Global Optimization through Rotation Space Search

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

Fredrik Kahl
Centre for Mathematical Sciences
Lund University, Sweden.

Abstract

This paper introduces a new algorithmic technique for solving certain problems in geometric computer vision. The main novelty of the method is a branch-and-bound search over rotation space, which is used in this paper to determine camera orientation. By searching over all possible rotations, problems can be reduced to known fixed-rotation problems for which optimal solutions have been previously given. In particular, a method is developed for the estimation of the essential matrix, giving the first guaranteed optimal algorithm for estimating the relative pose using a cost function based on reprojection errors. Recently convex optimization techniques have been shown to provide optimal solutions to many of the common problems in structure from motion. However, they do not apply to problems involving rotations. The search method described in this paper 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. The approach has been implemented and tested on a number of both synthetically generated and real data sets with good performance.

1 Introduction

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. Optimal (minimum cost) solutions will be given to these problems under the geometrically meaningful L_∞ cost function. Although these problems have been well studied in the past, no previous solutions have claimed optimality under any sort of meaningful geometric error model.

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 $\mathtt{P}$. A solution to this problem is given by the DLT algorithm ([5], 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 commonly solved by computing the fundamental or essential matrix; a commonly used algorithm is the 8-point algorithm ([11] or [5], 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 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

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

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.

The main technical obstactle for solving these kinds of problems is that they are by nature non-convex and may contain several local optima. This is true for other simpler multiview problems like triangulation as well, but unlike triangulation - for which there are rarely any local minima in practice (see [7]) - the pose problems we consider are known to be plagued by local minima and ambiguous solutions for real image problems, see [13, 16, 14]. Our solution for dealing with the non-convexity is accomplished by (i) efficiently searching the rotational manifold and (ii) exploiting the quasiconvexity of the problems when rotations are assumed to be known. The framework is based on branch-and-bound in the rotation space and hence one needs bounding functions to control the error. In order to speed up the computations, we also show that the relative pose problem can be solved using just Linear Programming (LP), and hence the computationally more expensive Second Order Cone Programming (SOCP) is not required.

1.1 Related Work

To this point, there has been no known efficient optimal solution to these problems. Solutions have been given in [12] for the pose problem, but the solution we give here is considerably faster and the approach in [12] does not generalize to the relative pose problem. To our knowledge, no optimal solution has been reported for the relative pose problem using a cost function based on reprojection errors. Optimal L_∞ solutions are given in this paper for both problems and the algorithms are efficient and fast. A preliminary version of the work has appeared in the conference paper [2].

The class of problems that can be solved globally with the L_∞-norm includes problems like n-view triangulation, uncalibrated camera pose, homography estimation and structure from motion with a reference plane and more, see [6, 10, 8]. The current paper further extends this class to include problems involving unkwown rotations. 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 [7] and [9]. 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, but with no guarantee of optimality. Neither of these two approaches has shown the ability to optimize over rotation space. A summary of research in this area is given in [3].

Another class of related problems that have been addressed using branch-and-bound techniques is geometric matching problems; see [1, 13] and the references there. The problems considered in [1] are harder in the sense that feature correspondences are not known a priori. On the other hand only problems with a small number of degrees of freedom seem to be tractable. Typically, a planar Euclidean transformation which maps one set of points to another is computed.

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

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

We will often be considering situations where a measured image point $\mathbf{v}$ is compared with a modelled point $\hat{\mathbf{v}}(\theta)$, where θ is a set of parameters that define the point $\hat{\mathbf{v}}$. We typically require that

the angle $\angle(\mathbf{v}, \hat{\mathbf{v}}(\theta))$ should be less than some error bound ϵ. Assuming that $\mathbf{v}^\top \hat{\mathbf{v}}(\theta) > 0$, this may be written as

$$\frac{\|\mathbf{v} \times \hat{\mathbf{v}}(\theta)\|}{\mathbf{v}^\top \hat{\mathbf{v}}(\theta)} \leq \tan(\epsilon) \ ,$$

or equivalently

$$\|\mathbf{v} \times \hat{\mathbf{v}}(\theta)\| - \epsilon' \mathbf{v}^\top \hat{\mathbf{v}}(\theta) \leq 0 \tag{1}$$

where $\epsilon' = \tan(\epsilon) \geq 0$. Note that this inequality implies the condition that $\mathbf{v}^\top \hat{\mathbf{v}}(\theta) > 0$. Now, it was observed in [6, 10, 8] that as long as $\hat{\mathbf{v}}(\theta)$ is expressed linearly in terms of the parameters θ, the condition (1) has the form of a second-order cone constraint. For a fixed ϵ', this is a convex constraint since the constraint function is the sum of a linear (hence convex) function $-\epsilon' \mathbf{v}^\top \hat{\mathbf{v}}(\theta)$, and the norm of a linear function.

Combining several such second-order constraints from different measurement vectors $\mathbf{v}_i$ leads to a so called *Second-Order Cone Program* (SOCP). For a given ϵ', one may ask to find any θ satisfying all the constraints. Since there is no objective function we have what is known as an *SOCP feasibility problem* which is easily solvable using commonly available software packages, for example, SeDuMi [15]. One can perform a binary search for the optimal ϵ' to solve the minimax optimization problem

$$\min_{\theta} \max_{i} \angle(\mathbf{v}_i, \hat{\mathbf{v}}_i(\theta)) \ . \tag{2}$$

by solving a series of SOCP feasibility problems, provided that each $\hat{\mathbf{v}}_i(\theta)$ is a linear expression in the parameters θ. For more details, see [8].

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

$$\min_{\mathtt{R},\mathbf{C}} \max_{i} \angle(\mathbf{v}_i, \mathtt{R}(\mathbf{X}_i - \mathbf{C})) \ . \tag{3}$$

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

This problem is of the form (2), but unfortunately, the vector $\mathtt{R}(\mathbf{X}_i - \mathbf{C})$ is not expressed linearly in terms of a set of parameters for the unknowns $\mathtt{R}$ and $\mathbf{C}$, so we have no direct solution to this problem using SOCP. It is interesting and relevant to note, however, that if the rotation $\mathtt{R}$ is known, then this problem is solvable. In fact, it is identical to the L_∞ triangulation problem. 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 [4]; SOCP was used to find the optimal solution in [6, 10, 8].

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

[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.

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.

$$\min_{\mathtt{R}_j, \mathbf{C}_j, \mathbf{X}_i} \max_{i,j} \angle(\mathbf{v}_{ij}, \mathtt{R}_j(\mathbf{X}_i - \mathbf{C}_j)) \ . \tag{4}$$

To simplify this problem, we may assume that the first camera has rotation $\mathtt{R}_1$ equal to the identity, 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, this problem is of the form given by (2), but it may not be formulated linearly in the parameters of the unknown quantities. However 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 [4]. As before, SOCP provides an efficient solution ([6, 8]).

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 ([7]).

Existence of a solution. The description of the minimization problems (3) and (4) above is written under the assumption that a minimum exists and is in fact attained. For the sake of mathematical rigour, we settle this issue here. A function attains its minimum if it is defined on a compact set and is continuous. However, the parameters $\mathbf{X}_i$ are defined to lie in $\mathcal{R}^3$, which is not compact (in the standard topology). To avoid this difficulty we simply compactify $\mathcal{R}^3$ by allowing points at infinity. In other words, we perform optimization over the oriented projective space $\mathcal{P}^{3+}$, defined as equivalence classes of non-zero homogeneous vectors $\{\mathbf{X} = (\mathtt{X}, \mathtt{Y}, \mathtt{Z}, \mathtt{T}) \neq \mathbf{0} \mid \mathtt{T} \geq 0\}$ where two such vectors are considered equivalent if they differ by a positive constant multiplier. Points for which $\mathtt{T} = 0$ are points at infinity, but unlike the usual projective space $\mathcal{P}^3$, points at infinity in opposite directions are not identified. This being the case, one may unambiguously extend the function $\angle(\mathbf{v}_i, \mathbf{X}_i - \mathbf{C}_j)$ to points at infinity $\mathbf{X}_i$.

It is easily seen that each equivalence class has a unique representative such that $\|\mathbf{X}\|^2 = \mathtt{X}^2 + \mathtt{Y}^2 + \mathtt{Z}^2 + \mathtt{T}^2 = 1$ and $\mathtt{T} \geq 0$. That is, $\mathcal{P}^{3+}$ is homeomorphic to the closed half-sphere S^{3+} in $\mathcal{R}^4$, and hence is compact.

A further small difficulty, that $\angle(\mathbf{v}_i, \mathbf{X}_i - \mathbf{C}_j)$ has an essential discontinuity when $\mathbf{X}_i = \mathbf{C}_j$ is avoided by defining $\angle(\mathbf{v}_i, \mathbf{0}) = 0$. Since we are always interesting in minimizing an L_∞ norm such as $\max_{ij} \angle(\mathbf{v}_{ij}, \mathbf{X}_i - \mathbf{C}_j)$, this function will be continuous unless all $\mathbf{X}_i$ and $\mathbf{C}_j$ are equal, which is easily avoided. In the relative orientation problem, for instance, we enforce a unit distance between the two cameras.

In this way, we may ensure that the objective function is continuous and defined on a compact domain, and hence achieves its minimum. From a practical implementation point of view, we find it unnecessary to use homogeneous coordinates in this way, since our method of solution is to approach the optimal solution through a sequence of feasibility problems. For this reason, we do not consider this issue throughout the rest of the paper.

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 $\mathcal{R}^n$, 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 $\mathcal{R}^3$.

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 $\epsilon_{\min}$ for 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 breadth-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 The Geometry of the Space of Rotations

The group of all rotations is often referred to as $SO(3)$, although strictly speaking this only refers to its representation as 3×3 orthogonal matrices. We will use $SO(3)$ to denote the group of rotations considered abstractly, but with a specific concrete representation in mind, namely in terms of the rotations' matrix representation.

Distance between two rotations. The group of rotations has a metric structure, defined by the angle metric, defined as follows. Note that 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 in $SO(3)$. We wish to measure the distance between these 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 be using the following inequality, which seems simple enough that we omit the proof.

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

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

We shall have cause also to consider two different representations of the set of rotations, quaternions and the angle-axis formulation of rotations, and we will be interested in the relationship between the angle metric, and natural metrics defined for these alternative representations.

Unit quaternions. A quaternion is a 4-vector $\mathbf{q} = (q_0, q_1, q_2, q_3)$ of real numbers with a defined non-commutative multiplication operation ([17] "Quaternion"). It may be verified that $\|\mathbf{q}_1\mathbf{q}_2\| = \|\mathbf{q}_1\| \, \|\mathbf{q}_2\|$ where $\|\cdot\|$ represents Euclidean norm in the 4-dimensional vector space. Thus, the unit quaternions form a group under multiplication. We denote the group of unit quaternions by H (in honour of Hamilton).

Connection with $3D$ rotations. A unit quaternion may be written as $\mathbf{q} = (\cos(\alpha/2), \sin(\alpha/2)\hat{\mathbf{r}})$ where $\hat{\mathbf{r}}$ is a unit vector and $0 \leq \alpha \leq 2\pi$. We identify this quaternion $\mathbf{q}$ with the rotation $R(\alpha, \hat{\mathbf{r}})$, namely the rotation through angle α about the axis $\hat{\mathbf{r}}$. It turns out that this mapping is a homomorphism from the unit quaternions onto the group of $3D$ rotations. One observes that a quaternion and its negative map to the same rotation. Thus, mapping from unit quaternions to rotations is a 2-to-1 mapping, inducing an isomorphism between the group of rotations and the unit quaternions modulo negation. Any rotation may be represented by a unit quaternion of this form with $0 \leq \alpha \leq \pi$.

Considered geometrically, the unit quaternions form a 3-dimensional sphere in 4-space. The upper hemisphere (those quaternions with $q_0 \geq 0$) is in one-to-one correspondence with the rotations, except at the boundary, where the correspondence is 2-to-1. We denote by $\tilde{H}$ the set of equivalence classes of unit quaternions, modulo the equivalence of a quaternion and its negative.

Distance in quaternion space. One may define a simple metric on the set of unit quaternions H as follows. Let $\mathbf{p}$ and $\mathbf{q}$ be unit quaternions, which may therefore be represented by points on the unit 3-sphere. We define the distance $d_g(\mathbf{p}, \mathbf{q})$ to be the geodesic distance on the sphere between points $\mathbf{p}$ and $\mathbf{q}$. In simpler terms, this is the distance between $\mathbf{p}$ and $\mathbf{q}$ along a great circle on the sphere. This great-circle distance is easily computed by computing the inner product of $\mathbf{p}$ and $\mathbf{q}$ as vectors. To avoid confusion with quaternion multiplication we write this inner product as $\langle \mathbf{p}, \mathbf{q} \rangle = \sum_{i=0}^{3} p_i q_i$. Since this inner product gives the cosine of the angle between $\mathbf{p}$ and $\mathbf{q}$, the distance metric is defined as

$$d_g(\mathbf{p}, \mathbf{q}) = \arccos\langle \mathbf{p}, \mathbf{q} \rangle \ ,$$

where $\arccos$ takes values between 0 and π. Since this distance function is defined in terms of geometric distance between points on the sphere, it is clearly a metric. The metric takes values between 0 and π. It is a basic property that this distance metric is invariant under quaternion multiplication, namely $d_g(\mathbf{pr}, \mathbf{qr}) = d_g(\mathbf{p}, \mathbf{q})$.

This metric induces a metric $\tilde{d}$ on the group $\tilde{H}$ of equivalence classes modulo negation. In particular given two equivalence classes $\tilde{\mathbf{p}} = \{\mathbf{p}, -\mathbf{p}\}$ and $\tilde{\mathbf{q}} = \{\mathbf{q}, -\mathbf{q}\}$, we define

$$\begin{aligned} \tilde{d}_g(\tilde{\mathbf{p}}, \tilde{\mathbf{q}}) &= \min\left(d_g(\mathbf{p}, \mathbf{q}), d_g(\mathbf{p}, -\mathbf{q})\right) \\ &= \min\left(d_g(\mathbf{p}, \mathbf{q}), \pi - d_g(\mathbf{p}, \mathbf{q})\right) . \end{aligned}$$

Isometry between $\tilde{H}$ and $SO(3)$. Note that the metric $\tilde{d}_g$ takes values between 0 and $\pi/2$, whereas the metric $d_\angle$ on rotations takes values between 0 and π. It should comes as no surprise that these metrics are closely related. In fact, the mapping from $\tilde{H}$ to the group of rotations is a scaled isometry with respect to the two metrics. In particular consider quaternions $\mathbf{p}$ and $\mathbf{q}$, and their corresponding rotations $\mathtt{R}_\mathbf{p}$ and $\mathtt{R}_\mathbf{q}$ respectively. The two metrics are related as follows:

$$d_\angle(\mathtt{R}_\mathbf{p}, \mathtt{R}_\mathbf{q}) = 2\,\tilde{d}_g(\tilde{\mathbf{p}}, \tilde{\mathbf{q}})\ .$$

In passing we observe that this relationship allows us to determine the "volume" of the group of all rotations. The group $\tilde{H}$ may be represented as the upper hemisphere of the unit sphere S^3, which has volume π^2 ([17], "Sphere"). Since the rotations are twice as big (in linear dimension), the set of all $3D$ rotations has volume $8\pi^2$ cubic radians.

3.1 The Angle-Axis Representation

We now discuss how best to represent rotations for our branch and bound application. A nice discussion of rotation representations is given in [17] (search for "rotation representation"). However, the required details of our chosen representation will be given below.

We have seen that a rotation may be represented by a quaternion $\mathbf{q} = (\cos(\alpha/2),\ \sin(\alpha/2)\hat{\mathbf{r}})$ where α is the angle of the rotation and $\hat{\mathbf{r}}$ is a unit vector representing the axis of the rotation. (We shall in general use $\hat{\mathbf{r}}$ to represent a unit vector.)

An alternative is to represent the rotation by the vector $\mathbf{r} = \alpha\hat{\mathbf{r}}$. This is a vector with magnitude α, the angle of the rotation, and with direction the axis of the rotation. This is a result of applying a so-called "azimuthal-equidistant" projection (in France, sometimes called the Postel projection, after Guillaume Postel, d 1581) to the unit quaternion sphere, according to the mapping:

$$(\cos(\alpha/2),\ \sin(\alpha/2)\hat{\mathbf{r}}) \mapsto \alpha\hat{\mathbf{r}}. \tag{5}$$

This mapping may be thought of as taking the upper quaternion hemisphere and flattening it, much as one might take a tennis ball cut in half and flatten it by pushing down to a plane. This causes tangential stretching at the periphery.

Because of the tangential stretching, we may intuitively observe that

$$2\tilde{d}(\tilde{\mathbf{q}}_1, \tilde{\mathbf{q}}_2) \le \|\mathbf{r}_1 - \mathbf{r}_2\|$$

where $\|\cdot\|$ represents Euclidean norm in $\mathcal{R}^3$, and $\mathbf{r}_i = \alpha_i\hat{\mathbf{r}}_i$ is the vector corresponding to quaternion $\mathbf{q}_i = (\cos(\alpha_i/2), \sin(\alpha_i/2)\hat{\mathbf{r}}_i)$.

This relationship may be proved rigorously by computing an infinitessimal metric on the quaternion sphere. Thus, let $\mathtt{J}$ be the Jacobian of the mapping $f : \mathbf{r} \mapsto \mathbf{q} = (\cos(\alpha/2), \sin(\alpha/2)\hat{\mathbf{r}})$, where $\mathbf{r} = \alpha\hat{\mathbf{r}}$. Then an infinitessimal metric on the quaternion unit sphere is given by $ds^2 = d\mathbf{r}^\top(\mathtt{J}^\top\mathtt{J})d\mathbf{r}$. The symmetric positive-definite matrix $\mathtt{J}^\top\mathtt{J}$ may be computed explicitly. Its eigenvalues are all at most equal to $1/2$. This means that a line segment from $\mathbf{r}_1$ to $\mathbf{r}_2$ maps under f to a path from $\mathbf{q}_1$ to $\mathbf{q}_2$ of length no greater than half the length $\|\mathbf{r}_1 - \mathbf{r}_2\|$ on the quaternion sphere. Since the geodesic distance $d_g(\mathbf{q}_1, \mathbf{q}_2)$ will be less than the length of this path, we have

$$\tilde{d}_g(\tilde{\mathbf{p}}, \tilde{\mathbf{q}}) \le d_g(\mathbf{p}, \mathbf{q}) \le \|\mathbf{r}_1 - \mathbf{r}_2\|/2$$

as required.

Finally, relating the Euclidean metric to the angle metric, we have the following relationship.

Lemma 3.2. *If $\mathbf{q}_i = (\cos(\alpha_i/2), \sin(\alpha_i/2)\hat{\mathbf{r}}_i)$ for $i = 1, 2$ and $\mathtt{R}_i$ are the corresponding rotations, and $\mathbf{r}_i = \alpha_i\hat{\mathbf{r}}_i$, then*

$$d_\angle(\mathtt{R}_1, \mathtt{R}_2) \le \|\mathbf{r}_1 - \mathbf{r}_2\|\ .$$

The important point in this lemma is that the angle distance is less than the Euclidean distance in the angle-axis representation.

An alternative way of thinking of the angle-axis representation of rotations is that a vector $\mathbf{r} = \alpha\hat{\mathbf{r}}$, where $\hat{\mathbf{r}}$ is a unit vector, represents the rotation

$$\begin{aligned} \mathtt{R} &= \exp([\mathbf{r}]_\times) = \mathtt{I} + [\mathbf{r}]_\times + [\mathbf{r}]_\times^2/2 + \dots \\ &= \mathtt{I} + \sin\alpha[\hat{\mathbf{r}}]_\times + (1-\cos\alpha)[\hat{\mathbf{r}}]_\times^2 \ , \end{aligned} \tag{6}$$

which is the Rodrigues formula (see [5]). Here $[\mathbf{r}]_\times$ denotes the 3×3 skew-symmetric matrix such that $\mathbf{r} \times \mathbf{v} = [\mathbf{r}]_\times \mathbf{v}$ for all 3-vectors $\mathbf{v}$. Through the association of the rotation $\mathtt{R}$ with the vector $\mathbf{r}$, we see that the set of rotations is represented by the ball of radius π in $\mathcal{R}^3$. The correspondence is one-to-one on the interior of the ball, whereas for vectors $\mathbf{r}$ with $\|\mathbf{r}\| = \pi$, the correspondence is 2-to-1, since both $\mathbf{r}$ and $-\mathbf{r}$ represent the same rotation. A rotation through angle π about an axis is the same as the rotation through angle π about the opposite axis.

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_π^3 of radius π in $\mathcal{R}^3$. We may enclose this ball in a cube $C_\pi^3 = [-\pi, \pi]^3$ in $\mathcal{R}^3$, 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^3 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_π^3. 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

$$r_D \leq \sqrt{3}\sigma \tag{7}$$

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

4 Feasibility Problems

The most important requirement in the branch-and-bound method is to be able to determine whether the function being minimized may attain a value less than the current minimum $\epsilon_{\min}$ on a restricted domain D in the parameter space. We will show how this is done in the case of our problems of interest.

First, we will detail how to obtain a bound for the relative pose problem in section 4.1 and then continue with the pose problem in section 4.2. In the two subsequent sections, we will show how one can improve computational efficiency in two different ways. In section 5, a more elaborate and efficient linear programming solution is derived for the relative pose problem. In section 6, a tighter bounding function than the one obtained in section 4.2 is derived for the pose problem.

4.1 The Relative Orientation Problem

We consider a feasibility problem motivated by the relative pose problem defined for a restricted rotation domain, D with radius r_D. We assume throughout that a set of correspondences $\mathbf{v}_i \leftrightarrow \mathbf{v}_i'$; $i = 1, \dots, N$ are given. 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:

$$\begin{array}{ll} \textbf{Given} & D, \epsilon_{\min} \\ \textbf{do there exist} & \mathbf{C}, \mathbf{X}_i, \mathtt{R} \in D \\ \textbf{such that} & \angle(\mathbf{v}_i, \mathbf{X}_i) \leq \epsilon_{\min} \\ \textbf{and} & \angle(\mathbf{v}'_i, \mathtt{R}(\mathbf{X}_i - \mathbf{C})) \leq \epsilon_{\min} \\ \textbf{for} & i = 1, \ldots, N \ ? \end{array} \tag{8}$$

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 (8) directly. Instead, we consider a slightly weaker, but solvable problem in which we fix the rotation and place a slightly weaker bound. Let $\mathtt{D}$ be a cube in rotation space with half-side length σ and centre representing the rotation $\hat{\mathtt{R}}$. Then, consider the problem

$$\begin{array}{ll} \textbf{Given} & \bar{\mathtt{R}}, \sigma, \epsilon_{\min} \\ \textbf{do there exist} & \mathbf{C}, \mathbf{X}_i \\ \textbf{such that} & \angle(\mathbf{v}_i, \mathbf{X}_i) \leq \epsilon_{\min} \\ \textbf{and} & \angle(\mathbf{v}'_i, \bar{\mathtt{R}}(\mathbf{X}_i - \mathbf{C})) \leq \epsilon_{\min} + \sqrt{3}\,\sigma \ ? \end{array} \tag{9}$$

Observe that the two constraints in problem (9) can written as second-order cone constraints of the type given in (1). Therefore, the problem is formulated as a SOCP feasibility problem, and the question may then be answered using an SOCP solver.

The two problems (8) and (9) are related as follows.

Lemma 4.3. *If problem (8) has an affirmative answer then so does problem (9). On the other hand, if the answer to problem (8) is negative on a domain D, then D may be split into subdomains D_i of sufficiently small radius such that problem (9) has a negative answer on every D_i.*

Proof. First we prove that if problem (8) has an affirmative answer then so does problem (9). Suppose that $\mathbf{C}$, $\{\mathbf{X}_i\}$ and $\mathtt{R}_{\text{opt}}$ constitute a feasible solution for problem (8). Then we show that $\mathbf{C}$ and $\{\mathbf{X}_i\}$ constitutes a solution to problem (9).

The first constraint $\angle(\mathbf{v}_i, \mathbf{X}_i) \leq \epsilon_{\min}$ is fulfilled, since it is the same as for problem (8). As for the second constraint, by the triangle inequality, we have

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

as required. Note where lemma 3.1 was used.

Now for the second part of the lemma. Suppose that problem (8) is infeasible, and that the minimum of all values of ϵ for which it is solvable is ϵ^+, instead of $\epsilon_{\min}$. Thus $\epsilon^+ > \epsilon_{\min}$ and the constraints $\angle(\mathbf{v}_i, \mathbf{X}_i) \leq \epsilon$ and $\angle(\mathbf{v}'_i, \mathtt{R}(\mathbf{X}_i - \mathbf{C})) \leq \epsilon$ can not be satisfied for any choice of $\mathbf{X}_i$, $\mathbf{C}$ and $\mathtt{R} \in D$ and any value of $\epsilon < \epsilon^+$. Choose σ such that $\epsilon_{\min} + \sqrt{3}\sigma < \epsilon^+$. Then problem (9) is infeasible on any subcube D_i of D with side half-length σ. The domain D may be divided into a finite number of such cubes so that problem (9) is infeasible on any one of them. This completes the proof of the lemma.

This lemma gives a strategy for showing that problem (8) is not feasible on domain $D = [-\sigma, \sigma]^3$. Letting $\bar{\mathtt{R}}$ be the rotation represented by the centre of D, we ask whether problem (9)

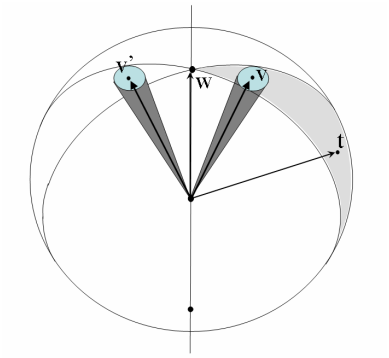

Figure 1: *The epipole lies in the wedge of a sphere bounded by the pair of crossed great circles, and containing the vector* $\mathbf{v}$.

is feasible. If the answer is no (problem is not feasible), then neither is problem (8). If on the other hand problem (9) is feasible, we subdivide domain D into smaller cubic domains and continue by testing these subdomains.

4.2 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

$$\begin{array}{ll} \textbf{Given} & \mathbf{X}_i, D, \epsilon_{\min} \\ \textbf{do there exist} & \mathbf{C}, \mathtt{R} \in D \\ \textbf{such that} & \angle(\mathtt{R}\mathbf{v}_i, \mathbf{X}_i - \mathbf{C}) \leq \epsilon_{\min} \ ? \end{array} \tag{10}$$

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

$$\begin{array}{ll} \textbf{Given} & \mathbf{X}_i, \bar{\mathtt{R}}, \sigma, \epsilon_{\min} \\ \textbf{does there exist} & \mathbf{C} \\ \textbf{such that} & \angle(\bar{\mathtt{R}}\mathbf{v}_i, \mathbf{X}_i - \mathbf{C}) \leq \epsilon_{\min} + \sqrt{3}\,\sigma \ ? \end{array} \tag{11}$$

This is essentially the triangulation problem, and it may be solved using SOCP. As in the relative pose case, problem (10) may be relaxed to (11), which is then used to solve the pose problem by the branch-and-bound technique.

5 A Linear Programming Solution for Relative Pose

For the relative pose 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 is presented now.

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 (8). 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 1. 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).

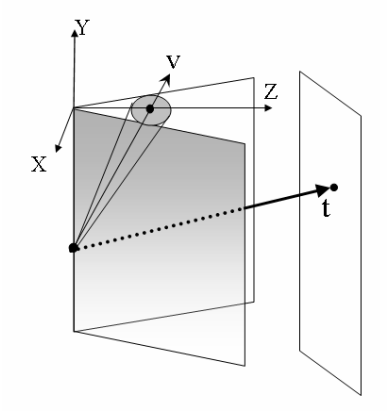

Figure 2: *The epipolar vector lies on the plane* $\mathrm{Z} = 1$ *in a coordinate frame determined by the pair of matched points* $\mathbf{v}$ *and* $\mathbf{v}'$*. In this frame, the* Y*-axis coincides with* $\mathbf{w}$.

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 2 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.

With 100 or even 1000 points involved in the reconstruction, the speed-up can be enormous, particularly as the problem must be solved repeatedly for differing values of ϵ in a binary search. Under the reasonable assumption that constrained programming is cubic in the size of the problem (although the commonly used Simplex algorithm can of course theoretically be exponential), and the *unreasonable* assumption that a SOCP problem takes the same amount of time as an LP programming problem of the same size, the difference in time represented by this new method on a problem involving 100 points, compared with the SOCP method is approximately a 650 times speedup.

5.1 More Details on Linear Programming

Let $\mathbf{t}$ be a unit direction vector from the camera centre $\mathbf{C}$ of the first camera to the centre $\mathbf{C}'$ of the second camera. This is the epipolar direction, which we wish to determine. Given a pair of image points $\mathbf{v}_i \leftrightarrow \mathbf{v}'_i$ corresponding to a $3D$ point $\mathbf{X}$, we have the equation

$$\mathbf{X} = \mathbf{C} + \lambda \mathbf{v}_i = \mathbf{C}' + \mu \mathbf{v}'_i \ .$$

for positive scalars λ and μ. This equation reduces to $\lambda \mathbf{v}_i - \mu \mathbf{v}'_i = \mathbf{C}' - \mathbf{C} = \nu \mathbf{t}$, where λ, μ, ν are all positive. Setting $\nu = 1$, we get the simple but key observation that

$$\mathbf{t} = \lambda \mathbf{v}_i - \mu \mathbf{v}'_i \text{ for positive constants } \lambda \text{ and } \mu \ . \tag{12}$$

Now, $\mathbf{v}_i$ and $\mathbf{v}'_i$ are vectors based at different basepoints. However, $\mathbf{v}'_i$ is simply a direction vector. We may move all of $\mathbf{v}_i$, $\mathbf{v}'_i$ and $\mathbf{t}$ to be based at the origin. They all then lie on the unit sphere, and in fact, being coplanar, all lie on a great circle. As before, expanding $\mathbf{v}_i$ and $\mathbf{v}'_i$ to cones, we see that the epipole $\mathbf{t}$ must lie on the section of the sphere bounded by the pair of crossed great circles. In fact, because of (12), the epipole must lie inside the wedge containing $\mathbf{v}_i$. This is

illustrated in Fig 1. If the two ϵ-circles about $\mathbf{v}_i$ and $\mathbf{v}'_i$ overlap, then there is no constraint (since any $\mathbf{t}$ would be feasible), and we ignore this corresponding pair.

Since we are dealing with a single correspondence at present, we drop the subscript i and write $\mathbf{v} \leftrightarrow \mathbf{v}'$. Now, since each great circle lies in a plane, it follows that $\mathbf{t}$ lies between the pair of planes of the two great circles, as shown in Fig 2. We relax the condition that $\mathbf{t}$ is a unit vector, and instead constrain it to lie on the plane $\mathrm{Z} = 1$ in a coordinate system chosen as follows.

1. The Y axis passes through the point $\mathbf{w}$ where the two great circles meet, between $\mathbf{v}$ and $\mathbf{v}'$.
2. The X-axis is in the direction of the cross product $\mathbf{w} \times \mathbf{v}$.
3. The Z-axis is $\mathbf{x} \times \mathbf{y}$ where $\mathbf{x}$ and $\mathbf{y}$ are the directions of the X and Y axes.

This choice of coordinate frame is carried out for the first matched point $\mathbf{v}_0 \leftrightarrow \mathbf{v}'_0$ only (provided it does yield a constraint, i.e. the ϵ-circles do not overlap). The other correspondences must be related to this coordinate frame. The easiest way is to rotate all the point correspondences $\mathbf{v}_i \leftrightarrow \mathbf{v}'_i$ for $i > 0$ to this coordinate frame, then compute the bounding planes in this coordinate frame, as in the next section.

Remark. Note that not all epipoles $\mathbf{t}$ lying between the two great circles, that is the marked wedge in Fig 1, are valid solutions. In fact the portion of the wedge lying between the $\mathbf{w}$-axis and the ϵ-cone around $\mathbf{v}$ corresponds to points that do not satisfy the constraint (12) for positive constants. This will result in a solution with reconstructed $3D$ points with negative depths (behind the cameras), which is geometrically invalid. The same is true for the symmetrically placed region lying between the negative $\mathbf{w}$ axis and the ϵ-cone about the vector $-\mathbf{v}'$. To remove such spurious solutions, further constraints could be added to the feasibility test. However, we have found experimentally and theoretically that this is not necessary in practice.

Indeed, by ignoring this small detail, the constraints on the vector $\mathbf{t}$ are made slightly less tight than they may be. This can conceivably result in a false positive answer to the feasibility test. This is not critical, since it can not result in the rejection of a postulated rotation value. Instead, in the branch-and-bound algorithm it will simply result in the current rotation cube being subdivided and deferred to the next level of resolution.

5.2 Computing the Great-Circles on the Unit Sphere

We want to compute the normals to the planes of the great-circles in Fig 1 tangent to the uncertainty cones for the measurements $\mathbf{v}_i$ and $\mathbf{v}'_i$.

5.2.1 General Algebraic Solution

It is possible to solve this easily in the case where the cones are elliptic, as follows. Observe that the planes of the great-circles in Fig 1 are tangent to the two cones. We find the bi-tangent planes as follows. The two cones may be expressed as equations $\mathbf{X}^\top \mathtt{Q} \mathbf{X} = 0$ and $\mathbf{X}^\top \mathtt{Q}' \mathbf{X} = 0$. where $\mathtt{Q}$ and $\mathtt{Q}'$ are 3×3 non-singular symmetric matrices. (For instance the case $\mathtt{Q} = \mathrm{diag}(1, -1, 1)$ represents a circular cone centred on the Y-axis.) The vector $\mathbf{X}$ represents a $3D$ point. Passing to the dual, we obtain the equations for the dual cones, represented by matrices $\mathtt{Q}^{-1}$ and $\mathtt{Q}'^{-1}$. A plane through the origin with normal $\mathbf{n}$ is tangent to the original cones if and only if $\mathbf{n}^\top \mathtt{Q}^{-1} \mathbf{n} = \mathbf{n}^\top \mathtt{Q}'^{-1} \mathbf{n} = 0$. Solving these two simultaneous quadratic equations results in 4 solutions corresponding to the four bi-tangent planes. We choose the solutions with appropriate sign to satisfy $\mathbf{n} \cdot \mathbf{v} > 0$ and $\mathbf{n} \cdot \mathbf{v}' < 0$. Geometrically, it is evident that there are just two such planes. Thus, we get two solutions $\mathbf{n}_1$ and $\mathbf{n}_2$ representing the planes in Fig 2. The linear constraints on the vector $\mathbf{t}$ are simply $\mathbf{n}_1 \cdot \mathbf{t} \geq 0$ and $\mathbf{n}_2 \cdot \mathbf{t} \geq 0$.

Setting $\mathrm{Z} = 1$ in these inequalities results in just two linear inequalities in X and Y representing constraints on the plane $\mathrm{Z} = 1$ in the coordinate system defined by the first point correspondence.

In this way, determining the existence or not of an epipole satisfying all constraints becomes an LP feasibility problem in only two variables.

5.2.2 A Closed-Form Expression

The method described above involves solution of a 4-th degree polynomial in order to find the bi-tangent planes. Instead, we show that it is possible to obtain a simple closed-form expression for the inequalities, as shown now. We consider the case where the cones are circular (we are interested in minimizing angle error on the sphere), but of possibly different diameters, given by angles ϵ and ϵ', centred on $\mathbf{v}$ and $\mathbf{v}'$ respectively. This is because problem (9) specifies different bounds for $\mathbf{v}_i$ and $\mathbf{v}'_i$.

The proof of the following results is given in an appendix. Refer to Fig 12 to aid in understanding this diagram.

1. First, $\mathbf{v} \cdot \mathbf{v}'$ is the cosine of the angle between $\mathbf{v}$ and $\mathbf{v}'$. If this is less than $\epsilon + \epsilon'$ then no constraint results from this correspondence, and we skip it. Let α be the angle between $\mathbf{v}$ and $\mathbf{v}'$.

2. The angle β between the plane containing $\mathbf{v}$ and $\mathbf{v}'$ and the crossed bi-tangent plane is given by

$$\sin^2 \beta = \frac{\sin^2(\epsilon) + 2\sin(\epsilon)\sin(\epsilon')\cos(\alpha) + \sin^2(\epsilon')}{\sin^2(\alpha)} \; . \tag{13}$$

3. The point $\mathbf{w}$ where the two great circles cross is

$$\mathbf{w} = \frac{\sin(\epsilon)\mathbf{v}' + \sin(\epsilon')\mathbf{v}}{\sin(\beta)\sin(\alpha)} \; . \tag{14}$$

4. The local coordinate system is defined by vectors $\mathbf{y} = \mathbf{w}$, $\mathbf{x} = (\mathbf{v} \times \mathbf{v}')/\|\mathbf{v} \times \mathbf{v}'\|$ and $\mathbf{z} = \mathbf{x} \times \mathbf{y}$.

5. The two normals are $\sin(\beta)\mathbf{z} \pm \cos(\beta)\mathbf{x}$.

5.3 Testing for Feasibility

The first point correspondence gives rise to a strip on the plane $\mathrm{Z} = 1$ between the lines $\mathrm{X} = \pm\tan(\beta)$. Thus, subsequent constraints bound a region in this strip. The question is whether there is a point in the strip that satisfies all the constraints. It would be possible to set this up as one LP programming problem consisting of $2n$ constraints in 2 variables, and then to solve it using an appropriate LP package. Here n is the number of point correspondences. If n is large, this can still take too long, particularly since we need to solve this problem a large number of times. We are looking for maximum speed.

In a search procedure there will be many instances in which the problem is infeasible, and it saves time to detect this early. It may be that with just a small number of constraints the problem already becomes infeasible. In the branch-and-bound algorithm, infeasible constraint problems are much more common than the feasible ones, so it is important to detect infeasible problems early.

After setting $\mathrm{Z} = 1$ in the constraints $\mathbf{n}_i \cdot \mathbf{t} \geq 0$, we obtain constraints of the form $a_i\mathrm{X} + b_i\mathrm{Y} + c_i \geq 0$, where $\mathbf{t} = (\mathrm{X}, \mathrm{Y}, 1)$. This can be rewritten (depending on the sign of b_i) as either $\mathrm{Y} \geq c_i + a_i\mathrm{X}$ or $\mathrm{Y} \leq c_i + a_i\mathrm{X}$ for new values of the constants a_i and c_i. Our algorithm is as follows. (Please refer to Fig 3).

1. Let X_- and X_+ be the two edges of the strip shown in Fig 3.

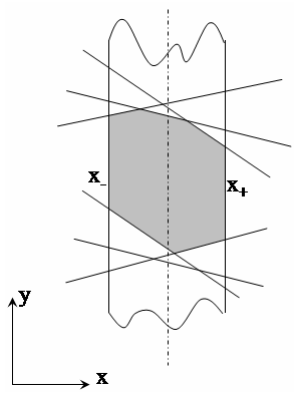

Figure 3: *Constraints on the position of the epipole* $\mathbf{t}$ *result in constraints on a strip in the plane* $\mathrm{Z} = 1$. *This strip is bounded by the lines* $\mathrm{X} = \pm\tan(\beta_0)$ *resulting from the first point correspondence. Considering the constraints one at a time, we compute loose bounds on the region, as discussed in the text.*

2. From $\mathrm{Y} \geq a_i + c_i\mathrm{X}$, we obtain a constraint $\mathrm{Y} \geq a_i + \min(c_i\mathrm{X}_-, c_i\mathrm{X}_+)$, which must hold for any point satisfying $\mathrm{Y} \geq a_i + c_i\mathrm{X}$. (Yes, this should be $\min$ and not $\max$, and of course the converse of this statement is not true.)

3. Similarly from $\mathrm{Y} \leq a_i + c_i\mathrm{X}$ we derive a constraint $\mathrm{Y} \leq a_i + \max(c_i\mathrm{X}_-, c_i\mathrm{X}_+)$.

4. Considering the constraints in order of arrival, we may compute loose running lower and upper bounds $\mathrm{Y}_{\mathrm{low}}$ and $\mathrm{Y}_{\mathrm{high}}$ for Y. If at any point $\mathrm{Y}_{\mathrm{low}} > \mathrm{Y}_{\mathrm{high}}$, it means that the constraints are infeasible, and we exit, reporting this fact.

This allows infeasible problems to be detected quickly. It also allows inactive constraints to be detected and eliminated, resulting in a smaller LP problem, should we need to run a complete LP algorithm. Specifically, any constraint that lies completely "below" the current value of $\mathrm{Y}_{\mathrm{low}}$ or above $\mathrm{Y}_{\mathrm{high}}$ does not need to be considered.

6 A First Order Bound for Pose

The bounds in section 4 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

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

where $\delta\mathbf{r}$ is the corresponding angle-axis representation of $\delta\mathtt{R}$. 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 $\mathtt{R}$ be a rotation and $\hat{\mathtt{R}}$ be its first order approximation about $\mathtt{R}_0$, where $d_\angle(\mathtt{R}, \mathtt{R}_0) < r_D < 0.76$. 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. The restriction $r_D < 0.76$ just implies that the size of the cubes in the branch-and-bound algorithm cannot be too large. This is easily handled by starting with a sufficiently fine subdivision of rotation space. This lemma will be proved in Appendix 2.

6.2 Revisiting the Pose Problem

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

$$\begin{array}{ll} \textbf{Given} & \mathbf{X}_i, \epsilon_{\min}, \bar{\mathtt{R}}, D, \text{ with } r_D < 0.76 \\ \textbf{do there exist} & \mathbf{C}, \hat{\mathtt{R}} \\ \textbf{such that} & \angle(\mathbf{v}_i, \hat{\mathtt{R}}(\mathbf{X}_i - \mathbf{C})) < \epsilon_{\min} + r_D^2/2 \ ? \end{array} \tag{15}$$

With essentially the same derivation as before, we can show that if this problem is infeasible, then so is problem (10). What remains to show is that we may find a solution to this problem. We may write

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

However, $\mathbf{C}$ is an unconstrained variable in problem (15), and as $\mathbf{C}$ runs over all values in $\mathcal{R}^3$, so does $(\mathtt{I} + [\delta\mathbf{r}]_\times)\mathbf{C}$. So the feasibility problem can be answered by solving

$$\begin{array}{ll} \textbf{Given} & \mathbf{X}_i, \bar{\mathtt{R}}, \epsilon_{\min} \\ \textbf{do there exist} & \mathbf{C}, \delta\mathbf{r} \in [-\sigma, \sigma]^3 \\ \textbf{such that} & \angle(\bar{\mathtt{R}}^\top \mathbf{v}_i, \mathbf{X}_i + \delta\mathbf{r} \times \mathbf{X}_i - \mathbf{C}) < \epsilon_{\min} + 3\sigma^2/2 \ ? \end{array} \tag{16}$$

The important point here is that the unknowns $\delta\mathbf{r}$ and $\mathbf{C}$ do not interact quadratically in this expression, and the problem is of the form (2), and so may be solved using SOCP.

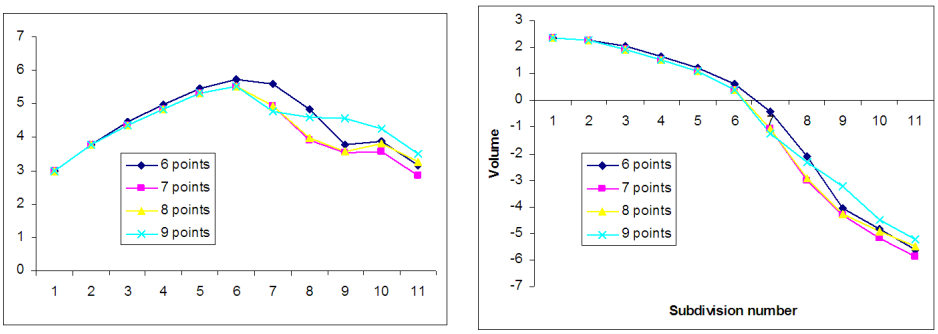

Figure 4: *For a typical problem, the number of remaining cubes (left) and the volume (right) are plotted (on a log-10 scale) against the number of subdivision phases. At the final subdivision, the cube half-side length is* 6.13×10^{-4} *radians, and the rotation is known to lie in a region with volume 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.)*

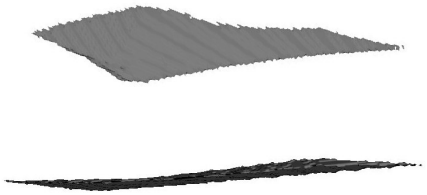

Figure 5: *For synthetic data with* $n = 20$ *points, the 3D shape of the remaining rotation region is shown after running the algorithm to resolution of* 0.001 *radians. Recall that the shape of the set of all rotations in the angle-axis representation is a closed* 3*-dimensional ball. The example in the figure is for a sideways motion of the camera with a field of view of about* $60°$*, and a motion equal to* 0.5 *times the distance to the closest point. 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.*

7 Verification and Testing

7.1 Relative Pose Estimation

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.

Initial rotation space is divided up into sufficiently small blocks. The exact number is not important; we use an $11 \times 11 \times 11$ subdivision. 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 3D shape of the rotation space is shown in a few examples (see Fig 5).

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

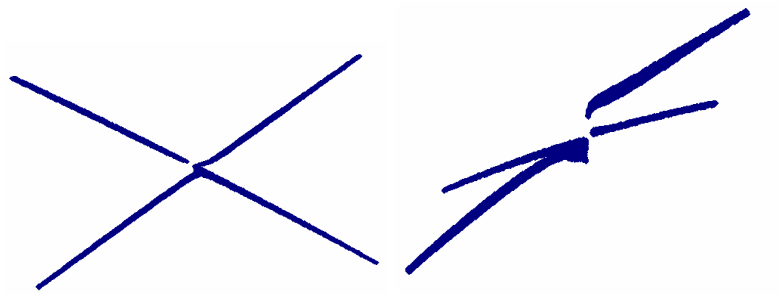

Figure 7: *Shape of the possible rotations for* 4 *point correspondences. The space of possible rotations forms a set of curves in* 3*-dimensional rotation space.*

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.

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. Fig 8 shows some typical timing information for various numbers of points.

Ordinary perspective cameras have a smaller field of view, and 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	29
time	6m 21s	16s	7s

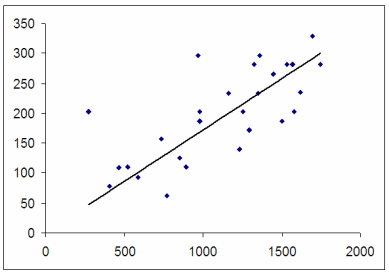

Figure 8: *Time in milliseconds plotted against number of points used to compute the relative pose, for* 360° *data from a Ladybug camera. Even very large problem sizes take less than 350 milliseconds.*

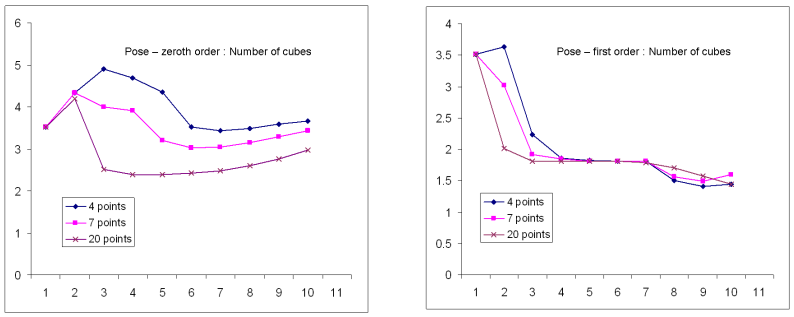

Figure 9: **Pose:** *The number of remaining cubes (on a log-*10 *scale) is plotted against the number of subdivision phases for camera pose computations on real data from the publicly available Oxford corridor sequence. In total, there are 11 images and several hundreds of pre-determined 2D-3D point correspondence in the data set. The average result of all* 11 *(independent) camera pose estimation problems in the sequence is given for varying number of randomly chosen 2D-3D 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 at each phase is less in the first order algorithm.*

7.2 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, but it is not evident how one should recast the SOCP feasibility problems using LP. At the initial subdivision of rotation space, the cube half-side length was set to $\pi/8$ radians.

In Fig 9 and Fig 10, camera pose computations are reported for both zero-th and first order bounds for the feasibility problems in (9) and (11), respectively. 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 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.

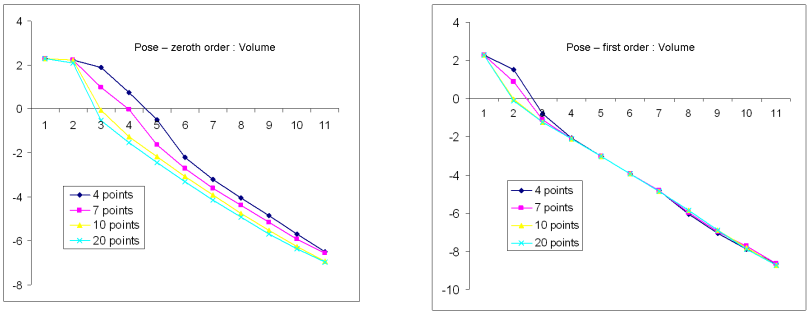

Figure 10: **Pose:** *The remaining volume (on a log-10 scale) of rotation space is plotted against the number of subdivision phases for camera pose computations on real data. The average of all* 11 *cameras in the corridor sequence is given for various numbers of randomly chosen point correspondences.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

8 Conclusions and Future Work

The algorithm for the relative pose problem works extremely well for 360° images, such as Ladybug^{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.

The method that we have proposed for solving the two-view motion problem (with known rotation) is optimal, and runs orders of magnitude faster than the existing optimal algorithms based on SOCP. It is impossible to use the old methods on very large problems, or ones that involve large numbers of invocations of the algorithm.

We have given two important improvements of the original branch-and-bound algorithm presented in sections 2-4 in order to speed up the computations. First, for the relative pose problem, an LP formulation of the feasibility problem was developed instead of the original SOCP formulation. Then, for the pose problem, a first order bound was derived which is tighter than the original zero-th order bound.

8.1 Ideas for Speedup

There is still room for improvements in terms of speed which would make the methods even more competitive compared to traditional local algorithms. These ideas are left for future research.

An LP formulation of the pose problem would most definitely result in faster execution times. Regarding the relative pose problem, we describe here other ideas of speeding up the calculations by accelerating the solution of the feasibility problems.

1. It was suggested in section 5.1 that the first (essentially an arbitrary) correspondence should be the one used to define the coordinate frame and determine a strip as in Fig 3 bounded by $\mathrm{x} = \pm\tan(\beta_0)$. A better idea would probably be to select the point pair for which the image correspondences were furthest separated, since in this case the angle β is smallest, resulting in a narrow strip and a more accurate feasibility test.

2. Since the feasibility test gives a way of identifying a feasibility problem with negative solution, it is not certain that we need to solve the LP problem at all. If the problem passes the feasibility

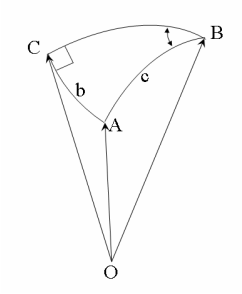

Figure 11: *The formula for the sine of an angle in a right-angled spherical triangle formed by arcs of great circles is* $\sin(B) = \sin(b)/\sin(c)$ *where* b *and* c *are the lengths of the arcs on the surface of the unit sphere.*

test, then we assume that the feasibility test has an affirmative answer, and subdivide the region D. This will mean that some regions are unnecessarily subdivided, but will not cause the algorithm to fail.

3. If we are willing to specify that one specific correspondence is exact, then we may do without the LP altogether. The fast feasibility test given in section 5.3 may be simplified, since instead of having a strip in which the epipole must lie, it is a straight line. This reduces the LP problem to a problem in one dimension, which may be solved trivially.

Appendix 1

We prove the formulas given in section 5.2.2. Parts 1 and 4 of the formulas given in section 5.2.2 are very simple. To prove the other results, we start with part 3, namely the formula for vector $\mathbf{w}$.

By symmetry, $\mathbf{w}$ is coplanar with $\mathbf{v}$ and $\mathbf{v}'$. We write $\mathbf{w} = a\mathbf{v} + b\mathbf{v}'$. Taking cross products with vectors $\mathbf{v}$ and $\mathbf{v}'$ and expressing the length of the resulting vector in two ways leads to

$$\begin{aligned}\sin(\gamma) &= \|\mathbf{w} \times \mathbf{v}\| = \|b\mathbf{v} \times \mathbf{v}'\| = b\sin(\alpha)\\ \sin(\gamma') &= \|\mathbf{w} \times \mathbf{v}'\| = \|a\mathbf{v} \times \mathbf{v}'\| = a\sin(\alpha)\end{aligned}$$

where γ and γ' are the angles separating $\mathbf{w}$ from $\mathbf{v}$ and $\mathbf{v}'$ respectively. From this we obtain

$$\mathbf{w} = \frac{\sin(\gamma')}{\sin(\alpha)}\mathbf{v} + \frac{\sin(\gamma)}{\sin(\alpha)}\mathbf{v}'. \tag{17}$$

We do not yet know the angles γ and γ'. At this point, we need an elementary result from spherical trigonometry (see Fig 11).

Lemma 8.5. *Let* ABC *be a spherical triangle in which* C *is a right-angle, and the edges be arcs of length* a*,* b *and* c *respectively, on a unit sphere. Then* $\sin B = \sin(b)/\sin(c)$.

This compares with the formula for a Euclidean triangle in which $\sin B = b/c$. We do not intend to prove this lemma.

Now, applying this to the triangles shown in Fig 12 we see that

$$\sin(\beta) = \frac{\sin(\epsilon)}{\sin(\gamma)} = \frac{\sin(\epsilon')}{\sin(\gamma')} \ .$$

Substituting for $\sin(\gamma)$ and $\sin(\gamma')$ in (17) gives the required formula (14) for $\mathbf{w}$.

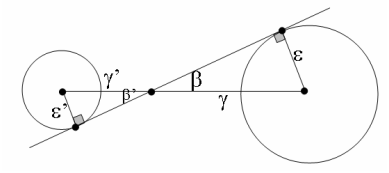

Figure 12: *Computing the angle between the plane bi-tangent to two cones and the plane containing the axes of the two cones. See the text for the computation.*

Next we wish to prove the formula (13) for β. This is simply a result of the fact that $\mathbf{w}$ is a unit vector. Computing the norm of $\mathbf{w}$ given by (14) yields

$$\|\mathbf{w}\|^2 = \frac{\sin^2(\epsilon) + 2\sin(\epsilon)\sin(\epsilon')\cos(\alpha) + \sin^2(\epsilon')}{\sin^2(\alpha)\sin^2(\beta)}$$

from which the result follows.

The final item in section 5.2.2, namely $\mathbf{n} = \sin(\beta)\mathbf{z} \pm \cos(\beta)\mathbf{x}$ is simply a statement that the angle between the tangent plane and the z-axis is β.

Appendix 2

We now prove lemma 3.2.

Proof. 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 $\delta\mathbf{r} = \delta\theta\hat{\mathbf{r}}$, with $\hat{\mathbf{r}}$ a unit vector be the vector representation of the rotation $\delta\mathtt{R}$. 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

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

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,

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

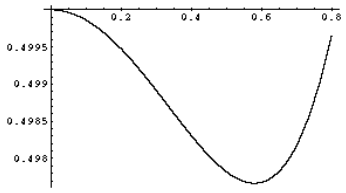

Figure 13: *Plot of the function* $\arcsin(\sqrt{(\sin x - x)^2 + (1-\cos x)^2})/x^2$*, verifying the last step of the proof of lemma 6.4.*

for $0 < \delta\theta < 0.76$. This is easily verified graphically (see Fig 13). A rigorous proof follows. Since both $\arcsin$ and square root are monotonic functions on the interval $[0, 1]$, it is equivalent to prove

$$(\sin x - x)^2 + (1 - \cos x)^2 \leq \sin^2(x^2/2) = (1 - \cos(x^2))/2$$

for x in the interval. Expanding both $\sin x - x$ and $1 - \cos x$ in a Taylor series we obtain bounds

$$\begin{aligned}(\sin x - x)^2 + (1 - \cos x)^2 &< (x^3/3!)^2 + (x^2/2 - x^4/4! + x^6/6!)^2 \\ &< x^4/4 - x^6/72 + x^8/320\end{aligned}$$

when $x > 0$. Similarly, from the Taylor series for $\cos(x^2)$ we obtain

$$x^4/4 - x^8/48 < (1 - \cos(x^2))/2 \ .$$

The lemma is completed by showing that $x^4/4 - x^6/72 + x^8/320 \leq x^4/4 - x^8/48$, which is seen to be true when $0 \leq x < \sqrt{40/69} = 0.76$.

References

[1] T.M. Breuel. Implementation techniques for geometric branch-and-bound matching methods. *Computer Vision and Image Understanding*, 90(3):258–294, 2003.

[2] R. Hartley and F. Kahl. Global optimization through searching rotation space and optimal estimation of the essential matrix. In *Proc. International Conference on Computer Vision*, October 2007.

[3] R. Hartley and F. Kahl. Optimal algorithms in multiview geometry. In *Proc. Asian Conference on Computer Vision*, volume 1, pages 13 – 34, November 2007.

[4] 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.

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision – 2nd Edition*. Cambridge University Press, 2004.

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

[7] F. Kahl, S. Agarwal, M. K. Chandraker, D. J. Kriegman, and S. Belongie. Practical global optimization for multiview geometry. *Int. Journal Computer Vision*, 79(3):271–284, 2008.

[8] F. Kahl and R. Hartley. Multiple view geometry under the L_∞-norm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(9):1603–1617, 2008.

[9] F. Kahl and D. Henrion. Globally optimal estimates for geometric reconstruction problems. *Int. Journal Computer Vision*, 74(1):3–15, 2007.

[10] Q. Ke and T. Kanade. Quasiconvex optimization for robust geometric reconstruction. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(10):1834–1847, 2007.

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

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

[13] C. Olsson, F. Kahl, and M. Oskarsson. Branch and bound methods for Euclidean registration problems. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2008. In Press.

[14] S. Soatto and R. Brockett. Optimal structure from motion: Local ambiguities and global estimates. In *Conf. Computer Vision and Pattern Recognition*, Santa Barbara, USA, 1998.

[15] J.F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12:625–653, 1999.

[16] R. Szeliski and S. B. Kang. Shape ambiguities in structure from motion. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(5), May 1997.

[17] Wikipedia. www.wikipedia.org.