right-angled triangles, see Figure 4.2. Let v denote the blue angle in that figure and v' the yellow one. Using Theorem 3 yields,

$$\sin v = \frac{\sin \epsilon}{\sin \theta} \text{ and } \sin v' = \frac{\sin \epsilon}{\sin \theta'}. \tag{4.17}$$

and if we define $w = v + v'$, we can write the constraint $|\varphi - \varphi'| < w$. Note that, if either $\sin\theta$ or $\sin\theta'$ is smaller than $\sin\epsilon$ then w is not defined. In these cases one of the

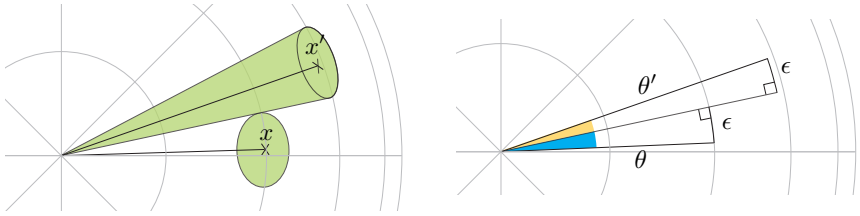

Figure 4.2: Case 1. Here the sphere from Figure 4.1 are viewed from above, i.e. the z-axis is pointing out of the paper. The green areas show the constraints on $\bar{x}$. For the two constraints to intersect they must not be further apart than this. The left image shows the setup for computing this limit angle.

triangles is degenerated and the intersection is non-empty regardless of the φ's. One way to describe this is to set $w = \pi$.

Case 2, $\theta' < \theta < \theta' + 2\epsilon$: Figure 4.3 illustrates the critical position. Using Theorem 2, we can compute w,

$$\cos\theta\cos\theta' + \sin\theta\sin\theta'\cos w = \cos 2\epsilon. \tag{4.18}$$

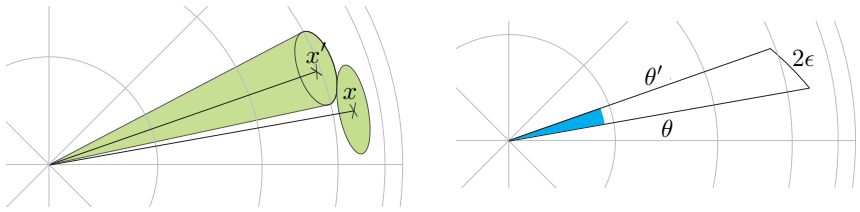

Figure 4.3: Case 2, see caption of Figure 4.2.

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

Theorem 15 gives us a relatively simple way to handle the constraints induced by one point-to-point correspondence. This will be used by both methods presented in this chapter. To get efficient methods we must not ignore that using two rotation matrices to represent a relative orientation is an overparameterization. In fact if S is a rotation about the z-axis, then (R, R') and (SR, SR') describe the same relative orientation. To see this we assume that

$$\lambda Rx = X, \quad \lambda' R'x' = X - t, \tag{4.19}$$

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

$$\lambda SRx = SX, \quad \lambda' SR'x' = SX - St = SX - t, \tag{4.20}$$

shows that a rotation around the z axis just changes the global coordinate system. We will soon see how to avoid this ambiguity.

To simplify the notation we introduce a function Γ that maps a given unit vector r to a rotation matrix, Γ_r, having r as its third row. Of course there are many such functions and any one of them will do. Now any relative orientation can be written as

$$R = S_\alpha \Gamma_r \text{ and } R' = \Gamma_{r'} \tag{4.21}$$

where S_α is a rotation by α about the z-axis. Hence a minimal set of parameters consists of unit vectors, r and r', together with an angle, α. It is worth to note that r and $-r'$ are actually the epipoles of the two cameras.

Now consider the spherical coordinates in (4.8). Only φ depends on α. Let $\varphi(r)$ denote the value if $\alpha = 0$. This changes the last constraint of Theorem 15 to

$$\varphi(r) + \alpha \in [\varphi'(r') - w, \varphi'(r') + w]. \tag{4.22}$$

4.3 An Optimal Algorithm

This section will present an algorithm for finding the largest consistent set of correspondences. The algorithm is based on an explicit search for unit vectors r and r' as defined in the previous section. The angle α on the other hand is eliminated by considering pairs of corresponding points. Let $\varphi_k(r)$ and $\varphi'_k(r')$ be the spherical coordinates of corresponding image points and define

$$\gamma_{jk}(r) = \varphi_j(r) - \varphi_k(r). \tag{4.23}$$

Then (4.22) implies that

$$|\gamma_{jk} - \gamma'_{jk}| < w_j + w_k. \tag{4.24}$$

Geometrically, γ_{jk} is the angle between two epipolar planes, see Figure 4.4. In the noise-free case, γ'_{jk} is exactly the same angle, so the difference should be small.

To see the connection between these pairwise constraints and Theorem 15, we need the following lemma from Chapter 2. For the proof we refer to Lemma 5.

Lemma 16. *Consider a set of intervals I_k on the unit circle such that $|I_k| < 2\pi/3$ for all k. If the intersection $I_j \bigcap I_k$ is non-empty for any pair (j, k) , then $\bigcap_k I_k$ is non-empty as well.*

If all pairwise correspondences fulfil (4.24) with uncertainty intervals less than $2\pi/3$, then Lemma 16 guarantees that there exists an angle α such that constraint (4.22) is fulfilled for each correspondence. Together with Theorem 15 this proves the following result.

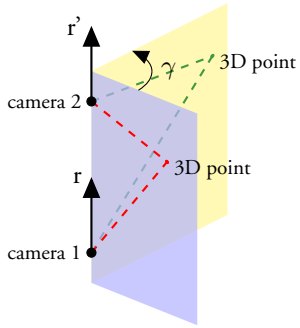

Figure 4.4: The angle between two epipolar planes.

Theorem 17. *Consider two unit vectors r and r'. If all corresponding points satisfy*

$$\theta_i < \theta'_i + 2\epsilon, \tag{4.25}$$

and all pairs of correspondences satisfy constraint (4.24) with $w_j + w_k < 2\pi/3$, then all correspondences are consistent with relative orientation (R, R'), where $R = S_\alpha \Gamma_r$ and $R' = \Gamma'_r$. Again S_α is a rotation about the z-axis.

The additional hypothesis that the uncertainty intervals should have length at most $2\pi/3$ is just an annoying technicality. Larger intervals would correspond to image points very close to the epipole (within angles $\epsilon < \theta < 2\epsilon$), and even if such points exist it is very likely that the conclusion of the lemma still holds. To get around this problem one could split long intervals and still get guaranteed convergence, but we have not found any problem instance where this is necessary in practice.

The pairwise constraints of Theorem 17 can be analyzed using graph methods. Using the techniques from Chapter 2 and 3 the maximum set of consistent correspondences can be found by finding a minimum vertex cover.

4.3.1 Branch-and-Bound Search

To solve Problem 6 we will seek the optimal values for r and r' using branch and bound. This means that we need a method for branching, i.e. splitting a problem into subproblems, as well as a method for bounding these subproblems.

Since the search space is the product space of two unit spheres, $S^2 \times S^2$, the branching will be performed by dividing these spheres into smaller an smaller spherical triangles. Figure 4.5 shows a spherical triangle, bounded by three great circles, and Figure 4.6 shows how one triangle can be split into four smaller ones.

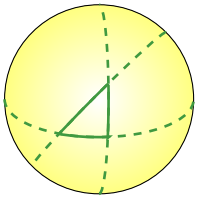

Figure 4.5: A spherical triangle is a triangular area on the sphere, bounded by great circles.

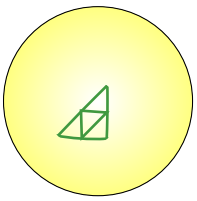

Figure 4.6: In the division step a triangle is divided into four smaller ones by dividing its sides at their midpoints.

For the bounding function, we will use the constraints of Theorem 17 together with vertex cover techniques from Chapter 2 and 3 to get a bound on the number of correspondences being consistent with a particular branch of the search tree. The tricky part here is computing upper and lower bounds on the γ_{jk}'s. We need a method to answer questions of the type: If r lies in this spherical triangle, what are the possible values of γ_{jk}? A first observation is that it is sufficient to consider the boundaries of the spherical triangle.

Theorem 18. *Given two image points x_j and x_k and a spherical triangle, T, of possible values for the epipole, r. If either $\pm x_j$, $\pm x_k$ lie in the triangle then γ_{jk} can assume any value. Otherwise the maximal and minimal values for γ_{jk} are obtained on the triangle boundary.*

To prove this claim we use an explicit formula for the angle γ (we temporarily drop the subindex). We choose a special coordinate system such that the two image points have coordinates $(0, \pm a, b)$ and define $r = (r_x, r_y, r_z)$. We assume $e_x \geq 0$ implying $\gamma \geq 0$. Let us first compute the normals of the two epipolar planes,

$$r \times x_j = \begin{pmatrix} br_y - ar_z \\ -br_x \\ ar_x \end{pmatrix}, \quad r \times x_k = \begin{pmatrix} br_y + ar_z \\ -br_x \\ -ar_x \end{pmatrix}. \tag{4.26}$$

Then consider their scalar product,

$$b^2 r_y^2 - a^2 r_z^2 + b^2 r_x^2 - a^2 r_x^2 = \cos\gamma \left(\sqrt{(br_y - ar_z)^2 + r_x^2} \sqrt{(br_y + ar_z)^2 + r_x^2} \right). \tag{4.27}$$

The fact that $r_x^2 + r_y^2 + r_z^2 = 1$ and some simple calculus yield

$$r_y^2 - a^2 + b^2 r_x^2 = 2abr_x \cot\gamma. \tag{4.28}$$

Proof. We start with the case case $r_x > 0$. We will show that γ as a function of the epipole has no local extrema and hence minimal and maximal values of γ are attained on the boundary. Since cotangent is a strictly monotone on $(0, \pi)$ it is sufficient to show that

$$g(r_x, r_y) = 2ab \cot \gamma = \frac{r_y^2 - a^2 + b^2 r_x^2}{r_x} \tag{4.29}$$

has no local extrema in the set $D = \{r_x^2 + r_y^2 \leq 1; r_x > 0\}$. We first note that $g'_y = 0$ only if $r_y = 0$ and that $g'_x = 0$ implies

$$0 = \frac{2b^2 r_x^2 - r_y^2 + a^2 - b^2 r_x^2}{r_x^2} = \frac{b^2 r_x^2 + a^2}{r_x^2}, \tag{4.30}$$

which is never satisfied. It remains to examine the boundary δD. If $r_x^2 + r_y^2 = 1$ we get

$$h(r_x) = 2ab \cot \gamma = \frac{1 - r_x^2 - a^2 + b^2 r_x^2}{r_x} = \frac{b^2 - a^2 r_x^2}{r_x} = \frac{b^2}{r_x} - a^2 r_x \tag{4.31}$$

$$h'(r_x) = -\frac{b^2}{r_x^2} - a^2 \tag{4.32}$$

which is never zero. Finally the end point $r_x = 1$, while being a local minimum to h is not an not an extremum to g. To see this note that $h'(r_x)$ is negative, whereas g'_x is positive.

This completes the proof for $r_x > 0$ and $r_x < 0$ can be handled with an almost identical proof. It remains to consider the set $r_x = 0$. This is a great circle through the points x_j and x_k. Let ℓ be the part of this great circle that lies inside T. It is easy to see that γ is either 0 or π on this great circle and that the value changes at $\pm x_j$ and $\pm x_k$. Thus if none of these points is in T, then γ is constant on ℓ so any value attained on ℓ is also attained on $\delta T \cup \ell$. This completes the proof. □

Theorem 18 means that to find upper and lower bounds for γ_{jk}, it is sufficient to consider the case when r lies on the boundary of a specific spherical triangle. Since the boundary consists of great circle arcs, we need to be able to compute upper and lower bounds for γ_{jk} when r moves on a great circle arc. Several techniques to do this has been tested, mainly based on setting up a system of polynomial equations. As often, one of the simplest turned out to be the most effective.

Consider a great circle parameterized by a single angle ξ. Let d_i be the angular distance from image point i to this great circle. We can compute γ_{ij} by first computing the angle relative to the great circle for each image point, see Figure 4.7, and take the difference between these angles. Hence we first compute,

$$\varphi_j(\xi) = \pm \operatorname{arccot} (\sin \xi \cot d_j), \tag{4.33}$$

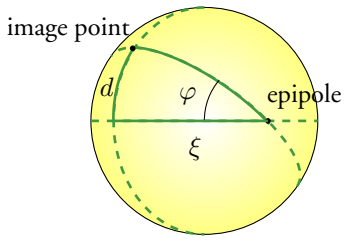

Figure 4.7: We first compute angles to the great circle and then take pairwise differences to yield the epipolar plane angles $\gamma_{ij} = \varphi_i - \varphi_j$.

where the formula follows from Theorem 2 and 3 applied to the right-angled triangle in Figure 4.7. The sign in (4.33) depends on which side of the great circle the image points lies. By subtracting two such angles, we get

$$\gamma_{jk}(\xi) = \pm \operatorname{arccot}(\sin\xi \cot d_j) \pm \operatorname{arccot}(\sin(\xi + \beta_k) \cot d_k). \tag{4.34}$$

Still at least two problems remain. First of all, finding the maximum and minimum values of (4.34) generally require solving a sixth degree polynomial equation. Furthermore, the number of functions to optimize is quadratic in the number of correspondences (since we are considering all pairs). To address both these problems we compute simple under- and overestimators for φ_i. These can then be used to produce under- and overestimators for the γ_{jk}'s in the first image (and similarly for $\bar{\gamma}_{jk}$ in the second image). This avoids the difficult optimization of γ_{jk} and most work is performed on the φ_j's and is thus linear in the number of image points.

Simple under- and overestimators for a C^3 function $\varphi(\xi)$ can be derived from the standard Taylor expansion. If

$$m \leq \frac{\delta^2\varphi}{\delta\xi^2} \leq M, \tag{4.35}$$

then

$$\begin{aligned}
\varphi(\xi) &\geq \varphi(\xi_0) + (\xi - \xi_0)\frac{\delta\varphi}{\delta\xi}(\xi_0) + \frac{(\xi - \xi_0)^2}{2} m \\
\varphi(\xi) &\leq \varphi(\xi_0) + (\xi - \xi_0)\frac{\delta\varphi}{\delta\xi}(\xi_0) + \frac{(\xi - \xi_0)^2}{2} M.
\end{aligned} \tag{4.36}$$

For m and M we will use conservative bounds based on the characteristics of trigonometric functions; e.g. $|\sin(x)| \leq 1$. In many cases these will show that γ_{jk} is in fact monotonic on the interval in question. In these cases it is sufficient to consider the endpoints (i.e. corners of the spherical triangle) and computations are very easy. Otherwise

we use the upper and lower bounds given by (4.36). Note that the approximation error decreases quadratically with the diameter of the triangles.

Algorithm 10 gives an overview of the approach. Let us quickly go through the different steps. To initiate the branch and bound search we divide the search space $S^2 \times S^2$ into 64 starting blocks. A block here is a pair of spherical triangles indicating the positions of r and r'. We also need a bound on the optimal solution. This is a starting guess for the number of outliers of the optimal solution and will be updated as the algorithm progresses. If the starting guess is too low the algorithm is restarted with a higher value.

Algorithm 10 Optimal Relative Orientation

Iterate until desired precision is reached:

1. Pick a box from the queue.
2. Try to detect and remove outliers.
3. Try to discard the box.
4. If the box cannot be discarded:
   - Divide the box and update the queue.
   - Try improve the bound on the optimum.
5. Remove the box from the queue.

Finally a word on updating the bound on the optimum, that is, finding better and better solutions. We do this by considering the centres of those boxes that could not be discarded. For these points we compute the constraints exactly and count the number of outliers. As the boxes get smaller and smaller, the optimal number of outliers will eventually be found.

4.3.2 Experiments

The algorithm was implemented in C++. Running times are for a 3.0 GHz Intel Dual-Core with 3 GB RAM.

The algorithm was evaluated on images from the Valbonne Church data set[1]. From the 15 images all pairs were formed. SIFT features were extracted and matched. The matching criterion of [58] was used with a threshold of 0.6; cf. Section 1.2.

The threshold was set to 0.0005 radians and the starting guess for the number of outliers to 5. The algorithm was terminated if the gap between lower and upper bound was less than 3 or if the average uncertainty of the epipole was less than 2 degrees.

The algorithm was also tested on the omnidirectional images used in [40]. Among other things, execution times depended on the distance between the cameras. This is natural since estimating the epipole - being the direction of translation - is difficult when the length of the translation is very small; cf. Chapter 6.

[1] The Valbonne data set was provided by ROBOTVIS, INRIA.

Figure 4.8: Example images from the Valbonne Church data set.

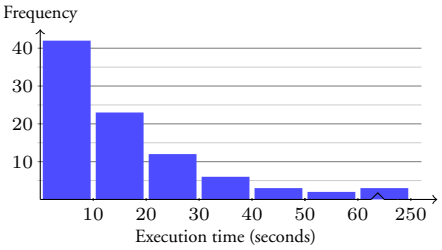

Figure 4.9: Execution times for the Valbonne data set. Between 10 and 100 points in each experiment (on average 64). Around 5% outliers.

For these experiments the error threshold was set to 0.002 radians and the starting guess for the number of outliers was initiated to 5. If there was no solution for the current threshold was increased with 5 until a solution could be found.

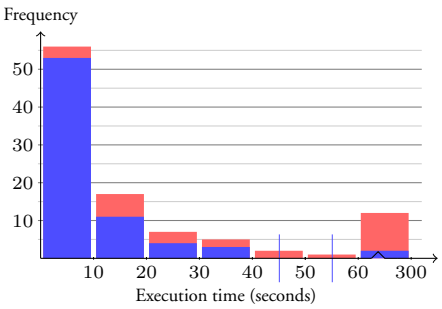

Figure 4.10: Execution times for the ladybug data set. Between 20 and 100 points in each experiment (on average 64). Around 10% outliers. Blue indicates camera pairs with a large camera distance and red the more difficult short distance cases.

4.4 A Brute-Force Algorithm

In this section we will look at algorithm that works with a discretization of the parameter space. First, we rewrite (4.22) as a constraint $\alpha \in [\alpha_{lo}, \alpha_{up}]$, where

$$\begin{aligned} \alpha_{lo} &= \varphi'(r') - \varphi(r) - w \\ \alpha_{up} &= \varphi'(r') - \varphi(r) + w. \end{aligned} \tag{4.37}$$

Each correspondence yields an interval of this kind. Sorting the lower and upper bounds of all such intervals, makes it easy to find a point that lies in as many intervals as possible,

see Algorithm 12.

The proposed algorithm is a brute force search over a discretized version of $S^2 \times S^2$. For every choice of $(r, r') \in S^2 \times S^2$, lower and upper bounds on α are computed and sorted, allowing the maximal number of inliers to be found. Algorithm 11 gives an overview of the different steps. The complexity will be $O(k^2 m \log(m))$ where k is the number of points in the discretization of S^2 and m is the number of correspondences.

Remark. *As it matters only rarely and complicates the description, we ignore case* (4.12) *of Theorem 15 in the computation of* w*. Thus* w *can always be divided into*

$$w = v + v' \tag{4.38}$$

where v *does not depend on* x' *and* v' *does not depend on* x*.*

Algorithm 11 Brute-Force Search

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

Compute a discretization, $\mathcal{D}$ *of* S^2*.*
For each $r \in \mathcal{D}$
  For each x
    Compute $\varphi(r)$*,* $\theta(r)$ *and* $v(r)$*.*
  For each x'
    Compute $\varphi'(r)$*,* $\theta'(r)$ *and* $v'(r)$*.*
Put $n_{max} = 0$
For each pair $(r, r') \in \mathcal{D} \times \mathcal{D}$
  For each correspondence (x, x')
    If $\theta'(r') + \epsilon > \theta(r) - \epsilon$
      Compute $w = v(r) + v'(r')$*.*
      A lower bound $\alpha_{lo} = \varphi'(r') - \varphi(r) - w$*.*
      An upper bound $\alpha_{up} = \varphi'(r') - \varphi(r) + w$*.*
  Find the max intersection n*, using Algorithm 12.*
  If $n > n_{\max}$
    Store the current parameters.
    Set $n_{\max} = n$*.*

4.4.1 Parallel Implementation

Naturally, the biggest concern about a brute-force algorithm lies in its computational performance. However, studying Algorithm 11 we note that the computations for different

Algorithm 12 Maximal Intersection

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

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

pairs (r, r') are independent, so we can easily parallelize the algorithm by a MapReduce model: In the Map step, the lower and upper bounds are sorted simultaneously and then intersections are computed simultaneously for all pairs (r, r'). In the Reduce step, the pair (r, r') that yields most inliers is picked out by reduction operations.

Nvidia's parallel computing architecture, CUDA, was used for the parallel implementation. Algorithm 12, was implemented in a 2-dimensional grid with k by k blocks, where k is again the number of points in the discretization. Each block executes the computation for one pair of epipoles, (r, r'). Inside each block, a parallel bitonic sorting algorithm with complexity $O(n \log(n)^2)$ is implemented since it is especially suited for sorting within a block using shared memory. To find the maximum intersection, each thread goes through the upper bound list to find the maximal intersection for the current lower bound. This is done using binary search.

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

4.4.2 Experiments

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

To get some data on the execution times, synthetic data was generated. First 100 random 3D point were generated in a cube centered at the origin, having side 300. The cameras were placed randomly at distance of approximately 1000. Gaussian noise with standard deviation 0.0002 was added to the image points. Figure 4.11 shows angular errors in rotation and translation when compared to the ground truth. The threshold

$\epsilon = 0.005$ was used with different degrees of discretization.

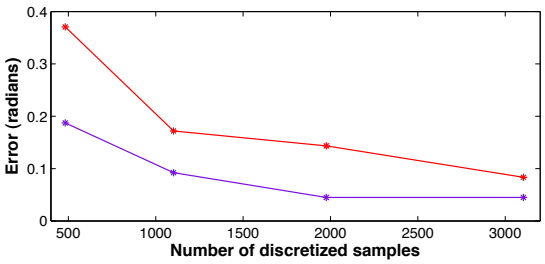

Figure 4.11: The plot shows errors for different discretizations. The error in rotation is shown in red and the error in translation is shown in blue.

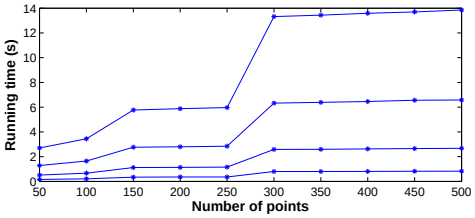

Figure 4.12: Execution times for different discretizations. Starting from below the curves were generated using 700, 1258, 1976 and 2862 points in the discretization of the unit sphere, S^2.

High rate of outliers. To test the proposed algorithm on data with a lot of outliers, synthetic data was generated in the following way. First 50 random 3D point were generated in a cube centered at the origin, having side 100. The cameras were placed randomly at distance of approximately 1000. Then 450 outliers were added to each image. They were generated in the same way as the inliers but separately for the two images. Gaussian noise with standard deviation 0.0002 was added to the image points. Figure 4.13 shows how many of the 50 inliers were found by the proposed algorithm. The threshold $\epsilon = 0.0005$ was used in the algorithm and the average computation time for the parallel implementation was 6s.

The outlier rate in these experiments was 90%. This means that using standard RANSAC and a five-point solver, the expected number of iterations before picking just one single set with 5 inliers is 100 000 and using reprojection errors that also means performing 50 million triangulations.

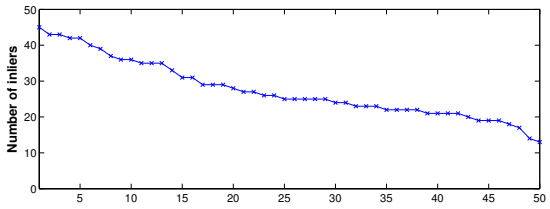

Figure 4.13: The obtained number of inliers for the 50 outlier experiments. The list was sorted for better visualization. In each example there were originally 50 inliers and 450 outliers. The error threshold was $\epsilon = 0.0005$ and 1976 points were used in the discretization of the sphere.

4.5 Restricted Motions

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

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

$$rf = 0, \quad r'f = 0 \quad \text{and} \quad \alpha = 0. \tag{4.39}$$

The first two constraints can easily be enforced in the discretization step. Only epipoles in these planes are generated. The third constraint reduces the set of angles that has to be considered in Algorithm 12.

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

$$\angle(r_3, r'_3) < \gamma_{max} \quad \text{and} \quad \alpha < \alpha_{max}. \tag{4.40}$$

These constraints reduce the number of pairs that have to be considered in Algorithm 11.

Experiments. The performance on planar scenes was tested on 64 image pairs from eniro.se. These are street-view images taken from a car so the motion is approximately planar. Since the images are given with direction information we could compute the deviation between the estimated rotation matrix and the ground truth. This deviation in radians is given in Figure 4.14. The results were produced using 100 points to discretize the unit circle and a threshold of 0.0005. The average execution time was 0.47 s for a sequential java implementation.

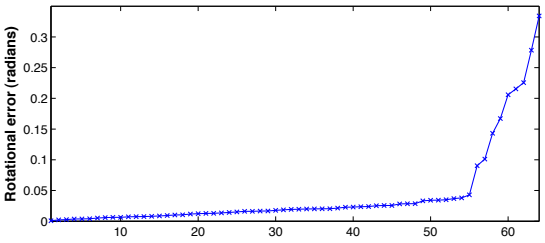

Figure 4.14: Angular error when comparing with the ground truth rotation. The data has been sorted for visualization purposes.

4.6 Other Cost Functions

So far we have simply counted the number of inliers to assess the quality of a relative orientation. Inliers are correspondences with the reprojection errors less than some prescribed threshold, ϵ. This approach is simple and generally yields good results, but it does have its limitations; see [42]. One problem is that the method might be sensitive to the choice of ϵ, but also that the distribution of the inlier errors is not considered. In [9] a more refined cost function is proposed. The assumption is that correct matchings have a clock-shaped error distribution similar to the Gaussian distribution, whereas incorrect matchings have approximately uniformly distributed errors. These assumptions lead to the cost function

$$C(d) = -\log\left(c + \exp\left(-d^2\right)\right) \tag{4.41}$$

where d is the reprojection error, see Figure 4.15. In the same book it is noted that an good approximation of this cost function can be obtained by truncating the ordinary squared error. A cost function of this kind cannot be handled directly by the proposed method,

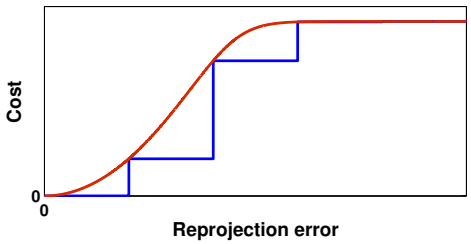

Figure 4.15: The robust cost function (red) suggested in [9], and a piecewise constant approximation of it (blue) which can be optimized using the proposed framework.

but one can approximate the function to arbitrary precision. An example of such an approximation is shown Figure 4.15. As the reprojection error increases it changes value three times. This means that when computing w in Algorithm 11, we should do so for three thresholds $\epsilon_1, \epsilon_2, \epsilon_3$. Consequently each correspondence will yield three intervals $I_1 \subset I_2 \subset I_3$ - one for each time the value of the cost function changes. The different types of intervals also get a weight indicating how much the cost function changes when entering this interval.

This means that in Algorithm 12 we get three lists of lower bounds $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ and similarly for the upper bounds. The different lists are sorted separately and then gone through like before. Passing a lower bound from $\mathcal{L}_i$, weight w_i is subtracted from the current cost, and passing an upper bound from $\mathcal{U}_i$ the same weight is added. The computational cost will be approximately linear in the number of steps in the cost function.

To verify the possibility of using other cost functions we tried it on some random data generated as described above. Using the appoximated truncated L_2 norm in the way described in Section 4.6 the rotational error decreased from the average 0.17 radians to an average of 0.11 radians. This was using 1100 points in the discretization. The threshold for the standard method was $\epsilon = 0.005$ and the thresholds for the approximate truncated L_2 was set to $\epsilon/2$, ϵ, $3\epsilon/2$ and 2ϵ.

4.7 Motion Segmentation

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

Algorithm 13 Multiple Motions

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

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

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

Therefore, we proceed in a sequential manner using Algorithm 3. The first step is to estimate N hypothetical motions. This is done in a sequential manner, using Algorithm 13. Typically, N is chosen significantly larger than the true number of motions not to miss any motion. The next step is to choose three of these N motions to perform the motion segmentation. We do this by going through all possible choices of three motions and choosing choosing the ones that yield the lowest number of outliers. Just as in Algorithm 13 outliers are tracks having an error larger than ϵ_2. Having decided on three motions we match each point track to that motion which yields the smallest errors.

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

4.7.1 Adding a Spatial Prior

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

$$C(\mathbf{x}) = \sum_{p \in \mathcal{V}} C_p(x_p) + \lambda \sum_{(p,q) \in E} C_{pq}(x_p, x_q). \tag{4.42}$$

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

$$C_{pq}(x_p, x_q) = \begin{cases} 0 & \text{if } x_p = x_q \\ \frac{d_{max} - d(p,q)}{d_{max}} & \text{if } x_p \neq x_q \end{cases} \tag{4.43}$$

where $d(p, q)$ denotes the Euclidean distance of point p and q and d_{max} is a threshold to define the size of the neighborhood. If $d(p, q) < d_{max}$, then $(p, q) \in E$. This smoothness term will penalize the case when two points lie close to each other but belong to different motions. The constant λ determines the balance between the data and smoothness term. Energy minimization was performed using α-expansions; see [13].

4.7.2 Results for Hopkins 155

We will now look at the performance of this 3D motion segmentation algorithm for rigid scenes from the Hopkins 155 database [85]. Current state-of-the-art results are reported

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

Table 4.1: Classification errors (%) for the 26 checkerboard sequences with 3 motions.

in [27] and all the top performers are included in the comparison below. In each sequence, there are typically 20-30 frames and a few hundred 2D feature tracks given. The number of motions in each sequence is also specified.

Some of the sequences contain articulated motions to which the presented framework does not apply. Therefore we focus on the subset of checkerboard sequences, 26 sequences with 3 motions and 78 sequences with 2 motions, hence 104 out of the 155 sequences are considered. Based on [27], one can conclude that the checkerboard sequences are the most difficult ones as the classification errors are significantly lower for the remaining ones.

All of the top performing algorithms are based on the affine camera model. Hence they are not dependent on the internal calibration of the cameras, whereas we assume calibrated cameras. To resolve this, the principal point is set to the middle of the image and the focal length to 700 pixels for images of size 480×640. This is the size for all sequences, but the last one, which has frame size 240×320 and consequently we halve the focal length for this case. Note that the true focal length is unknown, so the chosen is only empirically motivated[2].

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

We compare with the following algorithms: Generalized Principal Component Analysis (GPCA) [89], Local Subspace Affinity (LSA) [90], RANSAC [30], Multi-Stage Learning (MSL) [77], Agglomerative Lossy Compression (ALC) [59] and two variants of Sparse Subspace Clustering (SSC) [27]. There are two versions of our brute-force algorithm. The first one (BF) is implemented according to the description in Section 4.7 and the second one is with the addition of a spatial prior (BF-S) as described in Section 4.7.1.

In Tables 4.1 and 4.2, the misclassification rates are presented. Our brute-force algorithm achieves very low error rates, both in terms of mean and median error rates. Note that even though we are only using three frames (the first, the middle and the last) in each sequence, we are able to obtain state-of-the-art results. Since we are actually recovering the 3D motion, it is very simple to add spatial regularization to the results. Still, even without such regularization, our approach outperforms the competitors, and with regularization, the error rates are significantly lower.

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

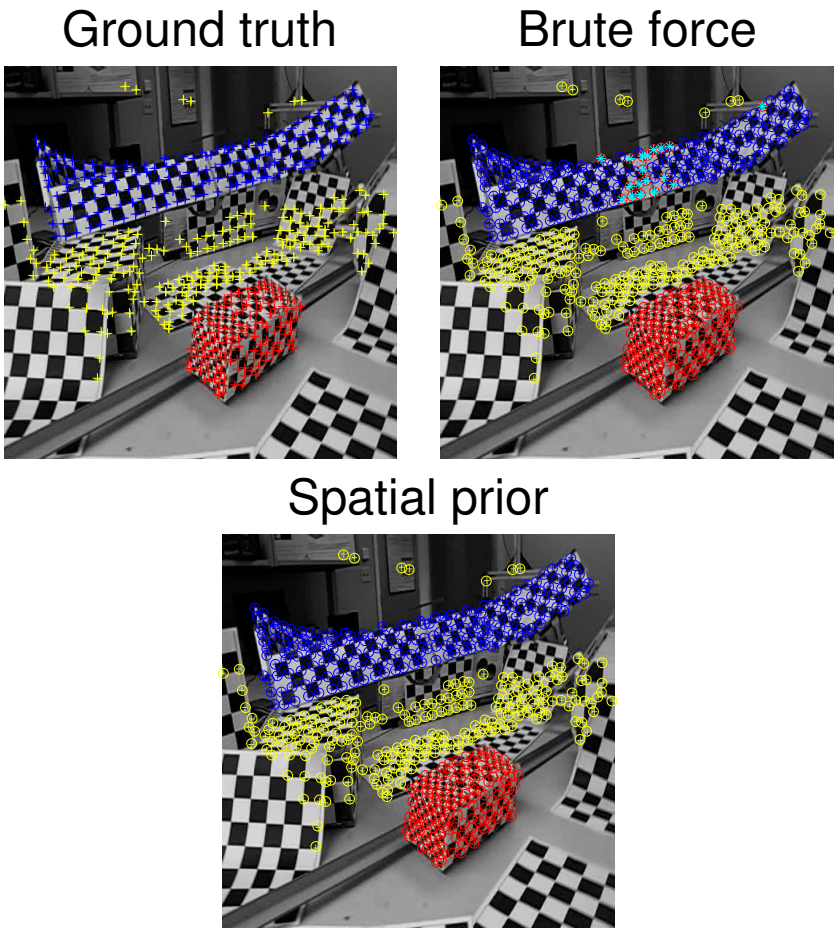

Figure 4.16: Example frame from one sequence with ground truth (left), brute-force (middle) and brute-force with spatial prior (right). The colors of the feature points indicate which motion class (blue, yellow, red). Feature points that are misclassified have been colored cyan (see middle figure). Note that the spatial prior is able to correct for all the errors.

4.8 Discussion

Using a brute-force algorithm for computing the relative orientation of two projections may seem like a step back considering the many sophisticated algorithms that have been developed over the years. But why is it that none of the best performing algorithms

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

Table 4.2: Classification errors (%) for the 78 checkerboard sequences with 2 motions.

for 3D motion segmentation uses a pinhole camera model? The results presented here shows that a pinhole model is the correct choice and the lack of perspective methods that perform well on the Hopkins 155 benchmark is likely due to algorithmic failure modes, for example, the incapability of handling planar scenes.

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

Chapter 5

Outliers and Quasiconvexity

In Chapter 1 we saw that a number of L_∞ optimization problems in multiple view geometry can be solved in polynomial time. But what happens if there are outliers among the correspondences? How hard is it to find the optimal solution with respect to the number of outliers? In this chapter we will see that in many cases the optimum can still be found in polynomial time. Unfortunately, the degree of the polynomial is quite high, so the polynomial-time algorithms are often too slow to use in practice. Still the result gives valuable theoretical insights. It helps us to understand why RANSAC is so successful for many applications. Also, it has given us guidelines for designing an alternative approach. Given a candidate solution, we present a procedure to verify whether this solution is optimal or not. This verification procedure can be seen as a guided search; either the candidate is verified or a better solution can be found.

In summary, we will show that many registration and reconstruction problems can be solved in polynomial time with a guarantee of global optimality. We derive practical algorithms for simultaneously (i) computing the optimal transformation and (ii) separating inliers from outliers. Still, there are some limitations to this framework. It is required that the residual errors as functions of the unknown transformation variables are quasiconvex; see Section 1.3.1. This is a weaker condition than convexity, but it does exclude most of the problems discussed in Chapters 2-4. Another weakness is that the proposed approach becomes computationally expensive when there is a large portion of outliers, say more than 80%. Typically, solutions with more than 50% inliers are sought for, and if there are no such solutions, then the method will report this.

5.1 Related Work

There is a large body of work on solving matching and registration problems [8, 6, 72, 87, 91]. For example, in [6, 51], the matching problem is formulated as an integer program and then solved by non-optimal methods. In [60], the correspondence problem is cast as an assignment problem and can thus be solved optimally. Matching problems have also been solved using ideas based on the Hough transform and branch-and-bound [67, 17, 46]. Perhaps the most popular paradigm is to use RANSAC [30] which has proven

to be successful in many practical situations. Our approach leverages on this idea to generate good candidates for global solutions. However, there are important differences. First of all, RANSAC gives no guarantee of optimality. Another problem is that if RANSAC returns an poor result, then it is unclear whether there is no better solution or whether RANSAC was just unable to find it. With our scheme, we can provide a certificate that there is no subset with, say, more than 50% inliers.

Recently, there has been a renewed interest in multiple view geometry problems aimed at optimal algorithms; cf. Section 1.3.1. Using the max-norm, it has been shown that many such problems can be efficiently solved using convex optimization. The problem of outliers has also been addressed in this context. In [50], a heuristic, non-optimal method is used to remove outliers. In [74], it is shown how to remove outliers but the method tends to remove a lot of inliers as well.

The work that is most closely related to this is [54], which can be seen as a refinement of [74] as actual outliers are detected and removed. This algorithm, can also be used for computing optimal solutions in polynomial time. Still, the work differs in severals aspects. For one thing, with the new approach the residual functions are not required to be strictly quasiconvex but only quasiconvex; cf. [74]. Moreover, the method in [54] is hardly practical except for very low number of outliers, whereas the verification scheme presented in this chapter can handle up to 50% outliers.

In computational geometry, there is a long tradition of providing performance bounds for different types of geometric optimization algorithms; see [1] for a survey. Some of the terminology and ideas in this chapter is borrowed from that research community.

5.2 Problem Formulation

The problem formulation is the same as in Chapter 2, but repeated for the reader's convenience.

Definition 5. *Consider two point sets, $\{x_i\}$ and $\{y_j\}$, a set of correspondences H and a transformation T. If H is one-to-one and*

$$d(T(x_m), y_\mu) \leq \epsilon, \tag{5.1}$$

for all $(m, \mu) \in I$, we say that H is ϵ-consistent with T.

Here, d refers to an appropriate metric. The problem that we want to solve is the following.

Problem 7. *Given two point sets and a set, H, of hypothetical correspondences between the point sets, find a subset $I \subset H$ maximizing $|I|$ subject to I being ϵ-consistent with some transformation T of a specified class $\mathcal{T}$.*

From now on we will assume that the transformation can be parameterized with a vector $t \in \mathbb{R}^n$. We also assume that the correspondence set is one-to-one and numbered such that x_i corresponds to y_i. Then errors can be written as $r_i(t) = d(T_t(x_i), y_i)$. Finally, we will assume that these residual functions are quasiconvex. This is true for many problems in multiple view geometry; see Section 1.3.1 and [49]. We look at an example that will fit with the framework.

Example: 2D similarity transformation. *The source points x_i are mapped to the target points y_i by a similarity transformation $\gamma R x_i + c$. This transformation can be parameterized with four parameters,*

$$\gamma R = \begin{pmatrix} r_1 & r_2 \\ -r_2 & r_1 \end{pmatrix}, \quad c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}. \tag{5.2}$$

As residual error we use the standard 2-norm $d(\gamma R x_i + t, y_i) = ||\gamma R x_i + c - y_i||$. It is easy to prove that this is a convex function of r and c.

5.3 Preliminaries

Since each correspondence in Problem 7 is linked to a residual function, we might as well consider sets of residual functions. The following definition will play an important role in the analysis.

Definition 6. *Given a set, S, of quasiconvex residual functions $r_i(t)$, we define*

$$e(S) = \min_t \max_{r_i \in S} r_i(t). \tag{5.3}$$

Since the residual functions are quasiconvex, this function can be computed efficiently. Hence a brute-force solution to Problem 7, would be to perform an exhaustive search over all subsets of H, compute $e(S)$ for each subset and check whether it is less than ϵ. However, the number of subsets is exponential so this approach normally not very practical. The following definition comes from computational geometry; cf. [1].

Definition 7. *Let S be a set of residual functions. A subset $B \subset S$ is a basis set of S, if $e(B) = e(S)$ and for any proper subset $B' \subset B$, we have $e(B') < e(B)$.*

The *combinatorial dimension* of a set S is the maximum size of a basis of S. The following theorem states that if the residual functions are quasiconvex, then the combinatorial dimension is always relatively small. Essentially the same proof can be found in [54].

Theorem 19. *Let S be a set of quasiconvex residual functions $r_i : \mathbb{R}^n \to \mathbb{R}_+$. If $|S| > n+1$ then there exists $B \subset S$ with $|B| = n + 1$ and $e(B) = e(S)$.*

Proof. Let $B_1, \ldots, B_K$ be an enumeration of all subsets of S with size $n+1$. Assume, for contradiction, that

$$\alpha = \max e(B_k) < e(S). \tag{5.4}$$

By Definition 7, this implies that the following set is non-empty

$$\bigcap_{r_i \in B_k} \{t : r_i(t) \leq \alpha\} \tag{5.5}$$

for any k. Note that the intersecting sets are sublevel sets of quasiconvex functions, so they are convex. Thus Helly's theorem implies that

$$\bigcap_{r_i \in S} \{t : r_i(t) \leq \alpha\}, \tag{5.6}$$

is non-empty as well, but by Definition 7 this implies that $\alpha \geq e(S)$, which contradicts (5.4). □

One implication of this theorem is that S cannot be a basis set, so the combinatorial dimension for this type of problems is bounded by $n+1$.

5.4 A Polynomial-Time Algorithm

This section will show that the optimal solution to Problem 7 can be found in polynomial time. Let S^* be the set of residuals corresponding to the optimal set of correspondences of Problem 7. Now consider the set of transformations that is ϵ-consistent with S^*. This set can be written as an intersection of the sublevel sets of all functions in S^*,

$$\bigcap_{r_i \in S^*} \{t : r_i(t) \leq \epsilon\}. \tag{5.7}$$

Finding any transformation in this set would solve Problem 7. Since it does not matter which one, we can add a goal function without changing the problem. This makes the solution unique.

Lemma 20. *Consider a set, S, of continuous quasiconvex residual functions, $r_i(t)$. Then there is a unique point in*

$$\bigcap_{r_i \in S} \{t : r_i(t) \leq \epsilon\}. \tag{5.8}$$

with minimal norm, $|t|$. We denote this point $t^*(S)$.

Proof. It is well-known that in a closed convex set there is a unique point with minimal norm; see e.g. [7]. Hence we need to prove that the set in (5.8) is closed and convex. First we note that the r_i's are continuous functions so the sublevel sets in (5.8) are closed. Moreover, the r_i's are quasiconvex, so the sublevel sets are convex. Finally, the intersection of closed convex sets is closed and convex. This completes the proof. □

The next theorem shows that we can find $t^*(S)$ for a large set by considering a relatively small subset.

Theorem 21. *Let S be a set of continuous quasiconvex residual functions defined on $\mathbb{R}^n$. If $t^*(S)$ is defined, then there is a subset of $B \subset S$ with $|B| \leq n+1$ such that $t^*(B) = t^*(S)$.*

Proof. Let $B_1, \ldots, B_K$ be all subsets of S with size $n+1$. Assume, for contradiction, that $|t^*(B_k)| < |t^*(S)|$ for every B_k. Hence, assume that

$$a = \max_k |t^*(B_k)| < |t^*(S)|. \tag{5.9}$$

This implies that for any k

$$\{t : |t| \leq a\} \bigcap \left(\bigcap_{r_i \in B_k} \{t : r_i(t)\} \right) \neq \emptyset. \tag{5.10}$$

Note that all the sets in (5.10) are convex and that (5.10) implies that any intersection of $n+1$ of them is non-empty. Hence, Helly's theorem implies that

$$\{t : |t| \leq a\} \bigcap \left(\bigcap_{r_i \in S} \{t : r_i(t)\} \right) \neq \emptyset \tag{5.11}$$

as well, which means that $|t^*(S)| \leq a$. This is a contradiction so the assumption in (5.9) must be wrong. Thus we have proven that $|t^*(S)| \leq |t^*(B)|$ for some subset, B, with $|B| = n+1$. However, $t^*(S)$ is feasible for any subset of S, so the unique minimizer $t^*(B)$ must be identical to $t^*(S)$; cf. Lemma 20. □

Remark. *A similar result is given in [55] for pseudoconvex functions. However, they assume that the minimizer of* (5.3) *is always unique. This is not true in general for pseudoconvex functions. The same mistake was made in the conference version of this work.*

Algorithm 14 Polynomial-Time Outlier Removal

Given a set, S, of continuous quasiconvex residual functions, defined on $\mathbb{R}^n$, solve Problem 7.

For each subset, $B \subset S$ of size $n+1$.
  Compute $t^*(B)$.
  Compute all residuals $r_i(t^*(B))$ and count the outliers.
  If this is the lowest number of outliers so far, save $t^*(B)$.

Theorem 21 means that we can use Algorithm 14 to solve Problem 7. Unlike the brute-force solution, Algorithm 14 only considers subsets of size $n+1$. There are

$O(|S|^{n+1})$ such sets. For each of them t^* is computed, but this can be done efficiently using convex optimization. Then all residuals are computed which has complexity $O(|S|)$. Hence the algorithm has complexity $O(|S|^{n+2})$, so it is polynomial in the number of residuals, but exponential in the dimension of the parameter space. For the problems that we are considering, that dimension is fixed, so it is natural to call it a polynomial-time algorithm.

5.5 Verifying Optimality

Even though Algorithm 14 is polynomial it is hardly practical for real problems. Thus we look at a dual approach. It turns out that it is possible to transform Problem 7 into a purely combinatorial problem. To do so, let $B_1, \dots B_K$ be all sets of exactly $n+1$ correspondences. For each B_k we determine if it is ϵ-consistent; see Definition 5. We then create a hypergraph in the following way. Make a vertex in V for each correspondence and create a edge for each B_k that was not ϵ-consistent. Theorem 19 tells us that a set of correspondences is ϵ-consistent if all its $n+1$-subsets are ϵ-consistent. Thus solving Problem 7 is equivalent to finding a minimum vertex cover for the hypergraph that we just constructed.

Clearly, this result is not very practical in itself. Not only is vertex cover for hypergraphs an NP-hard problem, but even setting up the graph requires solving $O(m^{n+1})$ convex feasibility problems. However, if we already have a strong solution, we can use methods from graph theory to verify that this solution is in fact the global optimum, or get a better solution.

To find candidate solutions, we use the popular RANSAC method; see Section 1.2. Its wide use in a variety of applications tells us that the method is often effective in finding a good solution. We use RANSAC to generate a first candidate solution I that is ϵ-consistent with some transformation. Then we try adding correspondences from $H \setminus I$ to this set. When no more correspondences can be added without violating consistency, we say that we have reached *a local optimum*. At this point we will be in one of the following situations.

- The optimal solution has been found and it has a high rate of inliers.
- There is no candidate solution with a high rate of inliers.
- The optimum has a high rate of inliers but it has not been found yet.

This section will describe algorithms to verify that the solution we have found is in fact the global optimum, if this is true, and show how the same algorithms can be used in the case when no good solution has been found. The algorithms can be seen as crude approximation algorithms for the vertex cover problem discussed above.

Starting from a set of hypothetical correspondences, H, assume that we have found a local maximum, i.e., a set $I_0 \subset H$ that is ϵ-consistent. Since it is a local maximum,

no more correspondences can be added to I_0. We wish to prove that I_0 is optimal in the sense of (7) or find a better candidate set. To verify that we have found the global optimum, we need to reject the hypothesis that there exists a set I^* that is ϵ-consistent with some transformation such that $|I^*| > |I_0|$.

We say that a correspondence i is an I_0-outlier if $I_0 \bigcup \{i\}$ is not ϵ-consistent. Our strategy to prove the optimality of I_0 is to show, for each I_0-outlier, that it cannot lie in I^*. Algorithm 15 shows the basic principle. Recall that n is the dimension of the parameter space; cf. Theorem 19.

Algorithm 15 Discarding a correspondence.

Let i be an I_0-outlier.
Let L be the number of detected inconsistencies.
Initialize $L = 0$.
Divide $H \setminus \{i\}$ into disjoint H_k with $|H_k| = n$.
For each H_k
  Perform a feasibility test for $H_k \cup \{i\}$.
  If the test fails set $L = L + 1$.
If $L > |H \setminus I_0|$
  Set $H = H \setminus \{i\}$.

The last step can be motivated as follows. If a set H_k is not consistent with i, then $i \in I^*$ implies that at least one correspondence from H_k is not in I^*. Since the H_k's are disjoint, the number of inconsistencies gives a lower bound on the number of I^*-outliers given that $i \in I^*$. If this bound is higher than the number of I_0-outliers, we can conclude that $i \notin I^*$.

Note the similarities between this algorithm and Algorithm 7 in Chapter 3. In that case we were attacking the vertex cover problem for a normal graph, whereas in this case we are dealing with a hypergraph.

Algorithm 15 is simple, but it only works for relatively small rates of outliers. We get approximately $|H|/n$ sets H_k, so we can verify I_0 must have less than $|H|/n$ outliers. Naturally we would like to improve this performance without having to do an exhaustive search. This can be done by choosing non-disjoint sets. In this way it is possible to generate more test sets. As the number of outliers grows the problem becomes more difficult until eventually it requires testing all sets of size $n + 1$, in which case the polynomial algorithm from the previous section is preferable.

A rather general approximation method is the following modification of Algorithm 15. Divide H into disjoint sets H_k of size $n + d$. For each H_k form all subsets of size n, denoted $H_{k,j}$. Now we check which of the sets $H_{k,j}$ that are consistent with i. The results tell us something about how many elements of H_k must be I^*-outliers provided that $i \in I^*$.

- If no subsets of H_k are consistent with i at least $d+1$ elements of H_k must be I^*-outliers provided $i \in I^*$.

- If less than $\binom{n+f}{f}$ of the sets are consistent at least d+1-f of the elements of H_k must be I^*-outliers provided $i \in I^*$.

Remark. *It is not necessary to have a candidate solution to start off with. The same algorithm can still be applied to prove a statement of the type* $|I^*| < \gamma|H|$ *for some* $0 < \gamma < 1$. *In this case we pick points one by one and show that they cannot be part of any* I *with* $|I| \geq \gamma|H|$.

5.6 Experiments

The verification approach has been tested on a number of problems from multiple view geometry. For all the experiments, SIFT was used to find corresponding points in the images; cf. Section 1.2. Then RANSAC was applied to generate a candidate solution I_0. If possible this was then verified as the global optimum, but in many cases, the initial solution first had to be improved. To do so, we tried adding correspondences one at a time and test if $I_0 \cup \{i\}$ was still consistent.

If the optimum could still could not be verified we used the following scheme. For each outlier that could not be discarded, we have a number of sets that are not consistent with this outlier and a number of sets that are; cf. Algorithm 15. Hence to find a new candidate solution, we pick an outlier that could not be discarded and a set of points that were consistent with this outlier. We then use all these points to calculate a new candidate solution.

For each of the experiments, results are compared to the performance of a standard RANSAC procedure. In the few cases where possible (within reasonable time) we also compare with the algorithm given in [54]. When this is not possible we give a worst case bound on the number of bases from [64]. The results are summarized in Table 5.1. The error tolerance ϵ was set to two pixels. The implementation was done in Matlab using SeDuMi to perform the feasibility tests.

Homography. There are many vision applications involving homographies (see [42]), for example, detecting a planar configuration viewed in two images. Given image correspondences of 3D points lying in a plane, there is a homography between the two image planes mapping corresponding points to each other. Other correspondences can be regarded as outliers. It was shown in [49] that the residual functions for a homography are indeed quasiconvex.

We tested our algorithm on two stereo image pairs with different outlier rates (see Table 5.1). Figure 5.1 shows the computed inliers and outliers for the first stereo pair. In

	Corr. $\lvert H\rvert$	Inliers RANSAC	Inliers optimal	Inlier rate	Tests for verification	Tests in [54] *worst case
Homography	513	430	432	0.84	5889	$6.7 \cdot 10^{15}$*
(Office Wall)	101	57	64	0.63	9468	$4.6 \cdot 10^{13}$*
3D Pose	353	264	270	0.76	10276	$4.0 \cdot 10^{21}$*
(Teddy Bear)	121	104	105	0.87	420	$9.3 \cdot 10^{13}$*
	69	49	50	0.72	1171	$5.6 \cdot 10^{14}$*
	86	55	57	0.66	800	$5.6 \cdot 10^{14}$*
	150	114	116	0.77	834	$6.5 \cdot 10^{15}$*
	65	42	50	0.77	1228	$4.8 \cdot 10^{13}$*
	14	9	9	0.64	1001	$2.7 \cdot 10^{5}$*
	105	87	87	0.83	418	$3.2 \cdot 10^{14}$*
	174	147	149	0.86	718	$1.0 \cdot 10^{16}$*
	263	244	245	0.93	187	$3.2 \cdot 10^{14}$*
2D Pose	217	194	199	0.92	396	$3.5 \cdot 10^{5}$*
(Books)	67	56	59	0.88	102	$4.9 \cdot 10^{5}$*
	76	70	71	0.93	20	574
	74	66	66	0.89	42	11177
	77	46	46	0.60	4563	$2.8 \cdot 10^{6}$*
	146	43	< 73	< 0.50	18335	$3.2 \cdot 10^{8}$*

Table 5.1: Summary of experimental results. Correspondences $|H|$ - the total number of hypothetical correspondences obtained from the SIFT descriptors. Inliers - the number of inliers detected by standard RANSAC and the local search method, respectively. Number of tests to verify optimum - number of convex feasibility tests required for verifying that the local maximum (output of the local method) is global. Number of tests in [54] - number of bisection programs required to discard the same number of outliers as the local method using the method from [54]. When it is not possible to run in practice a worst case bound from [64] is given. A bisection program consists of approximately 10-15 feasibility tests. All solutions except 2D pose image id 6 were verified to be optimal. In this case we verified that there was no solution with more than 50% inliers.

this case the number of inliers are 432 and the number of outliers are 81. The solution was verified to be optimal; see Table 5.1.

Uncalibrated Camera Pose. In this experiment we try to determine the camera pose of a 3D object from image data; see Figure 5.2. It was shown [49] that when the internal camera parameters are unknown, then the camera pose problem is quasiconvex.

For the experiments, we used images from the publicly available database introduced in [52]. The model, plotted in the left of Figure 5.2, was created from two stereo images of the front of the Teddy bear. Using SIFT correspondences we then estimated the pose in 10 test images where the Teddy is partly visible. In all cases the optimality of solution could be verified; see Table 5.1.

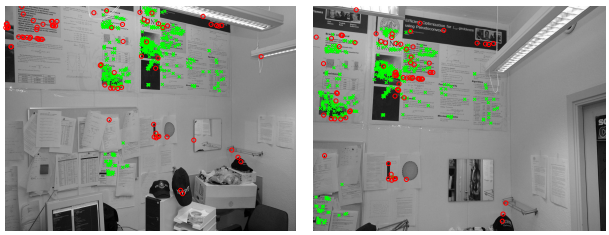

Figure 5.1: Inliers (green) and outliers (red) in a homography estimation. The solution is optimal.

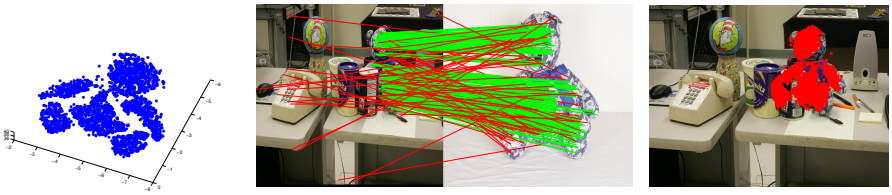

Figure 5.2: The uncalibrated camera pose problem. Left: The model obtained from two images. Middle: Correspondences between the test image and one of the stereo images (green - outliers, red - inliers). Right: The model points projected onto the test image under the optimal projection.

2D-2D Registration. In this experiment we took 6 images of books on a table, and tried to find the optimal similarity transformation between pairs of images. In one case we matched a model image with only one book to a more complex image with the same book appearing several times. We also tried matching the more complex images to each other. Interestingly, we were able to verify the optimal solution in cases where another strong solution existed, and in cases where no strong solution existed, we were able to prove this; see Figure 5.3 and Table 5.1.

Triangulation. We also tested our method on a triangulation example. We took 25 images of a painted vase and matched SIFT features from different views. The camera positions were estimated using markers. Then we tried to verify the solutions obtained by performing RANSAC on the correspondence data. There were 269 points that had been matched to at least seven images. Of these, we successfully verified 190 solutions to be optimal that had at least 50% inliers. In the remaining 79 cases we could verify that there was no solution with 50% inliers or more.

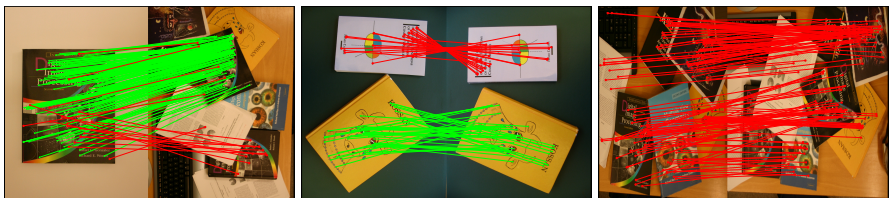

Figure 5.3: Images of books used for estimating similarity transformations. Green lines mark the inlier correspondences of the globally optimal transformation and red lines mark outliers. The middle images show that we can verify the optimal transformation despite the presence of another strong transformation, and to the right we have an example of a case where we can verify that there is no transformation with at least 50 % inliers.

Figure 5.4: The triangulation problem. Two images with points used for calculating the reconstruction to the right. Green points (inliers) are used for the triangulation while red points are detected as outliers.

5.7 Discussion

In contrast to most previous work dealing with outliers, this chapter presented a framework for estimating globally optimal solutions. For a large number of applications, we have seen both theoretically and experimentally that this is indeed a tractable problem. Another conclusion that we can draw, is that in most cases RANSAC works rather well. We have given both theoretical evidence and practical experiments which show that it is a sound method. Hopefully the presented framework can be useful when benchmarking other heuristic methods for dealing with outliers.

Part II

Estimation from Multiple Views

Chapter 6

Non-Sequential Structure from Motion

We study structure from motion estimation from multiple calibrated views. Given a set of images with known calibration data, we want to estimate scene structure as well as camera positions. Although this problem has been studied extensively over the years, no fully satisfactory solution exists. Among the things that make this problem so challenging, one can mention the high dimension of the space of unknowns and the difficulty in correctly matching features between views. Yet another challenge is the existence of repetitive or planar structures, short baselines between views or moving objects in the scene. Unlike ordinary mismatches that will cause random outliers in the data, repetitive structures can cause locally consistent geometries that do not agree with the global geometry. This can lead to two-view geometries supported by a large number of point correspondences, but not reflecting the underlying true geometry; see Figure 6.1 for an example.

Figure 6.1: ROAD ROLLER. In this image pair, 28 seemingly correct correspondences (green lines) are obtained in the estimation of the epipolar geometry. Even though the epipolar geometry is plausible and perfectly valid, it does not correspond to the true geometry.

6.1 Structure from Motion Approaches

Many methods for multi-view structure from motion start by estimating the geometry of two views. Often, a minimal solver is applied in combination with RANSAC; see e.g. [66]. The 3D geometry estimated from these two views is used to estimate the pose of another camera which in turn improves the quality of the reconstruction. This way more cameras are added incrementally. The reconstructions are often improved using local optimization; see [78]. We will refer to this approach as sequential structure from motion. One weakness of this approach is that the quality of the reconstruction might depend heavily on the choice of the initial pair. This is addressed in [80] with a heuristic approach based on the covariance of the structure and the CIRC criterion [82].

Another weak point of sequential methods is the iterative process of adding new cameras. It might be that the quality of the final reconstruction depends on the order in which cameras are added. Furthermore, due to their sequential nature these methods suffer from drift (error build-up) [21], instead of distributing the error evenly throughout the sequence. Recently an automated sequential system was presented [75], showing impressive results of large-scale reconstructions. The system, known as BUNDLER, will be compared to our work.

A different approach is taken by the methods based on factorization. In [81] a solution for the affine camera model is provided and [76] gives an extension to perspective cameras. The missing data problem and sensitivity to outliers are major concerns in this approach and it has been the object of study in subsequent papers, e.g., [79]. Hierarchical methods [31, 65, 71, 32, 28] organize images in a hierarchical cluster tree, and do the reconstruction from root to leafs.

In this chapter, a non-sequential method for estimating the geometry of multiple views is suggested. The method consists roughly of three parts. First the orientations of all cameras are estimated with a method being robust to low-level noise as well as large errors from the matching. Then the robust L_∞ method from Section 1.3.1 is used to solve the structure and motion with known orientations and finally the reconstruction is improved using bundle adjustment. Only the orientations are taken from the pairwise estimations and as we will see, these are often quite accurately estimated.

This places our approach in the same category as [62], but they estimate camera orientations using an over-parameterized linear least-squares formulation. As is well-known, linear least squares estimation, can be very sensitive if there is a large amount of outliers in the data. After that, various heuristics for identifying 4 inlier points are applied and finally, these 4 points are fed to the convex optimization scheme to recover camera translations and the 3D coordinates of the 4 points. The requirement of identifying 4 correct matches in multiple views makes this step of their algorithm sensitive to outliers. In contrast, we remove incorrect pairwise rotations before estimating camera orientations and do not rely on identifying 4 good matches.

Our approach also has clear similarities to [92]. They use short cycles and a Bayesian

model to predict which of the estimated rotations are correct. However, camera positions and scene structure are estimated in a sequential manner. Another method to discard erroneous rotation estimates is given in [35]. There a random sampling over spanning trees is suggested.

In [73], cycles are also used as a means to estimate camera orientations. From a spanning tree they generate a set of fundamental cycles in the camera graph. For each of these cycles they compute its error, that is, the rotational deviation from the identity. This error is distributed over the respective rotations in the cycle to form a consistent cycle. Unlike our approach they cannot handle large errors (outliers) among the relative rotations.

6.2 Overview

The following is an outline of the proposed approach.

1. Feature extraction using SIFT [58] and matching between pairs of views.

2. Estimation of the relative orientation for pairs of views. A standard 5-point solver [66] is used in a RANSAC loop to get an initial solution, which is improved using local optimization.

3. Detection and removal of large errors among the relative rotations.

4. Estimation of camera orientations using the remaining relative rotations.

5. 3D reconstruction using the estimated camera orientations. The reconstruction is computed using L_∞ optimization with auxiliary variables are used to handle outliers; see Section 1.3.1.

6. Standard bundle adjustment to improve the 3D reconstruction.

The chapter is organized as follows. Section 6.3 shows that, in contrast to traditional systems, our system does not suffer from geometries with short baselines. In fact, it is important to use these geometries since they give accurate estimations for camera rotations. Section 6.4 gives some results concerning the cycles in the camera graph and rotational consistency. In Section 6.5 we present a robust method of estimating camera rotations in a non-sequential way. Combined with a robust method for simultaneous estimation of camera positions and 3D structure, this gives us a reliable initial solution, that is fed into the bundle adjustment routine. Section 6.6 presents comparisons with BUNDLER for a number of data sets. After a short discussion, Section 6.8 presents some theoretic results for fixed rotation axis.

6.3 Estimation with Short Baseline

Geometries with short baseline are problematic for most approaches to multiple view geometry. Reconstructing a scene from two views with short baseline will tend to give high uncertainty. If the initial pair in sequential structure from motion has short baseline the estimated structure will be poor and it will be hard to correctly estimate the pose of the next camera. Hence, many methods that use a sequential approach for reconstruction [15, 75], will try to find a view pair with a large baseline to start from. This is achieved by picking a pair views which is poorly explained by a homography transformation, see e.g. [75]. Still, short baseline can pose a problem at later stages as well. Estimating the pose of a new camera, many of the 3D points used in this estimation might be reconstructed from a short baseline and this is likely to cause a poor estimation.

Although the hierarchical methods are less sensitive to drift, degenerate geometries still need to be avoided since the structure is used for reconstruction. As noted in [65] the loss of feature points and the necessity of a reasonable baseline creates a trade off. That is, there is a sweet spot in terms of view separation, where calculation of multi-view geometries is best performed.

As the proposed approach prefer pairwise geometries having a lot of corresponding points it is likely that short baseline view pairs will be used. The general view of these geometries is that they are quite useless since the structure and translation cannot be well estimated. In contrast, this section will show that the rotational part of the camera is well defined even if the distance between the cameras is small or even zero. This is important to motivate the proposed approach.

Let us first look at the estimation of the relative orientation of two cameras with the same camera centre and no noise. It is easy to see that if we allow 3D points at infinity, any translation direction will do. However, as the following theorem shows it is only in some rare degenerate configurations that the rotation is not uniquely determined.

Before stating the theorem, we recall the definition of an essential matrix. Consider two views of the same scene. Let x and y be the projections in the two images of a 3D point X. By aligning the global coordinate system with the first camera, we can write the equations

$$\begin{aligned} \lambda x &= X, \quad \lambda > 0 \\ \gamma y &= R(X - t), \quad \gamma > 0, \end{aligned} \tag{6.1}$$

where R is the relative rotation and t is the translation; cf. Section 1.1. It should be clear from this that x, $R^T y$ and t are coplanar. Consequently,

$$x \cdot (t \times R^T y) = 0, \tag{6.2}$$

which can be written in matrix form as

$$x^T [t]_\times R^T y = 0. \tag{6.3}$$

Here $[t]_\times R^T$ is the so called essential matrix which relates corresponding image points in the two views.

Theorem 22. *Let x_i and y_i be unit vectors representing image points from a calibrated camera. Assume that x_i and y_i are related by a pure rotation*

$$y_i = Qx_i. \tag{6.4}$$

Now consider an essential matrix $[t]_\times R$ with $t \neq 0$ such that

$$y_i^T [t]_\times R x_i = 0 \tag{6.5}$$

Then $R = Q$ unless the image points y_i lie on a quadratic surface $y_i^T A y_i = 0$, where A has the eigenvalues λ_1,λ_2 and $\lambda_1 + \lambda_2$ with $\lambda_1\lambda_2 \leq 0$ and $A \neq 0$.

Proof. We let $E = [t]_\times RQ^T$ and note that

$$y_i^T E y_i = y_i^T [t]_\times RQ^T y_i = y_i^T [t]_\times R x_i = 0. \tag{6.6}$$

Consequently, $y_i^T (E + E^T) y_i = 0$ as well. Since E has the form of an essential matrix it follows from a theorem in [48] that $A = E + E^T$ has eigenvalues λ_1, λ_2 and $\lambda_1 + \lambda_2$, with $\lambda_1\lambda_2 \leq 0$.

It remains to show that $A \neq 0$. To this end, let n be a rotation axis of $S = RQ^T$. If $A = 0$ then

$$0 = An = (E + E^T)n = [t]_\times Sn - S^T [t]_\times n = t \times n - S^T t \times n, \tag{6.7}$$

and hence S has two orthogonal eigenvectors, n and $t \times n$ with eigenvalue 1. The only rotation matrix with this property is the identity. Hence, $I = S = RQ^T$ and $R = Q$. □

This theorem shows that in the noise-free case, the rotation can generally be determined even with zero camera distance. To see what happens when there is image noise a simple synthetic experiment was performed. A set of 50 3D-points were randomly generated in the unit cube $[-1, 1]^3$ and two calibrated cameras were placed in the points $(0, 0, 10) \pm r$ where r are random vectors of different lengths. Gaussian noise with standard deviation $= 0.0001$ was added to the image projections. Then the relative orientation of the cameras was estimated using a minimal 5-point solver followed by bundle adjustment. More precisely the minimal solver from [66] was used in a 50-iteration RANSAC loop. The solution having most inliers was picked as a starting point for bundle adjustment. This was repeated for different camera distances and the results were averaged over 250 runs. Figure 6.2 shows an example of the geometry and comparisons between the ground truth and the estimated relative rotations and translations. Note that since the scale of the reconstruction is arbitrary we can only compare the translation direction. Still the figures shows quite clearly that, as the camera distance decreases, the

translation error increases drastically while the rotation remains stable. Figure 6.2 also shows how the quality of the structure estimation depends on the baseline. By setting the scale according to the ground truth camera distance we can measure the euclidean error of each reconstructed 3D point. Since only a few points would be required to pose in another camera, the minimum error over the 50 points was deemed to be most relevant.

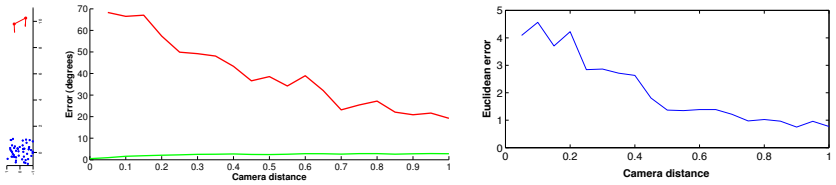

Figure 6.2: *Stability of the rotation estimate*. Left: An example of the geometry. Middle: Average error in degrees for the translation estimation (red line) and the rotation estimation (green line) for different camera distances. Right: Average minimum error for the structure estimation; see text for details.

This experiment shows that there is no reason not to use view pairs with short baseline in the rotation estimation. In fact, when the baseline is small the number of matchings is generally large and hence the rotation estimate will often be more accurate than with a larger baseline.

6.4 Cycles and Consistency

In this work we will use relative orientations, obtained from pairwise geometries, to estimate the absolute orientations of the cameras. The relative orientations induces a camera graph with edges where a relative orientation is available. Computing the product of rotations along a cycle in this graph should give roughly the identity matrix. Large deviations, inconsistencies, indicates an incorrectly estimated geometry.

Cycles have been used to detect inconsistency in camera graphs not only in this work but also in e.g. [35, 92, 73]. This section presents some results concerning the connection between cycles and consistency. First we provide a more precise definition of consistency. Note that d refers to the standard metric in $SO(3)$; see Section 1.3.3.

Definition 8. *Given a camera graph $G = (V, E)$ and error tolerance ϵ, we say that G is consistent if there exist orientations $R_1, \ldots, R_N$ satisfying*

$$d(R_i, \tilde{R}_{ij} R_j) \leq \epsilon \text{ for all } (i, j) \in E. \tag{6.8}$$

Our first result gives a necessary constraint on cycles for a graph to be consistent. Essentially the same result can also be found in [73] but we present a shorter proof. Let

us first recall that $\tilde{R}_{ij}$ is the estimated rotation between camera i and j and that each estimated rotation corresponds to an edge in the camera graph.

Theorem 23. *Consider a camera graph G that consists of a single simple cycle $i_1, i_2, i_3 \dots i_n, i_1$. If the estimated rotations along this cycle satisfy*

$$d(\tilde{R}_{i_1 i_2}\tilde{R}_{i_2 i_3}\dots\tilde{R}_{i_n i_1}, I) = \omega, \tag{6.9}$$

then $\epsilon = \omega/n$ is the smallest ϵ such that G is consistent with Definition 8.

Proof. To prove that G is consistent with $\epsilon = \omega/n$, it is sufficient to find rotations R_{ij} such that $d(R_{ij}, \tilde{R}_{ij}) < \omega/n$ and $R_{12}R_{23}\dots R_{n1} = I$. To find R_{12}, let $D = \tilde{R}_{12}\dots\tilde{R}_{n1}$. This is a rotation ω radians around some axis. Let $D_{\omega/n}$ be a rotation around the same axis but $-\omega/n$ radians and set $R_{12} = D_{\omega/n}\tilde{R}_{12}$. Then

$$d(R_{12}\tilde{R}_{23}\dots\tilde{R}_{n1}, I) = \omega - \omega/n. \tag{6.10}$$

By repeating this scheme with $D = \tilde{R}_{23}\dots\tilde{R}_{n1}R_{12}^T$ we can compute R_{23} such that the error decreases to $(n-2)\omega/n$ and the result follows by induction. □

Let C denote a cycle and $|C|$ the length of this cycle. Theorem 23 shows that if the error in some cycle is larger than $|C|\epsilon$ then the graph is not consistent. However, even with smaller errors it is likely that the cycle contains outlier rotations. The following experiment illustrates how the error in a cycle depends on the length of the cycle if there are no outlier rotations. If the error in a cycle is significantly larger than this, then there is probably at least one outlier rotation in that cycle.

Fifty pairs of views were generated in the exact same manner as in the experiment of the previous section; see that section for details. For each pair a rotation was estimated using RANSAC followed by bundle adjustment. This rotation was saved as well as the ground truth. Let R_i be the ground truth rotation for the ith view pair and let $\tilde{R}_i$ be the estimated rotation. For each k the error $d(R_1R_2\dots R_k, \tilde{R}_1\tilde{R}_2\dots\tilde{R}_k)$ was measured. The error for different k's is shown in Figure 6.3 averaged over 50 runs. As might be expected, the error is approximately proportional to the square root of the cycle length.

6.5 Robust Estimation of Orientations

We will now describe how to robustly estimate the orientations of all views with respect to a global coordinate system. This is done by considering relative rotations estimated from pairs of views. We start with a set of point-to-point correspondences between pairs of views. Any feature descriptor can be used to determine these, e.g. SIFT [58]. For each pair with sufficiently many corresponding points, we estimate the relative orientation using a RANSAC algorithm based on a minimal five-point solver [66]. The estimated relative orientation is improved by standard local optimization, i.e. bundle adjustment.

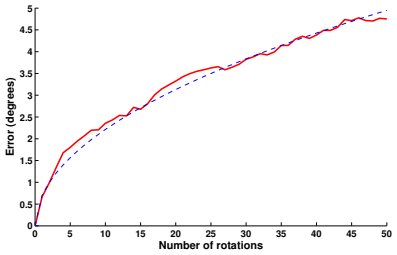

Figure 6.3: Synthetic experiments to show how the error in a cycle depends on the length of that cycle. The red curve shows the average over 50 experiments and the blue dashed curve shows a fitted function $y = c\sqrt{x}$.

Let $\tilde{R}_{ij}$ be the thus estimated rotation between camera i and j. We also save the number of inliers from this estimation as a measure of certainty.

Ideally there would exist absolute orientations R_i such that

$$R_i = \tilde{R}_{ij} R_j \text{ for all } (i,j) \in E. \tag{6.11}$$

Due to uncertainty in the matching process and camera model, this will not be the case. Instead we will have to deal with low-level noise as well as completely inconsistent rotations.

6.5.1 Handling Low-Level Noise

To deal with low-level noise, we will use the method from [34], but with a small modification. This approach is based on the quaternion representation of rotations; see Section 1.3.3. Let $\tilde{q}_{ij}$ be the quaternion representation of the estimated relative rotation $\tilde{R}_{ij}$. Rotation composition corresponds to quaternion multiplication, which is linear in the quaternion coordinates. Thus (6.11) can be written linearly as

$$\tilde{Q}_{ij} q_j - q_i = 0, \tag{6.12}$$

where $\tilde{Q}_{ij}$ is a 4×4 matrix corresponding to quaternion multiplication by $\tilde{q}_{ij}$. In [34] it is suggested to solve these equations in a least squares sense.

$$\min_{q_1,\ldots,q_N} \sum_{(i,j)\in E} |\tilde{Q}_{ij} q_j - q_i|^2. \tag{6.13}$$

To motivate this, look at two unit quaternions p and q corresponding to rotations R_p and R_q such that $d(R_p, R_q) = \alpha$. Assuming that α is small, (1.25) yields,

$$|p-q|^2 = \langle p-q, p-q \rangle = 2 - 2\cos(\alpha/2) = 4\sin^2(\alpha/4) \approx \alpha^2/4. \tag{6.14}$$

Thus the sum in (6.13) is approximately proportional to the sum of squared angular errors. However, optimizing (6.13) with the norm constraint, $|q_i| = 1$ is difficult so [34] suggests relaxing this constraint. The relaxation allows some constraints to be weighted down, but experiments indicate that this is a minor problem.

It was noted in [23] that the quaternion ambiguity (q and $-q$ represents the same rotation) may cause this method to fail. Say for example that the camera has moved around a building, while rotating an angle 2π. Let $q_1 = (1, 0, 0, 0)$. If we choose quaternion representation in the standard way, we will constrain the orientation quaternions q_i to move smoothly on the unit sphere of quaternions. But this means that when we are back where we started the orientation has just moved halfway around the sphere of unit quaternions. So for the linear equations to hold we have to represent q_1 with $(1, 0, 0, 0)$ in some equations and $(-1, 0, 0, 0)$ in others.

The approach presented in the next section to remove inconsistent relative rotations will also provide estimates $\bar{q}_i$ for the camera orientations. This gives us a way to resolve the ambiguity problem:

For all $(i, j) \in E$:
1. Represent $\tilde{R}_{ij}$ with a quaternion $\tilde{q}_{ij}$.
2. Compute $\tilde{Q}_{ij}$ as the matrix representation of $\tilde{q}_{ij}$, see (6.13).
3. If $|\bar{q}_i - \tilde{Q}_{ij}\bar{q}_j| > |\bar{q}_i + \tilde{Q}_{ij}\bar{q}_j|$, set $\tilde{Q}_{ij} = -\tilde{Q}_{ij}$.

6.5.2 Handling Large Errors

Inconsistent feature matching can lead to large errors among the estimated rotations. A typical example is given Figure 6.5, where the pairwise geometry has captured similar structures on the different sides of a road roller. To handle this type of problems we need some way to detect and remove large errors among the relative rotations.

If there are outliers among the estimated relative rotations, then the camera graph will not be consistent with respect to Definition 8. In these cases we would like to find a large consistent subgraph of the camera graph. Moreover it is natural to take into account the reliability of each estimated rotation. Let p_{ij} be the probability that the estimated relative rotation $\tilde{R}_{ij}$ is an outlier. In Section 6.3 we saw that the accuracy of the estimated rotations did not depend directly on the camera distance. Thus it is reasonable to model p_{ij} as a decreasing function of the number of inliers of that estimation, i.e. w_{ij}. For simplicity, we choose to optimize the sum of w_{ij}'s rather than trying to estimate the probabilities. Hence, we seek those camera orientations which are supported by the maximum number of point correspondences. However, the same approach can be used when estimates of the probabilities exist.

Problem 8. *Given a connected camera graph $G = (V, E)$ and edge weights w_{ij}, we want*

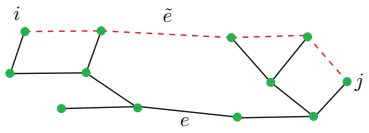

Figure 6.4: Black edges lie in T and red dashed edges lie in $\tilde{P}$. Removing e from the spanning tree creates two components.

to find a consistent subgraph $G_c = (V, E_c)$ *that maximizes*

$$\sum_{(i,j)\in E_c} w_{ij}. \tag{6.15}$$

This formulation involves finding both the absolute rotations R_i and the set of consistent rotations E_c. As we saw in Section 6.4 we can attack this problem by considering cycles in the camera graph. We assume that the camera graph is connected, otherwise we consider a subgraph. Considering all cycles in the graph is rarely feasible. To achieve reliable orientation estimations, we start from a spanning tree; cf. Section 1.3.2. If there are cycles in the graph there are also multiple ways to choose spanning trees. Govindu [35] used a RANSAC-type algorithm, sampling over different spanning trees. Instead, we look for a maximally reliable spanning tree and improve this by working directly with the camera graph.

Following Problem 8, it seems natural to seek a maximum-weight spanning tree. Such a tree is easily found using standard algorithms; see Section 1.3.2. Since a single outlier rotation will ruin the whole estimation a reasonable alternative is to maximize the weakest edge used for estimation. It turns out that the two formulations have the same solution; see Theorem 24.

Theorem 24. *If T is a maximum spanning tree and i and j are arbitrary nodes, then there is a path $P(i, j)$ between them such that $R(P) = \min_{(i,j)\in P} w_{ij}$ is maximal and $P \subseteq T$.*

Proof. Consider two nodes i and j. Since T is a spanning tree it contains a path, P, between these nodes. Now assume for contradiction that there is a path $\tilde{P} \not\subseteq T$ such that

$$R(\tilde{P}) > R(P). \tag{6.16}$$

Let e be the weakest edge in P. If we remove e from T it will divide the spanning tree into (i) nodes connected with i and (ii) nodes connected with j, see Figure 6.4. Clearly $\tilde{P}$ contains some edge $\tilde{e}$ connecting these components and by (6.16), $w_{\tilde{e}} > w_e$. But then, replacing e with $\tilde{e}$ creates a spanning tree with higher total edge weight than T. Since T is a maximum spanning tree, this is a contradiction. □

Assuming that the generated spanning tree contains no outlier rotations, we now have means to detect outliers among the other relative rotations. It easy to see that adding

any edge to a spanning tree will generate a cycle. If multiplying the rotations along this cycle yields a result far from the identity rotation, then the cycle must contain an outlier rotation. Let C denote a cycle and let R_C be the composition of all rotations along the cycle. Motivated by the experiment in Section 6.4 a cycle is considered inconsistent if

$$d(R_C, I) > \sqrt{|C|}\epsilon, \tag{6.17}$$

and in this case we will not use that relative rotation; see Algorithm 16. Note that the spanning tree also enables us to get initial estimates for all the camera orientations, as required by the scheme in Section 6.5.1.

Algorithm 16 Consistent orientations

Compute a maximum spanning tree, T.
Set $E_c = T$.
for each $e \in E \setminus T$
  Let C be the cycle formed by e and T.
  if the error in C is less than $\sqrt{|C|}\epsilon$
    $E_c = E_c \cup e$
Estimate camera orientations from E_c using the algorithm in Section 6.5.1.

The absolute orientations yielded by this Algorithm 16 can be viewed as a new spanning tree, so we repeat steps 2 to 4 to yield a better solution to Problem 8. We also use some other search heuristics:

- Add an outlier rotation and estimate new absolute rotations.

- Set the weight of an outlier rotation to a very high number, compute a new maximum spanning tree and re-estimate the orienatations.

As the experiments will show, these simple heuristics often work remarkably well. Since the spanning tree will consist of those relative rotations that had the highest number of inliers, large errors in the spanning tree are very unlikely.

6.6 Experiments

The developed approach was tested on a number of real image sequences. Figures 6.5, 6.6 and 6.7 show some screen shots. This section presents some performance statistics and a comparison to the state-of-the-art software BUNDLER.

Data. The set of image collections have been obtained (i) from the Internet, in particular, FLICKR and (ii) by taking photos with standard digital cameras. Image sizes vary from a couple of hundred to up to 3000 pixels in width. The number of images in a sequence ranges from 34 to a couple of hundreds. Only images where it has been possible to extract the focal length from the EXIF tag has been processed. The principal point is assumed to be in the middle of the image. The skew is set to zero and the aspect ratio to one.

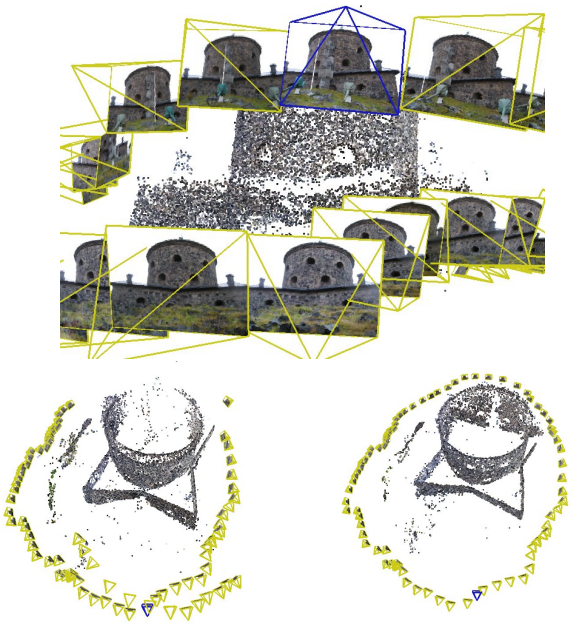

Figure 6.5: *Castle*. The top and left figures shows the 3D result from BUNDLER and on the right, our result is given. By careful inspection, one can see that the top and bottom image rows of the top figure display *different* facades of the castle. (A window is blocked by stairs in the top row.) This confusion of facades yields an incomplete and false reconstruction. Using rotational consistency, a complete trajectory is obtained.

Implementation details. For the feature extraction and matching, as for the extraction of focal lengths, we use exactly the same setting as in BUNDLER. More precisely, the matching stage is based on standard SIFT matching with default settings. Note that the input to our system and BUNDLER is identical since the same software is used. For the estimation of pairwise epipolar geometries, we allow 1000 RANSAC iterations. The

threshold is set to 3 pixels for a point correspondences (measured from the epipolar line in each image). If more than 10 point correspondences are obtained for an image pair, the two-view geometry is kept for later processing. We always check for cheirality (positive depths). When computing camera translation and 3D points, we allow a larger error, namely 10 pixels, as the initial estimates of rotations may be slightly off.

All our algorithms have been implemented in MATLAB. Given estimated two-view geometries, running times are typically between 5-10 minutes depending on the number of images, and the number of extracted feature points. For all sequences, the same parameter settings has been used.

Closed loop sequences. For our first experiments we have chosen to use closed loop sequences. For these datasets it is easy to detect if the method fails by investigating the ability to close the loop. Note that since there is no independent system that is guaranteed to give the true reconstruction it is very difficult to obtain the ground truth. The only way that we really can determine the quality of the reconstruction is by visual inspection. Note that, in our system, the ordering of the images is *nowhere* used. However, the images are taken in order so this gives us a way to check if correct epipolar geometries have been computed.

The closed loop sequences are the CASTLE (see Figure 6.5), ROAD ROLLER (see Figure 6.1) and RAILROAD. In all of these three sequences there are repeated textures introducing false two view geometries. Because of these false geometries BUNDLER fails to reconstruct the loop. Figure 6.5 shows that the front and the back of the castle are confused. Figure 6.6 shows the failed reconstructions for the other two datasets.

Figure 6.8 shows the two-view geometries that passed the RANSAC stage with at least 10 correct correspondences are plotted. An edge in the (circular) camera graph corresponds to such a two-view geometry. As can be seen, there are many false edges occurring for cameras far away from each other. By enforcing rotational consistency, hence computing a solution for Problem 8, the camera graph given in the second image of Figure 6.8 is obtained.

Leave one out test. STREET. In our method, all images are handled in a uniform manner. This is in contrast to BUNDLER which selects an initial epipolar geometry to base the reconstruction on and then sequentially adds new images. To test this dependency, we tried to reconstruct the same sequence with one image removed. This was repeated for all 99 images. The reconstructions were validated by registering to the original reconstruction. While our method is unaffected by removing one image in all cases, there are two cases for which BUNDLER fails to reconstruct the whole scene (and in these cases, only 45 and 57 cameras are reconstructed, respectively), see Figure 6.7.

Regular scenes. APARTMENT. In this two bedroom apartment, there are (natural) weak geometry links between different rooms. It is difficult to detect any difference from

the two apartment images in Figure 6.6. Both methods provide satisfactory results, and the differences are minor, however it turns out that there are 6 images of the bathroom that BUNDLER is not able to incorporate into the reconstruction. The final data set is the Cathedral of Linköping. Both methods produces satisfactory results for this data set.

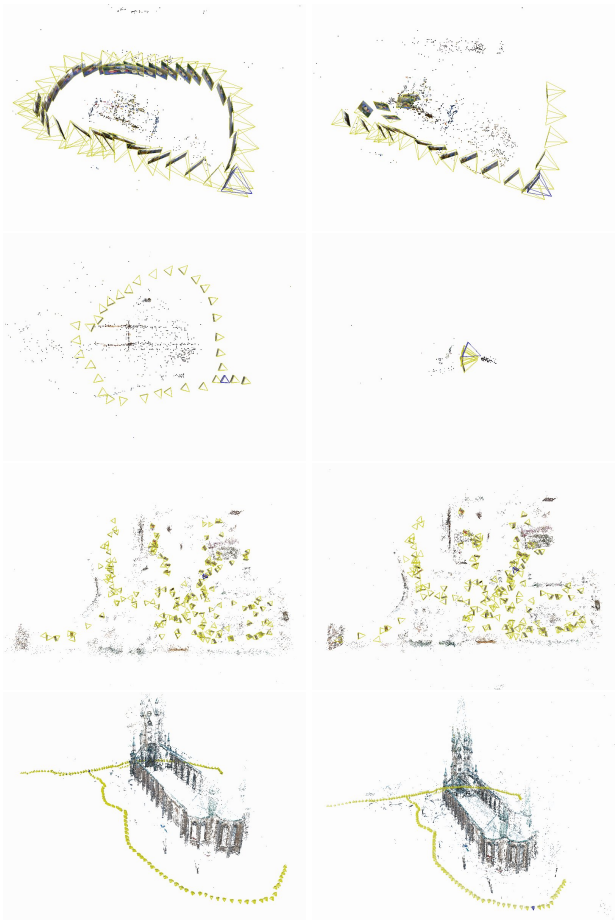

Figure 6.6: Reconstructions for the Road Roller, Railroad, Apartment and Cathedral data sets. To the left is our reconstruction and to the right the reconstruction obtained using BUNDLER.

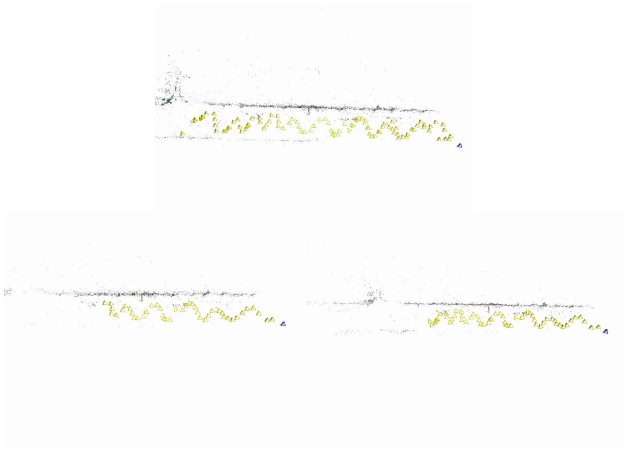

Figure 6.7: Three solutions from the leave-one-out test. Top: all cameras. Bottom: two incomplete reconstructions; see text for details.

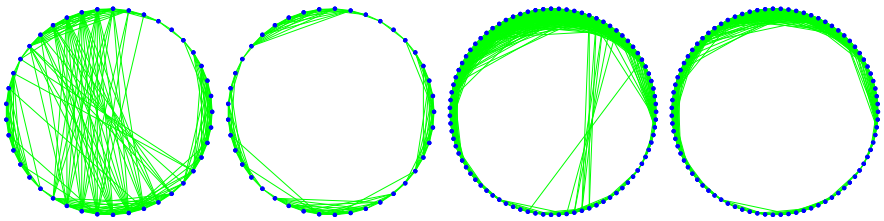

Figure 6.8: The first two (and the last two) figures show camera graphs before and after removing inconsistent rotations, respectively, for the ROAD ROLLER and CASTLE sequences. Each edge between two camera nodes corresponds to a valid epipolar geometry. For this illustration the cameras have been placed on a circle using prior knowledge of the true geometry, but this prior knowledge has *not* been used to detect the erroneous rotations.

6.7 Discussion

We have presented a system for large-scale 3D reconstruction for unordered images. Compared to standard incremental approaches, we have shown that short baselines do not cause any failures, on the contrary, they are reliable building blocks in our system, since matching between such views is relatively simple. Further, in our non-sequential system, we have demonstrated improvements with respect to state-of-the-art regarding loop-closing, detecting repetitive structures and obtaining a good global solution without depending in a specific base pair. These features have been supported both by theoretical results as well as experimental comparisons on real data.

A possible disadvantage is computational effort which may hinder the ability to scale to several thousands of images. However, the bottleneck for our system as well as many sequential systems is the pairwise image matchings. Another weakness is that repetitive structures with consistent rotations, for example, two parallel billboards will not be detected by rotational consistency and would have to be handled by the robust estimation stage. We have not encountered such a failure case in practice and we leave it to future work.

6.8 A Result for Fixed Rotation Axis

Quite frequently, e.g. when working with vehicle-mounted cameras, the rotation axis of the cameras is known and fixed. In these cases, we can prove a significantly stronger result than Theorem 23. This result is provided here as bonus material.

Theorem 25. *Assume that the rotation axis is fixed and that $N\epsilon < \pi/3$ where N is the number of cameras, then the solution to Problem 8 is the maximum weight subgraph $G_c \subset G$ such that any simple cycle $i_1, i_2, i_3, \ldots, i_n, i_1$ in G_c satisfies*

$$d(\tilde{R}_{i_1 i_2}\tilde{R}_{i_2 i_3}\ldots\tilde{R}_{i_n i_1}, I) \leq n\epsilon. \tag{6.18}$$

Lemma 26. *Given an $\epsilon > 0$, rotations R_k and natural numbers $n_k \in \mathbb{N}$ such that $n_k\,\epsilon < \pi/3$ for all k and*

$$d(R_i, R_j) \leq (n_i + n_j)\,\epsilon \tag{6.19}$$

for any pair i, j, then there exists an R such that

$$d(R_k, R) \leq n_k\,\epsilon \text{ for all } k. \tag{6.20}$$

Proof. We can represent R with a unit 2-vector q. For each k (6.20) restricts q to an interval, I_k, on the unit circle. It should be clear from (6.19) that any pair of I_k's have non-empty intersection and then Theorem 16 implies that $\bigcap_k I_k$ is non-empty as well. □

For the next part of the proof we will need the following relation for rotations R_i and S_i

$$\begin{aligned} &d(R_1R_2, S_1S_2) = d(R_1, S_1S_2R_2^T) \leq \\ &d(R_1, S_1) + d(S_1, S_1S_2R_2^T) = d(R_1, S_1) + d(R_2, S_2), \end{aligned} \tag{6.21}$$

where we used the triangular inequality for $SO(3)$. Note that the relation generalizes to longer sequences.

Lemma 27. *Assuming that prerequisites of Theorem 25 are satisfied, consider two simple paths P and Q from node 1 to node k, having lengths n_P and n_Q respectively. Let $\tilde{R}^P$ and $\tilde{R}^Q$ be the composition of all relative rotations along these paths. Then $d(\tilde{R}^P, \tilde{R}^Q) \leq (n_P + n_Q)\epsilon$.*

Proof. If P and Q constitute one simple cycle, then the result is almost immediate. If not, then we can divide P and Q into n parts such that each pair P_i, Q_i make out a simple cycle (or are identical). To see this we follow the path P until we reach a node which is also in Q. This is the first division point. We remove these first parts from P and Q and continue along P until we find the next node which is also in $Q \setminus Q_1$. This procedure will divide paths P and Q in the desired manner. Assuming such a partitioning and using (6.21), we get

$$d(\tilde{R}^P, \tilde{R}^Q) \leq d(\tilde{R}^{P_1}, \tilde{R}^{Q_1}) + \ldots + d(\tilde{R}^{P_n}, \tilde{R}^{Q_n}) \tag{6.22}$$

and since these pairs make out simple cycles satisfying (6.18) and every edge in P or Q is in exactly one of these cycles, the lemma follows. □

Proof. (Theorem 25) It is clear that if E_c is the edge set solving Problem 8 then any cycle in E_c satisfies (6.18). We need to prove that given an edge set E_c such that all cycles satisfy (6.18), then there exists $\{R_1, \ldots, R_N\}$ satisfying (6.8).

Let $R_1 = I$. Pick any node c. Each path P_k from node 1 to node c gives an estimate of the absolute orientation R_c. We denote it $\tilde{R}_{P_k}$. According to Lemma 27 any pair of such estimates satisfy $d(\tilde{R}_{P_j}, \tilde{R}_{P_k}) \leq (n_j + n_k)\epsilon$. Thus using Lemma 1, we can introduce a new edge from node 1 to node c equipped with a rotation R_c such that all cycles this generates, satisfy $d(\tilde{R}_{i_1 i_2} \ldots \tilde{R}_{i_n i_1}, I) \leq (n-1)\,\epsilon$. If we repeat this procedure for all nodes, we will get a graph such that any cycle satisfies

$$d(\tilde{R}_{i_1 i_2} \ldots \tilde{R}_{i_n i_1}, I) \leq (n-k)\epsilon, \tag{6.23}$$

where k is the number of *new* edges in the cycle. Specifically,

$$d(R_i^T \tilde{R}_{ij} R_j, I) \leq (3-2)\epsilon = \epsilon \tag{6.24}$$

so (6.8) holds. □

This theorem implies that we can obtain G_c in Problem 8 by solving a minimum vertex cover problem for a hypergraph; see Section 1.3.2. The nodes of the hypergraph are the estimated relative rotations $\tilde{R}_{ij}$ and each cycle in the camera graph that does not satisfy (6.18) is represented by a edge in the hypergraph, connecting all the relative rotations in that cycle. To get a consistent set of rotations while keeping as much data as possible, we want to find a minimum vertex cover for this hypergraph.

Chapter 7

Structure from Motion for 1D Cameras

In the previous chapter, we used the fact that the structure from motion problem with known rotations can be attacked by means of convex optimization; cf. Section 1.3.1. This chapter shows similar results for one-dimensional (1D) cameras. It turns out that several important problems in 1D vision can be solved using linear programming.

7.1 Introduction

One-dimensional cameras are used to provide inexpensive and reliable navigational systems for autonomous vehicles; see Figure 7.1. Strips of reflector tape are put on walls and objects along the route of the vehicle; cf. [45]. A laser scanner measures the direction from the vehicle to the different strips of tape, but not the distance. This is the 1D camera. These angles can be used to determine the position of the vehicle.

The use of 1D cameras in navigation is the main motivation for our work. However, an understanding of the one-dimensional case can be of use with ordinary cameras as well. In the case of an ordinary camera moving and rotating in a plane, 1D vision can be used as an efficient and accurate approximation; cf. [29]. The disadvantage of approximating is balanced against the possibility of obtaining globally optimal estimates, using the techniques from this chapter. In [2], a similar approximation is used for autocalibration.

Another interesting application was given in [70], where it is shown that structure from motion problems using line features and an affine camera model can be reduced to the structure from motion problem in 1D vision.

Apart from all this there are theoretical insights to gain from the study of 1D vision and it has already yielded several ideas on how to handle similar problems in higher dimensions.

One of the key problems in 1D as well as in ordinary vision is the structure from motion problem; cf. Section 1.2. In the case of autonomous vehicles, this is normally done when the system is installed to create a map which can then be used for localization. High accuracy is needed since the precision of the navigational system can never be higher

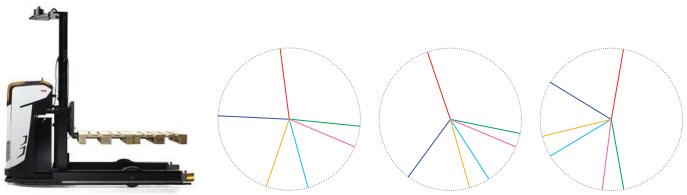

Figure 7.1: Left: An autonomous guided vehicle that uses one-dimensional vision to navigate. Right: Three example 1D images of 7 object points in the plane.

than that of the map.

This chapter presents a method for finding the globally optimal solution to this problem with respect to the L_∞ norm of the reprojection errors. The approach can also handle underdetermined problems, minimal cases, and missing data. Using the same framework, we also show how to find optimal solutions to the triangulation and camera pose problems.

As an illustration, Figure 7.1 shows 3 spherical images of 7 object points and Figure 7.2 shows the optimal geometry obtained with the proposed method.

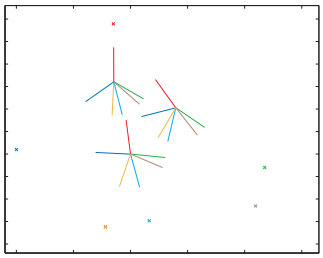

Figure 7.2: The globally optimal solution in the L_∞ sense to the structure from motion problem for the images in Figure 7.1.

The chapter is organized as follows. Section 7.2 gives a brief introduction to the geometry of the problem and Section 7.3 discusses the problems of triangulation and camera pose showing that they can be solved efficiently. An optimization method for the structure from motion problem is presented in Section 7.4 along with the required theoretic results. Finally, Section 7.5 presents some experiments illustrating the performance of the optimization method.

7.2 1D Retina Vision

We will now give a brief introduction to 1D vision as used in autonomous vehicle navigation. A laser navigated vehicle is shown in Figure 7.1. Mounted on top of the vehicle is a laser scanner. A vertical laser beam generated in the scanner is deflected by a rotating mirror at the top of the scanner. When the laser beam hits a strip of retroreflective tape, a beacon, a large part of the light is reflected back to the scanner. The reflected light is processed to find sharp intensity changes. When this happens the bearing of the laser beam relative to a fixed direction in the scanner is stored. Note that only the bearing and not the distance to the beacon is measured.

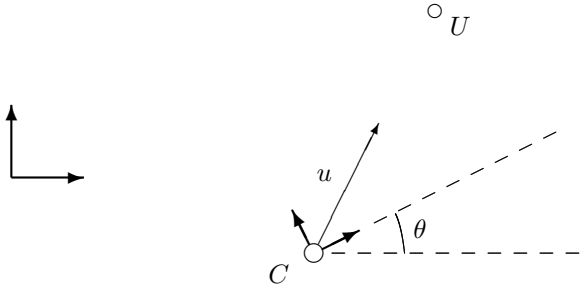

Figure 7.3: The measured bearing u (in a camera frame) as a function of scanner position C, scanner orientation θ, and object position U.

Figure 7.3 shows the setup with one camera and one reflector. We introduce a world coordinate frame which will be held fixed with respect to the scene. The position of the camera in the world coordinate system is given by $C \in \mathbb{R}^2$ and the position of an object point (a beacon) in the same coordinate frame is given by U. We choose to represent the measured bearing of a beacon with a unit vector u. Note that this is given in a local camera frame. The relationship between this local frame and the world coordinate frame is given by a rotation matrix R parameterized by a single angle θ.

In some cases it is convenient to identify the pair (R, C) with a camera matrix

$$P = \begin{pmatrix} a & -b & c \\ b & a & d \end{pmatrix} = k\,(R \mid -RC), \quad k > 0. \tag{7.1}$$

Note that every camera matrix of this form where $a \neq 0$ or $b \neq 0$ is associated with exactly one pair (R, C). If we also allow solutions with $a = b = 0$, which corresponds to a camera position at infinity and will not yield any good solutions in practice, we can identify the set of all camera matrices with $\mathbb{R}^4 \setminus \{\mathbf{0}\}$. This will be useful when discussing quasiconvexity. Note that we do not allow the degenerate case when all elements of P

are zero. When working with camera matrices we will use homogenous coordinates for the object (or scene) points, denoted by $\bar{U}$. Image points are represented by unit vectors u corresponding to the measured bearing. Note that the image vectors should not be regarded as a homogeneous quantity.

If there had been no errors whatsoever, we would have

$$\lambda\, u = R\,(U - C) = P\bar{U}, \tag{7.2}$$

for some positive depth λ. Since we are mostly interested in overdetermined problems, (7.2) cannot be satisfied exactly. Instead we are forced to solve an optimization problem. Motivated by the previous section we choose to minimize the L_∞ norm of the reprojection errors. The reprojection error ϵ is the angle between the measured bearing and the modeled bearing. Let $\angle(u, v)$ denote the angle between two vectors u and v, that is, $\arccos(u \cdot v)$. Then the reprojection error is given by

$$\epsilon = \angle\left(u, R\,(U - C)\right) = \angle\left(u, P\bar{U}\right). \tag{7.3}$$

7.3 Triangulation and Camera Pose Estimation

Before moving on to the general structure from motion problem, which is the main subject of this chapter, we consider the simpler problems of triangulation and camera pose estimation.

Consider a number of cameras seeing the same object. If the positions and orientations of the cameras are known, the goal is to determine the position of the object. This is called the triangulation problem.

Problem 9. *Given bearings $u_1, \ldots u_m$ of a single object from m different cameras $P_1, \ldots, P_m$, the* $\boldsymbol{L_\infty}$ **triangulation problem** *is to reconstruct the point U while minimizing*

$$f_{int}(U) = \max_i \angle\left(u_i, P_i\bar{U}\right). \tag{7.4}$$

If instead the positions of a number of objects are known, then the goal is to determine the position and orientation of the camera seeing these objects. This is the camera pose problem.

Problem 10. *Given n bearings $u_1, \ldots, u_n$ and the corresponding object points $U_1, \ldots, U_n$ the* $\boldsymbol{L_\infty}$ **camera pose problem** *is to find the camera matrix P such that*

$$f_{res}(P) = \max_j \angle\left(u_j, P\bar{U}_j\right) \tag{7.5}$$

is minimized.

These two problems are in a sense easy to solve. We shall see that both of them can be formulated as quasiconvex problems.

Lemma 28. *For any $\Delta < \pi/2$ the sets*

$$\{U \mid f_{int}(U) < \Delta\} \subset \mathbb{R}^2 \quad \text{and} \quad \{P \mid f_{res}(P) < \Delta\} \subset \mathbb{R}^4$$

are convex.

Proof. First note that if $u \cdot P\bar{U} \leq 0$ then the angle between u and $P\bar{U}$ is at least $\pi/2$ or $P = \mathbf{0}$. Thus we can assume $u \cdot P\bar{U} > 0$.

For a given U, P and corresponding u we have

$$\left| \frac{u \times P\bar{U}}{u \cdot P\bar{U}} \right| = \tan \angle \left(u, P\bar{U}\right), \tag{7.6}$$

where we define the cross-product for 2D vectors to be the scalar $u \times v = u_x v_y - v_x u_y$. Since $u \cdot P\bar{U} > 0$, checking whether $\angle \left(u, P\bar{U}\right) \leq \Delta$ is equivalent to

$$|u \times P\bar{U}| \leq (u \cdot P\bar{U}) \tan \Delta. \tag{7.7}$$

In the triangulation case, u and P are known and in camera pose case, u and U are known. Hence these equations are linear in the unknowns, and thus the constraints correspond to the intersection of half-planes and are convex. Note that in the camera pose case, these sets do not include the degenerate $P = \mathbf{0}$. □

Note that if we use the bisection algorithm, this result also tells us that the feasibility problems can be put as linear programs.

7.4 Structure from Motion

In the next problem we will assume that neither the positions of the objects nor the positions and orientations of the cameras are known. We will assume that the correspondence problem is solved, i.e., that it is known which measured bearings correspond to the same object. If the problem is deduced from ordinary vision, this correspondence can be decided using features in the two-dimensional image. In case of one-dimensional cameras the correspondence can be estimated with a RANSAC-type algorithm [30].

To simplify notations we introduce bold-face letters, for example, $\mathbf{R}$, which denotes a set of rotation matrices $\mathbf{R} = (R_1, \ldots, R_m)$.

Problem 11. *Consider n different points visible in m cameras. Given the bearings u_{ij} of point j in camera i, the* $\boldsymbol{L_\infty}$ **structure from motion problem** *is to find the cameras $(\mathbf{R}, \mathbf{C})$ and the object positions $\mathbf{U}$ that minimizes the maximal reprojection error,*

$$f\left(\mathbf{R}, \mathbf{C}, \mathbf{U}\right) = \max_{i,j} \angle \left(u_{ij}, R_i(U_j - C_i)\right). \tag{7.8}$$

Unfortunately this problem does not have the nice properties of the triangulation and camera pose problems in the previous section. The reason is that when both P and U are unknown (7.7) is normally not a convex condition. Nonetheless, quasiconvexity will play an important role for this problem as well.

The basic idea of our optimization scheme is to first consider optimization with fixed camera orientations, and then use branch and bound over the space of possible orientations. A problem here is that, especially with many cameras, the manifold of possible orientations is large. A method to reduce this manifold using linear conditions on the orientations is presented in Section 7.4.3.

7.4.1 Optimization with Fixed Orientations

In this section we prove that if we fix orientations in the structure from motion problem, we get a quasiconvex goal function.

Definition 9. *We define the function*

$$d(\boldsymbol{\theta}) = \min_{\mathbf{C},\mathbf{U}} f\big(\mathbf{R}(\boldsymbol{\theta}), \mathbf{C}, \mathbf{U}\big) \tag{7.9}$$

where $\mathbf{R}(\boldsymbol{\theta})$ *are the rotation matrices corresponding to* $\boldsymbol{\theta}$*.*

Lemma 29. *For any* $\Delta < \pi/2$ *and a given* $\boldsymbol{\theta}$*, the problem of determining whether*

$$d(\boldsymbol{\theta}) \leq \Delta \tag{7.10}$$

can be cast as a linear programming feasibility problem.

Proof. Since the orientations are fixed we can without loss of generality assume that orientations have been corrected for and simply assume that $R_i = I$ for all $R_i \in \mathbf{R}$. Then, for $\epsilon = \angle\big(u, P\bar{U}\big)$,

$$\frac{u \times (U - C)}{u \cdot (U - C)} = \frac{|u||(U - C)| \sin\epsilon}{|u||(U - C)| \cos\epsilon} = \tan\epsilon.$$

The constraint that

$$\epsilon \leq \Delta$$

is equivalent to

$$|u \times (U - C)| \leq \tan\Delta\,(u \cdot (U - C)),$$

which constitutes two linear inequality constraints in the unknowns U and C. □

This means that we can use linear programming to determine if the minimal L_∞ norm is less than some certain bound Δ. Moreover, using bisection we can get a good estimate of the minimal L_∞ norm.

To get better convergence we modify the normal bisection algorithm slightly. The idea is to seek a solution (C^*, U^*) to the problem in Lemma 29 that lies in the interior of the feasible space. For such a solution the L_∞ norm of the reprojection errors might be smaller than the current Δ, say $\Delta^\star$. Then one knows that the minimal L_∞ norm, $d(\boldsymbol{\theta})$ must be smaller than this $\Delta^\star$. To find such an interior solution we introduce a new variable k and try to maximize k under the constraints

$$|u \times (U - C)| + k \leq (u \cdot (U - C)) \tan \Delta.$$

We can now present an algorithm for finding the minimal L_∞ norm for fixed orientations $\boldsymbol{\theta}$.

1. Check if there is a feasible solution with all reprojected errors less than $\pi/2$. This corresponds to $\tan \Delta = \infty$ in the equations above. This can be solved by a simpler linear programming feasibility test. Use only $(u \cdot (U - C)) > 0$. If this is feasible then continue, otherwise return $d_{min} > \pi/2$.

2. Let $\Delta_l = 0$ and $\Delta_h = \pi/2$ be lower and upper bounds on the minimal L_∞ error norm.

3. Set $\Delta = (\Delta_h + \Delta_l)/2$. Examine whether $d(\boldsymbol{\theta}) \leq \Delta$. If this is the case calculate $\Delta^* = f(\mathbf{R}(\theta), \mathbf{C}^*, \mathbf{U}^*)$ for the feasible solution and set $\Delta_h = \Delta^\star$. Otherwise set $\Delta_l = \Delta$.

4. Iterate step 3 until $\Delta_h - \Delta_l$ is below a predefined threshold.

An example on how $d(\boldsymbol{\theta})$ might look is shown in Figure 7.7.

7.4.2 Lipschitz Continuity

To get further, we need an idea of how $d(\boldsymbol{\theta})$ depends on the camera orientations in $\boldsymbol{\theta}$. This is given by the following lemma.

Lemma 30. *The function d satisfies*

$$d(\boldsymbol{\phi}) - d(\boldsymbol{\theta}) \leq \max_j |\phi_j - \theta_j| \tag{7.11}$$

which implies that it is Lipschitz continuous with Lipschitz constant 1.

Proof. The value of d is calculated as the minimum of $f(\mathbf{R}(\boldsymbol{\theta}), \mathbf{C}, \mathbf{U})$ over all $\mathbf{C}$ and $\mathbf{U}$. Let $(\mathbf{C}^*, \mathbf{U}^*)$ be the minimizing camera positions and object coordinates for the orientations $\boldsymbol{\theta}$. Since (7.11) is equivalent to

$$\min_{\mathbf{C},\mathbf{U}} f(\mathbf{R}(\boldsymbol{\phi}), \mathbf{C}, \mathbf{U}) \leq \min_{\mathbf{C},\mathbf{U}} f(\mathbf{R}(\boldsymbol{\theta}), \mathbf{C}, \mathbf{U}) + \max_j |\phi_j - \theta_j|, \tag{7.12}$$

we see that, to prove (7.11) it is sufficient to find one pair $(\mathbf{C}, \mathbf{U})$ such that

$$f(\mathbf{R}(\boldsymbol{\phi}), \mathbf{C}, \mathbf{U}) \leq \min_{\mathbf{C},\mathbf{U}} f(\mathbf{R}(\boldsymbol{\theta}), \mathbf{C}, \mathbf{U}) + \max_j |\phi_j - \theta_j|,$$

since the minimun $\min_{\mathbf{C},\mathbf{U}} f(\mathbf{R}(\boldsymbol{\phi}), \mathbf{C}, \mathbf{U})$ is obviously smaller than the left-hand side. We will now argue that $(\mathbf{C}^*, \mathbf{U}^*)$ satisfies this condition.

Considering an arbitrary reprojection error for $(\mathbf{C}^*, \mathbf{U}^*)$ but with orientations $\boldsymbol{\phi}$ instead of $\boldsymbol{\theta}$ it is easy to see that

$$\begin{aligned}
&\angle\left(u_{ij}, R(\phi_i)(U_j^* - C_i^*)\right) \\
&\quad = \quad \angle\left(u_{ij}, (R(\theta_i) - R(\theta_i) + R(\phi_i))\,(U_j^* - C_i^*)\right) \\
&\quad \leq \quad \angle\left(u_{ij}, R(\theta_i)(U_j^* - C_i^*)\right) + \angle\left(R(\phi_i)(U_j^* - C_i^*), R(\theta_i)(U_j^* - C_i^*)\right) \\
&\quad \leq \quad \angle\left(u_{ij}, R(\theta_i)(U_j^* - C_i^*)\right) + |\phi_i - \theta_i|. \qquad (7.13)
\end{aligned}$$

Thus the maximal reprojection error is bounded by

$$f(\mathbf{R}(\boldsymbol{\phi}), \mathbf{C}^*, \mathbf{U}^*) \leq d(\boldsymbol{\theta}) + \max_i |\phi_i - \theta_i|,$$

which proves inequality (7.11). □

Using the fact that the function $d(\boldsymbol{\theta})$ can be evaluated and that it is Lipschitz continuous according to the above lemma, we will show how to solve globally for structure from motion. The basic idea is to perform branch and bound over the space of rotations. The Lipschitz property can be used to bound the function in the neighbourhood of a considered point. Before giving the details of the branch and bound algorithm, we will discuss how to limit the initial search space.

7.4.3 Initial Constraints on Orientations

When working with multiple views, the high dimension of the space of rotations can pose a problem in the branch and bound setting. This section shows how to derive simple constraints on the orientations, which can be used to limit the space we have to examine using branch and bound.

Consider two cameras, one with orientation $\theta = 0$ and position C and the other with orientation $\theta' = \phi$ and position C'. Let u and u' be the measured bearings of an object point in the two cameras. Then $C' - C$ must lie in the cone shown in Figure 7.4. This is stated in the following lemma.

Lemma 31. *There exists $a \geq 0$ and $b \geq 0$ such that*

$$C' - C = a\,R(0)^T u - b\,R(\phi)^T u'. \qquad (7.14)$$

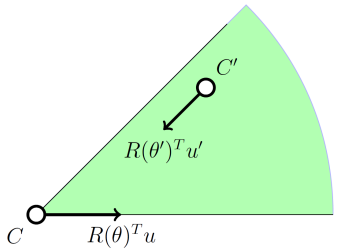

Figure 7.4: For the beams to intersect the right camera has to lie in the green area.

Proof. Let U be the coordinates of the object point. Then, from the projection equation (7.2), we get

$$C + a\, R(0)^T u = U = C' + b\, R(\phi)^T u' \quad a, b \geq 0$$

and the result follows. □

It is more efficient to work with angles rather than vectors. Hence, we define angles α, α' and c, such that

$$u = \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} \quad \text{and} \quad \frac{C' - C}{|C' - C|} = \begin{pmatrix} \cos c \\ \sin c \end{pmatrix}. \tag{7.15}$$

Inserting in (7.14) and using the addition formulas for the sine and cosine we get

$$\begin{pmatrix} \cos c \\ \sin c \end{pmatrix} = a \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} - b \begin{pmatrix} \cos(\alpha' + \phi) \\ \sin(\alpha' + \phi) \end{pmatrix}. \tag{7.16}$$

Let us first assume that $\phi = 0$ and $\alpha = 0$. Then we get the simplified relation

$$\begin{pmatrix} \cos c \\ \sin c \end{pmatrix} = a \begin{pmatrix} 1 \\ 0 \end{pmatrix} - b \begin{pmatrix} \cos\alpha' \\ \sin\alpha' \end{pmatrix} = a \begin{pmatrix} 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} \cos(\alpha' - \pi) \\ \sin(\alpha' - \pi) \end{pmatrix}. \tag{7.17}$$

We get two cases. If α' can be chosen in $[\pi, 2\pi]$ we get the case on the left in Figure 7.5. The constraint that a and b be positive implies that

$$c \in [0, \alpha' - \pi]. \tag{7.18}$$

If instead α' can be chosen in $[0, \pi]$ we get the case on the right Figure 7.5 and

$$c \in [\alpha' - \pi, 0]. \tag{7.19}$$

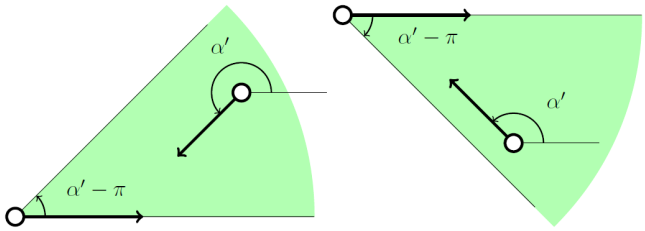

Figure 7.5: The two possible cases. Details are given in the text.

In the general case, when α and ϕ are not zero, the same argument leads to

$$c \in [\alpha, \alpha' - \pi + \phi] \quad \text{or} \tag{7.20}$$

$$c \in [\alpha' - \pi + \phi, \alpha]. \tag{7.21}$$

Note that angle representations must be chosen such that interval lengths are positive but less than π. Also an angle β lies in an interval if for some $n \in \mathbb{Z}$, $\beta + n \cdot 2\pi$ does.

For each point which is visible in two cameras, we get a constraint either of type (7.20) or (7.21). We introduce an index so that α_k represents the bearing of point k in the first camera and α'_k the bearing of the same point in the second camera. For a given ϕ to be feasible there must exist a c that satisfies (7.20) or (7.21) for every α_k. The following discussion will show how this can be used to produce bounds on ϕ.

First, note that (7.20) and (7.21) are linear in ϕ. Problems occur only when switching between the two types of intervals, (7.20) and (7.21). This happens when

$$\phi = \alpha_k - \alpha'_k \quad \text{or} \quad \phi = \alpha_k - \alpha'_k - \pi. \tag{7.22}$$

For each index k we get two such angles. Let

$$\{\phi^{(m)}\}, \quad m = 1 \dots M \tag{7.23}$$

be a sorted list of these angles. This divides the unit circle into M intervals. We will consider these intervals one by one. If

$$\phi \in [\phi^{(m)}, \phi^{(m+1)}], \tag{7.24}$$

then for each point we know if (7.20) or (7.21) is the relevant form of the constraint. Let K_1 be the set of the indices that generate constraints of type (7.20) and K_2 contain the

indices that generate constraints of type (7.21). To determine if there is an angle c that satisfies all these constraints we need to check if the intersection

$$\left(\bigcap_{k \in K_1} [\alpha_k, \alpha'_k - \pi + \phi] \right) \quad \bigcap \quad \left(\bigcap_{k \in K_2} [\alpha'_k - \pi + \phi, \alpha_k] \right), \tag{7.25}$$

is non-empty. To make the problem cleaner we introduce variables ℓ_k for the lower interval limits and h_k for the upper limits. We get

$$\left(\bigcap_{k \in K_1} [\ell_k, h_k + \phi] \right) \quad \bigcap \quad \left(\bigcap_{k \in K_2} [\ell_k + \phi, h_k] \right). \tag{7.26}$$

Now, Algorithm 7.4.3 can be used to check for which ϕ this intersection is non-empty. Naturally, only those ϕ's that also satisfy (7.24) are relevant. We repeat the algorithm for each of the M intervals defined by (7.23). Each interval yields different index sets K_1 and K_2 and thus different input to the algorithm.

---

1. Choose one interval $[\ell_0 + \phi, h_0]$.
2. Adjust all angle representations such that
   $h_0 - 2\pi \leq \ell_k < h_0$, $\ell_k < h_k < \ell_k + 2\pi$.
3. Let $L^a = \max_{k \in K_2} l_k, \quad L^b = \max_{k \in K_1} l_k$
   $H^a = \min_{k \in K_2} h_k, \quad H^b = \min_{k \in K_1} h_k$.

We have reduced the problem to
$[L^a, H^a + \phi] \quad \bigcap \quad [L^b + \phi, H^b]$.
If $L^a > H^b$ or $L^b > H^a$ the problem is infeasible,
otherwise it is feasible for $L^a - H^a < \phi < H^b - L^b$.

Modifying this algorithm to handle an error tolerance is simple. In step 3 each interval is simply widened with the tolerance on each side. Note that when we have multiple views, we can use this method for any pair of cameras.

7.4.4 Branch and Bound Algorithm

In this section we present a practical algorithm to solve for structure and motion with one-dimensional cameras. The input to our algorithm is a set of m one-dimensional images. Each image is a set of bearings of object points. We assume that correspondences between the different images are known, but not that all object points are visible in all cameras.

As a consequence of Lemma 29, structure from motion estimation for fixed orientations is a quasiconvex problem and can be solved efficiently. Thus we propose a branch

and bound algorithm over the space of orientations. With m cameras the space of orientations can be identified with $[0, 2\pi]^{m-1}$ since the first camera can be set to $R_1 = I$ by fixing the orientation of the world coordinate system. We represent the space of orientation with regions covering the whole set of feasible orientations.

To reduce this covering we first use the pairwise constraints of Section 7.4.3. Bounds $\theta_i - \theta_1 \in I_i$, where I_i is some interval, can be used to reduce the search space from $[0, 2\pi]^{m-1}$ to $\Pi_i I_i$. With omnidirectional data, which is the normal case in one-dimensional vision, this is often a considerable reduction.

After this initialization step we are left with a set of rectangular regions in the space of orientations. To discard such regions we use the Lipschitz continuity of the goal function (cf. Lemma 30) as follows. Assume that the best solution that we have found has L_∞ norm error d_{best} and that the largest side of the region we are considering to be w. Set the orientations to the centre of the region, denoted $\boldsymbol{\theta}_c$. Now assume that the goal function fulfills

$$d(\boldsymbol{\theta}_c) > d_{best} + \frac{w}{2}. \tag{7.27}$$

Then according to the lemma, no orientations $\boldsymbol{\theta}$ within the region can have $d(\boldsymbol{\theta}) \leq d_{best}$ and thus we can discard this region. According to Lemma 29, checking (7.27) is a linear feasibility problem so it can be solved efficiently.

Finally we need a method to find better and better solutions and thus updating d_{best}. One method is to seek among the centres of the feasible regions. That is, when a region has passed (7.27) we also check whether $d(\boldsymbol{\theta}_c) < d_{best}$. If this is true then we have found a better solution. The overall procedure is summarized in Algorithm 7.4.4.

Starting with an upper bound on the optimal solution d_{best}.
Iterate until desired precision is reached:

1. Pick a region from the queue.
2. Check feasibility using (7.27).
3. If the region cannot be discarded:
   - Divide the region and update the queue.
   - Try to update the lower bound on the optimum.
6. Remove the region from the queue.

In the initialization we only used a subset of the constraints discussed in Section 7.4.3, namely those concerning $\theta_i - \theta_1 = \theta_i$. To further increase speed it is straightforward to use general constraints on $\theta_i - \theta_j$. In step 2 of Algorithm 7.4.4, we simply check feasibility using these bounds before using (7.27).

7.5 Experiments

In this section the developed theory is experimentally validated and the practical aspects of Algorithms 7.4.3 and 7.4.4 are investigated on both synthetic and real data. Further, we give examples of multiple local minima under the L_2 norm error function.

Our implementation is done in MATLAB using `linprog` for linear programming (LP) feasibility problems. For all the below experimental setups, one LP problem took around (or less) $0.05s$ to execute on a standard Pentium 2.8GHz processor.

7.5.1 Illustration of Typical Three View Problems: Synthetic Data

In a first example, study the problem of 3 views of 7 points with measured angles (in radians)

$$\alpha = \begin{pmatrix} 3.1 & -1.9 & -0.1 & -1.9 & -1.3 & 1.7 & -0.4 \\ -2.2 & -1.3 & -0.2 & -1.3 & -1 & 1.9 & -0.4 \\ 2.6 & -2.9 & -1.4 & -2.9 & -2.6 & 1.4 & -1.7 \end{pmatrix}, \tag{7.28}$$

where rows represent different views and columns represent different points; see also Figure 7.1 where the 3 images are plotted. The optimization problem consists of finding the camera orientations and positions as well as the 7 object points. In the branch-and-bound scheme, given by Algorithm 7.4.4, we can assume that the first camera orientation θ_1 is set to zero since we are free to choose the coordinate system.

Suppose that there exists a solution with $d_{best} \leq 0.05$ radians, that is, a solution with angular reprojection error no worse than 0.05 radians. We initialize the queue with a square region with a centre point at $(\theta_2, \theta_3) = (\pi, \pi)$ and width 2π, hence covering all possible orientations. In the first iteration of Algorithm 7.4.4, this region (naturally) passes the feasibility test for the bound $0.05 + \pi$ and consequently it is divided into smaller ones. This splitting leads to a quadtree structure. In the next two iterations of the algorithm there are 4 and 16 regions, respectively. None of these can be outruled. At the next level, 60 out of 64 regions of width $\pi/4$ can be outruled.

In Table 7.1, we summarize the first 10 iterations of the algorithm by describing (i) the number n_{sq} of feasible regions there are left at each level and (ii) how much area A out of the total area $A_{tot} = (2\pi)^2$ these regions represent. After 10 iterations of the algorithm the optimal solution is bounded by $5.94 \leq \theta_2 \leq 5.98$ and $0.74 \leq \theta_3 \leq 0.86$.

iteration	1	2	3	4	5	6	7	8	9	10
n_{sq}	4	16	4	8	20	28	40	68	104	92
$\log(A/A_{tot})$	0.0	0.0	-1.2	-1.5	-1.7	-2.2	-2.6	-3.0	-3.4	-4.1

Table 7.1: Progress of the branch and bound algorithm for the synthetic example in (7.28). See text for details.

Figure 7.6 illustrates the progress of the algorithm. Note that the the method quickly focuses in on the optimal solution of the problem and orientations far away from the optimum are discarded early in the iterative process. A plot of the goal function $d(\boldsymbol{\theta})$ is given in Figure 7.7. The optimal solution is plotted in Figure 7.2.

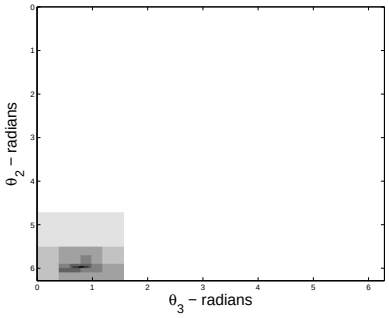

Figure 7.6: The quadtree map of the goal function. White regions are discarded early in the branch and bound algorithm and darker areas are kept longer.

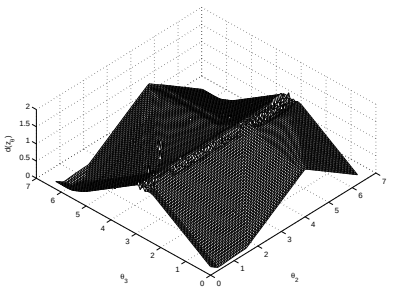

Figure 7.7: The figure shows the goal function $d(\boldsymbol{\theta})$ as a function of θ_2 and θ_3 for the example with 3 views of 7 points in Section 7.5.1. Notice that the function is periodic.

See Figure 7.8 for illustrations of other typical random, synthetic three-view examples with varying number of points. In certain cases there may be several local optima and even in underconstrained cases (meaning less equations than unknowns) one can often locate the global minimum to a small region of parameter space.

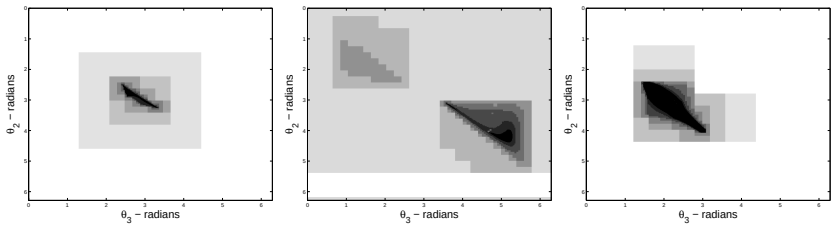

Figure 7.8: Evolution of the quadtree map. See Figure 7.6 for explanation. Left: 4 points, 3 cameras (underconstrained). Middle: 5 points, 3 cameras with two local optima. Right: 6 points, 3 cameras (overconstrained). Note that even though the solution is underconstrained in the left example, one can locate the global optimum to a small region of parameters space.

7.5.2 Omnidirectional Cameras: Real Data

In the standard setup for 1D vision, a rotating laser (as depicted in Figure 7.1) measures the angles between strips of reflective tape. This is an omnidirectional camera as measurements (or bearings) can be detected from a 360 degree field of view. All the experiments in this section are performed with such a sensor.

Our data were collected in a series of four experiments with varying number of object points and camera positions. The first three experiments were performed in a single room with 5 reflective tapes on the walls of the room, with bearing measurements at 7, 8 and 21 different camera positions, respectively. In the fourth experiment, 14 reflective tapes were placed inside an ice hockey rink and the camera captured bearings to these points from 70 positions. The resolution of the angular meter is roughly 0.8 mrad.

There is no ground truth available for any of these experiments. However, as the same set of 5 reflective tapes are measured in the first three independent runs of the laser truck, the reconstruction of the scene geometry should be the same for all three experiments. This will be used to validate that the reconstructions are plausible.

To measure the uncertainty of the solution space, we use $(\frac{V}{V_0})^{1/n}$, where V is the remaining volume of rotation space, $V_0 = (2\pi)^n$ is the total volume and n the dimension. Note that the measure is normalized with respect to dimension and it can be regarded as the (normalized) geometric mean of the width of each dimension. In order to examine the running times, random subsets of 3, 4 and 5 camera positions of the ice hockey rink data were tested by measuring uncertainty as a function of the number of LP problems solved. The result is graphed in Figure 7.9. Note that already after a few hundred programs, a large portion of the rotation space can be ruled out. Also note that the execution times increase as the number of dimensions go up, as can be expected from a branch and bound scheme. The overhead for setting up the LP problems is negligible.

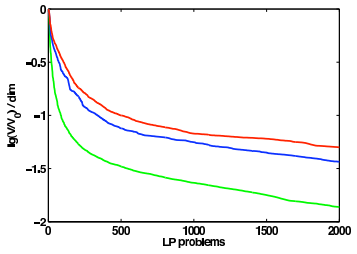

Figure 7.9: The plot shows the uncertainty as a function of the number of linear programs for 3 views (green), 4 views (blue) and 5 views (red). The remaining volume is normalized with respect to the dimension of the space of rotations, i.e., $V^{1/n}$ is used as a measure of this uncertainty. The data are averages over respectively 25, 15 and 9 experiments.

In the first (second) experiment of the single room setup, there are $m = 7$ ($m = 8$)

camera positions involved which means that one needs to perform branch and bound in $n = 6$ ($n = 7$) dimensions, respectively. The initial rotation constraints of Algorithm 7.4.3 reduce the potential rotation volume considerably to $(\frac{V}{V_0})^{1/6} = 0.17$ and $(\frac{V}{V_0})^{1/7} = 0.13$, respectively. Still, the rotational uncertainty is as much as 0.7 (0.7) radians for the most uncertain dimension.

After 5274 (6738) feasibility programs, there remain 2626 (4072) potential regions in the branch and bound queue, with relative volume $(\frac{V}{V_0})^{1/6} = 0.11$ and $(\frac{V}{V_0})^{1/6} = 0.10$, respectively, and a maximum width of 0.13 (0.13) radians for the most uncertain dimension of all rotation regions. The best solution found has an L_∞ error of 1.1 (1.3) mrad. Hence, even though solutions with low errors are obtained, it takes a long time to reduce the rotational uncertainty for such high dimensional problems. This is about the limit what is practically possible for branch and bound. The two computed solutions are plotted in the left and middle of Figure 7.10.

For the remaining two experiments with 21 and 70 camera positions we will have to modify our strategy to a more practical approach. By combining optimal structure from motion with alternating camera pose estimation and triangulation, it is possible to solve for many cameras and object points in an efficient manner. The guaranteed global optimality is of course lost, but our experiments indicate that this strategy still gives very precise results.

For the third experiment with 21 camera positions viewing 5 object points we use the following two-step scheme:

1. We select the first 5 camera positions, and use Algorithm 7.4.3 for initialization (which gives rotational uncertainty $(\frac{V}{V_0})^{1/4} = 0.18$) followed by Algorithm 7.4.4 for computing a global solution (which results in rotational uncertainty $(\frac{V}{V_0})^{1/4} = 0.05$ after 2620 LP programs). The L_∞ error is 1.5 mrad for the partial structure from motion solution.

2. Then, since all 5 object points are reconstructed, we can perform optimal resection on the remaining 16 camera positions. The computed solution can be iteratively improved by alternating optimal triangulation and optimal camera pose estimation. This results in a solution with L_∞ error 2.6 mrad. The iterative alternation scheme (typically) convergences in just a few iterations.

The three structure from motion solutions of the single room experiment are plotted in Figure 7.10. Note that the 2D object points have been identically reconstructed (modulo a choice of coordinate system) in all three experiments.

As a final omnidirectional experiment, with measurements from an ice hockey rink of 70 images of 14 points, we solve the structure from motion problem using the same two-step procedure as above. The result is shown in Figure 7.11 and the solution has a L_∞ error of 5.2 mrad.

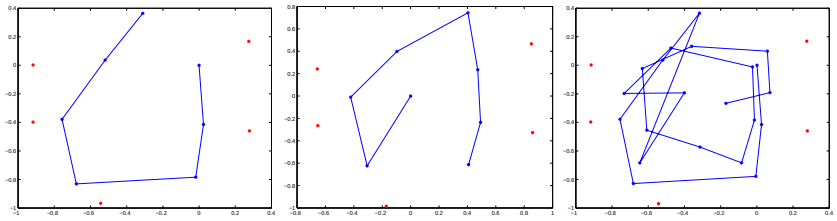

Figure 7.10: Structure and motion solutions obtained from measurements of 5 points in 7 cameras (left), 8 cameras (middle) and 21 cameras (right), respectively.

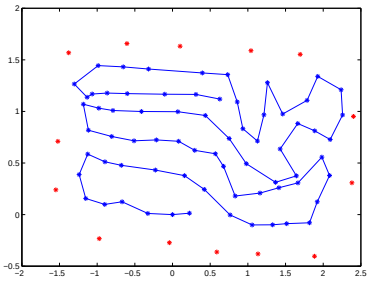

Figure 7.11: Structure and motion for the ice hockey experiment calculated by solving a series of camera pose and triangulation problems.

7.5.3 Cameras with Limited Field of View: Real Data

Pinhole cameras can only view objects in front of the camera and thus have a limited field of view. For 1D cameras having a limited field of view, finding the optimal solution to structure from motion turns out to be computationally much more demanding than for omnidirectional cameras. The results in this section are based on real data measurements, but we have also validated that similar conclusions can be drawn based on (random) synthetic data with limited field of view.

The setup for the experiment is as follows. First we placed 10 black-and-white markers in the scene. Then a camera was mounted on a trolley and moved around to create a planar motion. Figure 7.12 shows five of the captured images. The blue stars show the detected markers in each image. Since the camera y-axis is not perfectly aligned with the upward direction in the images we had to reestimate the coordinate system from objects in the image that can be assumed to be aligned with the upward direction, such as the door frame and other lines parallel to it. The green lines show the estimated coordinate system and the red stars show the projections of the computed bearings. The field of view is less than 45 degrees (that is, the angle difference between maximum and minimum

measurements).

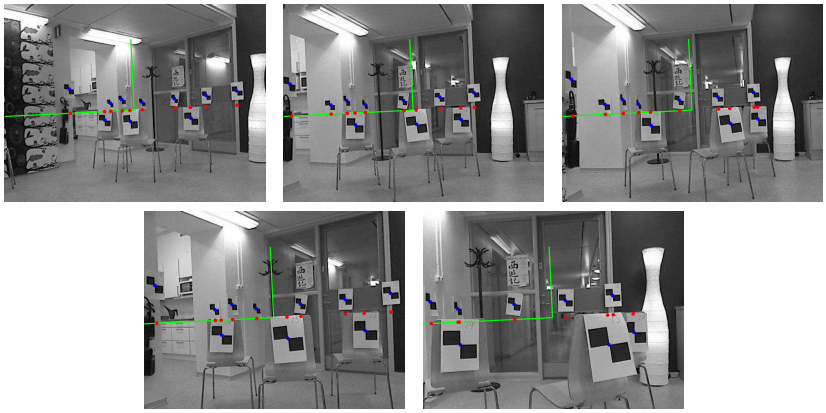

Figure 7.12: Setup for the experiment with planar moving cameras. The blue stars show the location of the detected markers, the green lines show the estimates of the x and y axes and the red stars show the calculated bearings of the markers. Note that not all markers are visible in all views.

In order to examine the execution times, similarly to what was done in the omnidirectional case, random subsets of 3 cameras were selected and the rotation uncertainty was studied as a function of LP problems solved. Figure 7.13 shows how Algorithm 7.4.4 *typically* works for one such example compared to a typical omnidirectional example. In all examples we have encountered, the convergence of the branch and bound algorithm is slow. Not surprisingly, Algorithm 7.4.3 fails to substantially reduce the initial rotaional uncertainty. For example, applying Algorithm 7.4.3 to the 5 images of 10 points described above yields $(\frac{V}{V_0})^{1/4} = 0.70$. After 3050 LP programs, the uncertainty has decreased to $(\frac{V}{V_0})^{1/4} = 0.57$ for the same instance. The best solution found at this point has L_∞ error 309 mrad. Apparently the optimization takes a lot longer time. Alternating optimal camera pose estimation and optimal triangulation improves the solution and a local optimal solution is obtained. The L_∞ error is 0.54 mrad.

To see if this difficulty is somehow inherent in the problem, we examined the slope of the goal function. As the branch and bound algorithm fails to efficiently discard regions, the goal function is flat around the optimal solution. Hence, there is a large neighbourhood of solutions that give almost the same reprojection error. Figure 7.14 shows two such solutions from one problem instance with 3 cameras viewing the 10 markers. Though the two solutions are very different the L_∞ errors are less than 2 mrad for both. In fact there is a whole continuum of low error solutions between these two. This means that the problem is very unstable. The behaviour is common in our data. Note especially that this example is not close to the minimal problem of five points in three views. Adding

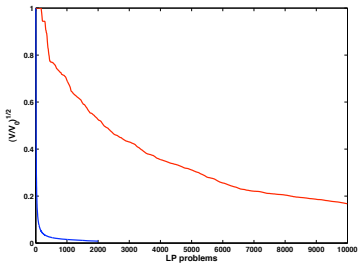

Figure 7.13: Remaining uncertainty as a function of the number of linear programming problems solved. The plot shows the average uncertainty per dimension as a fraction of the original uncertainty. The blue curve relates to a typical omnidirectional example, whereas the red curve comes from a camera having a limited field of view.

more views will reduce the ambiguity in determining the solution, but it is still present for smaller number of views (less than 5). This observation explains the poor performance of using narrow field of view cameras (cf. Figure 7.13).

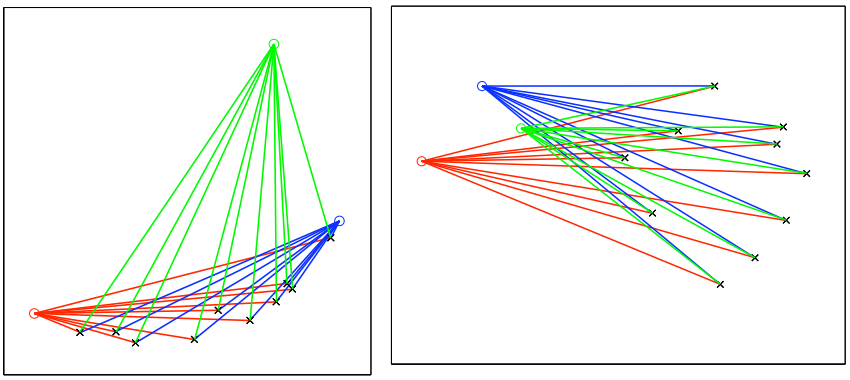

Figure 7.14: The plots show two solutions for the same three view problem. Both solutions have a L_∞ error of less than 2 mrad.

Do local minima occur? We conclude with an experiment that shows that local minima may occur, and in some cases even quite frequently.

Using MATLAB's built in function `fmincon` we tried to minimize the L_2 norm error of the angles for the 5 images of 10 markers. We found 16 local minima in total from random initializations. However many of them can easily be discarded since one or two

markers are far away (towards infinity) which is unreasonable in this setting. Figure 7.15 shows five of the detected minima, and Table 7.2 shows the corresponding values of the goal function. The first minimum is the one with lowest value out of the 16 cases. The solution computed above with the L_∞ approach yielded an error of 0.54 mrad, and the reconstructed scene is almost identical (modulo coordinate system) to the best L_2 solution, plotted in the top left of Figure 7.15.

local minima:	1	2	3	4	5
L_2-error:	0.0013	0.0832	0.2270	0.1907	0.2145

Table 7.2: The L_2 angular error (in radians) for the minima shown in Figure 7.15.

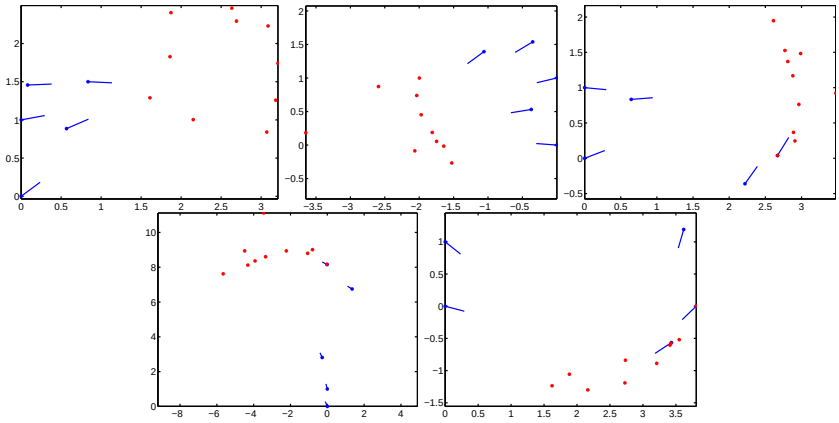

Figure 7.15: Five detected local minima. Note that some of the reconstructions appear to have negative depths. This is because not all of the markers are visible in all views.

7.6 Discussion

This chapter examined the problem of finding global minima to the structure from motion problem (SLAM, surveying) for 1D retina cameras using the L_∞ norm on reprojected angular errors. We have seen how the problem of known camera orientations can be reduced to a series of linear programming feasibility tests. We have also shown that the L_∞ goal function as a function of orientation variables has slope less than one. This Lipschitz property gives a way to efficiently search the orientation space for the optimal solution using branch and bound, resulting in a globally optimal algorithm with good

empirical performance.

Apart from algorithmic developments for the 1D structure from motion problem, we have also experimentally shown that the goal function is very flat for cameras with small field of view. In particular, this means that there may exist many solutions with similar reprojection errors. Hence, such problems are ill-posed and computationally hard to solve.

Bibliography

[1] P.K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. *ACM Comput. Surv.*, 1998.

[2] M. Armstrong, A. Zisserman, and R. Hartley. Self-calibration from image triplets. In *European Conf. on Computer Vision*, 1996.

[3] K. B. Atkinson. *Close Range Photogrammetry and Machine Vision*. Whittles Publishing, 1996.

[4] S. Axler. *Linear Algebra Done Right*. Springer-Verlag, 1997.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer Vision and Image Understanding*, 2008.

[6] A.C. Berg, T.L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *Conf. Computer Vision and Pattern Recognition*, 2005.

[7] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[8] P.J. Besl and N.D. McKay. A method for registration two 3-d shapes. *Pattern Analysis and Machine Intelligence*, 1992.

[9] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, USA, 1987.

[10] R.C. Bolles and R.A. Cain. Recognizing and locating partially visible objects: The local-feature-focus method. *Int. Journal Robotics Research*, 1982.

[11] J. A. Bondy and U. S. R. Murty. *Graph Theory*. Springer-Verlag, 2008.

[12] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[13] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence*, 2001.

[14] T.M. Breuel. Implementation techniques for geometric branch-and-bound matching methods. *Computer Vision and Image Understanding*, 2003.

[15] M. Brown and D.G. Lowe. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *Int. Conf. 3-D Digital Imaging and Modeling*, 2005.

[16] T.S. Caetano, T. Caelli, D. Schuurmans, and D.A.C. Barone. Graphical models and point pattern matching. *Pattern Analysis and Machine Intelligence*, 2006.

[17] T.A. Cass. Polynomial-time geometric matching for object recognition. *Int. Journal of Computer Vision*, 1999.

[18] A. Chiuso, R. Brockett, and S. Soatto. Optimal structure from motion: local ambiguities and global estimates. *IJCV*, 2000.

[19] O. Chum and J. Matas. Optimal randomized ransac. *Pattern Analysis and Machine Intelligence*, 2000.

[20] O. Chum and J. Matas. Matching with prosac - progressive sample consensus. In *Conf. Computer Vision and Pattern Recognition*, 2005.

[21] K. Cornelis, F. Verbiest, and L. Van Gool. Drift detection and removal for sequential structure from motion algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2004.

[22] A.D.J. Cross and E.R. Hancock. Graph matching using a dual step em algorithm. *Pattern Analysis and Machine Intelligence*, 1998.

[23] Y. Dai, J. Trumpf, H. Li, N. Barnes, and R. Hartley. Rotation averaging with application to camera-rig calibration. In *Asian Conf. Computer Vision*, 2009.

[24] A. Dalalyan and R. Keriven. L1-penalized robust estimation for a class of inverse problems arising in multiview geometry. In *Neural Information Processing Systems*, 2009.

[25] P. David, D. DeMenthon, R. Duraiswami, and H. Samet. SoftPOSIT: Simultaneous pose and correspondence determination. *IJCV*, 2004.

[26] J. Eckhoff. Helly, radon and carathéodory type theorems. In *Handbook of Convex Geometry*. North-Holland, Amsterdam, 1993.

[27] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *Conf. Computer Vision and Pattern Recognition*, 2009.

[28] M. Farenzena, A. Fusiello, and R. Gherardi. Structure-and-motion pipeline on a hierarchical cluster tree. In *Computer Vision Workshops (ICCV)*, 2009.

[29] O. D. Faugeras, L. Quan, and P. Sturm. Self-calibration of a 1d projective camera and its application to the self-calibration of a 2d projective camera. In *European Conf. on Computer Vision*, 1998.

[30] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 1981.

[31] A. Fitzgibbon and A. Zisserman. Automatic camera recovery for closed or open image sequences. In *European Conf. on Computer Vision*, 1998.

[32] R. Gherardi, M. Farenzena, and A. Fusiello. Improving the efficiency of hierarchical structure-and-motion. In *Conf. Computer Vision and Pattern Recognition*, 2010.

[33] S. Gold, A. Rangarajan, C. Lu, and E. Mjolsness. New algorithms for 2d and 3d point matching: Pose estimation and correspondence. *Pattern Recognition*, 1998.

[34] V. M. Govindu. Combining two-view constraints for motion estimation. In *Conf. Computer Vision and Pattern Recognition*, 2001.

[35] V. M. Govindu. Robustness in motion averaging. In *Asian Conf. Computer Vision*, 2006.

[36] E. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints*. MIT Press, 1990.

[37] J. A. Grunert. Das pothenot'sche problem in erweiterter gestalt; nebst bemerkungen über seine anwendung in der geodäsie. *Grunert Archiv der Mathematik und Physik*, 1841.

[38] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, 1988.

[39] R. Hartley. Defense of the eight-point algorithm. *Pattern Analysis and Machine Intelligence*, 1997.

[40] R. Hartley and F. Kahl. Global optimization through rotation space search. *Int. Journal of Computer Vision*, 2009.

[41] R. Hartley and F. Schaffalitzky. L_∞ minimization in geometric reconstruction problems. In *Conf. Computer Vision and Pattern Recognition*, 2004.

[42] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[43] R. Horaud and T. Skordas. Stereo correspondence through feature grouping and maximal cliques. *Pattern Analysis and Machine Intelligence*, 1989.

[44] D.P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *Int. Journal of Computer Vision*, 1990.

[45] K. Hyyppä. Optical navigation system using passive identical beacons. In *Proc. Intelligent autonomous systems, Amsterdam*, 1987.

[46] D. Jacobs. Matching 3-d models to 2-d images. *Int. Journal of Computer Vision*, 1997.

[47] F. Jurie. Solution of the simultaneous pose and correspondence problem using gaussian error model. *Computer Vision and Image Understanding*, 1999.

[48] F. Kahl and R. Hartley. Critical curves and surfaces for Euclidean reconstruction. In *European Conf. on Computer Vision*, 2002.

[49] F. Kahl and R. Hartley. Multiple view geometry under the L_∞-norm. *Pattern Analysis and Machine Intelligence*, 2008.

[50] Q. Ke and T. Kanade. Quasiconvex optimization for robust geometric reconstruction. *Pattern Analysis and Machine Intelligence*, 2007.

[51] M. P. Kumar, P. H. S. Torr, and A. Zisserman. Solving markov random fields using second order cone programming relaxations. In *Conf. Computer Vision and Pattern Recognition*, 2006.

[52] A. Kushal and J. Ponce. Modeling 3d objects from stereo views and recognizing them in photographs. In *ECCV*, 2006.

[53] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2007.

[54] H. Li. A practical algorithm for L_∞ triangulation with outliers. In *Conf. Computer Vision and Pattern Recognition*, 2007.

[55] H. Li. Efficient reduction for solving l-infinity problems in multiview geometry. In *Conf. Computer Vision and Pattern Recognition*, 2009.

[56] H. Li and R. Hartley. The 3D – 3D registration problem revisited. In *ICCV*, 2007.

[57] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 1981.

[58] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 2004.

[59] Y. Ma, H. Derksen, W. Hong, and J. Wright. Segmentation of multivariate mixed data via lossy data coding and compression. *Pattern Analysis and Machine Intelligence*, 2007.

[60] J. Maciel and J.P. Costeira. A global solution to sparse correspondence problems. *Pattern Analysis and Machine Intelligence*, 2003.

[61] M. Marques, M. Stosic, and J. Costeira. Subspace matching: Unique solution to point matching with geometric constraints. In *Int. Conf. Computer Vision*, 2009.

[62] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *Conf. Computer Vision and Pattern Recognition*, 2007.

[63] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conf.*, 2002.

[64] J. Matousek. On the geometric optimization with few violated constraints. *Symposium on Computational Geometry*, 1994.

[65] D. Nistér. Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. In *European Conf. on Computer Vision*, 2000.

[66] D. Nistér. An efficient solution to the five-point relative pose problem. *Pattern Analysis and Machine Intelligence*, 2004.

[67] C.F. Olson. A general method for geometric feature matching and model extraction. *Int. Journal of Computer Vision*, 2001.

[68] C. Olsson, F. Kahl, and M. Oskarsson. Optimal estimation of perspective camera pose. In *Int. Conf. Pattern Recognition*, 2006.

[69] S. Papert. The summer vision project. *Artificial Intelligence Group Vision Memo*, 100, 1966.

[70] L. Quan and T. Kanade. Affine structure from line correspondences with uncalibrated affine cameras. *Pattern Analysis and Machine Intelligence*, 1997.

[71] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?". In *European Conf. on Computer Vision*, 2002.

[72] F. Schaffalitzky and A. Zisserman. Automated location matching in movies. *Computer Vision and Image Understanding*, 2003.

[73] G. Sharp, S. Lee, and D. Wehe. Multiview registration of 3d scenes by minimizing error between coordinate frames. *Pattern Analysis and Machine Intelligence*, 2004.

[74] K. Sim and R. Hartley. Removing outliers using the L_∞-norm. In *Conf. Computer Vision and Pattern Recognition*, 2006.

[75] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from Internet photo collections. *Int. Journal of Computer Vision*, 2008.

[76] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *European Conf. on Computer Vision*, 1996.

[77] Y. Sugaya and K. Kanatani. Geometric structure of degeneracy for multi-body motion segmentation. In *In Workshop on Statistical Methods in Video Processing*, 2004.

[78] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, 2010.

[79] J.-P. Tardif, A. Bartoli, M. Trudeau, N. Guilbert, and S. Roy. Algorithms for batch matrix factorization with application to structure-from-motion. In *Conf. Computer Vision and Pattern Recognition*, 2007.

[80] T. Thormaehlen, H. Broszio, and A. Weissenfeld. Keyframe selection for camera motion and structure estimation from multiple views. In *ECCV*, 2004.

[81] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *Int. Journal of Computer Vision*, 1992.

[82] P. Torr, A. Fitzgibbon, and A. Zisserman. The problem of degeneracy in structure and motion recovery from uncalibrated image sequences. *IJCV*, 1999.

[83] P. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 2000.

[84] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. In *European Conf. on Computer Vision*, 2008.

[85] R. Tron and R. Vidal. A benchmark for the comparison of 3-d motion segmentation algorithms. In *Conf. Computer Vision and Pattern Recognition*, 2007.

[86] P. Tu, T. Saxena, and R. Hartley. Recognizing objects using color-annotated adjacency graphs. In *Lecture Notes in Computer Science; Shape, Contour and Grouping in Computer Vision*, 1999.

[87] T. Tuytelaars and L. Van Gool. Matching widely separated views based on affine invariant regions. *Int. Journal of Computer Vision*, 2004.

[88] A. Vedaldi, G. Guidi, and S. Soatto. Moving forward in structure from motion. In *Conf. Computer Vision and Pattern Recognition*, 2007.

[89] R. Vidal and R. Hartley. Motion segmentation with missing data using powerfactorization and gpca. In *Conf. Computer Vision and Pattern Recognition*, 2004.

[90] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and nondegenerate. In *European Conf. on Computer Vision*, 2006.

[91] G. Yang, C.V. Stewart, M. Sofka, and C.-L. Tsai. Registration of challenging image pairs: Initialization, estimation, and decision. *Pattern Analysis and Machine Intelligence*, 2007.

[92] C. Zach, M. Klopschitz, and M. Pollefeys. Disambiguating visual relations using loop constraints. In *Conf. Computer Vision and Pattern Recognition*, 2010.

[93] M. Zaslavskiy, F. Bach, and J.-P. Vert. A path following algorithm for the graph matching problem. *Pattern Analysis and Machine Intelligence*, 2009.