# A Fast Resection-Intersection Method for the Known Rotation Problem

Qianggong Zhang q.zhang.au@gmail.com
Tat-Jun Chin tat-jun.chin@adelaide.edu.au
Huu Minh Le huu.le@adelaide.edu.au
The University of Adelaide, Adelaide, SA, Australia

## Abstract

*The known rotation problem refers to a special case of structure-from-motion where the absolute orientations of the cameras are known. When formulated as a minimax ($\ell_\infty$) problem on reprojection errors, the problem is an instance of pseudo-convex programming. Though theoretically tractable, solving the known rotation problem on large-scale data (1,000's of views, 10,000's scene points) using existing methods can be very time-consuming. In this paper, we devise a fast algorithm for the known rotation problem. Our approach alternates between pose estimation and triangulation (i.e., resection-intersection) to break the problem into multiple simpler instances of pseudo-convex programming. The key to the vastly superior performance of our method lies in using a novel minimum enclosing ball (MEB) technique for the calculation of updating steps, which obviates the need for convex optimisation routines and greatly reduces memory footprint. We demonstrate the practicality of our method on large-scale problem instances which easily overwhelm current state-of-the-art algorithms*[1].

## 1. Introduction

Given a number of scene points that were viewed in a number of images, the goal of structure-from-motion (SfM) is to estimate the 3D coordinates of the scene points based on their measured 2D coordinates in the images. The implicit geometric constraints underpinning the imaging scenario also requires the pose of the cameras (each defined by a rotation and translation) that captured the images to be recovered jointly with the 3D scene points.

Many current SfM pipelines employ bundle adjustment (BA) [35] as a core routine. BA refers to the task of jointly refining the scene points and camera poses, and it is usually formulated as a non-linear least squares problem. Most BA implementations are based on the Levenberg-Marquardt algorithm, which enables convergence up to local optimality. In practice, it is vital for the target variables to be initialised well to avoid convergence to bad local optima.

An alternative SfM pipeline [14, 31, 36, 18, 23, 16, 13] that has begun to receive attention is as follows: first, estimate rotations by a rotation averaging method, then, keeping the rotations fixed, estimate the scene points and translations; the latter problem is called the *known rotation problem (KRot)*. The strength of this approach is twofold: first, multiple-rotation averaging can often be solved much more easily (in certain cases, up to global optimality [27, 29, 12]); second, when formulated as a minimax ($\ell_\infty$) problem, KRot becomes an instance of pseudo-convex programming, which is also amenable to exact global solutions [20].

In this paper, we focus on KRot. Although the problem is tractable, as we will demonstrate later, most existing algorithms [19, 21, 30, 28, 4] are not practical on large-scale inputs involving 1,000's of views and 10,000's scene points, in contrast to modern BA packages, e.g., [33, 22], which have been applied successfully on such sizes. The inefficiency of the previous KRot algorithms stems from their dependence on convex optimisation to drive the iterative updates (see Sec. 2.2 for details). Although convex solvers are theoretically efficient, in practice, they incur much computational and memory overheads. Arguably this has also hindered the usability of the alternative SfM pipeline.

**Contributions** We propose a fast algorithm for KRot. The overall structure of our method interleaves the calculation of scene points and translations [25]—akin to the resection-intersection approach for BA [35].

The primary feature of our algorithm that enables its superior performance over previous techniques lies in a novel method for solving each pseudo-convex sub-problem. Instead of relying on convex routines to compute the update, our method calculates descent directions in closed-form based on a novel minimum enclosing ball (MEB) technique. This leads to a fast and self-contained algorithm (does not require external convex solvers). The resection-intersection structure also makes it inherently parallelisable. As we will show in Sec. 4, our algorithm can scale up to input sizes that are beyond the reach of existing methods.

[1] See the supplementary material for demo program.

## 2. Known rotation problem

We assume calibrated cameras. Let $M$ be the number of scene points and $L$ be the number of cameras. Let $\mathbf{s}_k$ be the 3D coordinates of the $k$-th scene point, and $\mathbf{u}_{j,k}$ be the 2D observation of $\mathbf{s}_k$ in the $j$-th image. The pose of the $j$-th camera is defined by the rotation and translation $(\mathbf{R}_j, \mathbf{t}_j)$. Given the 2D observations $\{\mathbf{u}_{j,k}\}$ and rotations $\{\mathbf{R}_j\}$, under the minimax formulation [20], we solve

$$\begin{aligned} \min_{\{\mathbf{t}_j\},\{\mathbf{s}_k\}} \quad & \max_{j,k} \left\| \mathbf{u}_{j,k} - \frac{\mathbf{R}_j^{1:2}\mathbf{s}_k + \mathbf{t}_j^{1:2}}{\mathbf{R}_j^{3}\mathbf{s}_k + \mathbf{t}_j^{3}} \right\|_p \\ \text{s.t.} \quad & \mathbf{R}_j^3\mathbf{s}_k + \mathbf{t}_j^3 > 0 \;\; \forall j,k, \end{aligned} \tag{KRot}$$

to estimate the scene points $\{\mathbf{s}_k\}$ and the camera positions $\{\mathbf{t}_j\}$. Here, $\mathbf{R}_j^{1:2}$ and $\mathbf{R}_j^3$ are respectively the first two rows and the third row of $\mathbf{R}_j$ (similarly for $\mathbf{t}_j^{1:2}$ and $\mathbf{t}_j^3$). Intuitively, KRot aims to minimise the maximum reprojection error over all 2D observations, and the constraints of the problem ensure that the estimated $\{\mathbf{s}_k\}$ lie in front of all the cameras. As established in [28], KRot is pseudo-convex.

**Missing data and outliers** Not every scene point is visible to all images; simply drop $(j,k)$ pairs that are irrelevant can handle missing data. Also, like most core SfM routines (including standard BA [35], KRot is not robust to outliers. This does not reduce the value of KRot techniques, since removing outliers is usually and effectively done earlier in the pipeline, e.g., use RANSAC to estimate relative poses.

**Choice of norm** We leave the choice of the $p$-norm $\|\cdot\|_p$ in (KRot) free since KRot is pseudo-convex for any $p \geq 1$, and our algorithm works for any $p \geq 1$. In practice, typical choices of the $p$-norm include $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$ [4, 8].

### 2.1. Resection-intersection

Instead of solving KRot directly, one can alternate between solving for $\{\mathbf{t}_j\}$ and $\{\mathbf{s}_k\}$. In fact, if $\{\mathbf{t}_j\}$ are fixed, the scene points can be optimised independently, viz.:

$$\begin{aligned} \min_{\mathbf{s}_k} \quad & \max_{j} \left\| \mathbf{u}_{j,k} - \frac{\mathbf{R}_j^{1:2}\mathbf{s}_k + \mathbf{t}_j^{1:2}}{\mathbf{R}_j^{3}\mathbf{s}_k + \mathbf{t}_j^{3}} \right\|_p \\ \text{s.t.} \quad & \mathbf{R}_j^3\mathbf{s}_k + \mathbf{t}_j^3 > 0 \;\; \forall j. \end{aligned} \tag{Int$_k$}$$

Problem (Int$_k$) is simply $\ell_\infty$ triangulation [17], i.e., intersecting back-projected image points. Conversely, if $\{\mathbf{s}_k\}$ are fixed, the translations can also be optimised separately, viz.:

$$\begin{aligned} \min_{\mathbf{t}_j} \quad & \max_{k} \left\| \mathbf{u}_{j,k} - \frac{\mathbf{R}_j^{1:2}\mathbf{s}_k + \mathbf{t}_j^{1:2}}{\mathbf{R}_j^{3}\mathbf{s}_k + \mathbf{t}_j^{3}} \right\|_p \\ \text{s.t.} \quad & \mathbf{R}_j^3\mathbf{s}_k + \mathbf{t}_j^3 > 0 \;\; \forall k. \end{aligned} \tag{Res$_j$}$$

**Algorithm 1** Resection-intersection method for KRot.

**Require:** Input data $\{\mathbf{R}_j\}_{j=1}^{L}$, $\{\mathbf{u}_{j,k}\}_{j=1,k=1}^{L,\;M}$.
1: Initialise $\{\mathbf{t}_j\}_{j=1}^{L}$ and $\{\mathbf{s}_k\}_{k=1}^{M}$.
2: **repeat**
3: For each $k = 1, \ldots, M$, update $\mathbf{s}_k$ via (Int$_k$).
4: For each $j = 1, \ldots, L$, update $\mathbf{t}_j$ via (Res$_j$).
5: **until** convergence
6: **return** $\{\mathbf{t}_j\}_{j=1}^{L}$ and $\{\mathbf{s}_k\}_{k=1}^{M}$.

Problem (Res$_j$) is a special case of the $\ell_\infty$ camera resectioning problem [20]. The above properties motivate the resection-intersection method summarised in Algorithm 1.

By performing what is effectively block-wise coordinate descent, Algorithm 1 ensures convergence to the global optimum of KRot. While resection-intersection is eschewed for BA due to its slower convergence [35], it is effective for KRot since each sub-problem is pseudo-convex [28]. By solving (Int$_k$) and (Res$_j$) exactly, the best "step size" is used in each descent, which leads to fast overall convergence.

Another advantage of Algorithm 1 is that the sub-problems within either (Int$_k$) or (Res$_j$) are mutually independent given that either structure or camera positions are fixed. Hence, parallel computation can easily be leveraged for speed-ups; as we will demonstrate later.

### 2.2. Previous works

Many previous algorithms for KRot attempt to solve the overall problem directly (e.g., [20, 28, 4, 6, 11]). This requires to simultaneously update all the $3(L+M)$ variables in each iteration, which is cumbersome for large-scale problems. The resection-intersection approach, first introduced in [25], allows to partition KRot into small sub-problems without affecting global optimality guarantees.

However, a more fundamental weakness of many previous methods [20, 21, 30, 28, 4] is that they need to execute convex optimisation in each step, e.g., linear programming (LP) or second-order cone programming (SOCP). Though theoretically efficient, there are significant overheads in calling these routines. Even though the methods can be repurposed to solve KRot via resection-intersection, for large $L$ and $M$ the (sub-)overheads quickly add up.

There exist approaches that do not depend on convex optimisation. Based on a primal-dual interior-point framework, Dai et al. [6] perform a Newton-like descent and step size-search in each iteration. Although they outperformed previous methods, the need to calculate Hessians for all the measurements is a significant per-iteration cost. In contrast, our method only requires the computation of MEB on at most four 3D points per iteration (see Sec. 3.1.2).

A proximal splitting approach for KRot was proposed by Eriksson and Isaksson [11]. In a nutshell, their method

performs a one-step bundle adjustment (i.e., non-linear least squares) followed by 1D bisection to evaluate the proximal operator. The speed of convergence depends on the rate of increase of a penalty parameter, which needs to be controlled properly to avoid divergence. In practice, we found that often a conservative rate is required for correct results.

Donne et al. [8] proposed a so-called *polyhedron collapse* method for $\ell_\infty$ triangulation ($\text{Int}_k$). Specialising for the case of $p = \infty$, they leverage the linearity of the constraints to calculate descent directions in closed-form. Our proposed algorithms for the sub-problems can compute descent directions in closed-form without restricting $p$.

## 3. Fast descent method

One major contribution is a fast algorithm for the sub-problems in Algorithm 1. With simple manipulations, both ($\text{Int}_k$) and ($\text{Res}_j$) can be expressed in the common form

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^3} \quad & \max_i \; r_i(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{c}_i^T\mathbf{x} + d_i > 0 \;\; \forall i, \end{aligned} \tag{1}$$

where $\mathbf{x}$ are the variables of interest (3D coordinates or camera position—both 3D quantities in each sub-problem);

$$r_i(\mathbf{x}) = \frac{\|\mathbf{A}_i\mathbf{x} + \mathbf{b}_i\|_p}{\mathbf{c}_i^T\mathbf{x} + d_i} \tag{2}$$

is the $i$-th pseudo-convex residual function, and

$$\mathbf{A}_i = \begin{bmatrix} \mathbf{a}_{i,1}^T \\ \mathbf{a}_{i,2}^T \end{bmatrix} \in \mathbb{R}^{2\times 3}, \quad \mathbf{b}_i = \begin{bmatrix} b_{i,1} \\ b_{i,2} \end{bmatrix} \in \mathbb{R}^2, \tag{3}$$

$\mathbf{c}_i \in \mathbb{R}^3$ and $d_i \in \mathbb{R}$ are constants derived from the data (see the supp. material for details of converting ($\text{Int}_k$) and ($\text{Res}_j$) into (1)). Since (2) is pseudo-convex [1], their point-wise maximum is pseudo-convex. Hence, (1) can be solved globally using iterative minimisation techniques.

Our algorithm, called *fast descent method (FDM)*, is summarised in Algorithm 2. The structure is simple—given an initial feasible estimate $\hat{\mathbf{x}}$, find a direction $\boldsymbol{\lambda}$ and step size $\alpha$ to adjust $\hat{\mathbf{x}}$ such that the cost (point-wise maximum residual) decreases; stop when a valid $\boldsymbol{\lambda}$ cannot be found. As mentioned in Sec. 1, the key feature of FDM that enables its superior performance lies in a closed-form method to compute $\boldsymbol{\lambda}$. Details of FDM are in the rest of this section.

### 3.1. Efficient computation of descent direction

Algorithm 3 describes our routine for finding a descent direction for a current estimate $\hat{\mathbf{x}}$. Let $\hat{r}$ be the value of the objective function at $\hat{\mathbf{x}}$, i.e.,

$$\hat{r} = \max_i \; r_i(\hat{\mathbf{x}}). \tag{4}$$

**Algorithm 2** Fast Descent Method (FDM) for (1).

**Require:** Input data $\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i, d_i\}_{i=1}^N$, initial soln. $\hat{\mathbf{x}}$.
1: $\boldsymbol{\lambda} \leftarrow$ Find descent direction using data and $\hat{\mathbf{x}}$ (Alg. 3).
2: **while** $\boldsymbol{\lambda}$ is not null **do**
3: $\quad\alpha \leftarrow$ Find step size using data, $\hat{\mathbf{x}}$ and $\boldsymbol{\lambda}$ (Alg. 4).
4: $\quad\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \alpha\boldsymbol{\lambda}$.
5: $\quad\boldsymbol{\lambda} \leftarrow$ Find descent direction using data and $\hat{\mathbf{x}}$.
6: **end while**
7: **return** $\hat{\mathbf{x}}$.

**Algorithm 3** Find descent direction.

**Require:** Input data $\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i, d_i\}_{i=1}^N$, current estimate $\hat{\mathbf{x}}$, threshold $\epsilon_0$.
1: $\mathcal{G} \leftarrow$ Norm. negative active gradient vectors at $\hat{\mathbf{x}}$ (6).
2: $\mathbf{m}^* \leftarrow$ Centre of MEB of $\mathcal{G}$.
3: **if** $\|\mathbf{m}^*\|_2 \leq \epsilon_0$ **then**
4: $\quad\boldsymbol{\lambda} \leftarrow$ null.
5: **else**
6: $\quad\boldsymbol{\lambda} \leftarrow \mathbf{m}^* / \|\mathbf{m}^*\|_2$.
7: **end if**
8: **return** $\boldsymbol{\lambda}$.

The routine begins by finding the set of *active residuals* $\mathcal{A}$, i.e., the set of residuals that have the value $\hat{r}$ at $\hat{\mathbf{x}}$, i.e.,

$$\mathcal{A} \leftarrow \{\ell \in \{1, \ldots, N\} \mid r_\ell(\hat{\mathbf{x}}) = \hat{r}\}. \tag{5}$$

Then, we compute the normalised negative gradient vectors $\mathcal{G}$ corresponding to the active residuals

$$\mathcal{G} = \left\{ \mathbf{g} \in \mathbb{R}^3 \;\middle|\; \mathbf{g} = -\frac{\nabla r_\ell(\hat{\mathbf{x}})}{\|\nabla r_\ell(\hat{\mathbf{x}})\|_2}, \forall \ell \in \mathcal{A} \right\} \tag{6}$$

(see the supp. material on deriving the gradient $\nabla r_i(\mathbf{x})$ for all $p \geq 1$).

A descent direction $\boldsymbol{\lambda} \in \mathbb{R}^3$ is computed as the *centre* of the *minimum enclosing ball (MEB)* of $\mathcal{G}$. Specifically, the centre of the MEB of $\mathcal{G}$ is the point $\mathbf{m}^* \in \mathbb{R}^3$ where

$$\mathbf{m}^* = \operatorname*{argmin}_{\mathbf{m}\in\mathbb{R}^3} \max_{\mathbf{g}\in\mathcal{G}} \|\mathbf{g} - \mathbf{m}\|_2. \tag{7}$$

In the following, we prove the validity of this approach before proposing an algorithm to calculate MEB.

#### 3.1.1 Validity of descent direction

First, we define some notations: a bolded lower case letter may refer to a vector or a point, depending on the context. For any $\mathbf{x}$ and $\mathbf{y}$, $|\mathbf{xy}|$ is the distance between the pair, i.e.,

$$|\mathbf{xy}| := \|\mathbf{x} - \mathbf{y}\|_2 . \tag{8}$$

$\triangle\mathbf{xyz}$ is the triangle formed by three points $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$, and $\angle\mathbf{xyz}$ is the angle between vectors $(\mathbf{x}-\mathbf{y})$ and $(\mathbf{z}-\mathbf{y})$, i.e.,

$$\angle\mathbf{xyz} := \arccos\left(\frac{(\mathbf{x}-\mathbf{y})^T(\mathbf{z}-\mathbf{y})}{\|\mathbf{x}-\mathbf{y}\|\cdot\|\mathbf{z}-\mathbf{y}\|}\right). \tag{9}$$

The following are basic results from optimisation. Interested readers can refer to [26] and the supp. material.

**Lemma 1.** *A direction $\boldsymbol{\lambda}$ is a descent direction of a function $f(\mathbf{x})$ at $\hat{\mathbf{x}}$ iff $\langle\boldsymbol{\lambda}, -\nabla f(\hat{\mathbf{x}})\rangle = -\nabla f(\hat{\mathbf{x}})^T\boldsymbol{\lambda} > 0$.*

**Lemma 2.** *Given a descent direction $\boldsymbol{\lambda}$ for a function $f(\mathbf{x})$ at $\hat{\mathbf{x}}$, the larger $\langle\boldsymbol{\lambda}, -\nabla f(\hat{\mathbf{x}})\rangle$ is, the faster $f(\mathbf{x})$ is reduced locally along the direction $\hat{\mathbf{x}} + \alpha\boldsymbol{\lambda}$ for $\alpha > 0$.*

Our main theorem is as follows.

**Theorem 1.** *If the centre $\mathbf{m}^*$ of the MEB of $\mathcal{G}$ is not equal to the origin, then $\mathbf{m}^*$ is a descent direction for (1) at $\hat{\mathbf{x}}$.*

*Proof.* Let $\mathbf{o}$ be the origin. If $\mathbf{m}^* \neq \mathbf{o}$, then by the definition of MEB (7), for all $\mathbf{g} \in \mathcal{G}$

$$|\mathbf{m}^*\mathbf{g}| < |\mathbf{og}|, \tag{10}$$

which, by the Law of Sines [2], implies

$$\angle\mathbf{m}^*\mathbf{og} < \angle\mathbf{om}^*\mathbf{g} \tag{11}$$

Since the inner angles of a triangle sum to $180°$,

$$\angle\mathbf{m}^*\mathbf{og} + \angle\mathbf{om}^*\mathbf{g} < 180°. \tag{12}$$

Combining (11) and (12), $\angle\mathbf{m}^*\mathbf{og} < 90°$ is established, and thus

$$\langle\mathbf{m}^*, \mathbf{g}\rangle = |\mathbf{og}||\mathbf{om}^*|\cos(\angle\mathbf{m}^*\mathbf{og}) > 0. \tag{13}$$

By Lemma 1, $\mathbf{m}^*$ is a descent direction for all active residuals at $\hat{\mathbf{x}}$, and thus $\mathbf{m}^*$ is also a descent direction for (1) at $\hat{\mathbf{x}}$ since it reduces the value of the maximum residuals. □

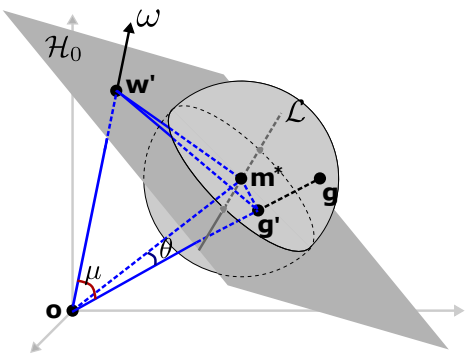


Figure 1. Diagram to support the proof of Theorem 2.

How good a descent direction is the centre of the MEB of $\mathcal{G}$? The following theorem shows that it is the optimal descent direction, in the sense of Lemma 2.

**Theorem 2.** *The centre $\mathbf{m}^*$ of the MEB of $\mathcal{G}$ is the descent direction that maximises the rate of decrease of (1) at $\hat{\mathbf{x}}$.*

*Proof.* Following Lemma 2, we aim to show that

$$\mathbf{m}^* = \underset{\mathbf{m}\in\mathbb{R}^3}{\operatorname{argmin}} \max_{\mathbf{g}\in\mathcal{G}} \angle\mathbf{gom}. \tag{14}$$

We first construct the following geometric objects to lay the foundation of the proof (refer to Fig. 1 for intuition):

1. Let $\mathcal{H}_0$ be the plane passing through $\mathbf{m}^*$ and orthogonal to the line segment $\overline{\mathbf{om}^*}$:
$$\overline{\mathbf{om}^*} \perp \mathcal{H}_0 \tag{15}$$
2. Let $\boldsymbol{\omega}$ an arbitrary direction from $\mathbf{o}$ intersecting $\mathcal{H}_0$ at $\mathbf{w}'$.
3. Let $\mathcal{H}_1$ be the plane passing through $\mathbf{m}^*$ and orthogonal to $\overline{\mathbf{m}^*\mathbf{w}'}$, and $\mathcal{H}_1$ intersects $\mathcal{H}_0$ at line $\mathcal{L}$; therefore
$$\mathcal{L} \perp \overline{\mathbf{m}^*\mathbf{w}'}. \tag{16}$$
4. By [5], there must exist a point $\mathbf{g}$ on the boundary of the MEB of $\mathcal{G}$ that satisfies
$$|\mathbf{w}'\mathbf{g}| \geq \sqrt{|\mathbf{m}^*\mathbf{w}'|^2 + |\mathbf{m}^*\mathbf{g}|^2}; \tag{17}$$
and let $\mathbf{g}'$ be the intersection of line $\overline{\mathbf{og}}$ and $\mathcal{H}_0$.
5. Define the angles
$$\mu = \angle\mathbf{w}'\mathbf{og}' \quad \text{and} \quad \theta = \angle\mathbf{m}^*\mathbf{og}'. \tag{18}$$

The geometric interpretation of (17) is that $\mathbf{g}$ and $\mathbf{w}'$ are in different half-spheres of the MEB of $\mathcal{G}$ divided by $\mathcal{H}_1$, which yields, that on the plane $\mathcal{H}_0$, $\mathbf{g}'$ and $\mathbf{w}'$ are on the different side of $\mathcal{L}$. Therefore, given (16), we get

$$\angle\mathbf{g}'\mathbf{m}^*\mathbf{w}' \geq 90°, \tag{19}$$

and the Law of Cosines [3] translates (19) into

$$|\mathbf{w}'\mathbf{g}'|^2 \geq |\mathbf{m}^*\mathbf{g}'|^2 + |\mathbf{m}^*\mathbf{w}'|^2. \tag{20}$$

(15) yields

$$\angle\mathbf{om}^*\mathbf{g}' = \angle\mathbf{om}^*\mathbf{w}' = 90°, \tag{21}$$

which, by Pythagorean theorem, implies

$$\begin{aligned} |\mathbf{og}'|^2 &= |\mathbf{om}^*|^2 + |\mathbf{m}^*\mathbf{g}'|^2 \\ \text{and} \quad |\mathbf{ow}'|^2 &= |\mathbf{om}^*|^2 + |\mathbf{m}^*\mathbf{w}'|^2. \end{aligned} \tag{22}$$

In $\triangle\mathbf{w}'\mathbf{og}'$, as in the Law of Cosines,

$$\cos(\mu) = \frac{|\mathbf{ow}'|^2 + |\mathbf{og}'|^2 - |\mathbf{w}'\mathbf{g}'|^2}{2|\mathbf{ow}'||\mathbf{og}'|}. \tag{23}$$

Substituting (22) into (23) yields

$$\cos(\mu) = \frac{|\mathbf{om}^*|^2 + |\mathbf{m}^*\mathbf{g}'|^2 + |\mathbf{om}^*|^2 + |\mathbf{m}^*\mathbf{w}'|^2 - |\mathbf{w}'\mathbf{g}'|^2}{2|\mathbf{ow}'||\mathbf{og}'|}. \tag{24}$$

Substituting (20) into (24) yields

$$\cos(\mu) \leq \frac{2|\mathbf{om}^*|^2}{2|\mathbf{ow}'||\mathbf{og}'|} = \frac{|\mathbf{om}^*|}{|\mathbf{ow}'|}\frac{|\mathbf{om}^*|}{|\mathbf{og}'|}. \tag{25}$$

By (21), $|\mathbf{om}^*|/|\mathbf{ow}'| < 1$, thus (25) yields

$$\cos(\mu) < \frac{|\mathbf{om}^*|}{|\mathbf{og}'|} = \cos(\theta) \quad \Longrightarrow \quad \mu > \theta, \tag{26}$$

therefore, we conclude that for any direction $\boldsymbol{\omega}$ other than $\mathbf{m}^*$, there always exists a point $\mathbf{g} \in \mathcal{G}$ satisfying

$$\angle \mathbf{go}\boldsymbol{\omega} > \angle \mathbf{gom}^*, \tag{27}$$

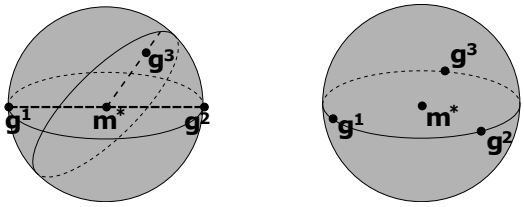
thus (14) is validated. □

#### 3.1.2 Closed-form MEB algorithm

The effort to calculate the MEB of $\mathcal{G}$ naturally depends on the size of (number of vectors in) $\mathcal{G}$. Fortunately, the combinatorial dimension of a problem with the form (1) has been established to be four [10, 32]. Thus the size of the active set $\mathcal{A}$, and hence the size of set $\mathcal{G}$, is *at most* four. This motivates a closed-form algorithm to calculate the MEB.

Let $\mathbf{m}^*$ and $f^*$ respectively be the centre and radius of the MEB of $\mathcal{G}$. Due to the small upper bound on the size of $\mathcal{G}$, the possible solutions for $\mathbf{m}^*$ and $f^*$ can be enumerated as follows:

- **Case 1**: $\mathcal{G} = \{\mathbf{g}_1\}$ is of size 1.
  Trivially, the centre $\mathbf{m}^* = \mathbf{g}_1$ and $f^* = 0$.
- **Case 2**: $\mathcal{G} = \{\mathbf{g}_1, \mathbf{g}_2\}$ is of size 2.
  The centre of the MEB must lie in the middle of the line segment $\overline{\mathbf{g}_1\mathbf{g}_2}$ (also a diameter of the MEB), i.e., $\mathbf{m}^* = (\mathbf{g}_1 - \mathbf{g}_2)/2$, and $f^*$ is simply $\|\mathbf{m}^* - \mathbf{g}_1\|_2$.
- **Case 3**: $\mathcal{G} = \{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3\}$ is of size 3.
  We need to check the following two possibilities to find the correct MEB (see Fig. 2 for an illustration):
  - **Case 3.1**: The MEB is formed by two of the three points (as in **Case 2** above) and the third point lies within the MEB. We need to solve **Case 2** on the three possible pairings of the points and check.
  - **Case 3.2**: The MEB is the ball that has $\mathbf{g}_1$, $\mathbf{g}_2$ and $\mathbf{g}_3$ on its surface (also on its great circle). The centre $\mathbf{m}^*$ and radius $f^*$ of the great circle (also of the ball) can be computed analytically [38, Chapter V].
- **Case 4**: $\mathcal{G} = \{\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_4\}$ is of size 4.
  Similar to **Case 3**, we need to check the following two possibilities to find the correct MEB:




(a) (b)

Figure 2. (a) **Case 3.1** when two points determine the MEB and the third point lies within the MEB. (b) **Case 3.2** when three points lie on the surface (also on a great circle) of the MEB.

  - **Case 4.1**: The MEB is formed by three of the four points (as in **Case 3** above) and the fourth point lies within the MEB. We need to solve **Case 3** on the four possible selections of triplets of the points and check.
  - **Case 4.2**: The MEB is the ball that contains $\mathbf{g}_1$, $\mathbf{g}_2$, $\mathbf{g}_3$ and $\mathbf{g}_4$ on its surface. However, recall that the items in $\mathcal{G}$ are unit vectors, hence the MEB on $\mathcal{G}$ must be centred at the origin ($\mathbf{m}^* = \mathbf{o}$) with radius $f^* = 1$.

**Degeneracies** In the degenerate cases where the size of $\mathcal{G}$ is greater than four, any existing MEB solver [37] can be employed to solve the degeneracy. Empirically, degeneracy did not affect Algorithm 3 since noisy data is rarely degenerate [24].

### 3.2. Optimising the step size

Once a descent direction $\boldsymbol{\lambda}$ is computed in Algorithm 2, we need to search for a step size $\alpha$ along $\boldsymbol{\lambda}$ to update $\hat{\mathbf{x}}$. The residual function parametrised by $\alpha$ is

$$\begin{aligned} r_i(\alpha) &= \frac{\|\mathbf{A}_i(\hat{\mathbf{x}} + \alpha\boldsymbol{\lambda}) + \mathbf{b}_i\|_p}{\mathbf{c}_i^T(\hat{\mathbf{x}} + \alpha\boldsymbol{\lambda}) + d_i} \\ &:= \frac{\|\mathbf{u}_i\alpha + \mathbf{v}_i\|_p}{w_i\alpha + z_i}, \end{aligned} \tag{28}$$

which remains pseudo-convex for $\alpha$ if $w_i\alpha + z_i > 0$, hencefore the search for the step size that provides the biggest reduction in the objective value can be formulated as the following one-dimensional pseudo-convex problem

$$\begin{aligned} \min_{\alpha \in \mathbb{R}_+} \quad & \max_i \ r_i(\alpha) \\ \text{s.t.} \quad & w_i\alpha + z_i > 0. \end{aligned} \tag{29}$$

Any of the previous methods for pseudo-convex programming (e.g., [28, 4, 20]) can be repurposed for (29), however, to avoid cumbersome convex sub-problems, we exploit the fact that (29) is a single variable problem and develop a modified bisection method searching over $\alpha$ to solve (29).

Algorithm 4 describes our approach; see also Fig. 3 for an illustration. The lower bound for $\alpha$ is initialised to 0, while the initial upper bound is obtained as the supremum

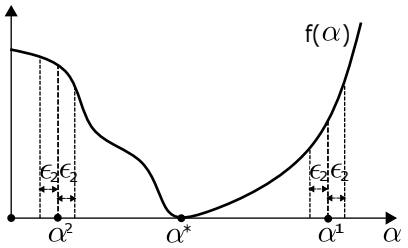



Figure 3. Illustration of Algorithm 4. If the current setting of $[lb, ub]$ yields $\alpha^1 = (lb + ub)/2$, then by $f(\alpha^1 - \epsilon_2) < f(\alpha^1) < f(\alpha^1 + \epsilon_2)$, $\alpha^1$ is known to be on the monotonously increasing part of $f(\alpha)$, thus the optimal $\alpha^*$ is smaller than $\alpha^1$ and $ub$ should be reduced; similarly, $lb$ need to be increased at $\alpha^2 = (lb+ub)/2$.

of $\alpha$ such that $w_i\alpha + z_i > 0$ is true for all $i$ (this can be done in linear time by incrementally intersecting the half-lines $w_i\alpha + z_i > 0$). The algorithm leverages the pseudo-convexity property of (29), namely, the objective function is strictly either increasing or decreasing on each side of the global minimum, to progressively refine the bound $[lb, ub]$ on $\alpha$. Two convergence thresholds $\epsilon_1$ and $\epsilon_2$ (set to respectively $10^{-6}$ and $10^{-8}$ in our experiments) that enable a termination up to any desired precision threshold[2].

**Algorithm 4** Modified bisection to find step size (29).

**Require:** Input data $\{\mathbf{u}_i, \mathbf{v}_i, w_i, z_i\}_{i=1}^N$, convergence thresholds $\epsilon_1$ and $\epsilon_2$.

$[lb, ub] \leftarrow$ lower and upper bound of $\alpha$ (see Sec. 3.2).
Define $f(\alpha) = \max_i\ r_i(\alpha)$.
**while** $ub - lb > \epsilon_1$ **do**
  $\hat{\alpha} = (ub + lb)/2$.
  $r_l \leftarrow f(\hat{\alpha} - \epsilon_2)$.
  $r \leftarrow f(\hat{\alpha})$.
  $r_r \leftarrow f(\hat{\alpha} + \epsilon_2)$.
  **if** $r_l > r > r_r$ **then**
    /* $\hat{\alpha}$ is on the decreasing part of $f(\alpha)$.*/
    $lb \leftarrow \alpha$.
  **else if** $r_l < r < r_r$ **then**
    /* $\hat{\alpha}$ is on the increasing part of $f(\alpha)$.*/
    $ub \leftarrow \alpha$.
  **else**
    /* $\hat{\alpha}$ is at a stationary point up to precision $\epsilon_2$.*/
    Break.
  **end if**
**end while**
**return** $\alpha = (ub + lb)/2$.

### 3.3. Convergence of FDM

FDM (Algorithm 2) terminates when the centre of the MEB of $\mathcal{G}$ is at the origin. Here, we show that this is the correct stopping criterion. First, we state another basic result before proving the main theorem.

**Lemma 3.** *Given two vectors* $\mathbf{a}$ *and* $\mathbf{b}$*, if* $\langle \mathbf{a}, \mathbf{b} \rangle > 0$*, then there exist a positive value* $\epsilon$ *that* $\|\epsilon\mathbf{b} - \mathbf{a}\|_2 < \|\mathbf{a}\|_2$.

**Theorem 3.** $\hat{\mathbf{x}}$ *is a stationary point of* (1) *iff the centre* $\mathbf{m}^*$ *of the MEB of* $\mathcal{G}$ *coincides with the origin* $\mathbf{o}$.

*Proof.* If $\hat{\mathbf{x}}$ is not a stationary point, then the values of the active residuals can be decreased simultaneously. This, by Lemma 1, implies the existence of a $\boldsymbol{\lambda}$ at $\hat{\mathbf{x}}$ that satisfies

$$\langle \boldsymbol{\lambda}, \mathbf{g} \rangle > 0 \quad \forall \mathbf{g} \in \mathcal{G}. \tag{30}$$

Therefore, by Lemma 3, there exists a non-zero $\epsilon$ such that

$$\|\epsilon\boldsymbol{\lambda} - \mathbf{g}\|_2 < \|\mathbf{g}\|_2 = 1 \quad \forall \mathbf{g} \in \mathcal{G}, \tag{31}$$

thus the origin $\mathbf{o}$ cannot coincide with $\mathbf{m}^*$.

If $\mathbf{m}^*$ does not coincide with $\mathbf{o}$, then by Theorem 1, $\mathbf{m}^*$ is a descent direction at $\hat{\mathbf{x}}$; $\hat{\mathbf{x}}$ is thus not a stationary point. □

Due to finite precision, a threshold $\epsilon_0$ ($\epsilon_0 = 10^{-8}$ in our experiments) is used in Algorithm 3 to test if $\mathbf{m}^*$ is numerically equal to $\mathbf{o}$. Finally, combining Theorem 3 and that a pseudo-convex function has only one stationary point, it is guaranteed that FDM will reach the global minimum in finite steps.

## 4. Experiments

There are two parts in our experiments: one is dedicated to benchmark the performance of FDM on triangulation sub-problem ($\text{Int}_k$), and the other is to benchmark the resection-intersection (henceforth, Res-Int) algorithm (1) with FDM solver on KRot. Experiments were done on a PC with a 3.7GHz Intel 4-core CPU and 16GB RAM.

**Choice of $p$-norm** Though our method is applicable to any $p \geq 1$ in (2), most of the previous works focussed on $p = 2$. Thus, we fixed $p = 2$ in our experiments. This however excluded [8], which is limited to $p = \infty$. In any case, [8] is only designed for the special case of triangulation but not our targeted problem—KRot.

**Datasets** We used 6 publicly available datasets from [9], covering small to large problem sizes to demonstrate the scalability of our method. Specifically, we used House (Small), Lund Cathedral (Small), Lund Cathedral (Large), Alcatraz Courtyard, Alcatraz Water Tower, and University of Western Ontario (Large). It is worth noting that for KRot, the state-of-the-art methods [4, 6, 11] were tested only on relatively small problems: up to M = 23,674 scene points and L = 67 cameras in [11]. Here, the data we used have one order magnitude more scene points and two orders of magnitude more cameras—as we will show later.

[2]Note that most globally optimal numerical schemes, including algorithms for KRot, guarantee optimality only up to a pre-defined threshold.

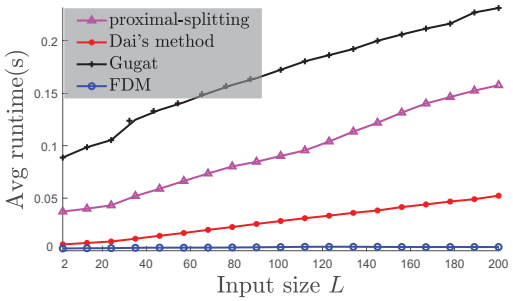



Figure 4. Runtime of competing methods on synthetic triangulation instances. For each input size $L$, the runtime was averaged over $M = 20$ random instances.

### 4.1. Triangulation

We compared FDM with (1) Gugat's algorithm [15], which was shown in [4] to outperform the other methods that solve convex sub-problems (bisection [20], Dinkelbach's method [7, 28]), (2) Dai's method [6], and (3) proximal splitting [11]. All methods (including FDM) were executed in Matlab[3]. We used SeDuMi [34] as the SOCP solver for the convex feasibility problems in Gugat.

To initialise a specific triangulation instance, we used the mid-point method (a closed-form solver) [17] on two randomly selected measurements to find the initial estimate.

**Synthetic data** We generated $L$ cameras with random poses, and varied $L$ from 2 to 200 (recall that the size of a triangulation instance is $L$). For each $L$ setting, we randomly generated $M = 20$ 3D scene points, projected them onto the cameras, and added Gaussian noise of $\sigma = 5$ to the projected points—using the projected point as data, this created $M$ random triangulation instances per $L$ setting. Fig. 4 shows the runtime of all competing methods

It is evident that FDM and Dai's method significantly outperformed Gugat and proximal splitting. Comparing just the two descent methods that do not employ convex sub-problems, FDM was not only faster than Dai's method, the former also scaled much better to large input sizes. As we will show later in Sec. 4.2, the good scaling property of FDM is essential for solving large-scale KRot instances.

**Real data** The real datasets used contain estimated camera poses, which we used to set up triangulation instances. Statistics of the datasets and runtime results are available in Table 1. Again, the excellent performance of FDM (avg runtime of ≈1 ms) was observed in real data. Note that, although practical triangulation instances are small ($L \leq 20$), it is still crucial to perform triangulation very efficiently due to the sheer number of triangulation instances.

[3]For Gugat, using Agarwal's implementation [4]. For [6, 11], using our own implementation since the original authors' code were not available.

### 4.2. Known rotation problem

We compared Res-Int (with FDM for the sub-problems) with Gugat's algorithm and proximal splitting, where the latter two were executed directly on the *full* KRot problem (i.e., each iteration updates all $3(L + M)$ variables). We did not compare against using Gugat, proximal splitting and Dai et al. [6] as sub-problem solvers in the Res-Int framework, since as demonstrated in Table 1, these three methods as sub-problem solvers are much slower than FDM. In fact, using Gugat and proximal splitting on the full KRot problem consumed much less time than using them as sub-problem solvers in Res-Int. For Res-Int, we also tested both sequential (seq) and parallel (par) versions (using 4 cores).

The major internal computations of the competitors are solved by third-party packages that were implemented in C/C++, e.g., SeDuMi for the SOCP feasibility tests in Gugat, and SBA [22] for the bundle adjustment subroutine in proximal splitting. Therefore, for a fair comparison, we also implemented FDM in C-Mex for this experiment.

To initialise a KRot instance, we ran 1 iteration of the bisection method given a loose upper bound $ub = 100$ pixels, as was done in [4].

Table 2 shows the runtime (in seconds) of the methods. To ensure that we did not exceed the capacity of SeDuMi in Gugat, we cull scene points that were observed in few images (i.e., if a scene point was observed by less than a certain number 'vis' of images, it was removed from the optimisation). Also, if an algorithm was not able to finish running within the cut-off limit of 3 hours, we terminated the program.

As in Table 2, Res-Int (both sequential and parallel variants) significantly outperformed Gugat and proximal splitting—in fact, Res-Int was able to scale up to input sizes that were beyond the reach of the two previous methods. Note that the 4 bigger data in Table 2 are significantly larger than the examples tested in [4] and [11].

For qualitative results of our method, see Fig. 5.

## 5. Conclusion

The proposed Res-Int algorithm for known rotation problem partitions the task into multiple 3-variable pseudo-convex optimisations and introduces a novel fast descent method based on minimum enclosing ball technique to solve the sub-problems. Not only the Res-Int algorithm achieves vastly superior performance to existing KRot methods, the FDM standalone also becomes the state-of-the-art triangulation solver. We hope to see both algorithms be applied to broader vision applications in the future.

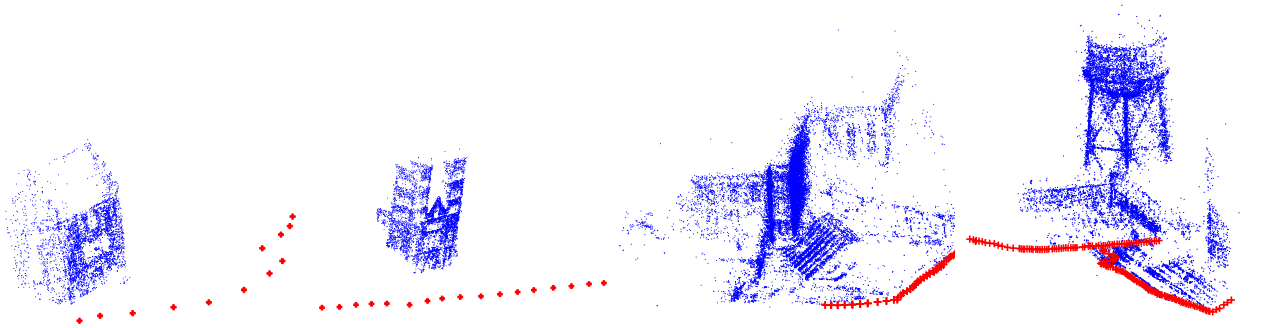
**Acknowledgements** This work was supported by ARC grant DP160103490 and CE140100016.

| Dataset statistics (for triangulation) | | | | Avg runtime (in milliseconds) | | | |
|---|---|---|---|---|---|---|---|
| Name | # points | # images | avg $L$ | Gugat [4] | Proximal [11] | Dai [6] | FDM |
| House (S) | 12,444 | 12 | 2.83 | 28.02 | 25.29 | 1.28 | **0.30** |
| Lund (S) | 16,878 | 17 | 2.68 | 28.17 | 22.01 | 1.28 | **0.29** |
| Yard | 23,674 | 133 | 13.58 | 57.67 | 44.14 | 3.20 | **1.06** |
| Tower | 14,828 | 172 | 11.43 | 53.90 | 47.57 | 2.74 | **0.85** |
| UWO (L) | 97,326 | 692 | 13.61 | 64.04 | 72.49 | 3.19 | **1.19** |
| Lund (L) | 159,055 | 1,208 | 14.60 | 73.04 | 99.75 | 3.58 | **1.25** |

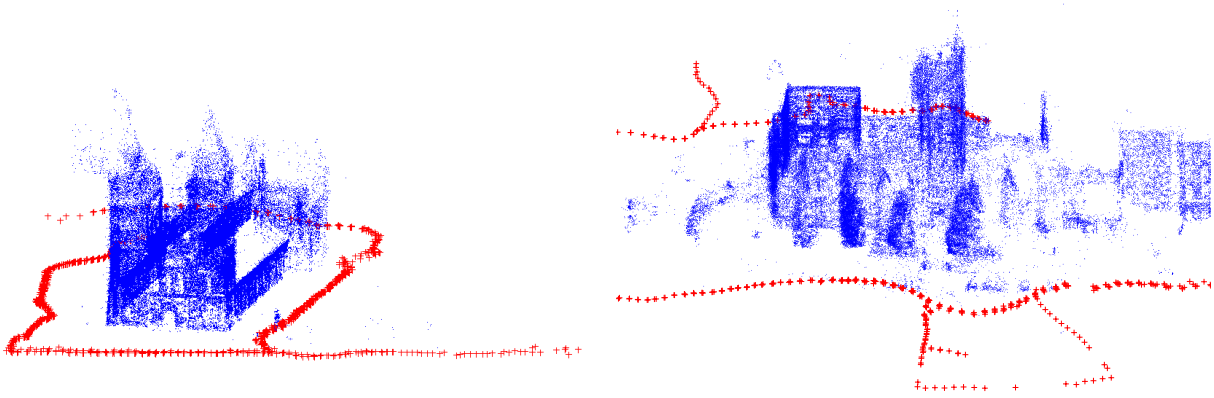
Table 1. Average runtime (in milliseconds) per triangulation instance on real data. '# points' is the total number of triangulation instances, and 'avg $L$' is the average size $L$ of the triangulation instances (NB: not all scene points are observed in each image).

| Dataset statistics (for KRot) | | | | | Avg runtime (in seconds) | | | |
|---|---|---|---|---|---|---|---|---|
| Name | vis | # points $M$ | # cameras $L$ | # obs | Gugat [4] | Proximal [11] | Res-Int (seq) | Res-Int (par) |
| House (S) | 4 | 2,174 | 12 | 12,037 | 27.80 | 19.18 | **3.15** | **2.97** |
| Lund (S) | 4 | 2,873 | 17 | 13,629 | 24.61 | 14.49 | **4.70** | **3.28** |
| Yard | 2 | 23,674 | 133 | 321,554 | n/a | 3,313.10 | **782.69** | **245.10** |
| Tower | 2 | 14,828 | 172 | 169,618 | n/a | 1,374.90 | **387.24** | **128.02** |
| UWO (L) | 2 | 97,326 | 692 | 1,324,698 | n/a | n/a | **2,347.70** | **698.84** |
| Lund (L) | 8 | 103,940 | 1,208 | 2,002,637 | n/a | n/a | **5,880.40** | **2,978.25** |

Table 2. Total runtime (in seconds) for KRot on real data. 'vis' is the threshold used to cull scene points that were observed in few images (i.e., if a scene point was observed in $<$vis images, it was removed from the optimisation). The purpose of culling is to reduce the overall problem size $3(M + L)$ and avoid exceeding the "capacity" of SeDuMi in Gugat. '# obs' is the total number of residual functions. If an algorithm was not able to finish running on an instance in 2 hours, we terminated the program and label their runtime as 'n/a' above.



(a) House (S) (b) Lund (S) (c) Yard (d) Tower



(e) Lund (L) (f) UWO (L)

Figure 5. Reconstruction results of Res-Int (Algorithm 1) on 6 real data. Each red '+' represents the position of a camera. Orientations of the cameras are not shown because they were not part of the optimisation variables of Res-Int.

## References


[1] https://en.wikipedia.org/wiki/Pseudoconvex_function. 3

[2] https://en.wikipedia.org/wiki/Law_of_sines. 4

[3] https://en.wikipedia.org/wiki/Law_of_cosines. 4

[4] S. Agarwal, N. Snavely, and S. M. Seitz. Fast algorithms for $L_\infty$ problems in multiview geometry. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 1, 2, 5, 6, 7, 8

[5] M. Badoiu and K. L. Clarkson. Smaller core-sets for balls. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 801–802. Society for Industrial and Applied Mathematics, 2003. 4

[6] Z. Dai, Y. Wu, F. Zhang, and H. Wang. A novel fast method for $L_\infty$ problems in multiview geometry. *Computer Vision–ECCV 2012*, pages 116–129, 2012. 2, 6, 7, 8

[7] W. Dinkelbach. On nonlinear fractional programming. *Management science*, 13(7):492–498, 1967. 7

[8] S. Donné, B. Goossens, and W. Philips. Point triangulation through polyhedron collapse using the l infinity norm. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 792–800, 2015. 2, 3, 6

[9] O. Enqvist, F. Kahl, and C. Olsson. Non-sequential structure from motion. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 264–271. IEEE, 2011. 6

[10] D. Eppstein. Quasiconvex programming. *Combinatorial and Computational Geometry*, 52(287-331):3, 2005. 5

[11] A. Eriksson and M. Isaksson. Pseudoconvex proximal splitting for l-infinity problems in multiview geometry. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 4066–4073. IEEE, 2014. 2, 6, 7, 8

[12] A. Eriksson, C. Olsson, F. Kahl, O. Enqvist, and T.-J. Chin. Why rotation averaging is easy. *arXiv preprint arXiv:1705.01362*, 2017. 1

[13] T. Goldstein, P. Hand, C. Lee, V. Voroninski, and S. Soatto. Shapefit and shapekick for robust, scalable structure from motion. In *European Conference on Computer Vision*, pages 289–304. Springer, 2016. 1

[14] V. M. Govindu. Lie-algebraic averaging for globally consistent motion estimation. In *CVPR 2004, IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2004. 1

[15] M. Gugat. A fast algorithm for a class of generalized fractional programs. *Management Science*, 42(10):1493–1499, 1996. 7

[16] P. Hand, C. Lee, and V. Voroninski. Exact simultaneous recovery of locations and structure from known orientations and corrupted point correspondences. *Discrete & Computational Geometry*, 59(2):413–450, 2018. 1

[17] R. Hartley and F. Schaffalitzky. $L_\infty$ minimization in geometric reconstruction problems. In *CVPR 2004, IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2004. 2, 7

[18] R. Hartley, J. Trumpf, Y. Dai, and H. Li. Rotation averaging. *International journal of computer vision*, 103(3):267–305, 2013. 1

[19] F. Kahl. Multiple view geometry and the $L_\infty$-norm. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1002–1009. IEEE, 2005. 1

[20] F. Kahl and R. Hartley. Multiple-view geometry under the $L_\infty$-norm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(9):1603–1617, 2008. 1, 2, 5, 7

[21] Q. Ke and T. Kanade. Quasiconvex optimization for robust geometric reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(10):1834–1847, 2007. 1, 2

[22] M. Lourakis and A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Aug. 2004. Available from http://www.ics.forth.gr/ lourakis/sba+. 1, 7

[23] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. 1

[24] J. Matoušek. On geometric optimization with few violated constraints. *Discrete & Computational Geometry*, 14(4):365–384, 1995. 5

[25] K. Mitra and R. Chellappa. A scalable projective bundle adjustment algorithm using the l infinity norm. In *Computer Vision, Graphics & Image Processing, 2008. ICVGIP'08. Sixth Indian Conference on*, pages 79–86. IEEE, 2008. 1, 2

[26] J. Nocedal and S. Wright. *Numerical optimization, 2nd Edition*. Springer, 2006. 4

[27] C. Olsson and O. Enqvist. Stable structure from motion for unordered image collections. *Image Analysis*, pages 524–535, 2011. 1

[28] C. Olsson, A. P. Eriksson, and F. Kahl. Efficient optimization for $L_\infty$-problems using pseudoconvexity. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007. 1, 2, 5, 7

[29] D. M. Rosen, L. Carlone, A. S. Bandeira, and J. J. Leonard. A certifiably correct algorithm for synchronization over the special euclidean group. *arXiv preprint arXiv:1611.00128*, 2016. 1

[30] Y. Seo and R. Hartley. A fast method to minimize $L_\infty$ error norm for geometric vision problems. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007. 1, 2

[31] K. Sim and R. Hartley. Recovering camera motion using $L_\infty$ minimization. In *CVPR 2006, IEEE Computer Society Conference on*, volume 1, pages 1230–1237. IEEE, 2006. 1

[32] K. Sim and R. Hartley. Removing outliers using the $L_\infty$ norm. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 485–494. IEEE, 2006. 5

[33] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring image collections in 3d. 2006. 1

[34] J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999. 7

[35] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustmenta modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999. 1, 2

[36] M. Uyttendaele, A. Criminisi, S. B. Kang, S. Winder, R. Szeliski, and R. Hartley. Image-based interactive exploration of real-world environments. *IEEE Computer Graphics and Applications*, 24(3):52–63, 2004. 1

[37] E. Welzl. Smallest enclosing disks (balls and ellipsoids). In *New results and new trends in computer science*, pages 359–370. Springer, 1991. 5

[38] D. Zwillinger. *CRC standard mathematical tables and formulae*. CRC press, 2002. 5