Global Optimization through Searching Rotation Space and Optimal Estimation of the Essential Matrix

Richard I. Hartley
Australian National University
and National ICT Australia*

Fredrik Kahl
Centre for Mathematical Sciences
Lund University, Sweden.

Abstract

This paper extends the set of problems for which a global solution can be found using modern optimization methods. In particular, the method is applied to estimation of the essential matrix, giving the first guaranteed optimal algorithm for estimating the relative pose under a geometric cost function, in this case, the L-infinity cost function. Convex optimization techniques has been shown to provide optimal solutions to many of the common problems in stucture from motion. However, they do not apply to problems involving rotations. In this paper, we introduce a search method that allows such problems to be solved optimally. Apart from the essential matrix, the algorithm is applied to the camera pose problem, providing an optimal algorithm.

1. Outline of the Proposed Method

In this paper, we will consider L_∞ optimization problems related to one-view or two-view geometry; in particular we will focus on two problems, the pose problem and the relative pose problem.

The pose problem is as follows. Given a set of $3D$ points with known position, and corresponding $2D$ image points, determine the location and pose of the camera. A little more formally: given $3D$ points $\mathbf{X}_i$ and corresponding image points, $\mathbf{v}_i$, determine the camera matrix $\mathtt{P}$ such that $\mathbf{v}_i = \mathtt{P}\mathbf{X}_i$ for all i. An algebraic solution to this problem is given by the DLT algorithm ([3], chapter 7).

The relative pose (or relative orientation) problem is to find the relative pose of two cameras, given a set of image correspondences, determined by unknown $3D$ points. Often the solution to this problem involves finding the positions of the $3D$ points as well. In other words, given image correspondences $\mathbf{v}_i \leftrightarrow \mathbf{v}'_i$, find two camera matrices $\mathtt{P}$ and $\mathtt{P}'$, along with $3D$ points $\mathbf{X}_i$, such that $\mathbf{v}_i = \mathtt{P}\mathbf{X}_i$ and $\mathbf{v}'_i = \mathtt{P}'\mathbf{X}_i$. This is the problem often solved by computing the fundamental or essential matrix; a commonly used algorithm is the 8-point algorithm ([6] or [3], chapter 11).

In the case where there is noise in the measurements, there is of course no exact solution to these problems, and generally it is considered optimal to find a solution to these problems that minimizes image error, namely the difference between the measured and modelled image points. The difference between the measured and modelled image measurements may be represented by a vector with one coordinate for each image measurement. The L_2 solution to the problem minimizes the L_2 norm of this error vector, namely the sum of squares of the image-measurement errors. The L_∞ solution, considered in this paper minimizes the L_∞ norm of the error vector, that is, it minimizes the maximum error, and is also called the minimax solution to the problem.

1.1. Related Work

To this point, there has been no known efficient optimal solution to these problems. Solutions have been given in [7] for the pose problem, but the solution we give here is considerably faster. No optimal solution has been reported for the relative pose problem. Optimal L_∞ solutions are given in this paper for both problems and the algorithms are efficient and fast.

This paper extends the class of problems that can be solved globally with the L_∞-norm. The class already includes problems like multiview triangulation, uncalibrated camera pose, homography estimation and structure from motion with a reference plane and more, see [4, 5].

In recent years there have been many attempts to compute globally optimal solutions for various geometric reconstruction problems. Using the L_∞-norm framework has perhaps been the most successful one, but other approaches include [1] and [4]. The former approach applies a branch and bound algorithm to compute L_1- and L_2-solutions for triangulation and uncalibrated pose and the latter one uses convex approximations for a set of geometric reconstruction problems, however, with no guarantee of optimality.

*NICTA is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

1.2. Getting Down to the Details

We consider calibrated cameras, and may therefore assume that the calibration matrix is the identity. As is commonly done with calibrated cameras, we find it convenient to consider image points as lying on an image sphere, rather than an image plane. Thus, an image measurement is a unit vector $\mathbf{v}_i$, representing the direction vector from the camera centre to the $3D$ point. Thus, a camera is represented by a rotation matrix $\mathtt{R}$, the orientation of the camera, and a position vector $\mathbf{C}$ representing the position of the camera centre. The image point corresponding to a point $\mathbf{X}$ is given by

$$\mathbf{v} = \frac{\mathtt{R}(\mathbf{X} - \mathbf{C})}{\|\mathtt{R}(\mathbf{X} - \mathbf{C})\|} .$$

The pose problem. Given a set of $3D$ points $\mathbf{X}_i$ and corresponding image "points" (actually unit vectors) $\mathbf{v}_i$, we seek the rotation $\mathtt{R}$ and camera centre $\mathbf{C}$ that realize the minimax cost function

$$\min_{\mathtt{R},\mathbf{C}} \max_i \angle(\mathbf{v}_i, \mathtt{R}(\mathbf{X}_i - \mathbf{C})) . \tag{1}$$

where $\angle(\cdot,\cdot)$ represents the angle between two vectors.[1]

It is interesting and relevant to observe that if the rotation $\mathtt{R}$ is known, then this problem is identical to the L_∞ triangulation problem as now explained. We observe that

$$\angle(\mathbf{v}_i, \mathtt{R}(\mathbf{X}_i - \mathbf{C})) = \angle(\mathtt{R}^\top \mathbf{v}_i, \mathbf{X}_i - \mathbf{C}) .$$

With known direction vectors $\mathtt{R}^\top \mathbf{v}_i$ and points $\mathbf{X}_i$ we seek the point $\mathbf{C}$ that minimizes the angular error. This is the "triangulation" problem, which was shown to be optimally solvable in L_∞ norm in [2]; Second Order Cone Programming (SOCP) may be used to find the optimal solution ([4, 5]). Thus, in principle, the pose problem may be solved by a search over all possible rotations $\mathtt{R}$ to find the rotation that gives the best solution. The challenge is to do this without having to test an infinite number of rotations.

The relative pose problem. The relative pose problem is to determine the relative position of two cameras given image point correspondences. We assume a set of image correspondences $\mathbf{v}_{i1} \leftrightarrow \mathbf{v}_{i2}$, where $\mathbf{v}_{i1}$ and $\mathbf{v}_{i2}$ are points in the first and second image respectively. We are required to find corresponding $3D$ points $\mathbf{X}_i$, rotation matrices $\mathtt{R}_j$ and camera centres $\mathbf{C}_j$ for $j = 1, 2$ that realize the minimum of the following cost function.

$$\min_{\mathtt{R}_j,\mathbf{C}_j,\mathbf{X}_i} \max_{i,j} \angle(\mathbf{v}_{ij}, \mathtt{R}_j(\mathbf{X}_i - \mathbf{C}_j)) . \tag{2}$$

[1]Thus, we formulate this problem as minimizing the angular error in measurements, instead of "pixel error" on an image plane. This is not an essential point; the problems could equally well be formulated in terms of image-plane error, but we find this formulation more natural for calibrated cameras.

To simplify this problem, we may assume that the first camera has rotation $\mathtt{R}_1$ equal to the identy, and camera centre $\mathbf{C}_1$ at the origin, leaving only the relative pose $(\mathtt{R}_2, \mathbf{C}_2)$ as well as the points $\mathbf{X}_i$ to be determined.

Once again, if the rotation is known, then the problem reduces to that of structure-and-motion with known rotations. This has also been shown to be solvable in L_∞ norm in [2]. As before, SOCP provides an efficient solution ([4]).

As this discussion shows, both the problems considered reduce to optimization over a space of Euclidean motions (rotation and translation). In both cases, the problem has a known solution if the rotation is known, so solving the general problem comes down to a search over all rotations. This search will be carried out using a branch-and-bound strategy ([1]).

2. Branch and Bound

We will discuss branch-and-bound optimization as a form of search over a parameter space. In our discussion, it will be assumed that the parameter space is some subset of a Euclidean space R^n, where n should not be too large. In the case of rotations, we may parametrize rotations using the angle-axis parametrization, to be described later, in which rotations are represented by 3-vectors. All $3D$ rotations may be represented by vectors in the closed ball of radius π in R^3.

In our version of branch and bound, we divide up the parameter space into cubic blocks, each block representing a set of similar rotations. Let D be a block in the parameter space. One now considers the optimization problem over the restricted set of parameters D. This will be referred to as the *restricted optimization problem*. Thus, for instance in the pose problem, one tries to find the minimum

$$\min_{\mathtt{R}\in D,\mathbf{C}} \max_i \angle(\mathbf{v}_i, \mathtt{R}(\mathbf{X}_i - \mathbf{C}))$$

where instead of trying to solve the problem over all rotations, one restricts to rotations in the block D.

The critical requirement in branch-and-bound is that although it may not be possible to solve the restricted problem exactly, it is at least possible to find a lower bound for the optimal solution to the restricted problem. The tighter this lower bound is, the better, and in any case it is necessary that as the size of the block D gets smaller, the lower bound becomes a closer and closer approximation to the optimal solution, and in the limit the lower bound converges to the value of the optimal minimum of the restricted problem.

The branch and bound algorithm now goes as follows.

1. Start with an initial approximate solution to the optimization problem, found by any method at all, and having cost $\epsilon_{\min}$ under the cost function being minimized.

2. Now, divide up the parameter space into blocks D_j. For each such block determine whether there is a solution to the restricted optimization problem on D_j having cost less than $\epsilon_{\min}$. This question may be formulated as a feasibility problem. If the answer is no (no solution with cost less than $\epsilon_{\min}$ exists on D_j), then block D_j can be excluded from further consideration.

3. Otherwise we take two steps:

   (a) evaluate the cost function for some value of the parameter inside the region D_j, and if this is less than $\epsilon_{\min}$, replace the value of $\epsilon_{\min}$ by this new current minimum.

   (b) Subdivide D_j into two or more smaller regions.

This algorithm terminates when the remaining blocks constrain the solution within the desired accuracy. Normally, this search is carried out depth-first – all blocks of a given size are considered before blocks at a finer resolution level are considered. However, under some circumstances (for instance in minimal cases where a solution with zero cost is known to exist), it may be preferable to carry out the search depth-first, since this method has a smaller memory requirement. We have implemented both search strategies; both work well.

This gives a general overview of the algorithm. In the particular problems we are interested in, the search is over all rotations, so the parameter space is 3-dimensional. Note that we do not subdivide the translation space.

In the next few sections of this paper, we will consider the details of our parametrization of rotations, and then the method for computing a lower bound for the restricted optimization problem.

3. Subdividing Rotation Space

3D rotations form a group with a metric structure, the angle metric defined as follows. Any rotation can be expressed as a rotation through a positive angle less than π about some axis. Let $\mathtt{R}$ and $\mathtt{R}'$ be two rotations. The distance $d_\angle(\mathtt{R},\mathtt{R}')$ is the angle θ lying in the range $0 \leq \theta \leq \pi$ of the rotation $\mathtt{R}'\mathtt{R}^{-1}$. Note that it does not matter whether we define this distance in terms of $\mathtt{R}'\mathtt{R}^{-1}$, or $\mathtt{R}^{-1}\mathtt{R}'$, or $\mathtt{R}'^{-1}\mathtt{R}$, or $\mathtt{R}\mathtt{R}'^{-1}$. The angle is the same.

We will use the following inequality, which seems simple enough that we omit the proof.

Lemma 3.1. *For any vector* $\mathbf{V}$,

$$\angle(\mathtt{R}\mathbf{V}, \mathtt{R}'\mathbf{V}) \leq d_\angle(\mathtt{R},\mathtt{R}') \; .$$

In the *angle-axis* representation of rotations, a rotation may also be represented by a 3-vector $\mathbf{v} = \alpha\hat{\mathbf{v}}$, where $\hat{\mathbf{v}}$ is a unit vector. This represents a rotation through an angle α about the axis $\hat{\mathbf{v}}$. Since any rotation may be represented by a rotation of at most π radians, the set of all rotations is represented by vectors in the 3-ball B^3_π of radius π. This parametrization is smooth. It is also one-to-one excepte on the boundary of B^3_π, where the mapping is two-to-one.

We will be interested in the relationship between the angle metric, and the Euclidean metric defined on the ball B^3_π. The desired result is as follows.

Lemma 3.2. *For each* $i = 1, 2$, *let* $\mathtt{R}_i$ *be the rotation corresponding to vectors* $\mathbf{v}_i$ *in the angle-axis representation. Then*

$$d_\angle(\mathtt{R}_1, \mathtt{R}_2) \leq \|\mathbf{v}_1 - \mathbf{v}_2\|$$

with equality only when $\mathbf{v}_1 = \mathbf{v}_2$.

It is a fact (though not specifically used here) that the angle-axis representation of rotations is to first order an isometry close to the origin (which represents the identity rotation). The important point in this lemma is that the angle distance is less than the Euclidean distance in the angle-axis representation.

Dividing up rotation space. The angle-axis representation gives a convenient way of dividing up rotation space into blocks. Rotations are represented by points in a ball B^3_π of radius π in $\mathbb{R}^3$. We may enclose this ball in a cube $C_\pi = [-\pi, \pi]^3$ in $\mathbb{R}^3$, and each point in this cube represents a rotation. The representation is redundant, since points outside the ball represent the same rotation as some point inside the ball, but this does not matter for our purposes.

Now, the cube C_π may easily be broken up into cubic blocks D_i of a given size. These blocks form the initial subdivision of the rotation space used in the branch-and-bound algorithm. When necessary, a cube D_i may be subdivided into 8 cubes of half the size. A simple test may be used to determine if a cube D_i contains any points lying inside the ball B^3_π. If it does not, then it is discarded and not used in the search. Given a cube D_i, we represent by $\bar{\mathtt{R}}$ the rotation corresponding to the centre of the cube, and by r_D the "radius" of the cube in terms of the angle metric. Thus $r_D = \max(d_\angle(\bar{\mathtt{R}}, \mathtt{R}))$, where $\mathtt{R}$ runs over all rotations represented by points in the cube D. From lemma 3.2 we have the inequality

$$r_D \leq \sqrt{3}\sigma \tag{3}$$

where σ is the half-side length of the cube D.

4. Feasibility Problems

We consider a feasibility problem motivated by the relative pose problem defined for a restricted rotation domain, D with radius r_D. By the radius of the region, is meant $\max d_\angle(\mathtt{R}, \bar{\mathtt{R}})$, where $\bar{\mathtt{R}}$ is the rotation at the centre of D.

We choose a coordinate system aligned with the first camera. Then $(\mathtt{R}, \mathbf{C})$ denote the relative pose of the second camera with respect to the first. The relevant feasibility problem for the relative pose problem is then:

$$\begin{array}{ll} \textbf{Do there exist} & \mathbf{C}, \mathbf{X}_i, \mathtt{R} \in D \\ \textbf{such that} & \angle(\mathbf{v}_{i0}, \mathbf{X}_i) < \epsilon_{\min} \\ \textbf{and} & \angle(\mathbf{v}_{i1}, \mathtt{R}(\mathbf{X}_i - \mathbf{C})) < \epsilon_{\min}. \end{array} \tag{4}$$

Generally, we will be interested in this problem in the case where we have a tentative solution to the relative pose problem with L_∞ error $\epsilon_{\min}$. A negative answer to this question means that the optimal solution can not be achieved with $\mathtt{R} \in D$.

Unfortunately, it is not possible to answer problem (4) directly. Instead, we consider a slightly weaker, but solvable problem in which we fix the rotation and place a slightly weaker bound.

$$\begin{array}{ll} \textbf{Do there exist} & \mathbf{C}, \mathbf{X}_i \\ \textbf{such that} & \angle(\mathbf{v}_{i0}, \mathbf{X}_i) < \epsilon_{\min} \\ \textbf{and} & \angle(\mathbf{v}_{i1}, \bar{\mathtt{R}}(\mathbf{X}_i - \mathbf{C})) < \epsilon_{\min} + \sqrt{3}\,\sigma\ . \end{array} \tag{5}$$

where σ is the half-side length of the cube D_i. Observe that, problem (5) may be solved, as shown in [2, 4] using for instance SOCP.

These two problems are related as follows.

Lemma 4.3. *If problem (4) has an affirmative answer then so does problem (5). On the other hand, if the answer to problem (4) is negative on a domain D, then D may be split into subdomains D_i of sufficiently small radius such that problem (5) has a negative answer on any D_i.*

Proof: First we prove that if problem (4) is feasible, then so is problem (5). Suppose that $\mathbf{C}$, $\mathbf{X}_i$ and $\mathtt{R}_{\text{opt}}$ constitute a feasible solution for problem (4). Then we show that $\mathbf{C}$ and $\mathbf{X}_i$ constitutes a solution to problem (5).

The first constraint $\angle(\mathbf{v}_{i0}, \mathbf{X}_i) < \epsilon_{\min}$ is fulfilled, since it is the same as for problem (4). As for the second constraint, by the triangle inequality, we have

$$\begin{aligned} \angle(\mathbf{v}_{i1}, \bar{\mathtt{R}}(\mathbf{X}_i - \mathbf{C})) &\leq \angle(\mathbf{v}_{i1}, \mathtt{R}_{\text{opt}}(\mathbf{X}_i - \mathbf{C})) \\ &\quad + \angle(\bar{\mathtt{R}}(\mathbf{X}_i - \mathbf{C}), \mathtt{R}_{\text{opt}}(\mathbf{X}_i - \mathbf{C})) \\ &\leq \epsilon_{\min} + d_\angle(\bar{\mathtt{R}}, \mathtt{R}_{\text{opt}}) \\ &\leq \epsilon_{\min} + r_D \\ &\leq \epsilon_{\min} + \sqrt{3}\sigma \end{aligned}$$

as required. Note where lemma 3.1 was used.

Now, the second part of the lemma. Suppose that problem (4) is infeasible, and that the minimum value of ϵ for which it is solvable is ϵ_0, instead of $\epsilon_{\min}$. Thus, $\epsilon_0 > \epsilon_{\min}$. Let $r_D = \epsilon_0 - \epsilon_{\min}$. Then problem (5) is infeasible on any subdomain D_i of D with radius less than r_D. The domain D may be divided into a finite number of subdomains of radius less than r_D and problem (5) is infeasible on any one of them. This completes the proof of the lemma.

This lemma gives a strategy for showing that problem (4) is not feasible on domain D.

5. The Pose Problem

The pose problem may be solved by the general branch-and-bound technique outlined in section 2 through solving the feasibility problem

$$\begin{array}{ll} \textbf{Do there exist} & \mathbf{C}, \mathtt{R} \in D \\ \textbf{such that} & \angle(\mathtt{R}\mathbf{v}_i, \mathbf{C} - \mathbf{X}_i) < \epsilon_{\min}. \end{array} \tag{6}$$

Again, it is not possible to solve this problem directly. Instead, we consider the problem

$$\begin{array}{ll} \textbf{Does there exist} & \mathbf{C} \\ \textbf{such that} & \angle(\bar{\mathtt{R}}\mathbf{v}_i, \mathbf{X}_i) < \epsilon_{\min} + \sqrt{3}\,\sigma\ . \end{array} \tag{7}$$

This is essentially the triangulation problem, and it may be solved using for instance SOCP. As in the rotation case, problem (7) may be reduced to (6) and used to solve the pose problem by the branch-and-bound technique.

6. First Order Bounds

The bounds in the last section were called zero-th order bounds because they used a zero-th order approximation to the rotations in a region D of rotation space, namely the centre of the rotation domain D. Although these bounds allowed us to formulate a branch-and-bound algorithm to solve the pose and relative pose problems, the bounds are somewhat pessimistic. The speed of the branch-and-bound algorithm depends on the number of cells D that need to be tested. A cell is eliminated from further consideration if the lower bound residual calculated for that cell exceeds the current minimum residual. If not, we need to subdivide the cell and repeat the calculation for the subdivided cells. It is critical to avoid subdividing unnecessarily, so the better the lower bound is, the fewer subdivisions will be necessary.

With the zero-th order rotation estimate, the uncertainty gap on our estimate of the residual is equal to the radius of the rotation cell. It will be seen that using a first order approximation to rotation, it is possible to decrease the gap to the order of the squared-radius of the rotation cell. For small cells, the gap will be very small. If this sounds hard to follow at present, wait until we describe the details.

6.1. First Order Approximation to Rotations

A rotation matrix $\mathtt{R}$ corresponding to a 3-vector $\mathbf{r}$ in the angle-axis representation can be represented in the exponential form as

$$\mathtt{R} = \exp([\mathbf{r}]_\times) = \mathtt{I} + [\mathbf{r}]_\times + [\mathbf{r}]_\times^2/2 + \dots\ .$$

Let $\bar{\mathtt{R}}$ be a rotation. A first order approximation to $\mathtt{R}$ about $\bar{\mathtt{R}}$ is defined as follows. Let $\delta\mathtt{R} = \bar{\mathtt{R}}^\top \mathtt{R}$ and

$$\delta\mathtt{R} = \exp([\delta\mathbf{r}]_\times) = \mathtt{I} + [\delta\mathbf{r}]_\times + [\delta\mathbf{r}]_\times^2/2 + \dots \ .$$

By truncating this series after the second term we get

$$\mathtt{R} = \bar{\mathtt{R}}\delta\mathtt{R} \approx \bar{\mathtt{R}} + \bar{\mathtt{R}}[\delta\mathbf{r}]_\times \ .$$

This last expression is the first order approximation to $\mathtt{R}$ about $\bar{\mathtt{R}}$, which will be denoted by $\hat{\mathtt{R}}$.

We need to evaluate how good an approximation this is to the rotation $\mathtt{R}$. The required relationship is as follows.

Lemma 6.4. *Let D be a rotation domain with centre $\bar{\mathtt{R}}$ and radius $r_D < 0.8$, let $\mathtt{R}$ be a rotation in D, and $\hat{\mathtt{R}}$ be its first order approximation about $\bar{\mathtt{R}}$. Then for any vector $\mathbf{V}$,*

$$\angle(\mathtt{R}\mathbf{V}, \hat{\mathtt{R}}\mathbf{V}) \leq r_D^2/2.$$

This compares with the zero-th order approximation $\bar{\mathtt{R}}$ to $\mathtt{R}$, for which $\angle(\mathtt{R}\mathbf{V}, \bar{\mathtt{R}}\mathbf{V}) \leq r_D$. When r_D is small, the first order approximation to $\mathtt{R}$ gives a significantly better result.

We prove this lemma. Since $\mathtt{R} = \bar{\mathtt{R}}\delta\mathtt{R}$ and $\hat{\mathtt{R}} = \bar{\mathtt{R}} + \bar{\mathtt{R}}[\delta\mathbf{r}]_\times$, we see that

$$\begin{aligned} \angle(\mathtt{R}\mathbf{V}, \hat{\mathtt{R}}\mathbf{V}) &= \angle(\bar{\mathtt{R}}\delta\mathtt{R}\mathbf{V}, \bar{\mathtt{R}}(\mathtt{I} + [\delta\mathbf{r}]_\times)\mathbf{V}) \\ &= \angle(\delta\mathtt{R}\mathbf{V}, (\mathtt{I} + [\delta\mathbf{r}]_\times)\mathbf{V}) \end{aligned}$$

We wish to bound this angle as $\mathbf{V}$ varies over all vectors. Since the magnitude of $\mathbf{V}$ is irrelevant, we may assume that $\mathbf{V}$ is a unit vector, in which case, so is $\delta\mathtt{R}\mathbf{V}$.

For vectors $\mathbf{A}$ and $\mathbf{B}$ for which $\|\mathbf{A}\| = 1$, and $\|\mathbf{A}-\mathbf{B}\| < 1$, observe that $\angle(\mathbf{A}, \mathbf{B}) \leq \arcsin(\|\mathbf{A} - \mathbf{B}\|)$, as may be seen by drawing a simple diagram. Apply this fact, we see that when $\mathbf{V}$ is a unit vector,

$$\angle(\mathtt{R}\mathbf{V}, \hat{\mathtt{R}}\mathbf{V}) \leq \arcsin\left(\|(\delta\mathtt{R} - \mathtt{I} - [\delta\mathbf{r}]_\times)\mathbf{V}\|\right) \ .$$

Now, let $\delta\mathbf{r} = \delta\theta\hat{\mathbf{r}}$, where $\hat{\mathbf{r}}$ is a unit vector. From Rodriques's formula we have

$$\delta\mathtt{R} = \mathtt{I} + \sin\delta\theta[\hat{\mathbf{r}}]_\times + (1 - \cos\delta\theta)[\hat{\mathbf{r}}]_\times^2$$

and so $(\delta\mathtt{R} - \mathtt{I} - [\delta\mathbf{r}]_\times)\mathbf{V}$ is equal to

$$\begin{aligned} &(\sin\delta\theta - \delta\theta)[\hat{\mathbf{r}}]_\times\mathbf{V} + (1 - \cos\delta\theta)[\hat{\mathbf{r}}]_\times^2\mathbf{V} \\ = \ &(\sin\delta\theta - \delta\theta)\hat{\mathbf{r}} \times \mathbf{V} + (1 - \cos\delta\theta)\hat{\mathbf{r}} \times (\hat{\mathbf{r}} \times \mathbf{V}) \ . \end{aligned}$$

Note that $\hat{\mathbf{r}} \times \mathbf{V}$ and $\hat{\mathbf{r}} \times (\hat{\mathbf{r}} \times \mathbf{V})$ are orthogonal and of the same length. Clearly, the magnitude of this vector is maximized (as $\mathbf{V}$ varies over all unit vectors) when $\hat{\mathbf{r}}$ and $\mathbf{V}$ are orthogonal, so that $\hat{\mathbf{r}} \times \mathbf{V}$ is a unit vector. In this case,

$$\begin{aligned} \angle(\mathtt{R}\mathbf{V}, \hat{\mathtt{R}}\mathbf{V}) &\leq \arcsin\left(\|(\delta\mathtt{R} - \mathtt{I} - [\delta\mathbf{r}]_\times)\mathbf{V}\|\right) \\ &= \arcsin\left(\sqrt{(\sin\delta\theta - \delta\theta)^2 + (1 - \cos\delta\theta)^2}\right) \\ &\leq \delta\theta^2/2 \end{aligned}$$

for $0 < \delta\theta < 0.8$. Rigorous justification of this last step is omitted, but it is easily verified graphically (see Fig 1).

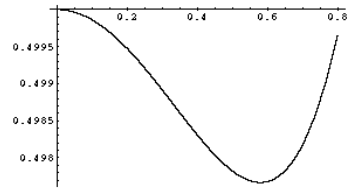

Figure 1. *Plot of the function $\arcsin(\sqrt{(\sin x - x)^2 + (1 - \cos x)^2})/x^2$, verifying the last step of the proof of lemma 6.4.*

6.2. The Pose Problem - First Order

We may formulate the feasibility problem for pose computation using the first order approximation to $\mathtt{R}$ as follows.

$$\begin{array}{ll} \textbf{Do there exist} & \mathbf{C}, \mathtt{R} \in D \\ \textbf{such that} & \angle(\mathbf{v}_i, \hat{\mathtt{R}}(\mathbf{X}_i - \mathbf{C})) < \epsilon_{\min} + 3\sigma^2. \end{array} \quad (8)$$

With essentially the same derivation as before, we can show that if this problem is infeasible, then so is problem (6). What remains to show is that we may find a solution to this problem. We may write

$$\begin{aligned} \angle(\mathbf{v}_i, \hat{\mathtt{R}}(\mathbf{X}_i - \mathbf{C})) &= \angle(\bar{\mathtt{R}}^\top\mathbf{v}_i, (\mathtt{I} + [\delta_r]_\times)(\mathbf{X}_i - \mathbf{C})) \\ &= \angle(\bar{\mathtt{R}}^\top\mathbf{v}_i, \mathbf{X}_i + \delta_r \times \mathbf{X}_i - (\mathtt{I} + [\delta_r]_\times)\mathbf{C})) \end{aligned}$$

However, $\mathbf{C}$ is an unconstrained variable in problem (8), and as $\mathbf{C}$ runs over all values in $\mathbb{R}^3$, so does $(\mathtt{I} + [\delta_r]_\times)\mathbf{C})$. So the constraint in problem (8) is equivalent to

$$\angle(\bar{\mathtt{R}}^\top\mathbf{v}_i, \mathbf{X}_i + \delta_r \times \mathbf{X}_i - \mathbf{C}) < \epsilon_{\min} + 3\sigma^2 \ .$$

The important point here is that the unknowns δ_r and $\mathbf{C}$ do not interact quadratically in this expression. Expressing the tangent of the angle in terms of vector cross and scalar products turns this into a SOCP constraint, as observed in ([4]).

7. Linear Programming

For the essential matrix problem, the SOCP-based method turns out to be far too slow to be practical. A method that gives orders of magnitude speed-up based on Linear Programming has been developed. Details of this method will be given in an expanded version of this paper, but a sketch is given below.

Consider a set of correspondences $\mathbf{v}_i \leftrightarrow \mathbf{v}'_i$ in two views, and suppose that the rotation is known. We address the question of whether a solution to the relative pose problem exists, fitting the data within a given tolerance $\epsilon_{\min}$ as in (4).

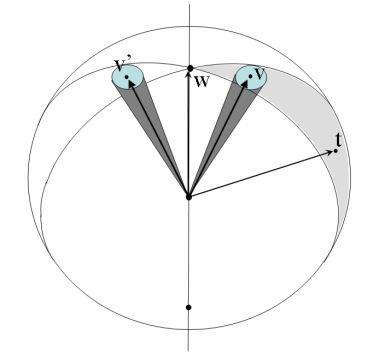

Figure 2. *The epipole lies in the wedge of a sphere bounded by the pair of crossed great circles, and containing the vector* $\mathbf{v}$.

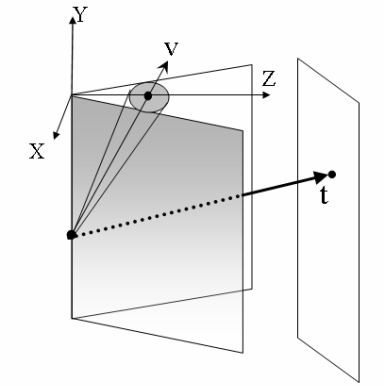

Figure 3. *The epipolar vector lies on the plane* $\mathrm{Z} = 1$ *in a coordinate frame determined by the pair of matched points* $\mathbf{v}$ *and* $\mathbf{v}'$.

Such a possible solution would involve an epipolar direction $\mathbf{t}$ giving the relative displacement of the second camera with respect to the first. We consider just a single correspondence $\mathbf{v} \leftrightarrow \mathbf{v}'$ from the set of correspondences. The three vectors $\mathbf{v}$, $\mathbf{v}'$ and $\mathbf{t}$ must be coplanar (the familiar coplanarity condition). With a given uncertainty $\epsilon_{\min}$ in the measurements $\mathbf{v}$ and $\mathbf{v}'$, we see that the vector $\mathbf{t}$ must lie in a wedge-shaped region, as shown in Fig 2. Thus, the vector $\mathbf{t}$ lies between two planes, which may be specified in terms of the corresponding pair $\mathbf{v} \leftrightarrow \mathbf{v}'$. In this way $\mathbf{t}$ is constrained by two linear constraints (one for each bounding plane). From a set of n correspondences, $2n$ linear constraints arise in this way. The existence of a solution within the desired tolerance $\epsilon_{\min}$ therefore becomes an LP feasibility problem in only 3 variables (the coordinates of $\mathbf{t}$). We can reduce this to two variables; since the length of $\mathbf{t}$ is indeterminate, and irrelevant, we can intersect the vector $\mathbf{t}$ with a plane, as in Fig 3 reducing the problem to two variables.

Thus, we have replaced an SOCP feasibility problem in $3n + 2$ variables and $2n$ constraints by an LP feasibility problem in only 2 variables and $2n$ constraints. The most important advantage is not simply the greater speed of LP versus SOCP, but rather that in the SOCP problem, the coordinates of the 3D points $\mathbf{X}_i$ appear as parameters, whereas we have eliminated them in the LP formulation. This makes for a much smaller feasibility problem that can be solved orders of magnitude faster.

8. Verification and Testing

We tested the algorithms thoroughly on real and synthetic data. Algorithms were coded in C++ and also Matlab. For the Matlab implementations, Second Order Cone Programming (Sedumi) was used to carry out the feasibility tests. For the C++ implementation we made use of the LP-based method sketched in section 7. For comparison only, we also implemented in C++ a method, based on SOCP, that approximated the circular cones by polyhedral cones. This allowed us to compare a quasi-SOCP based method coded in C++. (No SOCP implementation in C++ is available to us.) It was determined that the LP-based method of section 7 often gave a speed up of over 1000 times over other methods. Timing data is therefore reported for this algorithm.

8.1. Essential Matrix

We tested the algorithm on many synthetic examples. Generally speaking, the speed of convergence of the algorithm was closely tied to field of view of the camera. For cameras with 360° field of view, the convergence was very fast.

Generally, the subdivision search for the optimal rotation was carried out using cubes in rotation space down to a predetermined resolution. This results in a finite region of rotation space in which the rotation must lie. Using a breadth-first search, we consider cubes of diminishing size. When all cubes of a given size have been considered (we call this a phase of the algorithm), the remaining cubes are subdivided and considered in the next phase. In Fig 4 the number of remaining cubes and the remaining volume after each phase is shown.

Next, the remaining shape of the rotation space is shown in a few examples, cf. Fig 5.

Reconstruction from small numbers of points. It is well known that the essential matrix can be computed from only 5 points, in which case up to 10 solutions may occur. However, it is also possible to attempt to compute the essential matrix from 3 and 4 point correspondences. In the case of 5 points, there is a 0-dimensional set of exact solutions. For 4 and 3 point correspondences, one finds 1 and 2 dimensional sets of possible rotations, embedded in the 3-dimensional rotation space. This is shown for synthetic data in Fig 6 and Fig 7 for 3 and 4 points, respectively.

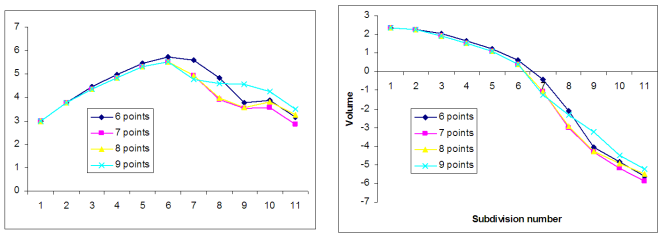

Figure 4. *Left: The number of remaining cubes (left) and the volume (right) are plotted (on a log-10 scale) against the number of subdivision phases. At the final subdivision, the cube half-side length is* 6.13×10^{-4} *radians, and the rotation is known to lie in a region with volume about* 10^{-6} *cubic radians. In other words, the rotation is known within about* 10^{-2} *radians. (The rotation could of course be computed with arbitrary accuracy.)*

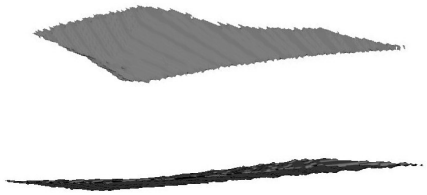

Figure 5. *For synthetic data with* $n = 20$ *points the shape of the remaining rotation region is shown (two views) after running the algorithm to resolution of* 0.001 *radians. The example is for a sideways motion of the camera. The shape of the rotation region is quite flat and elongated. This is explained by the known translation/rotation ambiguity, that translation and rotation of a camera are at times difficult to distinguish, particularly for small fields of view.*

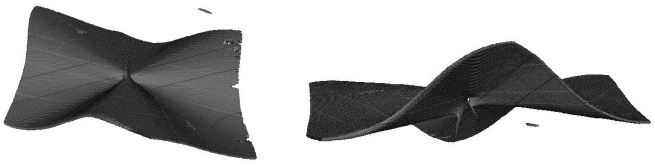

Figure 6. *Shape of the possible rotations for* 3 *point correspondences. The space of possible rotations forms a surface in rotation space.*

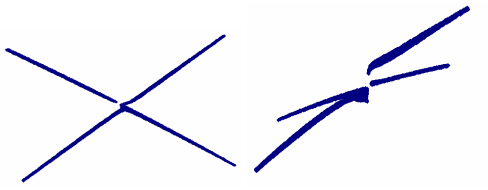

Figure 7. *Shape of the possible rotations for* 4 *point correspondences. The space of possible rotations forms a set of curves in rotation space.*

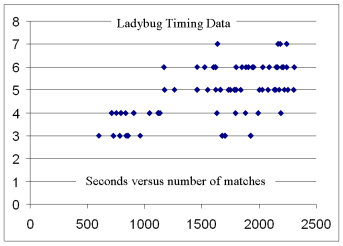

Figure 8. *Time in seconds plotted against number of points used to compute the essential matrix, for* 360° *data from a Ladybug camera. Even very large problem sizes take less than 10 seconds.*

Timing information The speed of convergence depends on many factors, most notably the field of view and the degree of perspective in the images. These factors vitally effect the branch-and-bound convergence rate, that is the number of cubes that need to be tested. Generally speaking, the computation time can be large for small numbers of points, because the number of required tests is high, since solutions with rotations far from the correct one can still have relatively small error – the data does not constrain the rotation so strongly. On the other hand, for large problems, the number of tests required is much smaller but each individual feasibility test is more expensive.

It has been mentioned that the feasibility problem involved in this method could be solved using SOCP. However, for problems of this size, the run time would be exorbitant. Instead, we found an optimal algorithm using linear programming (LP), which runs orders of magnitude faster. This algorithm will be discussed elsewhere; timing Fig 8 shows some typical timing information for various numbers of points, using the LP algorithm.

Ordinary perspective cameras have a smaller field of view, and are hence the algorithm is slower, since it becomes harder to eliminate rotation cubes. Here are some example times for image pairs taken from the Notredame data set provided by Noah Snavely.

correspondences	6572	794	50
time	1h 33m	6m 14s	56s

8.2. Pose estimation

The camera pose estimation has been implemented in Matlab using SOCP feasibility tests in the branch and bound algorithm. Timings reported below should take this into consideration. We believe that a fast LP implementation would result in a speed-up of a factor 10 to 100.

In Fig 9 and Fig 10, camera pose computations are reported for both zero-th and first order bounds. Note that there is a large difference between the two ways of bounding the error. The number of remaining cubes after each subdivision phase is considerably larger by a factor of at

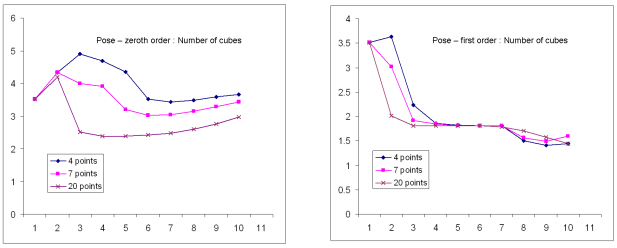

Figure 9. *The number of remaining cubes (on a log-*10 *scale) is plotted against the number of subdivision phases for camera pose computations on real data. The average of all* 11 *cameras in the publicly available Oxford Corridor sequence is given for various number of randomly chosen point correspondences. Left: the zero-th order algorithm; Right: the first order algorithm. The number of cubes examined at each iteration is around* 100 *times less for the first order algorithm. Also note that the number of cubes considered decreases with the size of the problem.*

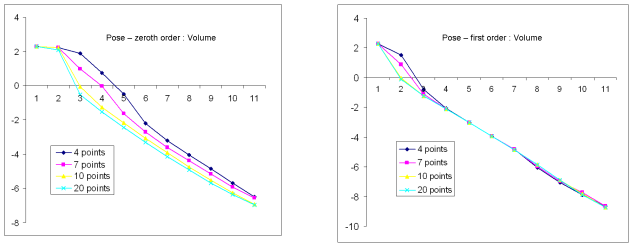

Figure 10. *The remaining volume (on a log-10 scale) of rotation space is plotted against the number of subdivision phases for camera pose computations on real data. The average of all* 11 *cameras in the corridor sequence is given for various numbers of randomly chosen point correspondences. Left: the zero-th order algorithm; Right: the first order algorithm. After* 11 *iterations, the remaining volume is a factor of around* 100 *less for the first order algorithm.*

least 100 for the weaker zero-th bound than the first order bound.

Each SOCP feasibility problem is slightly more complicated for a first order bound compared to a zero-th order bound since the dimension of the problem is higher due to the first order terms. Each individual feasibility test takes approximately 20% longer time to execute, but the total time gained with the first order method is considerable. On the average, for a 4-point pose problem in the corridor sequence, the execution time is around 1 hour for the zero-th order, but only 2 minutes for the first order method. Corresponding numbers for a 10-point pose problem are 10 minutes (zero-th order) and 1.5 minutes (first order), respectively.

9. Conclusion

The algorithm for the essential matrix works extremely well for 360° images, such as Ladybug$^{\text{TM}}$images, less quickly, but still reasonably for narrower field of view images. In some instances this algorithm will be preferable to known algorithms in real instances where real time speed is not an issue. Another important role for this algorithm is in giving a bench-mark optimal solution against which other algorithms may be judged.

The method of searching over rotations proposed here has applications on other problems that we are still exploring. It has the potential for wide applicability.

References

[1] S. Agarwal, M.K. Chandraker, F. Kahl, S. Belongie, and D.J. Kriegman. Practical global optimization for multiview geometry. In *European Conf. Computer Vision*, pages 592–605, Graz, Austria, 2006.

[2] R. Hartley and F. Schaffalitzky. L_∞ minimization in geometric reconstruction problems. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition, Washington DC*, pages I–504–509, June 2004.

[3] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision – 2nd Edition*. Cambridge University Press, 2003.

[4] F. Kahl. Multiple view geometry and the L_∞-norm. In *Proc. International Conference on Computer Vision*, pages 1002–1009, 2005.

[5] Q Ke and T Kanade. Quasiconvex optimization for robust geometric reconstruction. In *Proc. International Conference on Computer Vision*, pages 986 – 993, 2005.

[6] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, September 1981.

[7] C. Olsson, F. Kahl, and M. Oskarsson. Optimal estimation of perspective camera pose. In *Int. Conf. Pattern Recognition*, Hong Kong, China, 2006.