Doctoral Thesis

Robust and Large-Scale Quasiconvex Programming in Structure-from-Motion

Submitted by:

Qianggong ZHANG

Supervised by:

Assoc. Prof. Tat-Jun CHIN

Prof. David SUTER

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Faculty of Engineering, Computer and Mathematical Sciences

School of Computer Science

May 2018

Declaration

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

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

I acknowledge that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

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

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Signed:

Date: 31 / 01 / 2018

THE UNIVERSITY OF ADELAIDE

Abstract

School of Computer Science

Doctor of Philosophy

Robust and Large-Scale Quasiconvex Programming in Structure-from-Motion

by Qianggong ZHANG

Structure-from-Motion (SfM) is a cornerstone of computer vision. Briefly speaking, SfM is the task of simultaneously estimating the poses of the cameras behind a set of images of a scene, and the 3D coordinates of the points in the scene.

Often, the optimisation problems that underpin SfM do not have closed-form solutions, and finding solutions via numerical schemes is necessary. An objective function, which measures the discrepancy of a geometric object (e.g., camera poses, rotations, 3D coordinates) with a set of image measurements, is to be minimised. Each image measurement gives rise to an error function. For example, the reprojection error, which measures the distance between an observed image point and the projection of a 3D point onto the image, is a commonly used error function.

An influential optimisation paradigm in SfM is the ℓ_∞ paradigm, where the objective function takes the form of the maximum of all individual error functions (e.g. individual reprojection errors of scene points). The benefit of the ℓ_∞ paradigm is that the objective function of many SfM optimisation problems become quasiconvex, hence there is a unique minimum in the objective function. The task of formulating and minimising quasiconvex objective functions is called *quasiconvex programming*.

Although tremendous progress in SfM techniques under the ℓ_∞ paradigm has been made, there are still unsatisfactorily solved problems, specifically, problems associated with large-scale input data and outliers in the data. This thesis describes novel techniques to tackle these problems.

A major weakness of the ℓ_∞ paradigm is its susceptibility to outliers. This thesis improves the robustness of ℓ_∞ solutions against outliers by employing the least median of squares (LMS) criterion, which amounts to minimising the median error. In the context of triangulation, this thesis proposes a locally convergent robust algorithm underpinned by a novel quasiconvex plane sweep technique. Imposing the LMS criterion achieves significant outlier tolerance, and, at the same time, some properties of quasiconvexity greatly simplify the process of solving the LMS problem.

Approximation is a commonly used technique to tackle large-scale input data. This thesis introduces the coreset technique to quasiconvex programming problems. The coreset technique aims find a representative subset of the input data, such that solving the same problem on the subset yields a solution that is within known bound of the optimal solution on the complete input set. In particular, this thesis develops a coreset approximate algorithm to handle large-scale triangulation tasks.

Another technique to handle large-scale input data is to break the optimisation into multiple smaller sub-problems. Such a decomposition usually speeds up the overall optimisation process, and alleviates the limitation on memory. This thesis develops a large-scale optimisation algorithm for the known rotation problem (KRot). The proposed method decomposes the original quasiconvex programming problem with potentially hundreds of thousands of parameters into multiple sub-problems with only three parameters each. An efficient solver based on a novel minimum enclosing ball technique is proposed to solve the sub-problems.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Associate Professor Tat-Jun Chin, for all his advice, comments, and critical revisions throughout this thesis and the articles we published during my Ph.D study. I really appreciate his interest in this research. Working with him has certainly been an enriching life experience. I would also like to thank my co-supervisor, Professor David Suter, for his advice, revisions and comments during research meetings.

I am also grateful to the Department of Education and Training for providing the scholarship to make my Ph.D study possible. I extend this thank to the School, the Faculty, and the University for facilitating my research.

Last but not least, my sincere appreciation to my family for their emotional support.

Contents

Publications

This thesis is in part result of the work presented in the following papers:

- Qianggong Zhang, Tat-Jun Chin and Huu Minh Le: A Fast Resection-Intersection Method for the Known Rotation Problem. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018.
- Qianggong Zhang, Tat-Jun Chin and David Suter: Quasiconvex Plane Sweep for Triangulation With Outliers. IEEE International Conference on Computer Vision (ICCV) 2017.
- Qianggong Zhang and Tat-Jun Chin: Coresets for Triangulation. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 2017.
- Qianggong Zhang and Tat-Jun Chin: An Efficient Meta-Algorithm for Triangulation. Asian Conference on Computer Vision (ACCV) Workshops, 148-161.

Chapter 1

Introduction

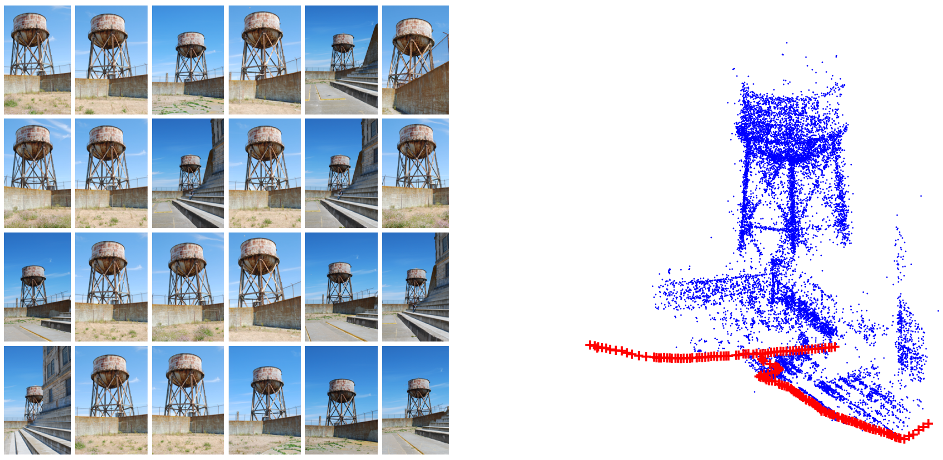
Computer vision is a discipline aiming to extract high-level information from digital images or videos. In many respects, computer vision can be regarded as a field of artificial intelligence, and over the last 40 years, researchers have endeavoured to develop computers with the capacity to 'see' the world like humans do. Building such intelligent computer vision systems is a challenging task. While biological vision systems are able to effortlessly extract rich information from the visual data, computer vision systems still lack the ability to fully handle the complexity of visual data.

However, computer vision is developing rapidly. The last decade has seen an unprecedented level of deployment of vision technologies. Some examples of computer vision applications include automatic face recognition, automated medical image analysis, robotic manufacturing, object recognition and tracking, augmented reality, autonomous vehicles, security and surveillance, to name a few. In some tasks, such as object classification and face recognition, computer vision can even outperform humans. It is conceivable that computer vision will become increasingly prevalent in the future.

1.1 SfM in computer vision

Many computer vision capabilities benefit from having the 3D structure of the environment. A primary way of acquiring 3D information is by SfM. Given a set of images (views) of a scene, SfM estimates the 3D coordinates of the points (structure) in the scene and the poses of the cameras (motion) behind the images. See Figure 1.1.

A typical SfM pipeline can be largely separated into two modules, the feature-extracting-and-matching module and the structure-and-motion-estimation module. Raw images are firstly fed into the feature-extracting-and-matching module where features (e.g., interest



(a) Input of SfM (b) Output of SfM

Figure 1.1: Demonstration of SfM. Panel (a) shows sample images from the Alcatraz water tower dataset [1]. Panel (b) is the output of SfM, where each red '+' represents the position of camera centres of the images, and the blue dots represent reconstructed scene points.

points) are extracted from the images and then matched across the images to produce a set of feature correspondences, see Figure 1.2. The feature correspondences are input to the structure-and-motion-estimation module to calculate the 3D structure of the scene and camera poses.

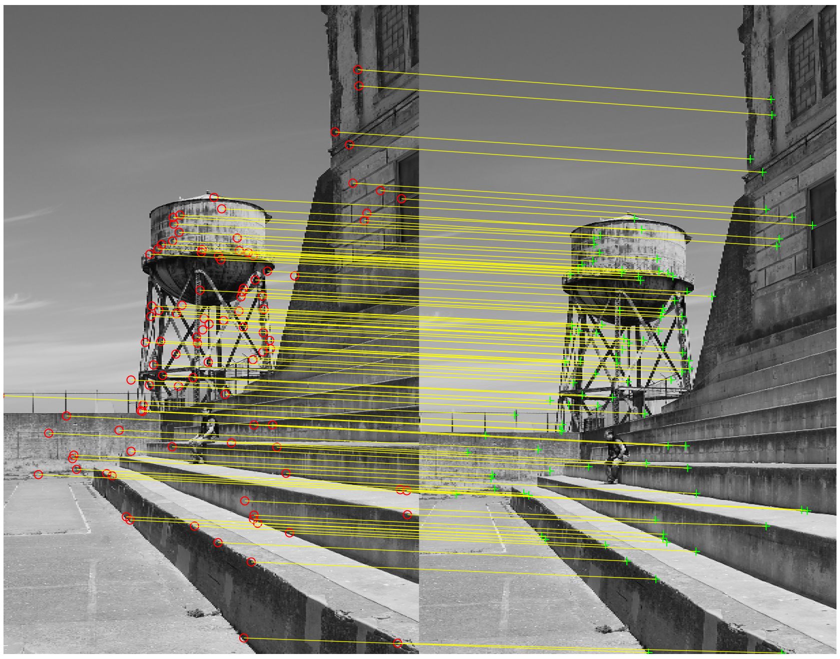
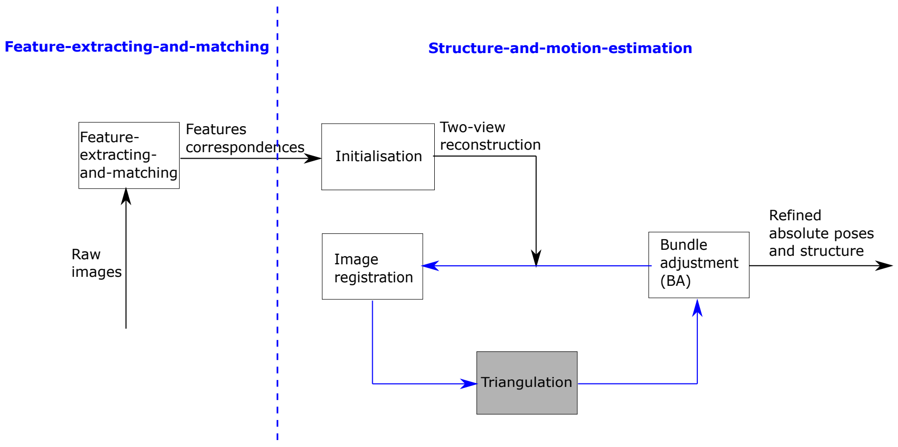
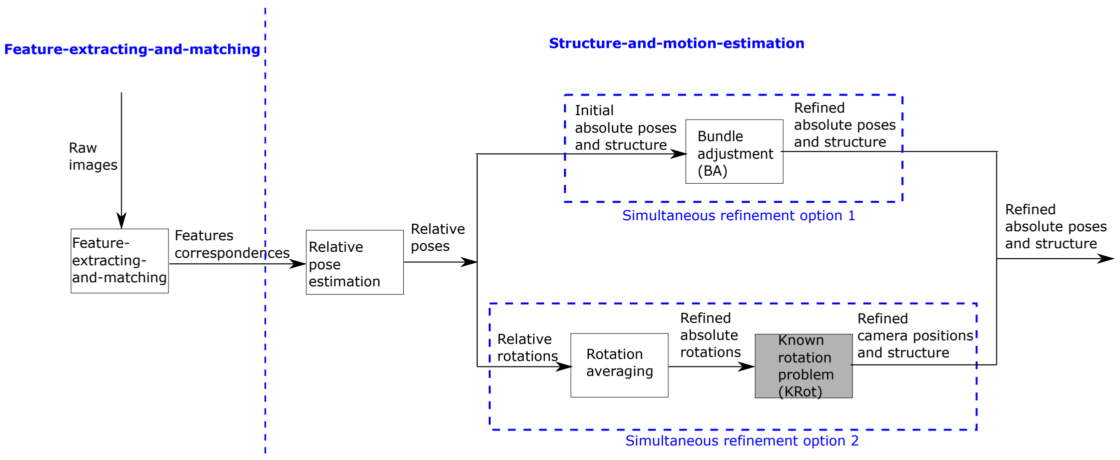


Figure 1.2: Feature-extraction-and-matching. Features extracted from the images on the left (red circles) are matched with features extracted from the image on the right (green crosses) as indicated by yellow lines.

In the structure-and-motion-estimation module, there are mainly two approaches, the incremental approach [51, 5, 47] and the global approach [42, 39, 16]. Figure 1.3 shows the two types of pipelines.


Feature-extracting-and-matching
Structure-and-motion-estimation
Feature-extracting-and-matching
Features correspondences
Initialisation
Two-view reconstruction
Raw images
Image registration
Bundle adjustment (BA)
Refined absolute poses and structure
Triangulation

(a) An incremental SfM pipeline


Feature-extracting-and-matching
Structure-and-motion-estimation
Raw images
Feature-extracting-and-matching
Features correspondences
Relative pose estimation
Relative poses
Initial absolute poses and structure
Bundle adjustment (BA)
Refined absolute poses and structure
Simultaneous refinement option 1
Refined absolute poses and structure
Relative rotations
Rotation averaging
Refined absolute rotations
Known rotation problem (KRot)
Refined camera positions and structure
Simultaneous refinement option 2

(b) A global SfM pipeline

FIGURE 1.3: Pipeline of SfM. Panel (a) shows an incremental SfM pipeline. Panel (b) shows a global SfM pipeline. Triangulation and the known rotation problem (KRot), which are in grey boxes in the two panels, are the two sub-problems in SfM that are focused upon in this thesis and will be introduced in detail in Section 1.2.2 and Section 1.2.3.

1.1.1 Incremental SfM pipeline

Figure 1.3a demonstrates a typical incremental SfM pipeline. The *initialisation* procedure is to initialise the reconstruction model with a carefully selected two-view reconstruction [10]. Choosing a suitable initial pair is critical, since the reconstruction may never recover from a bad initialization. New images can be added to the current reconstruction in succession to gradually complete the reconstruction of the whole scene.

Each time a new image I is to be added, it first goes through the *image registration* procedure where the pose of I is obtained by solving the Perspective-n-Point (PnP) problem. The input of the PnP problem is the feature correspondences to the scene points whose 3D coordinates have already been estimated in the current reconstruction (the registered scene points), and the RANSAC technique [24] is applied here to remove wrong feature correspondences (outliers).

The newly registered image I observes some registered scene points and some unregistered scene points. If any unregistered scene point is also observed by as least one more registered image, then it can be reconstructed by *triangulation*. Triangulation is a crucial step in the incremental SfM; it amounts to estimating the 3D coordinates of a scent point from the observations of the point in a set of images when the camera poses of the images are known. For the registered scene points, if any of them is also observed by I, then its 3D coordinates is refined by triangulation to incorporate the additional measurement from I.

An incremental pipeline needs further refinement to prevent drifts to a non-recoverable state [15]. *Bundle adjustment* [54] (BA) is the procedure to provide necessary refinement of the poses of registered images and the 3D coordinates of registered scene points. BA is a non-linear least squares problem that minimises the sum of squared reprojection errors (see Section 1.2.2.1), and Levenberg-Marquardt method is the standard method for solving BA.

The incremental approach suits SfM problems with a large collection of images and has demonstrated impressive results on internet-scale image sets [5], however, it is typically slow, vulnerable to drifting errors, and dependent on good initialisations. In contrast, the global approach has better potential in efficiency and accuracy [16].

1.1.2 Global SfM pipeline

In a typical global SfM pipeline, as shown in Figure 1.3b, the relative poses between all pairs of images are estimated simultaneously in the *relative pose estimation* procedure. As in the incremental pipeline, outliers are removed in this procedure.

Starting from here, there are various strategies for refining the camera poses and the structure; here, two representative strategies are presented. The first option performs the BA procedure. With the relative poses available, a rough estimation of absolute poses of all cameras can be obtained by fixing an arbitrary camera as the reference camera and inferring the poses of others; then an initial estimation of the structure (the 3D coordinates of the scene points) is obtainable by triangulation. This provides necessary initialisation for BA to simultaneously refine the absolute poses of cameras and the structure.

The second option is to first get a refined estimation of the absolution rotations of all the cameras via a *rotation averaging* procedure [28], then solve a *known rotation problem* (KRot) [32] to simultaneously estimate the camera translations and the structure (see Section 1.2.3 for details). This thesis focuses on the second option.

1.1.3 Structure densification

What the SfM pipeline has achieved thus far might be a sparse/semi-dense reconstruction of the texture-rich areas of a scene, with gaps in the low-texture regions. This sparse/semi-dense reconstruction can be densified by multi-view stereo (MVS) techniques (e.g., patch-based multi-view stereo (PMVS) [25]) if a dense reconstruction is desirable. SfM + MVS has been one of the standard pipelines for dense reconstruction [34].

The following introduces the core concepts behind some of the building blocks of the pipeline in Figure 1.3 that are relevant to this thesis.

1.2 Fundamentals of SfM

1.2.1 Pinhole camera model

In SfM, the commonly used camera model is the pinhole camera model. Observe Figure 1.4 where a camera with focal length f is placed in a 3D Euclidean coordinate system — here, the world coordinate system (with axes $\mathrm{X_W}$, $\mathrm{Y_W}$ and $\mathrm{Z_W}$) coincides with the camera coordinate system (with axes $\mathrm{X_C}$, $\mathrm{Y_C}$ and $\mathrm{Z_C}$), such that the camera centre $\mathbf{C}$ is at the origin $\mathbf{O}$ of the world coordinate system and the axes of the camera coordinate system align with the axes of the world coordinate system. The *principal point* $\mathbf{p}$ of the camera is at the origin of the 2D Euclidean coordinate system (with axes x and y) on the image plane (or image) $\mathcal{P}$. A 3D point $\mathbf{X}$ in the world coordinate system and the

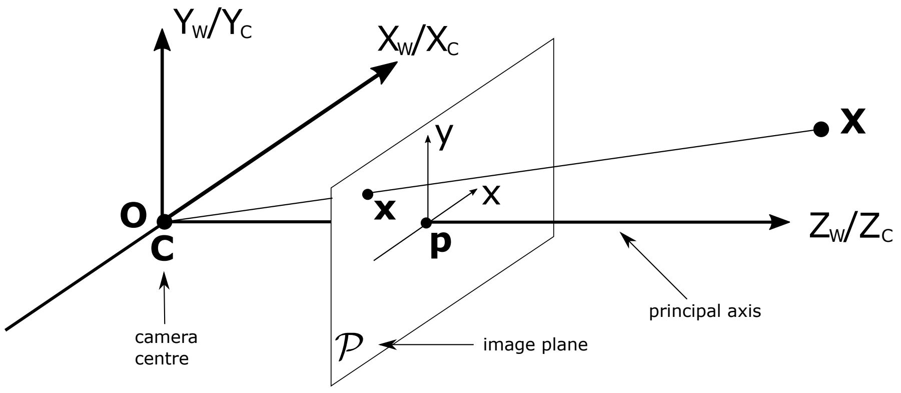


FIGURE 1.4: Pinhole camera model. **C** is the camera centre and **p** the principal point. The world coordinate system ($\mathrm{X_W}$, $\mathrm{Y_W}$, $\mathrm{Z_W}$) coincides with the camera coordinate system ($\mathrm{X_C}$, $\mathrm{Y_C}$, $\mathrm{Z_C}$). **C** is placed at the origin **O** of the world coordinate system. **X** and **x** are respectively the 3D scene point and its projected image point on the image $\mathcal{P}$. The distance between **C** and **p** equals the focal length f.

2D coordinates **x** of the 2D projection of **X** onto $\mathcal{P}$ comply with the equation:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{fX_1}{X_3} \\ \frac{fX_1}{X_3} \end{bmatrix},$$

$$\text{where} \quad \mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \in \mathbb{R}^3 \text{ is the 3D coordinates of } \mathbf{X}, \tag{1.1}$$

$$\text{and} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2 \text{ is the 2D coordinates of } \mathbf{x} \text{ on } \mathcal{P}.$$

Letting $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{x}}$ represent the homogeneous coordinates of **X** and **x** respectively, then (1.1) can be rewritten as

$$\tilde{\mathbf{x}} = \begin{bmatrix} fX_1 \\ fX_2 \\ X_3 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\mathbf{X}} = P\tilde{\mathbf{X}}. \tag{1.2}$$

The matrix P that transforms $\tilde{\mathbf{X}}$ to $\tilde{\mathbf{x}}$ in (1.2) is called the *camera matrix.* The camera matrix can be further decomposed into three parts

$$P = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \middle| \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right] = K[R|\mathbf{t}], \tag{1.3}$$

$$\tag{1.4}$$

where

$$K = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.5}$$

is called the *camera calibration matrix*, and

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{t} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{1.6}$$

are respectively the rotation and translation that define the orientation and position of the camera coordinate system with respect to the world coordinate system. Note that the rotation matrix R is an identity matrix and the translation $\mathbf{t}$ is a zero vector in this example because of the coincidence between the world coordinate system and the camera coordinate system. In the general case, R can be any valid rotation matrix, and $\mathbf{t}$ will follow

$$\mathbf{t} = -R\,\hat{\mathbf{t}}, \tag{1.7}$$

where $\hat{\mathbf{t}}$ is the 3D coordinates of the camera centre $\mathbf{C}$ in the world coordinate system. R and $\mathbf{t}$ together define the *pose* of a camera.

Often, the principal point $\mathbf{p}$ is not at the origin of the camera coordinate system on $\mathcal{P}$, in which case, a more general camera calibration matrix is

$$K = \begin{bmatrix} f & 0 & p_1 \\ 0 & f & p_2 \\ 0 & 0 & 1 \end{bmatrix}, \quad \text{where } \mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \in \mathbb{R}^2 \text{ is the 2D coordinates of } \mathbf{p} \text{ on } \mathcal{P}. \tag{1.8}$$

Normally, camera calibration matrices K are estimated by a calibration process [29, Section 8.5]; this thesis assumes that K for all images are known.

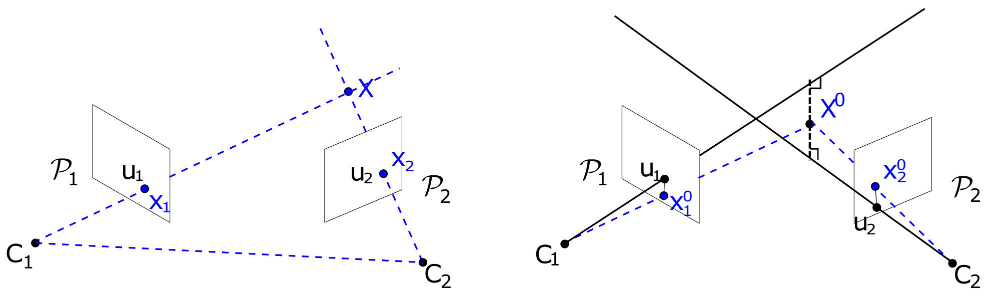
1.2.2 Triangulation

One fundamental task in SfM is triangulation, which aims to recover the 3D coordinates of a scene point from observations of the point in a set of images, given that the camera matrices P of the images are known. A basic 2-view triangulation instance is presented as follows to introduce the idea as well as some optimisation concepts that are commonly used in SfM.

Consider a scenario where the 3D coordinates of a single scene point need to be estimated from two images. The two images $\mathcal{P}_1$ and $\mathcal{P}_2$ are in front of their respective camera centre $\mathbf{C}_1$ and $\mathbf{C}_2$, and the 3D point is imaged by the two cameras, resulting in two 2D *measured* image points (or *measurements*) — $\mathbf{u}_1$ on $\mathcal{P}_1$ and $\mathbf{u}_2$ on $\mathcal{P}_2$ (see Figure 1.5a). Suppose the camera matrices of the two cameras, P_1 and P_2, are known. The task is to retrieve the 3D coordinates of the scene point from the two measurements $\mathbf{u}_1$ and $\mathbf{u}_2$.

In the absence of noise, estimating the 3D coordinates of the scene point is trivial. Suppose $\mathbf{X}$ is the true 3D coordinates of the scene point. Given P_1 and P_2, by (1.2), it is viable to project $\mathbf{X}$ onto the two images to produce two image points — $\mathbf{x}_1$ on $\mathcal{P}_1$ and $\mathbf{x}_2$ on $\mathcal{P}_2$. Then line $\overline{\mathbf{C}_1\mathbf{x}_1}$ and line $\overline{\mathbf{C}_2\mathbf{x}_2}$ must intersect at $\mathbf{X}$ in 3D space. This is shown in Figure 1.5a. Without noise, the measured image points $\mathbf{u}_1$ and $\mathbf{u}_2$ will coincide with the projected image points $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively, such that $\mathbf{X}$ can be calculated as the intersection of the two lines $\overline{\mathbf{C}_1\mathbf{u}_1}$ and $\overline{\mathbf{C}_2\mathbf{u}_2}$.

However, noise is inevitable in practice. On each image (e.g., $\mathcal{P}_1$), the existence of noise would cause a discrepancy between the measured image point (e.g., $\mathbf{u}_1$) and the projected image point (e.g., $\mathbf{x}_1$). In that case, in the example demonstrated in Figure 1.5b, line $\overline{\mathbf{C}_1\mathbf{u}_1}$ and line $\overline{\mathbf{C}_2\mathbf{u}_2}$ will not intersect, thus the simple method based on intersecting lines will fail. As illustrated in Figure 1.5b, suppose $\mathbf{X}^0$ is the current estimation of $\mathbf{X}$, then projecting $\mathbf{X}^0$ onto the two images gives two image points $\mathbf{x}_1^0$ on $\mathcal{P}_1$ and $\mathbf{x}_1^0$ on $\mathcal{P}_2$. Notice that, the measured image points $\mathbf{u}_1$ and $\mathbf{u}_1$ do not coincide with $\mathbf{x}_1^0$ and $\mathbf{x}_2^0$.



(a) No noise (b) With noise

Figure 1.5: A demonstration of 2-view geometry. The left panel demonstrates the simple intersection method for triangulation in the absence of noise. The right panel demonstrates the discrepancy between the noisy measurements and the projected points for a candidate 3D estimate $\mathbf{X}^0$.

1.2.2.1 Reprojection error

Reprojection error measures the discrepancy between the measured image point and the projected image point on each image. Formally, let $P_i \in \mathbb{R}^{3\times4}$ be the camera matrix of the i^{th} image, $\mathbf{u}_i$ the 2D measured image point of the 3D scene point $\mathbf{X}$ on the i^{th}

image, and $\mathbf{x}_i$ be the 2D projection of $\mathbf{X}$ on the i^{th} image. Then the reprojection error $f_i(\mathbf{X})$ caused by scene point $\mathbf{X}$ on the i^{th} image is defined as:

$$f_i(\mathbf{X}) = \|\mathbf{u}_i - \mathbf{x}_i\|_q = \left\| \mathbf{u}_i - \frac{P_i^{1:2}\tilde{\mathbf{X}}}{P_i^3\tilde{\mathbf{X}}} \right\|_q, \tag{1.9}$$

where $P^{1:2}$ is the first-two rows of P, and P^3 is the third row of P. The symbol $\|\cdot\|_q$ indicates a valid q-norm [2]; usually q is taken to be 1, 2 or ∞. In this thesis, q is left unspecified because most algorithms that are surveyed or proposed in this thesis can apply to any q, except for the polyhedron collapse triangulation algorithm surveyed in Section 2.2.3.1, where $q = \infty$. Other errors do exist, for example, the angular error (as used in [45, 44]) and the photometric error (as used in [40, 18]). However, this thesis will focus on the reprojection error since it is more commonly used.

In the presence of noise, it is desirable that the estimate $\mathbf{X}$ causes small reprojection errors. This creates an optimisation problem. The exact form of the optimisation problem for triangulation depends on how the multiple reprojection errors (incurred by the 3D scene point $\mathbf{X}$ on multiple images) are aggregated. The objective function $F(\mathbf{X})$ is used to aggregate the individual reprojection errors.

The following shows some examples of $F(\mathbf{X})$.

1.2.2.2 The ℓ_2 formulation of triangulation

A commonly used $F(\mathbf{X})$ is the sum of squared errors

$$F(\mathbf{X}) = \|f_i(\mathbf{X})\|_2 = \sqrt{\sum_i f_i(\mathbf{X})^2}, \tag{1.10}$$

and the optimisation problem that minimises $F(\mathbf{X})$ is also called the ℓ_2 formulation of triangulation.

The ℓ_2 formulation of triangulation can be interpreted as *maximum likelihood estimation*. However, the major weakness of the ℓ_2 formulation is that the objective function $F(\mathbf{X})$ contains multiple minima, and finding the global optimal solution is often intractable [31].

1.2.2.3 The ℓ_∞ formulation and quasiconvex programming

A more recently used $F(\mathbf{X})$ is the maximum over individual reprojection errors:

$$F(\mathbf{X}) = \|f_i(\mathbf{X})\|_\infty = \max_i \; f_i(\mathbf{X}). \tag{1.11}$$

This approach was first introduced in [27]. This thesis focuses on the ℓ_∞ paradigm and the associated optimisation problem. In a triangulation instance where a 3D scene point $\mathbf{X}$ is to be estimated from N images, the input of the triangulation instance includes camera matrices $\{P_i\}_{i=1}^N$ and 2D measurements $\{\mathbf{u}_i\}_{i=1}^N$ on the images, and the output is the 3D coordinates of $\mathbf{X}$. The optimisation problem is

$$\begin{aligned} \min_{\mathbf{X}\in\mathbb{R}^3} \quad & \max_i \quad f_i(\mathbf{X}), \\ \text{s.t.} \quad & P_i^3\tilde{\mathbf{X}} > 0, \;\; \forall i \in \{1,\ldots,N\}, \end{aligned} \tag{1.12}$$

where $f_i(\mathbf{X})$ is as defined in (1.9), and the constraints $P_i^3\tilde{\mathbf{X}} > 0$ ensure that $\mathbf{X}$ lies in front of all the cameras (also call *cheirality*).

Using the ℓ_∞ norm to aggregate the errors is useful mainly because $F(\mathbf{X})$ has a unique minimum. As established in Hartley and Schaffalitzky's seminal paper [27], the reprojection error is quasiconvex [3], hence their point-wise maximum is also quasiconvex. Quasiconvex functions have unique minima. Section 2.1 in Chapter 2 will define quasiconvex functions more formally.

On the other hand, there are weaknesses in the ℓ_∞ formulation. One weakness is its susceptibility to outliers. Another disadvantage of the ℓ_∞ formulation is that the efficiency of algorithms for optimisation deteriorates for large input data size since the algorithms often employ costly convex optimisation routines. This thesis makes progress towards addressing the above two issues.

1.2.3 KRot

KRot is the task of simultaneously estimating the 3D coordinates of scene points ($\mathbf{X}$) and cameras translations ($\mathbf{t}$), given that the rotations (R) of the cameras are known. It is part of an SfM pipeline where the orientations of the cameras have been estimated first using rotation averaging (see Figure 1.3). The strength of this approach is twofold: first, estimating the rotations of the cameras is a comparatively easy task. In some circumstances, rotation averaging can be solved globally [42, 22]. Second, when formulated as an ℓ_∞ problem, KRot becomes quasiconvex programming, which is also amenable to global solutions [32].

Formally, in a KRot instance that involves M 3D scene points $\{\mathbf{X}_k\}_{k=1}^M$ and N images, the input includes camera rotation matrices $\{R_j\}_{j=1}^N$ and 2D measurements $\{\mathbf{u}_{j,k}\}_{j=1,k=1}^{N,\;\;M}$, where $\mathbf{u}_{j,k}$ represents the 2D measurement of the k^{th} 3D scene point $\mathbf{X}_k$ on the j^{th} image, and the output is the 3D coordinates of the scene points $\{\mathbf{X}_k\}_{k=1}^M$ and the translations $\{\mathbf{t}_j\}_{j=1}^N$ of cameras. The reprojection error incurred by the k^{th} 3D scene point on the

j^{th} image is defined as

$$f_{j,k}(\mathbf{Y}) = \left\| \mathbf{u}_{j,k} - \frac{P_j^{1:2}\tilde{\mathbf{X}}_k}{P_j^3\tilde{\mathbf{X}}_k} \right\|_q = \left\| \mathbf{u}_{j,k} - \frac{K_j^{1:2}(R_j^{1:2}\mathbf{x}_k + \mathbf{t}_j^{1:2})}{K_j^3(R_j^3\mathbf{x}_k + \mathbf{t}_j^3)} \right\|_q, \tag{1.13}$$

where $P_j^{1:2}$ and P_j^3 are respectively the first two rows and the third row of P_j (same for $K_j^{1:2}$ and K_j^3, $R_j^{1:2}$ and R_j^3, $\mathbf{t}_j^{1:2}$ and $\mathbf{t}_j^3$), $\mathbf{Y}$ is the aggregation of parameters $\{\mathbf{t}_j\}_{j=1}^N$ and $\{\mathbf{X}_k\}_{k=1}^M$, and the symbol $\|\cdot\|_q$ represents the type of norm.

Under the ℓ_∞ formulation, the optimisation problem corresponding to KRot is

$$\begin{aligned} &\min_{\mathbf{Y}} \quad \max_{j,k} \quad f_{j,k}(\mathbf{Y}), \\ &\text{s.t.} \quad K_j^3(R_j^3\mathbf{X}_k + \mathbf{t}_j^3) > 0, \ \ \forall \ \ j \in \{1, \dots, N\}, k \in \{1, \dots, M\}, \end{aligned} \tag{1.14}$$

which estimates the scene points $\{\mathbf{X}_k\}_{k=1}^M$ and the camera translations $\{\mathbf{t}_j\}_{j=1}^N$ simultaneously. Intuitively, KRot aims to minimise the maximum reprojection error over all the 2D observations, and the constraints of the problem ensure that the estimated $\{\mathbf{x}_k\}$ lie in front of all the cameras. As established in [43], the ℓ_∞ KRot is a quasiconvex optimisation problem since (1.13) is quasiconvex.

1.2.4 Other optimisation problems

Note that there are other optimisation problems in the SfM pipeline, such as relative pose estimation, rotation averaging, and bundle adjustment, as shown in Figure 1.3. This thesis focuses on ℓ_∞ triangulation and KRot mainly because these two problems are representative of quasiconvex problems in computer vision.

1.3 Thesis outline

In Chapter 3, an efficient meta-algorithm for ℓ_∞ triangulation is introduced.[1] By exploiting the quasiconvexity of ℓ_∞ triangulation, the algorithm efficiently finds the global optimum, usually within 10 iterations regardless of the input size.

To deal with outliers, Chapter 4 introduces a locally optimal algorithm (Q-sweep) for robust triangulation. Q-sweep converges to a local minimum in a comparable runtime with the prevalent random methods. A novel quasiconvex plane sweep technique underpins the superior performance of Q-sweep. This is the first locally optimal algorithm for LMS triangulation.

[1]After this meta-algorithm was published, the authors realised it was similar to Seo and Hartley's work [48].

Chapter 5 shows that ℓ_∞ triangulation admits a coreset scheme and develops a coreset algorithm for large-scale triangulation. The proposed algorithm produces a representative subset (i.e., coreset) of the whole input such that solving triangulation on this coreset will produce a solution within known bound of the optimal solution on the whole input. This provides a theoretically justifiable approximate approach of dealing with large-scale input.

Finally, Chapter 6 proposes an intersection-resection algorithm (Res-Int) for KRot. The Res-Int algorithm breaks the original optimisation task (potentially with hundreds of thousands of parameters) into multiple sub-problems (with only 3 parameters), which can then be efficiently solved by a novel fast descent method (FDM) based on a minimum enclosing ball technique. The Res-Int algorithm is not only significantly faster than the current state-of-art methods, but also can scale up to problem sizes that are beyond the reach of existing methods.

Chapter 2

Literature Review

This chapter surveys existing algorithms for ℓ_∞ triangulation and ℓ_∞ KRot. To give an exhaustive list of existing solvers would be difficult, thus this chapter aims to survey the state-of-the-art algorithms that either are closely related to this thesis or can serve as competitors to the algorithms proposed in this thesis. For brevity, in this chapter, 'triangulation' implies 'ℓ_∞ triangulation' and 'KRot' implies 'ℓ_∞ KRot'.

Since most quasiconvex optimisation algorithms are applicable to both triangulation and KRot, this chapter first reviews existing methods for triangulation, including standard algorithms and robust algorithms to deal with outliers, then summarises the aspects in which KRot is different from triangulation in regard to optimisation.

In addition, several concepts from computational geometry that are relevant to this thesis, i.e., minimum enclosing ball (MEB) problem , coreset, and plane sweep, are also introduced.

2.1 Quasiconvexity

As in [14, Chapter 3.4], one definition of a quasiconvex function is as follows

Definition 2.1 (Quasiconvexity)**.** A function $e(\mathbf{x}) : \mathbb{R}^\rho \to \mathbb{R}$ is quasiconvex if its domain $\mathcal{Y}$ and all its δ-sublevel sets

$$\mathcal{S}_\delta = \{\mathbf{x} \in \mathcal{Y} \mid e(\mathbf{x}) \le \delta\}, \quad \delta \in \mathbb{R} \tag{2.1}$$

are convex.

An illustration of a 1D quasiconvex function $e(x) : \mathbb{R} \to \mathbb{R}$ is shown in Figure 2.1a.

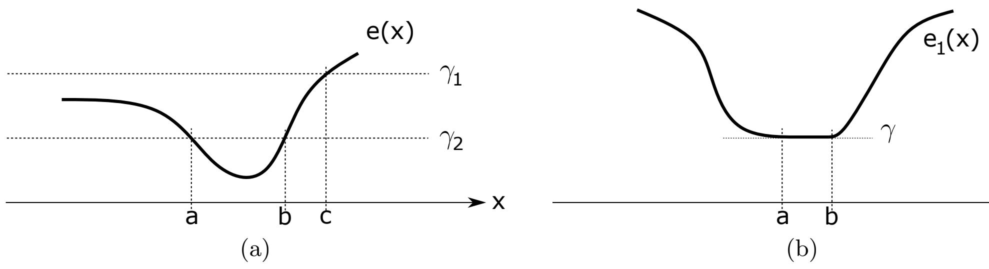


FIGURE 2.1: Panel (a): a quasiconvex function $e(x) : \mathbb{R} \to \mathbb{R}$. For each δ, the δ-sublevel set $\mathcal{S}_\delta$ is convex, i.e., a contiguous interval in the 1D case. The sublevel set $\mathcal{S}_{\delta_1}$ is the interval [a,b]. The sublevel set $\mathcal{S}_{\delta_2}$ is the interval $[-\infty$,c]. Panel (b): a quasiconvex function $e_1(x) : \mathbb{R} \to \mathbb{R}$ has a unique minimum γ, while any $x \in$ [a,b] is a global minimiser.

A quasiconvex function has a unique minimum (i.e., the globally minimal value), though its global minimisers can be multiple, as shown in Figure 2.1b. Another useful property of quasiconvexity is that the maximum of quasiconvex functions (i.e., $E(\mathbf{x}) = \max\{e_1(\mathbf{x}), ..., e_n(\mathbf{x})\}$) is also quasiconvex [3].

2.2 Algorithms for triangulation

2.2.1 Bisection method

Since each δ-sublevel set $\mathcal{S}_\delta$ of a quasiconvex function is convex, the quasiconvex optimisation problem (1.12) can be solved by a sequence of convex programming problems. To see that, (1.12) is firstly transformed into an equivalent formulation

$$\begin{aligned} \min_{\mathbf{X}\in\mathbb{R}^3,\delta\in\mathbb{R}} \quad & \delta \\ \text{s.t.} \quad & f_i(\mathbf{X}) \leq \delta, \\ & P_i^3\tilde{\mathbf{X}} > 0, \\ & \forall i \in \{1, \dots, N\}. \end{aligned} \tag{2.2}$$

The optimal δ^* in (2.2) can be found by solving the feasibility test

$$\begin{aligned} \text{find} \quad & \mathbf{X} \in \mathbb{R}^3, \\ \text{s.t.} \quad & f_i(\mathbf{X}) \leq \delta, \\ & P_i^3\tilde{\mathbf{X}} > 0, \\ & \forall i \in \{1, \dots, N\} \end{aligned} \tag{2.3}$$

with different values of δ in a bisection process. Figure 2.2 provides an illustration.

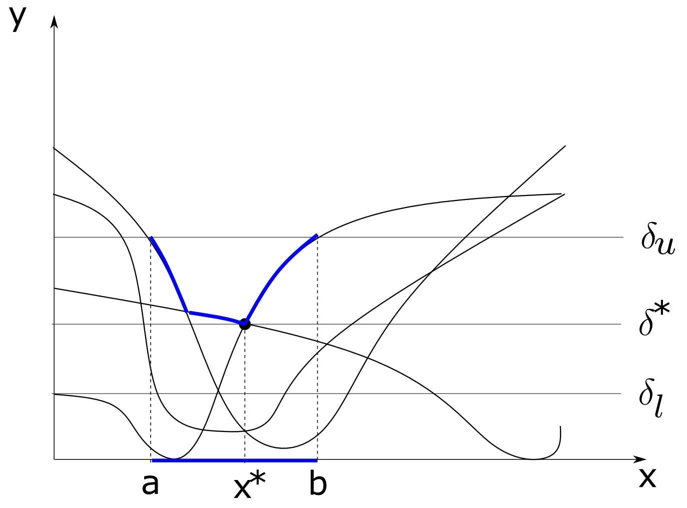


FIGURE 2.2: A demonstration of bisection to solve a 1D quasiconvex optimisation problem that minimises the maximum over four quasiconvex functions. The quasiconvex functions have unique minima, and their point-wise maximum is the upper-envelope, which is also quasiconvex and has a unique minimum achieved at x^*. δ^* is the optimal δ. A feasible $\delta = \delta_u$ produces a non-empty feasible region [a,b], $x^* \in$[a,b] (marked in blue on the x-axis), which indicates $\delta^* \leq \delta_u$, thus a smaller δ should be tested in the next iteration. On the other hand, a non-feasible $\delta = \delta_l$ results in an empty feasible region, which indicates $\delta^* > \delta_l$, thus a larger δ is tested in the next iteration. Every feasibility test narrows the range of feasible δ, and eventually, δ converges to δ^* and $\mathbf{x}$ converges to x^*.

As in Definition 2.1, the quasiconvexity of each $f_i(\mathbf{X})$ implies the feasible region of $\mathbf{X}$ in (2.2) for each δ (i.e., sublevel set $\mathcal{S}_\delta$) is convex, thus (2.3) (which finds the intersection of convex regions) is a convex feasibility problem. Actually, depending on the choice of q in $f_i(\mathbf{X})$ (see (1.9)), (2.3) is either a second order cone programming feasibility problem (for $q = 2$), or a linear programming feasibility problem (for $q = 1$ or $q = \infty$).

As outlined in Algorithm 2.1, the bisection algorithm maintains an interval $[\delta_l, \delta_u]$ that encloses the optimal δ^*, and, with each iteration, the updated δ is calculated as the middle value of δ_l and δ_u as in Step 2. Variable $\mathbf{X}^*$ and δ^* store the latest feasible $\mathbf{X}$ and δ; as the gap between δ_l and δ_u diminishes wish each iteration, $\mathbf{X}$ and δ converge to the optimal values.

2.2.2 Generalised fractional programming

Problem (2.2) is in the form of a generalised fractional programming (GFP) problem, as shown in the follows.

Algorithm 2.1 `BISECTION`: bisection method for triangulation in the form of (2.2).

```
Require: Input data $\{P_i, \mathbf{u}_i\}_{i=1}^{N}$, an initial interval $[\delta_l, \delta_u]$ containing the optimal $\delta^*$,
    and a tolerance $\epsilon$.
 1: while $\delta_u - \delta_l > \epsilon$ do
 2:     $\delta \leftarrow (\delta_u + \delta_l)/2$.
 3:     Feasibility test: solve (2.3) to get $\mathbf{X}$.
 4:     if feasible then
 5:         $\delta_u \leftarrow \delta, \mathbf{X}^* \leftarrow \mathbf{X}, \delta^* \leftarrow \delta$.
 6:     else
 7:         $\delta_l \leftarrow \delta$.
 8:     end if
 9: end while
10: return $\delta^*, \mathbf{X}^*$.
```

Taking the denominator of $f_i(\mathbf{X})$ out of the $\|\cdot\|_q$ operation in (1.9) yields

$$f_i(\mathbf{X}) = \left\| \mathbf{u}_i - \frac{P_i^{1:2}\tilde{\mathbf{X}}}{P_i^3\tilde{\mathbf{X}}} \right\|_q = \frac{\left\| \mathbf{u}_i P_i^3\tilde{\mathbf{X}} - P_i^{1:2}\tilde{\mathbf{X}} \right\|_q}{P_i^3\tilde{\mathbf{X}}} = \frac{g_i(\mathbf{X})}{h_i(\mathbf{X})}. \tag{2.4}$$

Then by (2.4), (2.2) can be rewritten as

$$\begin{aligned} \min_{\mathbf{X}\in\mathbb{R}^3, \delta\in\mathbb{R}} \quad & \delta \\ \text{s.t.} \quad & \frac{g_i(\mathbf{X})}{h_i(\mathbf{X})} \leq \delta, \\ & h_i(\mathbf{X}) > 0, \\ & \forall i \in \{1, \dots, N\}, \end{aligned} \tag{2.5}$$

which is in the form of a GFP problem [6].

Bisection method is a special case of GFP type algorithms. Similar to bisection, GFP type algorithms solve (2.5) by repeating feasibility test (2.3) as in Algorithm 2.1, however, different algorithms apply different strategies for updating δ and $\mathbf{X}$. Here, two other GFP type algorithms, i.e., Dinkelbach's method [19] and Gugat's method [26], are surveyed.

2.2.2.1 Dinkelbach's method

Although bisection method is simple to implement and analyse, it has two drawbacks. Firstly, the convergence rate is only linear [6], and secondly, the $\mathbf{X}$ obtained from the previous iteration does not contribute towards a more informed update of δ. In these two aspects, Dinkelbach's method [19] is superior.

The basic Dinkelbach's method is similar to bisection method except that, with each iteration, δ_u is updated based on the value of the last feasible $\mathbf{X}$:

$$\delta_u = \max_i \left\| \mathbf{u}_i - \frac{P_i^{1:2}\tilde{\mathbf{X}}}{P_i^3\tilde{\mathbf{X}}} \right\|_q . \tag{2.6}$$

Such a δ_u progresses to δ^* more rapidly than the δ_u maintained in bisection method. Although the provable convergence rate of the basic Dinkelbach's method is still linear, a few varieties of the basic version can achieve a better convergence rate. For example, the so called *scaled Dinkelbach's algorithm* and *dual Dinkelbach's algorithm* are proven to have super-linear convergence [6].

2.2.2.2 Gugat's method

Following a similar spirit to Dinkelbach's method, Gugat's method [26] employs a more advanced updating rule on δ. It is necessary for Gugat's method to alter the formulation of (2.3) as follows.

Each inequality constraint (i.e., $g_i(\mathbf{X})/h_i(\mathbf{X}) \leq \delta$, as opposed to the strict inequality constraint such as $h_i(\mathbf{X}) > 0$) can be developed into

$$g_i(\mathbf{X}) - \delta h_i(\mathbf{X}) \leq 0, \tag{2.7}$$

which can be further transformed into

$$g_i(\mathbf{X}) - \delta h_i(\mathbf{X}) \leq \omega, \tag{2.8}$$

where $\omega \in \mathbb{R}$ indicates the degree of violation of constraint (2.8). Then by (2.8), (2.3) is transformed into

$$\begin{aligned} \min_{\mathbf{X}\in\mathbb{R}^3,\omega\in\mathbb{R}} \quad & \omega \\ \text{s.t.} \quad & g_i(\mathbf{X}) - \delta h_i(\mathbf{X}) \leq \omega, \\ & h_i(\mathbf{X}) > 0, \\ & \forall i \in \{1, \dots, N\}, \end{aligned} \tag{Q^δ}$$

and (2.5) is feasible if the optimal ω of (Q^δ) is less than 0.

In each iteration of Gugat's method, δ is updated based on not only the last feasible $\mathbf{X}$ but also the solution $\boldsymbol{\lambda}$ of the dual problem of (Q^{δ})

$$\delta = max\{\ \delta_l,\ min\{\delta + \omega/\boldsymbol{\lambda}\mathbf{g}(\mathbf{X}),\ \delta_u\}\ \}, \tag{2.9}$$

where δ_1 and δ_u are calculated as in Dinkelbach's method, ω and $\boldsymbol{\lambda}$ are respectively the primal solution and dual solution of (Q^{δ}), and $\mathbf{g}(\mathbf{X})$ is a column vector with the i^{th} element to be the value of $g_i(\mathbf{X})$ in (Q^{δ}), which is evaluated on the last feasible $\mathbf{X}$.

The GFP type algorithms for triangulation are efficient for moderate input size. As input size increases, the efficiency of this type of algorithms can drop dramatically. That is because, in each iteration, the convex feasibility problem (Q^{δ}) (or (2.3)) can incur significant computational and memory overheads (although convex optimisation is tractable, in practice, running a convex solver on large number of constraints is computationally expensive).

2.2.3 Descent methods

To obviate the reliance on solving convex optimisation problems for feasibility test, researchers have devised descent type algorithms. The underlying principle of descent type algorithms is to iteratively find a descent direction $\mathit{\Delta}\mathbf{X}$ then update the solution $\mathbf{X}$ along $\mathit{\Delta}\mathbf{X}$ by a step size α

$$\mathbf{X} = \mathbf{X} + \mathit{\Delta}\mathbf{X} * \alpha. \tag{2.10}$$

Different descent type algorithms use different strategies for finding the descent direction and step size.

2.2.3.1 Polyhedron collapse method

The polyhedron collapse method proposed by Donne et al. [20] is a descent type method. Geometrically, the process of $\mathbf{X}$ converging to $\mathbf{X}^*$ with each iteration is analogous to a 3D polyhedron $\mathcal{H}$ collapsing to a 3D point (i.e., $\mathbf{X}^*$). To make the analogy more explicit, [20] first reformulates the i^{th} reprojection error (1.9) as

$$f_i(\mathbf{X}) = \left\| \mathbf{u}_i - \frac{P_i^{1:2}\tilde{\mathbf{X}}}{P_i^{3}\tilde{\mathbf{X}}} \right\|_q = \frac{\|\mathbf{A}_i\mathbf{X} + \mathbf{b}_i\|_q}{\mathbf{c}_i^T\mathbf{X} + d_i}, \tag{2.11}$$

$$\text{where}\ \ \mathbf{A}_i = \begin{bmatrix} \mathbf{a}_{i,1}^T \\ \mathbf{a}_{i,2}^T \end{bmatrix} \in \mathbb{R}^{2\times 3},\ \ \mathbf{b}_i = \begin{bmatrix} b_{i,1} \\ b_{i,2} \end{bmatrix} \in \mathbb{R}^2, \tag{2.12}$$

$\mathbf{c}_i \in \mathbb{R}^3$ and d_i are constants calculated from the data P_i and $\mathbf{u}_i$. By choosing $q = \infty$, (2.11) is rewritten into

$$f_i(\mathbf{X}) = \max\left(\frac{|\mathbf{a}_{i,1}^T\mathbf{X} + b_{i,1}|}{\mathbf{c}_i^T\mathbf{X} + d_i}, \frac{|\mathbf{a}_{i,2}^T\mathbf{X} + b_{i,2}|}{\mathbf{c}_i^T\mathbf{X} + d_i}\right), \tag{2.13}$$

which can be further developed into

$$\begin{aligned} f_i(\mathbf{X}) &= \max\left(\frac{\mathbf{a}_{i,1}^T\mathbf{X} + b_{i,1}}{\mathbf{c}_i^T\mathbf{X} + d_i}, \frac{-\mathbf{a}_{i,1}^T\mathbf{X} - b_{i,1}}{\mathbf{c}_i^T\mathbf{X} + d_i},\right. \\ &\qquad \left.\frac{\mathbf{a}_{i,2}^T\mathbf{X} + b_{i,2}}{\mathbf{c}_i^T\mathbf{X} + d_i}, \frac{-\mathbf{a}_{i,2}^T\mathbf{X} - b_{i,2}}{\mathbf{c}_i^T\mathbf{X} + d_i}\right) \qquad (2.14) \\ &= \max\left(f_i^1(\mathbf{X}), f_i^2(\mathbf{X}), f_i^3(\mathbf{X}), f_i^4(\mathbf{X})\right). \qquad (2.15) \end{aligned}$$

Each component of the reprojection errors

$$f_i^j(\mathbf{X}) = \frac{\mathbf{a}_{i,j}^T\mathbf{X} + b_{i,j}}{\mathbf{c}_i^T\mathbf{X} + d_i}, \qquad i \in \{1, \dots, N\},\ j \in \{1, 2, 3, 4\} \tag{2.16}$$

forms a constraint in another equivalent formulation of (2.5)

$$\begin{aligned} \min_{\mathbf{X}\in\mathbb{R}^3, \delta\in\mathbb{R}} \quad & \delta \\ \text{s.t.} \quad & f_i^j(\mathbf{X}) = \frac{\mathbf{a}_{i,j}^T\mathbf{X} + b_{i,j}}{\mathbf{c}_i^T\mathbf{X} + d_i} \le \delta, \\ & \mathbf{c}_i^T\mathbf{X} + d_i > 0, \\ & \forall i \in \{1, \dots, N\},\ j \in \{1, \dots, 4\} \end{aligned} \tag{2.17}$$

With fixed δ, each constraint in (2.17) essentially defines a half-space. All these half-spaces together construct a feasible region of $\mathbf{X}$ in 3D space, and this feasible region is denoted as the polyhedron $\mathcal{H}$. The optimal $\mathbf{X}^*$ must reside inside $\mathcal{H}$; and the current $\mathbf{X}$ is provably on an edge of $\mathcal{H}$.

In each iteration, a direction $\mathit{\Delta}\mathbf{X}$ starting from $\mathbf{X}$ is derived from just a small number (generally within 4) of *active constraints* (the constraints satisfying $f_i^j(\mathbf{X}) = \delta$) in (2.17), then $\mathcal{H}$ collapses along $\mathit{\Delta}\mathbf{X}$ by a step size α, which is computed by a line search process.

The polyhedron collapse method is one of the most efficient triangulation solvers, however, it is only applicable to the case where $q = \infty$ in (2.11). Moreover, in each iteration, both the descent direction $\mathit{\Delta}\mathbf{X}$ and the step size α produced by the algorithm are not optimal, thus the convergence rate of the method has not reached its full potential.

2.2.3.2 Relax method

The Relax method proposed by Dai et al. [17] is another descent type algorithm. By investigating the behaviour of Gugat's method, the authors discovered two hurdles that affect the performance of the method. Let $Q_t^{\delta_t}$ denote the optimisation task of solving (Q^δ) with $\delta = \delta_t$ in the t^{th} iteration of Gugat's method, and let $\mathbf{X}_t$ be the solution of $Q_t^{\delta_t}$, then these two hurdles are summarised as

1. **Hurdle 1**: in each iteration, the convex feasibility problem $Q_t^{\delta_t}$ of the current iteration t solves $\mathbf{X}_t$ from scratch, and the solution $\mathbf{X}_{t-1}$ obtained from the previous iteration is not fully utilised — in a sense that $\mathbf{X}_{t-1}$ is only used to guide updating δ_t, rather than used to initialise the current $Q_t^{\delta_t}$;

2. **Hurdle 2**: for each $Q_t^{\delta_t}$, starting from an initialisation $\mathbf{X}_t^0$, the solution $\mathbf{X}_t$ is solved via multiple Newton steps. However, it is observed that normally only the first Newton step makes the largest progress towards the $\mathbf{X}_t$. The following Newton steps not only make weak contribution towards optimality, but also are likely to push the solution to the boundary of the feasible region of $Q_t^{\delta_t}$.

For Hurdle 1, $\mathbf{X}_{t-1}$ obtained from $Q_{t-1}^{\delta_{t-1}}$ is not used to hot-start $Q_t^{\delta_t}$ because $\mathbf{X}_{t-1}$ is in general on the boundary of the feasible region of $Q_{t-1}^{\delta_{t-1}}$, such that it is generally not a feasible solution (or interior point) of $Q_t^{\delta_t}$.

The remedies are twofold. Firstly, the algorithm 'relaxes' δ_{t-1} in $Q_{t-1}^{\delta_{t-1}}$ mildly by replacing δ_{t-1} with a slightly larger value δ'_{t-1}, where $\delta'_{t-1} > \delta_{t-1}$, in the hope that by solving $Q_{t-1}^{\delta'_{t-1}}$, the resulting solution $\mathbf{X}'_{t-1}$ is more likely to be an interior point (feasible solution) of $Q_t^{\delta_t}$.

The second remedy, which solves both Hurdle 1 and Hurdle 2, is to terminate $Q_{t-1}^{\delta_{t-1}}$ after only one Newton step. This strategy not only significantly reduces the chance of pushing $\mathbf{X}_{t-1}$ out of the feasible region of $Q_t^{\delta_t}$, but also removes the runtime wasted on those insignificant Newton steps.

Thus, the core idea of Relax method is to perform only one Newton step on the relaxed version of (Q^δ) with each iteration, so it is essentially a descent type algorithm with the descent direction $\Delta\mathbf{X}$ and step size α derived from the standard Newton method.

Relax algorithm demonstrates better performance than the GFP type algorithms in terms of efficiency and scalability. However, the need for calculating Hessians for all the measurements (in the Newton step in every iteration) is a significant per-iteration cost.

2.2.4 Algorithms exploiting generalised linear programming properties

Problem (2.5) is a generalised linear programming (GLP) problem, and this has been extensively exploited by many modern algorithms.

2.2.4.1 GLP properties

It has been established that, the triangulation problem (e.g., (1.12)) belongs to a broader class of problems — GLP problems [8] if $q \geq 1$ in (1.9). Many algorithms exploit two particular GLP properties, which, stated in the context of triangulation, are as follows.

Property 1 (Monotonicity). Let $\mathcal{X} = \{1, ..., N\}$ index the set of data $\{P_i, \mathbf{u}_i\}_{i=1}^{N}$, then for any $\mathcal{C} \subseteq \mathcal{X}$,

$$\min_{\mathbf{X}} \max_{i \in \mathcal{C}} \; f_i(\mathbf{X}) \leq \min_{\mathbf{X}} \max_{i \in \mathcal{X}} \; f_i(\mathbf{X}) \tag{2.18}$$

given the appropriate cheirality constraints on both sides. □

Property 2 (Support set). Let $\mathcal{X} = \{1, ..., N\}$ index the set of data $\{P_i, \mathbf{u}_i\}_{i=1}^{N}$, and let $\mathbf{X}^*$ and δ^* respectively be the minimiser and minimised objective value of (1.12). There exists a subset $\mathcal{Q} \subseteq \mathcal{X}$ with $|\mathcal{Q}| \leq 4$, such that for any $\mathcal{C}$ that satisfies $\mathcal{Q} \subseteq \mathcal{C} \subseteq \mathcal{X}$, the following holds

$$\delta^* = \min_{\mathbf{X}} \max_{i \in \mathcal{Q}} \; f_i(\mathbf{X}) = \min_{\mathbf{X}} \max_{i \in \mathcal{C}} \; f_i(\mathbf{X}) = \min_{\mathbf{X}} \max_{i \in \mathcal{X}} \; f_i(\mathbf{X}) \tag{2.19}$$

given the appropriate cheirality constraints. In fact, the three problems in (2.19) have the same minimiser $\mathbf{X}^*$. Further,

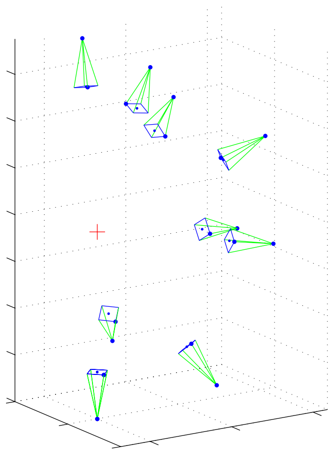
$$f_i(\mathbf{X}^*) = \delta^* \quad \text{for any } i \in \mathcal{Q}. \tag{2.20}$$

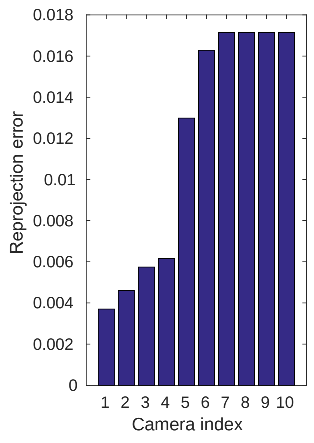
The subset $\mathcal{Q}$ is called the 'support set' of the problem. □

See [8] for details and proofs related to the above properties. Intuitively, (2.20) states that, at the minimiser of (1.12), the minimised maximum error occurs at the support set $\mathcal{Q}$ whose size is bounded by 4; Figure 2.3 illustrates. Further, (2.19) states that solving (1.12) amounts to solving the same problem on $\mathcal{Q}$.

2.2.4.2 QuickPseudo algorithm

QuickPseudo algorithm [36] applied Sharir and Welzl's algorithm [49] for GLP to triangulation. The algorithm aims to find the support set $\mathcal{Q}$. In the algorithm, $\mathcal{Q}$ is


(a)



(b)

Figure 2.3: Triangulating a point $\mathbf{X}$ observed in 10 views by solving (1.12) with $q = \infty$. The red '+' in panel (a) is the minimiser $\mathbf{X}^*$. The reprojection errors incurred by $\mathbf{X}^*$ on all images are shown in panel (b). Observe that there are at most 4 data with the same largest reprojection error at $\mathbf{X}^*$; these data are the support set $\mathcal{Q}$ of this triangulation instance.

initialised with an arbitrary input datum ('a datum' means a pair of input — $\{P_i, \mathbf{u}_i\}$ — that belongs to a particular camera). QuickPseudo method iterates through the data and checks for violations to the current support set $\mathcal{Q}$ (a violation occurs when the reprojection error f_i produced by the i^{th} datum is greater than the reprojection error produced by the current support set $\mathcal{Q}$). If the i^{th} datum violates the current $\mathcal{Q}$, a new support set is calculated from the data indexed by $\{\mathcal{Q} \cup i\}$. The algorithm terminates when none of the data violate the current $\mathcal{Q}$, which implies that $\mathcal{Q}$ is the support set of the whole input data. QuickPseudo method was proven to have sub-exponential runtime [37].

2.2.5 Other methods

Olsson et al. [43] introduced two globally convergent triangulation algorithms after establishing that triangulation problem was actually a pseudoconvex programming problem. Pseudoconvexity is a stronger condition than quasiconvexity in the sense that a pesudoconvex function has only one stationary point, and this implies that a pesudoconvex function has a unique global minimiser (as opposed to the existence of multiple global minimisers in a quasiconvex function, as seen in Figure 2.1b). The first algorithm proposed in [43] uses LOQO algorithm [55], an interior point algorithm for general non-convex problems, to solve $\mathbf{X}^*$ and δ^* jointly. The main contribution of Olsson et al. is an analytical proof of global convergence of the approach. The second algorithm is an interior point method that numerically solves the Karush-Kuhn-Tucker equations of (2.5), and this algorithm is proven to be equivalent to the scaled Dinkelbach's algorithm mentioned in Section 2.2.2.1 [6].

Eriksson and Isaksson [21] proposed a proximal splitting approach. Briefly, in each iteration, the proposed method performs a one-step bundle adjustment followed by a 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, often a conservative rate is required to produce correct results. Proximal splitting method demonstrates better efficiency than GFP type algorithms in practice.

2.3 Robust triangulation

As mentioned in 1.2.2.3, a major weakness of the ℓ_∞ formulation is being vulnerable to outliers. In the context of triangulation, problem (1.12) is essentially to fit the outliers because the maximum reprojection error is in general caused by the outliers (i.e., wrongly associated features). To deal with data that are contaminated by outliers, a robust technique usually needs to be applied.

Commonly used robust techniques include least median of squares (LMS) [46], maximum consensus [24], M-estimators [30], to name a few. M-estimator technique transforms the objective function into an intrinsically robust cost function (e.g., Huber function) to achieve robustness; the maximum consensus criterion aims to detect as many inliers as possible; while the goal of LMS is to minimise the median squared error. It is hard to conclude which technique is the 'best' because each technique has its own strength in specific respects.

2.3.1 LMS triangulation

This thesis focuses on the LMS criterion. In the context of triangulation, the LMS criterion entails solving

$$\begin{aligned} \min_{\mathbf{X}\in\mathbb{R}^3} \quad & \underset{i\in\{1,\dots,N\}}{\text{median}} \quad f_i(\mathbf{X}), \\ \text{s.t.} \quad & P_i^3\tilde{\mathbf{X}} > 0, \;\; \forall i \in \{1,\dots,N\}, \end{aligned} \tag{2.21}$$

i.e., minimising the median reprojection error. LMS provably has a breakdown point of 0.5, which means that it can tolerate up to 50% of outliers [46, Chap. 3]. However, taking the median removes the nice properties of the ℓ_∞ paradigm. Chiefly, the median of the reprojection errors is not quasiconvex, and problem (2.21) becomes intractable in general.

The non-differentiability of the median also complicates the usage of standard gradient-based optimization [41]. However, as we will see in Chapter 4, the quasiconvexity of the individual reprojection errors (i.e., (1.9)) can still be exploited for optimisation.

2.3.1.1 Random sampling method

A popular LMS algorithm is by random sampling [46]. In the context of triangulation, a random sampling type LMS algorithm is outlined in Algorithm 2.2. In each iteration of Algorithm 2.2, a minimal subset $\mathcal{E}$ (4 elements for triangulation) of the input $\mathcal{X} = \{1, ..., N\}$ is sampled, then an estimation of $\mathbf{X}$ is obtained by solving (1.12) on the subset $\mathcal{E}$.

Next, reprojection errors evaluated on $\mathbf{X}$ are calculated for the whole data $\mathcal{X}$, and the median reprojection error is obtained to evaluate this sample. This procedure of random-sampling-then-verifying is repeated for a number of times (estimated based on the highest expected outlier rate of 0.5) and the sample that produces the least median error is selected to compute the final solution $\mathbf{X}^*$.

This method is simple and generally efficient, however, it could become inefficient on large-scale datasets. Additionally, the randomized heuristic has no guarantee of optimality.

Algorithm 2.2 `RANDOMLMS`: a random sampling LMS method for triangulation.

```
Require: Input data $\{P_i, \mathbf{u}_i\}_{i=1}^N$, $\mathcal{X} = \{1, ..., N\}$ index the set of data, and $Rep$ the
    number of repetition.
 1: $med^* \leftarrow +\infty$.
 2: for $t = 1, ..., Rep$ do
 3:   $\mathcal{E} \leftarrow$ RANDOMSAMPLE$(\mathcal{X}, 4)$.
 4:   Solve (1.12) on $\mathcal{E}$ to get $\mathbf{X}$.
 5:   if $med^* > \operatorname{median}_{i \in \mathcal{X}} f_i(\mathbf{X})$ then
 6:     $med^* \leftarrow \operatorname{median}_{i \in \mathcal{X}} f_i(\mathbf{X})$.
 7:     $\mathbf{X}^* \leftarrow \mathbf{X}$.
 8:   end if
 9: end for
10: return $\mathbf{X}^*$.
```

2.3.1.2 Ke and Kanade's method

Ke and Kanade [33] used the bisection technique endowed with a *non-convex* feasibility test to solve general quasiconvex LMS problems, which includes (2.21). A brief survey of this method, in the context of (2.21), is as follows.

Let $F_m(\mathbf{X})$ denote the median reprojection error in problem (2.21), then (2.21) is rewritten to

$$\begin{aligned} \min_{\mathbf{X}\in\mathbb{R}^3,\theta\in\mathbb{R}} \quad & \theta \\ \text{s.t.} \quad & F_m(\mathbf{X}) \leq \theta, \\ & P_i^3\tilde{\mathbf{X}} > 0, \\ & \forall i \in \{1,\dots,N\}, \end{aligned} \tag{2.22}$$

which, if to be solved by bisection method, amounts to repeating the feasibility problem

$$\begin{aligned} \text{find} \quad & \mathbf{X} \in \mathbb{R}^3 \\ \text{s.t.} \quad & F_m(\mathbf{X}) \leq \theta, \\ & P_i^3\tilde{\mathbf{X}} > 0, \\ & \forall i \in \{1,\dots,N\} \end{aligned} \tag{2.23}$$

for different θ.

As established by the authors, solving the feasibility problem (2.23) is equivalent to solving the following integer programming problem

$$\begin{aligned} \min_{\mathbf{X}\in\mathbb{R}^3,\boldsymbol{\delta}\in\mathbb{R}^N} \quad & \delta_1 + \delta_2 + \cdots + \delta_N \\ \text{s.t.} \quad & g_i(\mathbf{X}) - \theta h_i(\mathbf{X}) \leq \delta_i, \\ & -h_i(\mathbf{X}) + \epsilon \leq \delta_i, \\ & \delta_i = \{0, v\}, \\ & \forall i \in \{1,\dots,N\}, \end{aligned} \tag{2.24}$$

where ϵ is a small positive number (to turn strict inequality constraints into inequality constraints), v is a large positive integer, $\boldsymbol{\delta}$ is the aggregation of δ_i, and the definitions of $g_i(\mathbf{X})$ and $h_i(\mathbf{X})$ follow (2.4).

Since integer programming is intractable, the proposed method solves a relaxed problem to (2.24):

$$\begin{aligned} \min_{\mathbf{X}\in\mathbb{R}^3,\boldsymbol{\delta}\in\mathbb{R}^N} \quad & \delta_1 + \delta_2 + \cdots + \delta_N \\ \text{s.t.} \quad & g_i(\mathbf{X}) - \theta h_i(\mathbf{X}) \leq \delta_i, \\ & -h_i(\mathbf{X}) + \epsilon \leq \delta_i, \\ & \forall i \in \{1,\dots,N\}. \end{aligned} \tag{2.25}$$

The proposed method finds an approximate solution of (2.22) by repeating the relaxed feasibility problem (2.25) in a bisection approach. This method can only converge to an approximate LMS solution without any certificate of optimality (either local or global).

2.3.1.3 Global method

On the other extreme, combinatorial search algorithms have been proposed to solve LMS exactly [53, 7]. For triangulation, Li [35] exploited the quasiconvexity of the reprojection error and devised a search algorithm that enumerates all local minima of the LMS problem. Despite the low-dimensionality of $\mathbf{X}$, the exact algorithms are computationally costly, and are practical only for small instances.

What is lacking in the field of LMS triangulation thus far is an algorithm that can provide some assurance of the quality of solutions while being much more tractable than prohibitively inefficient global methods.

2.3.2 An ℓ_∞ outlier-removal scheme

The outlier-removal scheme proposed by Sim and Hartley [50] is relevant to this thesis. The authors established that a support set for (1.12) contains at least one outlier if there is any, thus removing this support set from the input guarantees to reduce the number of outliers. The principle of this outlier-removal scheme is to repeatedly compute a support set of current input and update the input by removing the computed support set. The algorithm terminates when the maximum reprojection error computed from the current input is within a pre-defined threshold. Since an uncertain number of inliers and outliers are removed in each iteration, this algorithm does not guarantee optimality for the LMS criterion.

2.4 KRot

As described in Section 1.2.2 and Section 1.2.3, KRot and triangulation have many aspects in common; most prominently, both problems can be solved by quasiconvex programming and both are GLP problems. Thus, it is not surprising to see many triangulation algorithms surveyed above can also be used to solve KRot.

For example, the GFP type algorithms (e.g., bisection method, Dinkelbach's method, and Gugat's method) mentioned in Section 2.2.2 can apply to KRot seamlessly, although the algorithms demonstrate unsatisfactory efficiency and scalability due to significantly increased number of variables associated with KRot (see below). Some algorithms, such as Relax method [17] and the proximal splitting method [21], were actually designed as dual-purpose solvers for both triangulation and KRot. They are in general more efficient than the GFP type algorithms, however, they still lack the capacity of dealing with large-scale KRot.

KRot is in general much harder to solve than triangulation. Primarily, two features of KRot account for the difficulty. One of the features is that the input size in a KRot instance is much larger than in a triangulation instance. In a reconstruction task involving M 3D scene points and N cameras, the number of measurements is up to $M \times N$ (when every 3D point is observed by every camera's image thus incurs a measurement); thus, if the reconstruction task is to be solved as a KRot instance, the input size of the KRot instance is also up to $M \times N$. At the same time, if camera translations (i.e., $\mathbf{t}$) are already known and only the M scene points need to be estimated, the reconstruction can be solved by M triangulation instances and the input size of each triangulation instance is bounded by N. The sheer volume of input size of KRot could significantly compromise the efficiency of the aforementioned quasiconvex programming algorithms.

The other feature that makes KRot hard to solve is the large number of parameters to be estimated. The parameter space of each triangulation instance is only 3D, implying a support set of size at most 4, which underpins the success of the triangulation algorithms that exploit GLP properties (e.g., QuickPseudo method in Section 2.2.4.1 and Sim's outlier-removal scheme in Section 2.3.2). In contrast, the parameter space in a KRot instance with M scene points and N cameras is $3 \times (M + N)$-dimensional, as shown in (1.14). As a result, the size bound of a support set for such a KRot instance is $3 \times (M + N) + 1$, which makes GLP algorithms impractical.

2.5 Other background

Some theories established in this thesis are inspired by a few concepts from computational geometry such as MEB, coreset, and plane sweep. For the purpose of a background introduction, a short review of these concepts is provided in this section.

2.5.1 MEB

The MEB problem refers to the task of finding the centre of a ball with minimal radius that encloses a given set of points (see Figure 2.4). Formally, let $\mathcal{B}(\mathbf{m}, r)$ denote an enclosing ball with centre $\mathbf{m}$ and radius r, and $\mathcal{B}^*(\mathbf{m}^*, r^*)$ denote the MEB, then given a set $\mathcal{G} = \{\mathbf{g}_i \in \mathbb{R}^\rho\}_{i=1}^N$ of N points $\mathbf{g}_i$ in a ρ-dimensional space, the centre $\mathbf{m}^*$ and the radius r^* of the MEB $\mathcal{B}^*(\mathbf{m}^*, r^*)$ is defined as

$$\begin{aligned} \mathbf{m}^* &= \operatorname*{argmin}_{\mathbf{m}\in\mathbb{R}^\rho} \max_{\mathbf{g}_i\in\mathcal{G}} \|\mathbf{g}_i - \mathbf{m}\|_2; \\ r^* &= \max_{\mathbf{g}_i\in\mathcal{G}} \|\mathbf{g}_i - \mathbf{m}^*\|_2, \end{aligned} \tag{2.26}$$

Some other commonly seen terms for MEB include *smallest circle* when $\rho = 2$ and *smallest sphere* when $\rho = 3$.

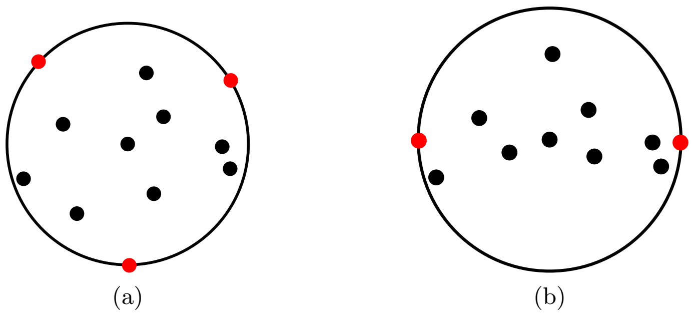


FIGURE 2.4: MEB in a ($\rho = 2$)-dimensional space. Panel (a) is of the general case where the size of the support set is $\rho + 1 = 3$, and panel (b) is of the degenerated case where the size of the support set is less than $\rho + 1$. In both panel, the set of the red dots represents the support set.

MEB has many important applications across a large landscape of research, thus remains to be an active research area. In the late 20th century, [56, 38] devised $O(n)$ complex algorithms for fixed dimension ρ. In 2003, [23] proposed a method that can deal with ρ up to $10,000$. There are also algorithms designed to find an approximate solution with bounded precision to the exact solution [9].

MEB problem also belongs to GLP problems, thus GLP properties (i.e. monotonicity, support set, etc.) also apply to MEB problem, as seen in Figure 2.4.

MEB related techniques are substantial components of this research, as seen in Chapter 3, Chapter 5 and Chapter 6.

2.5.2 Coreset

For a problem with input size too large to handle, or when an exact solution is unnecessary, an approximation solution with guaranteed accuracy becomes desirable. The term *(1 + ϵ)-approximation* refers to an approximate solution that is no 'worse' than the exact solution by a factor of ϵ. Being 'better' or 'worse' is subject to pre-defined metrics for measuring the quality of a solution [4]. For example, in the context of MEB problems, the radius r is usually used as a quality measurement. In which case, an approximate solution $\mathcal{B}'(\mathbf{m}', r')$ is qualified to be a $(1+\epsilon)$-approximation to the exact solution $\mathcal{B}^*(\mathbf{m}^*, r^*)$ if

$$r^* \leq r' \leq (1+\epsilon)r^*. \tag{2.27}$$

Associated with the concept of $(1+\epsilon)$-approximation is the concept of *coreset*. A coreset is a subset $\mathcal{Z}$ of the original input $\mathcal{X}$, $\mathcal{Z} \subseteq \mathcal{X}$, such that solving the same problem on $\mathcal{Z}$ yields a $(1+\epsilon)$-approximation of the solution obtained from solving the problem on $\mathcal{X}$.

Chapter 5 establishes the necessary mathematical underpinnings of a coreset algorithm for triangulation.

2.5.3 Plane sweep

Plane sweep is one of the key techniques in Computational Geometry. The central idea of a plane sweep algorithm is to use an imaginary line to sweep or move across the plane, stopping at some *event points*. Normally, the solution of the to-be-solved problem lies in one of the event points, and the solution is found once the imaginary line has passed over the plane. This family of algorithms can be adapted to solve many related computational geometric problems, such as finding intersecting polygons, finding the intersections of line segments [11], Delaunay triangulation [13], to name a few.

Amongst the sweeping type algorithms, the LMS line fitting algorithm developed by Souvaine and Steele [52] is mostly relevant to this thesis. The algorithm boils down to using a sweep line to pass through all the intersections of multiple linear functions (as illustrated in Fig. 2.5). Benefiting from the plane sweep technique, the algorithm solves the LMS line fitting problem in $O(n^2 log(n))$ time (where n is the number of linear functions). By comparison, a brute-force method solves the same problem in $O(n^3)$ time.

The algorithm proposed in Chapter 4 for LMS triangulation also benefits from employing the plane sweep technique to sweep multiple quasiconvex functions.

2.6 Summary of contributions

As surveyed in Section 2.2.2, the GFP type algorithms do not scale well to large input size. Nevertheless, if a large-scale triangulation instance can be broken into multiple smaller-scale sub-problems, GFP type algorithms are still useful as sub-problem solvers; this is demonstrated in Chapter 3. The meta-triangulation algorithm developed in Chapter 3 exploits the quasiconvexity of (1.12), and the algorithm efficiently finds

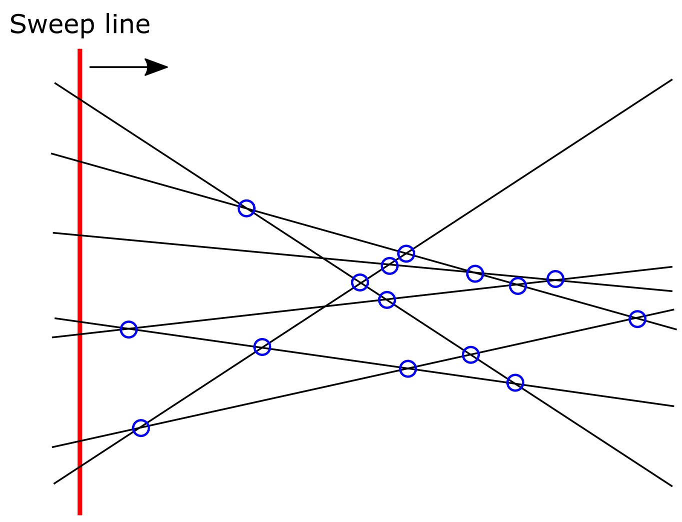


FIGURE 2.5: Plane sweep algorithm for the dual form of LMS line fitting — to find the intersection in a line arrangement that can cover half of the lines in the arrangement with minimal vertical distance. The red vertical line is the imaginary sweep line sweeping from the left to the right of the plane; the blue circles represent the intersections (i.e., event points) in this line arrangement; the algorithm finds the 'best' intersection by sweeping over all the intersections one by one.

the global optimum usually within 10 iterations regardless of the input size. Being a meta-algorithm, it relies on other triangulation solvers (e.g., the GFP type algorithms) to solve smaller-scale sub-problems, however, extensive experiments suggests it solves a triangulation instance more efficiently than invoking an underlying triangulation solver on the whole input data directly.[1]

The coreset algorithm proposed in Chapter 5 is also a powerful tool for large-scale triangulation. It produces a representative subset (i.e., coreset) of the whole input such that solving triangulation on this subset will produce a solution that is within known bound of the optimal solution on the whole input. The coreset algorithm is equipped with an 'any-time stopping' feature. Whenever the algorithm stops, a guaranteed approximate accuracy can be derived from how many iterations have been performed upon termination of the algorithm.

To deal with outliers, this thesis devises a novel locally optimal algorithm called Q-sweep that guarantees deterministic convergence to a local minimum for LMS triangulation. Q-sweep algorithm fills the gap between the methods with no optimality assurance and the globally optimal algorithms that are prohibitively inefficient. Experiments demonstrated in Chapter 1 shows that Q-sweep algorithm constantly yields better solutions than random sampling methods with comparable runtime and is much more practical than globally optimal methods.

[1]After this meta-algorithm was published, the authors realised it was similar to Seo and Hartley's work [48].

For KRot, in response to large input size and large number of to-be-estimated parameters, Chapter 6 devises a novel resection-intersection (Res-Int) KRot solver that breaks optimisation problem (1.14) into multiple sub-problems which are then efficiently solved by a novel fast descent method (FDM). Empirically, the Res-Int algorithm exhibits superior performance to the state-of-the-art algorithms, and can scale up to large input data that easily overwhelm existing algorithms. Moreover, the fast descent method (FDM) by its own becomes a state-of-the-art triangulation solver.

Chapter 3

An Efficient Meta-Algorithm for Triangulation

The work contained in this chapter has been published as the following paper

Qianggong Zhang and Tat-Jun Chin: An Effcient Meta-Algorithm for Triangulation. Asian Conference on Computer Vision (ACCV) Workshops, 148-161.

The published paper is available at
https://link.springer.com/chapter/10.1007/978-3-319-54427-4_12

After publication, the authors realised this method was similar to Seo and Hartley's work. [48]

Statement of Authorship

Title of Paper	An Efficient Meta-Algorithm for Triangulation
Publication Status	[✓ Published] Accepted for Publication Submitted for Publication Unpublished and Unsubmitted w ork w ritten in manuscript style
Publication Details	Zhang, Q., & Chin, T. J. (2016, November). An Efficient Meta-Algorithm for Triangulation. In Asian Conference on Computer Vision (pp. 148-161). Springer, Cham.

Principal Author

Name of Principal Author (Candidate)	Qianggong Zhang		
Contribution to the Paper	Performed experiments, interpreted data, wrote manuscript and acted as corresponding author		
Overall percentage (%)	60%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	22/11/2017

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Tat-Jun Chin		
Contribution to the Paper	Supervised development of work and revised manuscript.		
Signature		Date	22/11/2017

An Efficient Meta-Algorithm for Triangulation

Qianggong Zhang(✉) and Tat-Jun Chin

The University of Adelaide, Adelaide, SA 5000, Australia
qianggong.zhang@adelaide.edu.au

Abstract. Triangulation by ℓ_∞ minimisation has become established in computer vision. State-of-the-art ℓ_∞ triangulation algorithms exploit the quasiconvexity of the cost function to derive iterative update rules that deliver the global minimum. Such algorithms, however, can be computationally costly for large problem instances that contain many image measurements. In this paper, we exploit the fact that ℓ_∞ triangulation is an instance of generalised linear programs (GLP) to speed up the optimisation. Specifically, the solution of GLPs can be obtained as the solution on a small subset of the data called the *support set*. A meta-algorithm is then constructed to efficiently find the support set of a set of image measurements for triangulation. We demonstrate that, on practical datasets, using the meta-algorithm in conjunction with all existing ℓ_∞ triangulation solvers provides faster convergence than directly executing the triangulation routines on the full set of measurements.

1 Introduction

Triangulation refers to the task of estimating the 3D coordinates of a scene point from multiple 2D image observations of the point, given that the pose of the cameras are known. Triangulation is fundamentally important to 3D vision, since it enables the recovery of the 3D structure of a scene. Whilst in theory structure and motion must be obtained simultaneously, there are many settings, such as large scale reconstruction [10,24] and SLAM [17], where the camera poses are first estimated with a sparse set of 3D points, before a denser scene structure is produced by triangulating other points using the estimated camera poses.

An established approach for triangulation is by ℓ_∞ minimisation [12]. Specifically, we seek the 3D coordinates that minimise the maximum reprojection error across all views. Unlike the sum of squared error function which contains multiple local minima, the maximum reprojection error function is quasiconvex and thus contains a single global minimum. Algorithms that take advantage of this property have been developed to solve such quasiconvex problems exactly [2,5,7,9,13,14,19]. In particular, Agarwal et al. [2] showed that some of the most effective algorithms belong to the class of generalised fractional programming methods [6,11].

Although algorithms for ℓ_∞ triangulation have steadily improved, there is still room for improvement. In particular, on large scale reconstruction problems or SLAM where there are usually a significant number of views per point (recall

© Springer International Publishing AG 2017
C.-S. Chen et al. (Eds.): ACCV 2016 Workshops, Part II, LNCS 10117, pp. 148–161, 2017.
DOI: 10.1007/978-3-319-54427-4_12

that the size of a triangulation problem is the number of 2D observations of a scene point), the computational cost of many of the algorithms [2] can be considerable; we will demonstrate this in Sect. 4. A major reason is that the algorithms need to repeatedly solve convex programs (LPs or SOCPs) constructed using *all the measurements* to determine the update direction. Although theoretically convex programs can be solved efficiently, the runtime of many existing solvers are still non-trivial given large input sizes. It is thus of significant practical interest to develop faster algorithms.

Contributions. We propose the usage of a meta-algorithm that works in conjunction with existing ℓ_∞ solvers to achieve fast triangulation. The meta-algorithm exploits the fact that ℓ_∞ triangulation is a GLP. Specifically, the solution of a GLP given a set of input data can be obtained as the solution of the same problem on a small subset of the data called the *support set*. The purpose of the meta-algorithm is to efficiently find the support set. In contrast to existing ℓ_∞ solvers, the meta-algorithm does not require to solve convex problems on the full data simultaneously. This property is desirable on large scale problems. We demonstrate the benefits of the meta-algorithm on publicly available large scale 3D reconstruction datasets.

2 Background

Let $\{\mathbf{P}_i, \mathbf{u}_i\}_{i=1}^N$ be a set of data for triangulation, consisting of camera matrices $\mathbf{P}_i \in \mathbb{R}^{3\times4}$ and observed image positions $\mathbf{u}_i \in \mathbb{R}^2$ of the same scene point $\mathbf{x} \in \mathbb{R}^3$. In this paper, by a "datum" we mean a specific camera and image point $\{\mathbf{P}_i, \mathbf{u}_i\}$, and let $\mathcal{X} = \{1, \ldots, N\}$ index the set of data. The ℓ_∞ technique estimates $\mathbf{x}$ by minimising the maximum reprojection error

$$\min_{\mathbf{x}} \max_{i \in \mathcal{X}} \quad r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i), \quad \text{where } r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i) = \left\| \mathbf{u}_i - \frac{\mathbf{P}_i^{1:2}\tilde{\mathbf{x}}}{\mathbf{P}_i^3\tilde{\mathbf{x}}} \right\|_p, \tag{1}$$
$$\text{subject to} \quad \mathbf{P}_i^3\tilde{\mathbf{x}} > 0, \quad \forall i \in \mathcal{X}$$

Here, $\mathbf{P}_i^{1:2}$ and $\mathbf{P}_i^3$ respectively denote the first-2 rows and 3rd row of $\mathbf{P}_i$, and $\tilde{\mathbf{x}}$ is $\mathbf{x}$ in homogeneous coordinates. The chirality constraints $\mathbf{P}_i^3\tilde{\mathbf{x}} > 0$ ensure that the estimated point lies in front of all the cameras. The reprojection error

$$r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i) = \left\| \mathbf{u}_i - \frac{\mathbf{P}_i^{1:2}\tilde{\mathbf{x}}}{\mathbf{P}_i^3\tilde{\mathbf{x}}} \right\|_p = \frac{\left\| \left(\mathbf{u}_i\mathbf{P}_i^3 - \mathbf{P}_i^{1:2} \right) \tilde{\mathbf{x}} \right\|_p}{\mathbf{P}_i^3\tilde{\mathbf{x}}} \tag{2}$$

is basically the distance between the observed point $\mathbf{u}_i$ and the projection of $\mathbf{x}$ onto the i-th image plane. We have left p in the reprojection error undefined, since it remains a "design choice"; the optimisation problem above, therefore, is dubbed (ℓ_∞, ℓ_p) to reflect this choice. Typical values of p are 1, 2 and ∞.

The (ℓ_∞, ℓ_p) triangulation problem can be re-expressed as

$$\begin{aligned} \min_{\mathbf{x}} \quad & \delta \\ \text{subject to} \quad & \frac{\left\| \left(\mathbf{u}_i \mathbf{P}_i^3 - \mathbf{P}_i^{1:2} \right) \tilde{\mathbf{x}} \right\|_p}{\mathbf{P}_i^3 \tilde{\mathbf{x}}} \leq \delta, \\ & \mathbf{P}_i^3 \tilde{\mathbf{x}} > 0, \quad \forall i \in \mathcal{X} \end{aligned} \tag{3}$$

where the optimal δ^* is precisely the minimised maximum reprojection error.

2.1 Bisection and Generalised Fractional Programs

Although Eq. (2) is not convex, it is quasiconvex in the region $\mathbf{P}_i^3 \tilde{\mathbf{x}} > 0$. Since the objective function in Eq. (1) is the maximum of a set of quasiconvex functions, it is also quasiconvex. This implies that for any *fixed* (and non-negative) δ, the following feasibility problem is convex for $p \geq 1$:

$$\begin{aligned} \text{does there exist} \quad & \mathbf{x} \\ \text{such that} \quad & \frac{\left\| \left(\mathbf{u}_i \mathbf{P}_i^3 - \mathbf{P}_i^{1:2} \right) \tilde{\mathbf{x}} \right\|_p}{\mathbf{P}_i^3 \tilde{\mathbf{x}}} \leq \delta, \\ & \mathbf{P}_i^3 \tilde{\mathbf{x}} > 0, \quad \forall i \in \mathcal{X} \end{aligned} \tag{4}$$

The method of bisection exploits the convexity of Eq. (4) to efficiently solve Eq. (3) [13]. Specifically, a binary search is conducted to find the minimum δ. In each step, the feasibility test Eq. (4) (that involves all of the data) is conducted.

Bisection can be viewed as an instance of generalised fractional programming (GFP) [2]. More efficient GFP algorithms exist, such as Dinkelbach's method [6] and Gugat's method [11]. However, all these methods share the similarity that a convex problem involving all of the data must be solved at each iteration. This represents a significant computational bottleneck for large problems [7].

2.2 Generalised Linear Programs

For $p \geq 1$, (ℓ_∞, ℓ_p) problem also belongs to a broader class of problems called generalised linear programs (GLP) [3]. Two useful properties of GLPs, stated in the context of (ℓ_∞, ℓ_p), are as follows.

Property 1 (Monotonicity). For any $\mathcal{C} \subseteq \mathcal{X}$,

$$\min_{\mathbf{x}} \max_{i \in \mathcal{C}} \; r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i) \leq \min_{\mathbf{x}} \max_{i \in \mathcal{X}} \; r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i) \tag{5}$$

given the appropriate chirality constraints on both sides. □

Property 2 (Support set). Let $\mathbf{x}^*$ and δ^* respectively be the minimiser and minimised objective value of (ℓ_∞, ℓ_p). There exists a subset $\mathcal{B} \subseteq \mathcal{X}$ with $|\mathcal{B}| \leq 4$, such that for any $\mathcal{C}$ that satisfies $\mathcal{B} \subseteq \mathcal{C} \subseteq \mathcal{X}$, the following holds

$$\delta^* = \min_{\mathbf{x}} \max_{i \in \mathcal{B}} \; r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i) = \min_{\mathbf{x}} \max_{i \in \mathcal{C}} \; r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i) = \min_{\mathbf{x}} \max_{i \in \mathcal{X}} \; r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i) \tag{6}$$

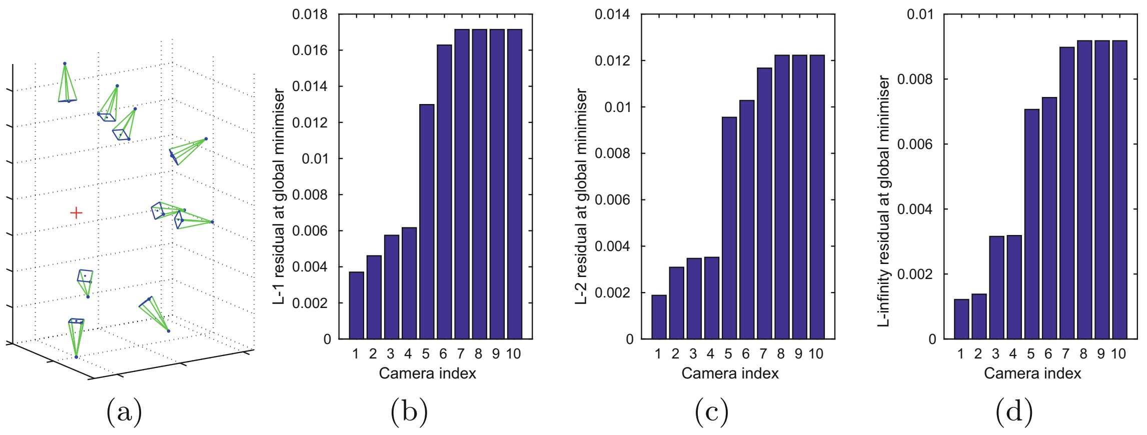


Fig. 1. Triangulating a point $\mathbf{x}$ observed in 10 views by solving (ℓ_∞, ℓ_p). The red '+' in panel (a) is the ℓ_∞ solution $\mathbf{x}^*$ for $p = 2$ in the reprojection error. The reprojection error of all the data at the solution $\mathbf{x}^*$ for respectively $p = 1$, 2 and ∞ are shown in panels (b), (c) and (d). Observe that there are at most 4 data with the same residual at $\mathbf{x}^*$; these data are the support set $\mathcal{B}$ of the respective problems. (Color figure online)

given the appropriate chirality contraints. In fact, the three problems in Eq. (6) have the same minimiser $\mathbf{x}^*$. Further,

$$r(\mathbf{x}^* \mid \mathbf{P}_i, \mathbf{u}_i) = \delta^* \quad \text{for any } i \in \mathcal{B} \tag{7}$$

The subset $\mathcal{B}$ is called the "support set" of the problem. □

See [3,15,23] for details and proofs related to the above properties. Intuitively, Eq. (7) states that, at the solution of (ℓ_∞, ℓ_p), the minimised maximum error occurs at the support set $\mathcal{B}$; Fig. 1 illustrates. Further, Eq. (6) states that solving (ℓ_∞, ℓ_p) amounts to solving the same problem on $\mathcal{B}$. Many classical algorithms in computational geometry [4,16,20,22] exploit this property to solve GLPs.

One of the most basic algorithms is due to Sharir and Welzl [22]. Li [15] more recently applied this algorithm to triangulation. Algorithm 1 summarises the method, in the context of (ℓ_∞, ℓ_p).

The algorithm finds the support set $\mathcal{B}$ by iterating through the data, and checking for violations to the current $\mathcal{B}$ (Step 8). If a datum i violates the current $\mathcal{B}$, a new support set is calculated from the data indexed by $\mathcal{B}$ and i (Steps 3 and 10), which implicitly also obtains the current estimate $\mathbf{x}^*$ and objective value δ^* from problem based on data $\mathcal{B}$ (Steps 4 and 11). Support set updating is conducted using a *primitive solver* - we refer the reader to [22] for details. The algorithm terminates when none of the data violate the current $\mathcal{B}$, which implies that $\mathcal{B}$ is the support set of the whole input data.

Matoušek et al. [16] proved that the algorithm of Sharir and Welzl [22] has sub-exponential runtime. Observe that, unlike the GFP algorithms (Sect. 2.1), each update iteration of Algorithm 1 involves only small subset of the data ($\mathcal{B} \cup \{i\}$). However, as we will show in Sect. 4, the runtime of Algorithm 1 can still be significant on large scale 3D reconstruction datasets.

Algorithm 1. Subexponential-time algorithm [22] for solving (ℓ_∞, ℓ_p).

```
Require: Input data $\{\mathbf{P}_i, \mathbf{u}_i\}_{i=1}^N$.
 1: Randomly permute the order of $\{\mathbf{P}_i, \mathbf{u}_i\}_{i=1}^N$, and define $\mathcal{X} = \{1, \ldots, N\}$.
 2: $\mathcal{B} \leftarrow \{1, 2, 3, 4\}$.
 3: $\mathcal{B} \leftarrow$ Support set of data indexed by $\mathcal{B} \cup \{i\}$.
 4: $(\mathbf{x}^*, \delta^*) \leftarrow$ Solution of Eq. (3) on data indexed by $\mathcal{B}$.
 5: while true do
 6:   $v \leftarrow 0$.
 7:   for $i = 1, \ldots, N$ do
 8:     if $r(\mathbf{x}^* \mid \mathbf{P}_i, \mathbf{u}_i) > \delta^*$ then
 9:       $v \leftarrow v + 1$.
10:       $\mathcal{B} \leftarrow$ Support set of data indexed by $\mathcal{B} \cup \{i\}$.
11:       $(\mathbf{x}^*, \delta^*) \leftarrow$ Solution of Eq. (3) on data indexed by $\mathcal{B}$.
12:       Optionally rearrange $\mathcal{X}$ by moving $i$ to the first position.
13:     end if
14:   end for
15:   if $v = 0$ then
16:     Break.
17:   end if
18: end while
19: return $\mathbf{x}^*$ and $\delta^*$.
```

3 Meta-Algorithm for ℓ_∞ Triangulation

The primary source of inefficiency in Algorithm 1 is that multiple passes over all N of the data are usually required. This is due to the fact that the algorithm keeps track of only at most 4 of the data at once (i.e., $\mathcal{B}$). Thus, a large number of updates are required before convergence. Secondly, the choice of the datum i for updating $\mathcal{B}$ is determined randomly (via the random permutation of $\mathcal{X}$ during initialisation). To hasten convergence, the selection should be made more strategically. Figure 2(a) provides an analogy of each iteration of Algorithm 1.

We propose Algorithm 2 as a more efficient technique for ℓ_∞ triangulation [21]. First, Algorithm 2 is a meta-algorithm, since it requires an underlying (ℓ_∞, ℓ_p) solver to carry out the updates (see Steps 3 and 10). Any of the previous algorithms [2,5,7,9,13,14,19] can be embedded. Thus, although Algorithm 2 appears simple, a lot of complexity due to solving (ℓ_∞, ℓ_p) has been abstracted away (in this sense, Algorithm 1 is also a meta-algorithm, since there remains the primitive solver routine whose specification is independent of the main structure).

Conceptually, instead of explicitly finding the support set $\mathcal{B}$ directly, Algorithm 2 seeks a representative subset of the data $\mathcal{C}$, which is no smaller than 4. The algorithm incrementally expands $\mathcal{C}$, by choosing the data q that most violates the current solution $\mathbf{x}^*$. This helps to accelerate convergence, since the "radius" of $\mathcal{C}$ (i.e., the minimised maximum reprojection error of the data in $\mathcal{C}$) is expanded quickly; see Fig. 2(b). Contrast this to Algorithm 1, whose selection of the "pivot" datum i is achieved effectively by random selection.

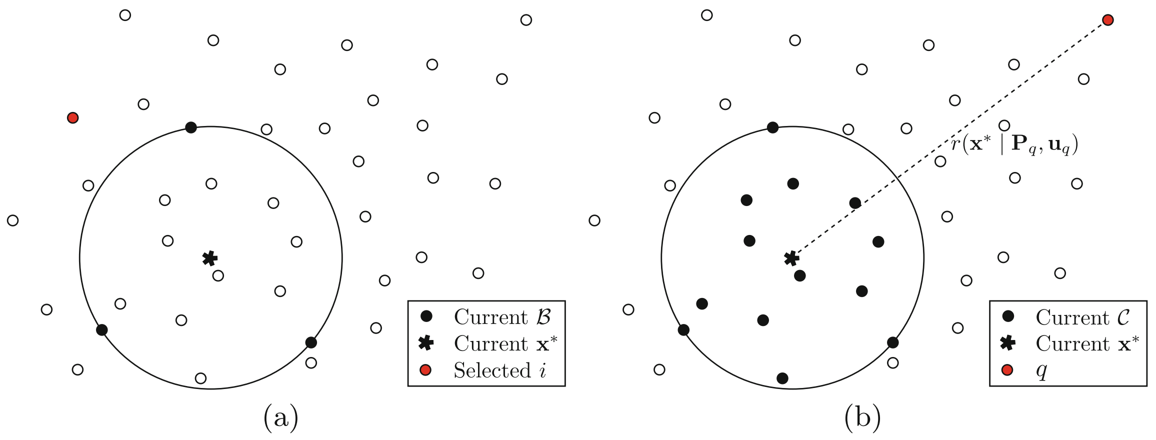


Fig. 2. Illustration of conceptual difference between Algorithms 1 and 2 on the minimum bounding circle problem, which is also a GLP and thus analogous to ℓ_∞ triangulation. (a) Algorithm 1 seeks the support set $\mathcal{B}$ of the data, and in each iteration, $\mathcal{B}$ is updated using the primitive solver [22] from the current $\mathcal{B}$ and a (randomly) selected datum i that violates $\mathcal{B}$. (b) Algorithm 2 seeks a representative subset $\mathcal{C}$ of the data, and in each iteration, $\mathcal{C}$ is updated using an (ℓ_∞, ℓ_p) solver [2,5,7,9,13,14,19] from the current $\mathcal{C}$ and the most violating datum q.

Algorithm 2 terminates when $\mathcal{C}$ equals $\mathcal{X}$, or when $\mathcal{C}$ contains the support set $\mathcal{B}$ of the whole input data. A formal statement is provided as follows.

Theorem 1. *Algorithm 2 finds* $\mathbf{x}^*$ *in at most* N *iterations.*

Proof. Given the current $\mathcal{C}$ with solution $\mathbf{x}^*$ and value δ^*, let q be obtained according to Step 5 in Algorithm 2.

- If $q \in \mathcal{C}$, then, by how $\mathbf{x}^*$ and δ^* were calculated in Step 10, the condition in Step 6 must be satisfied and $\mathbf{x}^*$ is the global minimiser.
- If $q \notin \mathcal{C}$ and the condition in Step 6 is satisfied, then Eq. (6) is implied and $\mathbf{x}^*$ is the global minimiser.
- If $q \notin \mathcal{C}$ and the condition in Step 6 is not satisfied, then Algorithm 2 will insert q into $\mathcal{C}$. There are at most N of such insertions (including the initial four insertions in the initialisation). In the worst case all of $\mathcal{X}$ will finally be inserted, and $\mathcal{C}$ converges to $\mathcal{X}$. $\square$

Clearly, for Algorithm 2 to be more efficient than executing an (ℓ_∞, ℓ_p) solver directly on the full input data, $\mathcal{C}$ must be expanded in a way that incorporates $\mathcal{B}$ into $\mathcal{C}$ quickly. This enables $\mathcal{C}$ and the number of iterations to be small, and, equally importantly, the embedded (ℓ_∞, ℓ_p) solver need only be invoked on small subsets $\mathcal{C}$ of $\mathcal{X}$. Section 4 demonstrates that, in practice, Algorithm 2 can in fact find the global minimiser much more efficiently than invoking an (ℓ_∞, ℓ_p) solver [2,5,7,9,13,14,19] in "batch mode" on the whole data.

3.1 Connections to Coreset Method

Algorithm 2 is in fact an instance of a coreset algorithm [1] (the representative subset $\mathcal{C}$ can be interpreted as a coreset of $\mathcal{X}$). Coreset methods are applied

Algorithm 2. Meta-algorithm for solving (ℓ_∞, ℓ_p).

```
Require: Input data $\{\mathbf{P}_i, \mathbf{u}_i\}_{i=1}^N$.
 1: Randomly permute the order of $\{\mathbf{P}_i, \mathbf{u}_i\}_{i=1}^N$, and define $\mathcal{X} = \{1, \ldots, N\}$.
 2: $\mathcal{C} \leftarrow \{1, 2, 3, 4\}$.
 3: $(\mathbf{x}^*, \delta^*) \leftarrow$ Solution of Eq. (3) on data indexed by $\mathcal{C}$.
 4: for $t = 1, \ldots, N - 4$ do
 5:     $q \leftarrow arg\,max_{i \in \mathcal{X}} r(\mathbf{x}^* \mid \mathbf{P}_i, \mathbf{u}_i)$.
 6:     if $r(\mathbf{x}^* \mid \mathbf{P}_q, \mathbf{u}_q) \leq \delta^*$ then
 7:         Exit for loop.
 8:     end if
 9:     $\mathcal{C} \leftarrow \mathcal{C} \cup \{q\}$.
10:     $(\mathbf{x}^*, \delta^*) \leftarrow$ Solution of Eq. (3) on data indexed by $\mathcal{C}$.
11: end for
12: return $\mathbf{x}^*$ and $\delta^*$.
```

frequently in discrete geometry to obtain ϵ-approximation solutions for *extent* problems, such as minimum enclosing balls. However, the bounds arising from current theoretical results are usually too loose to be of practical use (e.g., to obtain a small approximation error ϵ, the maximum number of iterations predicted is often far larger than available data in real-life problems).

Nonetheless, Algorithm 2 remains a very efficient meta-algorithm for (ℓ_∞, ℓ_p), as we will demonstrate in the next section.

4 Experiments

We conducted experiments to investigate the performance of Algorithm 2 as a global minimiser to the ℓ_∞ triangulation problem. We used a standard machine with 3.2 GHz processor and 16 GB main memory.

4.1 Datasets and Initialisation

We tested on publicly available datasets for large scale 3D reconstruction, namely, Vercingetorix Statue, Stockholm City Hall, Arc of Triumph, Alcatraz, Örebro Castle [8,18], and Notre Dame [24]. The *a priori* estimated camera poses and intrinsics supplied with these datasets were used to derive camera matrices. For triangulation, the size of an instance is the number of observations of the target 3D point. To avoid excessive runtimes, we randomly sampled 10% of the scene points in each dataset - this reduces the number of problem instances, but not the size of each of the selected instances. Figure 3 illustrates the distribution of problem sizes in each of the datasets used.

As shown in the respective pseudo-codes, Algorithms 1 and 2 were initialised by randomly choosing four data to instantiate $\mathbf{x}^*$ by solving (ℓ_∞, ℓ_p).

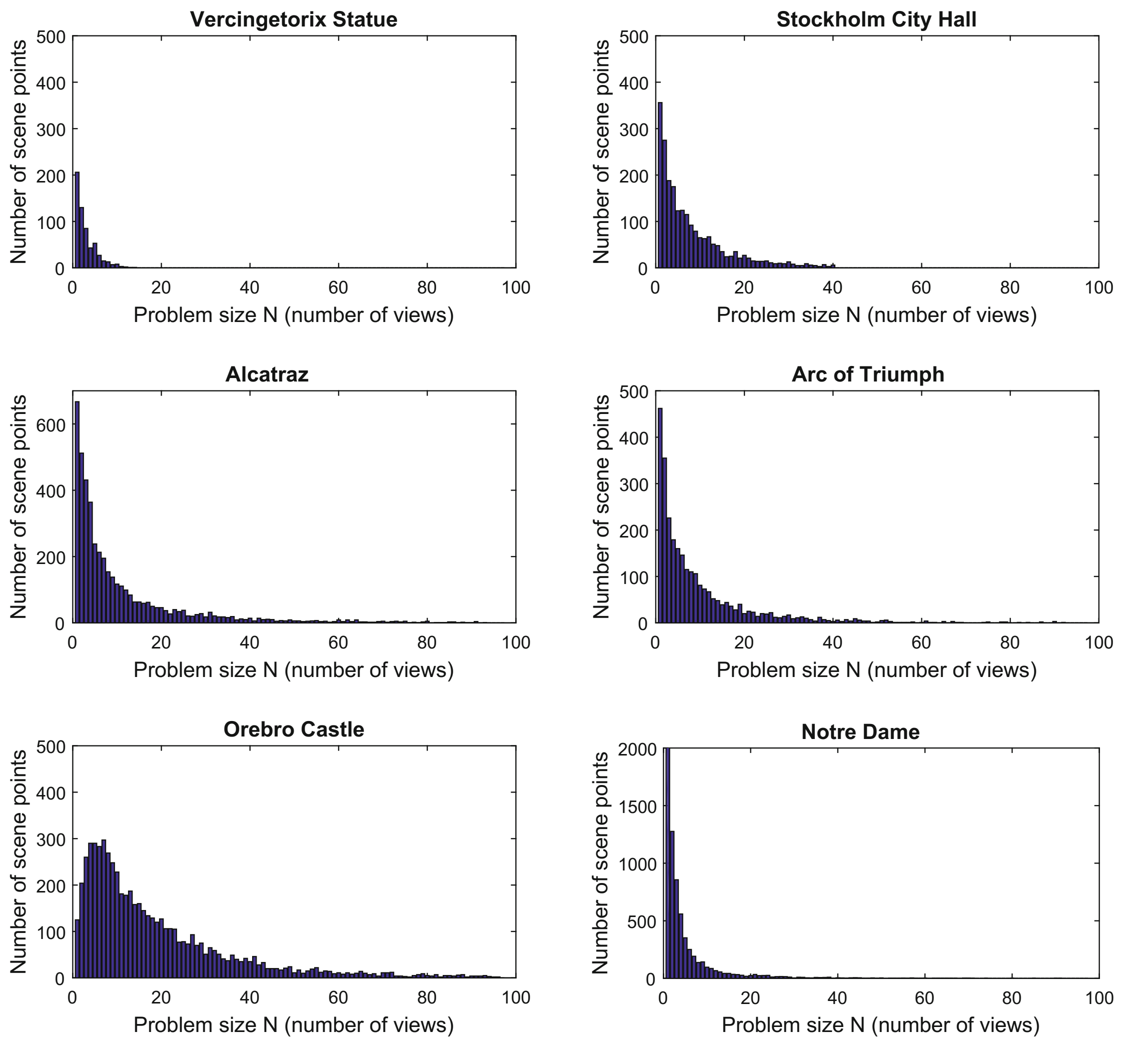


Fig. 3. Histograms of problem size.

4.2 Comparing Algorithm 1 and Algorithm 2

As mentioned above, Algorithms 1 and 2 can both be regarded as meta-algorithms, in that they require an embedded solver (the primitive solver [22] for Algorithms 1 and an (ℓ_∞, ℓ_p) solver for Algorithm 2), and the main structure of the algorithm is independent from the specification of the embedded solvers. Further, the solvers are only executed on a subset of the data in each update. In this section, we compare Algorithms 1 and 2 in terms of the number of updates (i.e., the number of calls to the embedded solver) required before convergence.

Synthetic Data. First, we generated synthetic data of varying sizes N (recall that the size of a triangulation problem is the number of views/measurements of the same 3D point). The camera centres were distributed uniformly around the 3D point, with the camera orientation pointing roughly towards the 3D point; Fig. 1 illustrates an instance of the synthetically generated data with $N = 10$. The observed point coordinates in each view were obtained by perturbing the true point with Normal noise of standard deviation 3 pixels. For brevity, we considered only the (ℓ_∞, ℓ_2) version in this experiment. The method of bisection was used as the embedded solver for Algorithm 2.

Columns 2 and 3 in Table 1 show the number of updates for $N = 10$, 100, $1k$, $10k$, $20k$ and $50k$, and Fig. 4 plots the results in a graph (number of updates versus N). Both algorithms performed only a very small number of updates compared to the actual size of the data before convergence. However, it is also clear that Algorithm 2 required far fewer updates than Algorithm 1. The growth of the number of updates for both methods also slowed down as N increased. In particular, the number of updates conducted by Algorithm 2 remained at 5 after $N = 1k$. This illustrates the ability of Algorithm 2 to quickly find and incorporate the support set $\mathcal{B}$ into $\mathcal{C}$.

Table 1. Comparison of number of updates and cumulative subproblem size between Algorithm 1 and Algorithm 2 on synthetically generated data of varying size N.

Data size N	Number of updates		Cumul. subproblem size	
	Algorithm 1	Algorithm 2	Algorithm 1	Algorithm 2
10	8	3	40	22
100	15	4	75	30
1000	35	5	175	39
10000	48	5	240	39
20000	62	5	310	39
50000	76	5	380	39

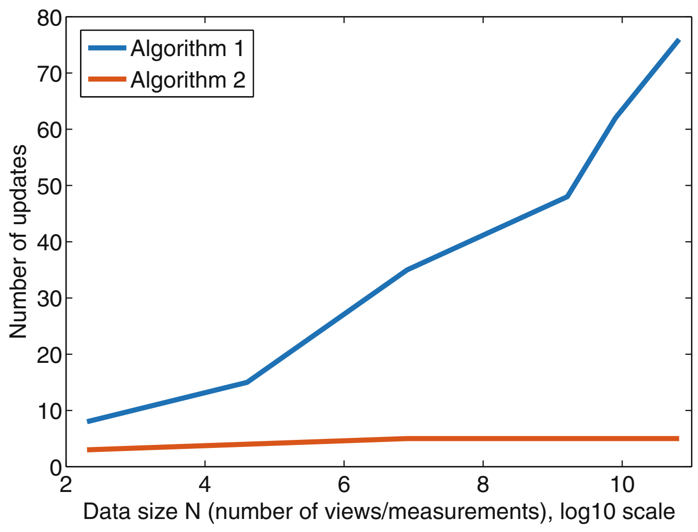


Fig. 4. Number of updates as a function of input size N (synthetic data) for Algorithm 1 and Algorithm 2.

A comparison of the number of updates will not be complete without comparing the total computational effort required. Instead of recording the actual runtime, however, which is dependent on the specific solver used and the maturity of its implementation, we compare the cumulative size of the subproblems that needed to be solved across the updates — intuitively, this is the amount of data that the algorithm accessed for triangulation. For Algorithm 1, since each update is invoked on the data of size 5 (i.e., $|\mathcal{B} \cup \{i\}|$), the cumulative subproblem size is simply 5 times the number of updates. For Algorithm 2, the subproblem size depends on the size of $\mathcal{C}$; the relationship between the cumulative subproblem size and the number of updates/iterations is

$$4 \text{ (initialisation)} + 5 \text{ (iter 1)} + 6 \text{ (iter 2)} + 7 \text{ (iter 3)} + \dots \tag{8}$$

Columns 4 and 5 in Table 1 show the cumulative subproblem sizes for both algorithms. Clearly Algorithm 2 remains superior in this respect, thus we expect Algorithm 2 to be faster than Algorithm 1 in actual applications.

Real Data. The experiment above was repeated on real data; here, the three variants of (ℓ_∞, ℓ_p) with $p = 1$, 2 and ∞ were tested. The Dinkelbach's method was used as the embedded (ℓ_∞, ℓ_p) solver in Algorithm 2. Tables 2 and 3 show respectively the *total* number of updates and *total* cumulative subproblem size of both algorithms across all triangulation instances in each dataset. The same conclusion can be drawn, i.e., Algorithm 2 is far more efficient in terms of number of updates and total amount of measurements accessed.

Table 2. Number of updates.

Dataset	(ℓ_∞, ℓ_1)		(ℓ_∞, ℓ_2)		$(\ell_\infty, \ell_\infty)$	
	Algorithm 1	Algorithm 2	Algorithm 1	Algorithm 2	Algorithm 1	Algorithm 2
Vercingetorix	2883	1065	2788	1069	2914	1104
Stockholm	14766	5811	13624	5350	14956	5915
Arc of Triumph	20796	7430	19147	7069	20676	7524
Alcatraz	36943	11028	34881	12533	36094	11966
Örebro Castle	63613	22506	57806	20912	61978	22380
Notre Dame	62577	10743	44758	13451	66469	12358

Table 3. Cumulative subproblem size.

Dataset	(ℓ_∞, ℓ_1)		(ℓ_∞, ℓ_2)		$(\ell_\infty, \ell_\infty)$	
	Algorithm 1	Algorithm 2	Algorithm 1	Algorithm 2	Algorithm 1	Algorithm 2
Vercingetorix	11532	4958	11152	4974	11656	5213
Stockholm	59064	30327	54496	26875	59824	30913
Arc of Triumph	83184	38970	76588	36236	82704	39616
Alcatraz	147772	57902	139524	66303	144376	63980
Örebro Castle	254452	126963	231224	114711	247912	126260
Notre Dame	250308	49034	179032	64200	265876	58721

4.3 Relative Speed-Up of Meta-Algorithm over Batch Solution

We compared running Algorithm 2 with a specific solver as a sub-routine, and the direct execution of the same solver in "batch mode" on the whole data. Since the runtime of Algorithm 2 depends on the efficiency of embedded solver, the key performance indicator here is the *relative speed-up* achieved by the meta-algorithm over batch execution.

Choice of Solvers. In this experiment, we tested the three variants of (ℓ_∞, ℓ_p), specifically (ℓ_∞, ℓ_1), (ℓ_∞, ℓ_2) and $(\ell_\infty, \ell_\infty)$.

Table 4. Runtime comparison between Algorithm 2 and batch execution for (ℓ_∞, ℓ_1), (ℓ_∞, ℓ_2) and $(\ell_\infty, \ell_\infty)$. Points: number of scene points in the dataset. Views: total number views involved. For Algorithm 2, the number in parentheses indicates the percentage amongst all triangulation instances where Algorithm 2 was faster than batch execution.

				Total runtime (s)	
(ℓ_∞, ℓ_1)	Dataset	Points	Views	Batch	Algorithm 2
	Vercingetorix	594	68	1.61	1.58 (13%)
	Stockholm	2176	43	10.49	9.87 (19%)
	Arc of Triumph	2744	173	20.57	12.00 (49%)
	Alcatraz	4431	419	166.34	40.12 (73%)
	Örebro Castle	5943	761	1011.96	56.01 (94%)
	Notre Dame	7149	715	1535.57	116.22 (87%)
				Total runtime (s)	
(ℓ_∞, ℓ_2)	Dataset	Points	Views	Batch	Algorithm 2
	Vercingetorix	594	68	23.65	17.43 (26%)
	Stockholm	2176	43	109.51	74.02 (32%)
	Arc of Triumph	2744	173	204.40	89.91 (56%)
	Alcatraz	4431	419	452.73	239.55 (47%)
	Örebro Castle	5943	761	1440.97	351.25 (76%)
	Notre Dame	7149	715	2399.22	582.64 (76%)
				Total runtime (s)	
$(\ell_\infty, \ell_\infty)$	Dataset	Points	Views	Batch	Algorithm 2
	Vercingetorix	594	68	1.82	1.83 (−1%)
	Stockholm	2176	43	12.20	11.77 (4%)
	Arc of Triumph	2744	173	23.57	14.15 (40%)
	Alcatraz	4431	419	146.26	28.93 (80%)
	Örebro Castle	5943	761	887.27	52.20 (94%)
	Notre Dame	7149	715	865.35	31.72 (96%)

For (ℓ_∞, ℓ_1) and (ℓ_∞, ℓ_2), we chose Dinkelbach's method [6] (equivalent to [19]). Although the best performing technique in [2] was Gugat's algorithm [11], our experiments suggested that it did not outperform Dinkelbach's method on the triangulation problem — in any case, the performance metric of interest here is the relative speed-up of the meta-algorithm (Algorithm 2) over batch execution. If a faster solver was used, it would likely improve both Algorithm 2 and batch solution by the same factor. SeDuMi [25] was used to solve the convex subproblems (LP for (ℓ_∞, ℓ_1) and SOCP for (ℓ_∞, ℓ_2)) in Dinkelbach's method. For $(\ell_\infty, \ell_\infty)$, we used the state-of-the-art polyhedron collapse solver [7].

Results and Analysis. Table 4 shows the total runtime of Algorithm 2 and batch execution on the large scale 3D reconstruction datasets described in Sect. 4.1. Clearly on most of the datasets, embedding the (ℓ_∞, ℓ_p) solver into Algorithm 2 significantly cuts down the total runtime. On datasets where the computational gains were not evident, this was because the datasets were too

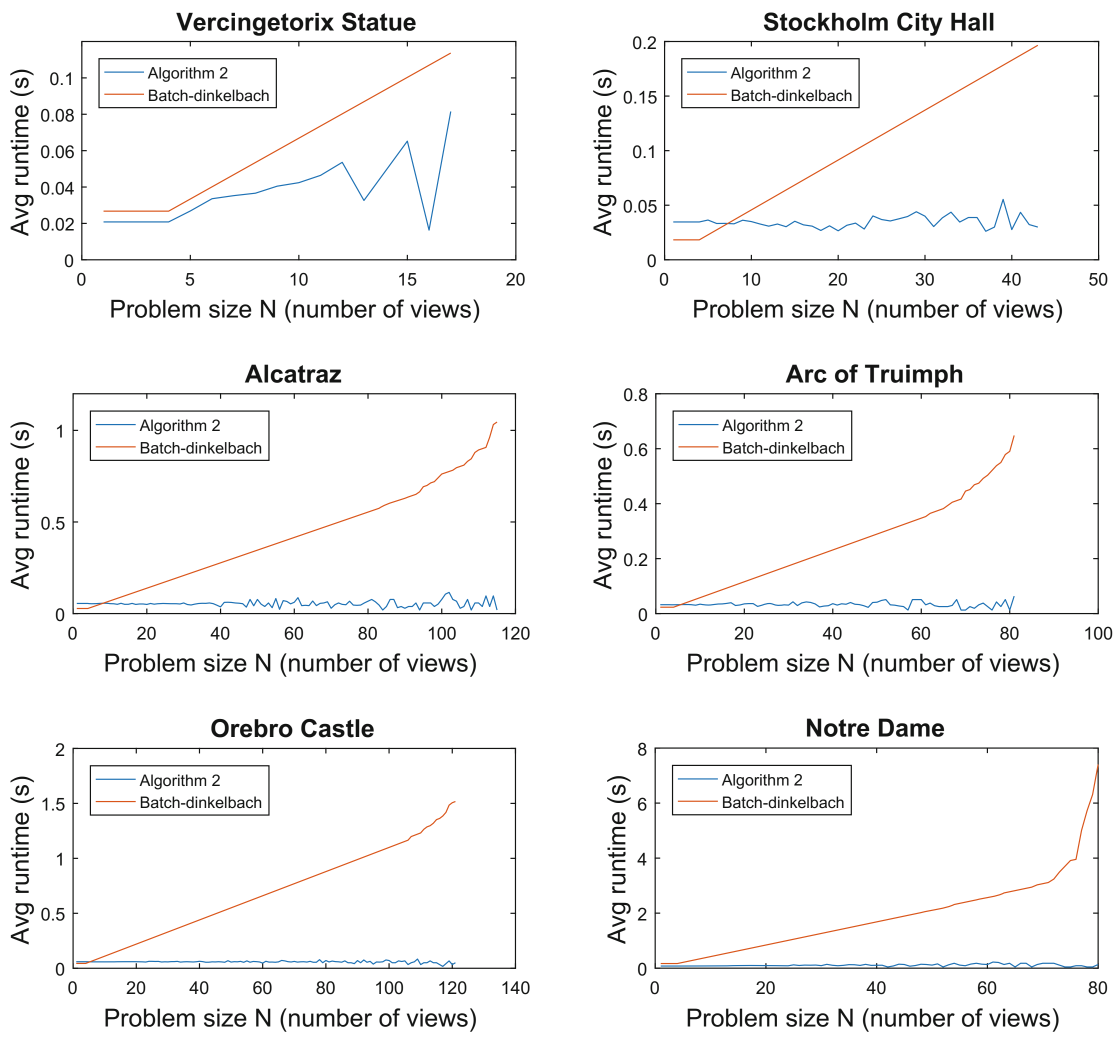


Fig. 5. Runtime comparison between batch method and Algorithm 2.

small (in terms of the number of triangulation instances and the size of the instances) for the benefit of Algorithm 2 to be exhibited.

For the case of (ℓ_∞, ℓ_2), the average runtimes of Algorithm 2 and batch solution as a function of problem size N (number of views/measurements) are plotted in Fig. 5. Observe that the runtime of batch increased linearly and then exponentially, whilst Algorithm 2 exhibited almost constant runtime — the latter observation is not surprising, since Algorithm 2 usually terminated at ≤ 10 iterations regardless of the problem size, as established in Sect. 4.2.

Of course, on all of the datasets, most of the triangulation instances are small, as summarised in the histograms in Fig. 3. However, in these datasets, there are sufficient numbers of moderate to large problem instances, such that the total runtime of Algorithm 2 is still much smaller than the total runtime of batch, as evidenced in Table 4.

5 Conclusions

In this paper, we propose a meta-algorithm for ℓ_∞ triangulation. By exploring the fact that ℓ_∞ triangulation is a GLP, the meta-algorithm offers significant acceleration over applying an underlying triangulation solvers in batch mode. We provided comprehensive experimental results that establish the practical value of the meta-algorithm on large scale 3D reconstruction datasets.

Acknowledgement. This work was supported by ARC Grant DP160103490.

References

1. Agarwal, P.K., Har-Peled, S., Varadarajan, K.R.: Geometric approximation via coresets. Discret. Comput. Geom. **52**, 1–30 (2005)
2. Agarwal, S., Snavely, N., Seitz, S.: Fast algorithms for l_∞ problems in multiview geometry. In: CVPR (2008)
3. Amenta, N.: Helly-type theorems and generalized linear programming. Discret. Comput. Geom. **12**, 241–261 (1994)
4. Clarkson, K.L.: Las Vegas algorithms for linear and integer programming when the dimension is small. J. ACM **42**, 488–499 (1995)
5. Dai, Z., Wu, Y., Zhang, F., Wang, H.: A novel fast method for L_∞ problems in multiview geometry. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7576, pp. 116–129. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33715-4_9
6. Dinkelbach, W.: On nonlinear fractional programming. Manag. Sci. **13**, 492–498 (1967)
7. Donné, S., Goossens, B., Philips, W.: Point triangulation through polyhedrom collapse using the l_∞ norm. In: ICCV (2015)
8. Enqvist, O., Olsson, C., Kahl, F.: Stable structure from motion using rotational consistency. Technical report (2010)
9. Eriksson, A., Isaksson, M.: Pseudoconvex proximal splitting for l_∞ problems in multiview geometry. In: CVPR (2014)

10. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multi-view stereopsis. IEEE TPAMI **32**, 1362–1376 (2010)
11. Gugat, M.: A fast algorithm for a class of generalized fractional programs. Manag. Sci. **42**, 1493–1499 (1996)
12. Hartley, R.I., Schaffalitzky, F.: l_∞ minimization in geometric reconstruction problems. In: CVPR (2004)
13. Kahl, F.: Multiple view geometry and the l_∞ norm. In: ICCV (2005)
14. Ke, Q., Kanade, T.: Quasiconvex optimization for robust geometric reconstruction. In: ICCV (2005)
15. Li, H.: Efficient reduction of l_∞ geometry problems. In: CVPR (2009)
16. Matoušek, J., Sharir, M., Welzl, E.: A subexponential bound for linear programming. Algorithmica **16**, 498–516 (1996)
17. Mur-Artal, R., Tardós, J.D.: Probabilistic semi-dense mapping from highly accurate feature-based monocular SLAM. In: RSS (2015)
18. Olsson, C., Enqvist, O.: Stable structure from motion for unordered image collections. In: Heyden, A., Kahl, F. (eds.) SCIA 2011. LNCS, vol. 6688, pp. 524–535. Springer, Heidelberg (2011). doi:10.1007/978-3-642-21227-7_49
19. Olsson, C., Eriksson, A., Kahl, F.: Efficient optimization for l_∞ problems using pseudoconvexity. In: ICCV (2007)
20. Seidel, R.: Small-dimensional linear programming and convex hulls made easy. Discret. Comput. Geom. **6**, 423–434 (1991)
21. Seo, Y., Hartley, R.I.: A fast method to minimize l_∞ error norm for geometric vision problems. In: ICCV (2007)
22. Sharir, M., Welzl, E.: A combinatorial bound for linear programming and related problems. In: Finkel, A., Jantzen, M. (eds.) STACS 1992. LNCS, vol. 577, pp. 567–579. Springer, Heidelberg (1992). doi:10.1007/3-540-55210-3_213
23. Sim, K., Hartley, R.: Removing outliers using the l_∞ norm. In: CVPR (2006)
24. Snavely, N., Seitz, S.M., Szeliski, R.: Modeling the world from internet photo collections. IJCV **80**, 189–210 (2007)
25. Sturm, J.F.: Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. Optim. Methods Softw. **11–12**, 625–653 (1999)

Chapter 4

Quasiconvex Plane Sweep for Triangulation with Outliers

The work contained in this chapter has been published as the following paper

Qianggong Zhang, Tat-Jun Chin and David Suter: Quasiconvex Plane Sweep for Triangulation With Outliers. IEEE International Conference on Computer Vision (ICCV) 2017.

The published paper is available at
http://ieeexplore.ieee.org/document/8237367/

Statement of Authorship

Title of Paper	Quasiconvex Plane Sweep for Triangulation with Outliers		
Publication Status	✓ Published	☐ Accepted for Publication	
	☐ Submitted for Publication	☐ Unpublished and Unsubmitted w ork w ritten in manuscript style	
Publication Details	Zhang, Q., Chin, T. J., & Suter, D. (2017). Quasiconvex Plane Sweep for Triangulation with Outliers. In Proceedings of the IEEE International Conference on Computer Vision.		

Principal Author

Name of Principal Author (Candidate)	Qianggong Zhang		
Contribution to the Paper	Developed the theoretical proof of key theorems, performed experiments and wrote manuscript.		
Overall percentage (%)	60%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	22/11/2017

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Tat-Jun Chin		
Contribution to the Paper	Supervised development of work and revised manuscript.		
Signature		Date	22/11/2017

Name of Co-Author	David Suter		
Contribution to the Paper	Revised manuscript.		
Signature		Date	22/11/2017

Please cut and paste additional co-author panels here as required.

Quasiconvex Plane Sweep for Triangulation with Outliers

Qianggong Zhang Tat-Jun Chin David Suter
The University of Adelaide
Adelaide, South Australia, 5005, Australia
qianggong.zhang@adelaide.edu.au, tat-jun.chin@adelaide.edu.au, david.suter@adelaide.edu.au

Abstract

Triangulation is a fundamental task in 3D computer vision. Unsurprisingly, it is a well-investigated problem with many mature algorithms. However, algorithms for robust *triangulation, which are necessary to produce correct results in the presence of egregiously incorrect measurements (i.e., outliers), have received much less attention. The default approach to deal with outliers in triangulation is by random sampling. The randomized heuristic is not only suboptimal, it could, in fact, be computationally inefficient on large-scale datasets. In this paper, we propose a novel locally optimal algorithm for robust triangulation. A key feature of our method is to efficiently derive the local update step by plane sweeping a set of quasiconvex functions. Underpinning our method is a new theory behind quasiconvex plane sweep, which has not been examined previously in computational geometry. Relative to the random sampling heuristic, our algorithm not only guarantees deterministic convergence to a local minimum, it typically achieves higher quality solutions in similar runtimes*[1].

1. Introduction

Triangulation is the task of estimating the 3D coordinates of a scene point from multiple 2D image observations of the point, given that the pose of the cameras are known [14]. The task is of fundamental importance to 3D vision, since it enables the recovery of the 3D structure of a scene.

Most 3D reconstruction pipelines estimate 3D structure and camera poses simultaneously (via bundle adjustment, factorization, or equivalent steps). However, triangulation can play an important role in densifying or refining the 3D structure, by estimating the 3D coordinates of additional image measurements (e.g., extracted from original high-resolution images) based on the optimized camera poses.

Since 2D feature detection and association methods are not perfect, they inevitably create wrong feature correspondences and tracks. In an SfM pipeline, outliers are removed during the robust relative pose estimation step. However, outliers will exist in the additional feature correspondences extracted post-SfM, since they were not subjected to the SfM pipeline (e.g. for efficiency reasons).

[1]See supplementary material for demo program.

For *non-robust* triangulation, the ℓ_∞ paradigm [13] has been influential. Given a set of N 2D image measurements $\{\mathbf{u}_i\}_{i=1}^N$ of the same scene point $\mathbf{x} \in \mathbb{R}^3$, and the associated camera matrices $\{\mathbf{P}_i\}_{i=1}^N$ with each $\mathbf{P}_i \in \mathbb{R}^{3\times 4}$, we estimate $\mathbf{x}$ by minimizing the maximum reprojection error

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^3} \quad & \underset{i\in\{1,\ldots,N\}}{\text{maximum}} \left\| \mathbf{u}_i - \frac{\mathbf{P}_i^{1:2}\tilde{\mathbf{x}}}{\mathbf{P}_i^3\tilde{\mathbf{x}}} \right\|_p, \\ \text{s.t.} \quad & \mathbf{P}_i^3\tilde{\mathbf{x}} > 0 \;\; \forall i \in \{1,\ldots,N\}, \end{aligned} \tag{1}$$

where $\mathbf{P}^{1:2}$ is the first-two rows of $\mathbf{P}$, $\mathbf{P}^3$ is the third row of $\mathbf{P}$, and $\tilde{\mathbf{x}}$ is $\mathbf{x}$ in homogeneous coordinates. The positivity constraints $\mathbf{P}_i^3\tilde{\mathbf{x}} > 0$ ensure that $\mathbf{x}$ lies in front of all the cameras. In the above, $\|\cdot\|_p$ indicates a valid p-norm; usually p is taken to be 1, 2 or ∞.

Algorithms to solve (1) take advantage of quasiconvexity to efficiently find the global minimizer $\mathbf{x}^*$ [15, 16, 25, 23, 3]. Recently, Donné et al. [9] showed that their *polyhedron collapse* algorithm (for $p = \infty$) is the fastest.

A major weakness of the ℓ_∞ paradigm, however, is that the estimate is easily biased by outlying measurements. To fix this issue, the usage of an inherently robust cost function is necessary. A popular robust criterion is least median squares (LMS) [24]; for triangulation, this entails solving

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^3} \quad & \underset{i\in\{1,\ldots,N\}}{\text{median}} \left\| \mathbf{u}_i - \frac{\mathbf{P}_i^{1:2}\tilde{\mathbf{x}}}{\mathbf{P}_i^3\tilde{\mathbf{x}}} \right\|_p, \\ \text{s.t.} \quad & \mathbf{P}_i^3\tilde{\mathbf{x}} > 0 \;\; \forall i \in \{1,\ldots,N\}, \end{aligned} \tag{2}$$

i.e., minimize the median error. LMS provably has a breakdown point of 0.5, which means that it can tolerate *up to* 50% of outliers [24, Chap. 3]. The drawback of LMS, however, is that the median of the reprojection errors is not quasiconvex, and problem (2) becomes intractable in general. The non-differentiability of the median also complicates the usage of standard gradient-based optimization [21].

Existing algorithms for LMS Most practitioners rely on the random sampling heuristic to approximately solve LMS [30, 24]. Specifically, we randomly sample minimal subsets of the measurements to estimate $\mathbf{x}$ (using, e.g., DLT for triangulation), then select the estimate with the lowest median error. A probabilistic upper bound of the number of samples to take can be deduced based on the highest expected outlier rate of 0.5 [1]. Apart from being non-deterministic, a noticeable weakness of random sampling is that it provides no optimality guarantees.

Ke and Kanade [16] used the bisection technique endowed with a *non-convex* feasibility test to solve general quasiconvex LMS problems, which includes (2). For tractability, a relaxed feasibility test which is more conservative is performed, thus the method can only converge to an approximate LMS solution without any certificate of optimality (either local or global).

On the other extreme, combinatorial search algorithms have been proposed to solve LMS exactly [28, 4]. For triangulation, Li [17] exploited the quasiconvexity of the reprojection error to devise a search algorithm that enumerates all local minima of the LMS problem. Despite the low-dimensionality of $\mathbf{x}$, the exact algorithms are computationally costly, and are practical only for small instances.

Our contributions We propose a novel *locally optimal* algorithm for LMS triangulation (2). At each iteration, our approach calculates the update via a 1D quasiconvex LMS problem, which can be solved efficiently via plane sweep. We develop the necessary theory and algorithm for quasiconvex plane sweep, and establish the convergence of the overall algorithm to a local minimum.

Experimentally, we show that our method consistently yields better solutions than random sampling with comparable runtimes. Further, our technique is much faster and practical than the globally optimal methods.

Differentiation against RANSAC Another popular robust technique in computer vision is RANSAC [12]. Unlike LMS, the aim of RANSAC is to maximize the number of inliers, given a threshold. Whether LMS or inlier maximization is the "better" criterion is debatable—certainly for robust triangulation, both are valid and widely used.

The optimization machinery in RANSAC is random sampling, thus, it shares the disadvantages of the randomized heuristic for LMS mentioned above. Of course, there are alternative methods for inlier maximization, e.g., RANSAC variants [6], branch-and-bound [18] and subset search [5]. We stress, however, that from an optimization viewpoint, these methods solve a different problem to LMS and are not strictly comparable to our algorithm (not to mention that global methods for inlier maximization would also be costly, similar to global methods for LMS).

2. Background

First, define and rewrite the i-th reprojection error as

$$r_i(\mathbf{x}) = \left\| \mathbf{u}_i - \frac{\mathbf{P}_i^{1:2}\tilde{\mathbf{x}}}{\mathbf{P}_i^3\tilde{\mathbf{x}}} \right\|_p = \frac{\|\mathbf{A}_i\mathbf{x} + \mathbf{b}_i\|_p}{\mathbf{c}_i^T\mathbf{x} + d_i}, \quad (3)$$

$$\text{where } \mathbf{A}_i = \begin{bmatrix} \mathbf{a}_{i,1}^T \\ \mathbf{a}_{i,2}^T \end{bmatrix} \in \mathbb{R}^{2\times 3}, \quad \mathbf{b}_i = \begin{bmatrix} b_{i,1} \\ b_{i,2} \end{bmatrix} \in \mathbb{R}^2, \quad (4)$$

$\mathbf{c}_i \in \mathbb{R}^3$ and d_i are constants calculated from the data $\mathbf{P}_i$ and $\mathbf{u}_i$. For $p \geq 1$, $r_i(\mathbf{x})$ is quasiconvex [11].

To enable direct comparison with the state-of-the-art polyhedron collapse method, we also base our method on the same $p = \infty$. The reprojection error thus becomes

$$r_i(\mathbf{x}) = \max\left(\frac{|\mathbf{a}_{i,1}^T\mathbf{x} + b_{i,1}|}{\mathbf{c}_i^T\mathbf{x} + d_i}, \frac{|\mathbf{a}_{i,2}^T\mathbf{x} + b_{i,2}|}{\mathbf{c}_i^T\mathbf{x} + d_i} \right), \quad (5)$$

which can be further developed into

$$\begin{aligned} r_i(\mathbf{x}) &= \max\left(\frac{\mathbf{a}_{i,1}^T\mathbf{x} + b_{i,1}}{\mathbf{c}_i^T\mathbf{x} + d_i}, \frac{-\mathbf{a}_{i,1}^T\mathbf{x} - b_{i,1}}{\mathbf{c}_i^T\mathbf{x} + d_i}, \right. \\ &\qquad \left. \frac{\mathbf{a}_{i,2}^T\mathbf{x} + b_{i,2}}{\mathbf{c}_i^T\mathbf{x} + d_i}, \frac{-\mathbf{a}_{i,2}^T\mathbf{x} - b_{i,2}}{\mathbf{c}_i^T\mathbf{x} + d_i} \right) \quad (6) \\ &= \max\left(r_{i,1}(\mathbf{x}), r_{i,2}(\mathbf{x}), r_{i,3}(\mathbf{x}), r_{i,4}(\mathbf{x})\right). \quad (7) \end{aligned}$$

For simplicity, we define $r_{i,j}(\mathbf{x})$, $j = 1, \ldots, 4$, as

$$r_{i,j}(\mathbf{x}) = \frac{\mathbf{a}_{i,j}^T\mathbf{x} + b_{i,j}}{\mathbf{c}_i^T\mathbf{x} + d_i}; \quad (8)$$

the reader should be reminded that constants $\mathbf{a}_{i,j}$ and $b_{i,j}$ should be taken with the appropriate sign from the input data. Henceforth, we call the $r_{i,j}(\mathbf{x})$'s "constraints".

Defining $r_i(\mathbf{x})$ as above, i.e., as a max over four linear fractional terms, will be crucial for clarifying the operations of our method later. For now, we re-express the LMS triangulation problem equivalently as

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^3} \quad & \underset{i\in\{1,\ldots,N\}}{\text{median}} \; r_i(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{c}_i^T\mathbf{x} + d_i > 0 \;\; \forall i \in \{1,\ldots,N\}, \end{aligned} \quad (9)$$

where the input data is $\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i, d_i\}_{i=1}^N$.

3. Locally optimal LMS triangulation

Algorithm 1 describes the proposed locally optimal method (called **Q-sweep**) to solve (9). The overall structure of Q-sweep is simple—given an initial feasible estimate $\hat{\mathbf{x}}$, find a direction $\Delta\mathbf{x}$ and step size α to adjust $\hat{\mathbf{x}}$ such that the median error decreases; stop when a valid $\Delta\mathbf{x}$ cannot be found. A similar overall structure exists in polyhedron collapse, and indeed in many techniques in the wider optimization literature [21]. Nonetheless, there are significant novelties in our work, namely, an efficient routine to compute the optimal step size α for LMS triangulation, and theoretical analyses on convergence and complexity.

Algorithm 1 Q-sweep method for LMS triangulation.

```
Require: Input data $\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i, d_i\}_{i=1}^N$, initial soln. $\hat{\mathbf{x}}$.
1: $\Delta\mathbf{x} \leftarrow$ DESCENTDIR$\big(\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i, d_i\}_{i=1}^N, \hat{\mathbf{x}}\big)$.
2: while $\Delta\mathbf{x}$ is not null do
3:     $\alpha \leftarrow$ STEPSIZE$\big(\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i, d_i\}_{i=1}^N, \hat{\mathbf{x}}, \Delta\mathbf{x}\big)$.
4:     $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \alpha\Delta\mathbf{x}$.
5:     $\Delta\mathbf{x} \leftarrow$ DESCENTDIR$\big(\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i, d_i\}_{i=1}^N, \hat{\mathbf{x}}\big)$.
6: end while
7: return $\hat{\mathbf{x}}$.
```

The rest of this section is devoted to fleshing out Algorithm 1 (details on initialization are postponed until Sec. 4).

3.1. Finding descent direction

Algorithm 2 describes the routine DESCENTDIR used in Q-sweep to find $\Delta\mathbf{x}$ for the current estimate $\hat{\mathbf{x}}$. The routine begins by finding the set of residuals $\mathcal{A}$ and constraints $\mathcal{J}$ that are active, i.e., has the same value as the *median* error for $\hat{\mathbf{x}}$. Given $\mathcal{J}$, the rest of the routine largely follows the procedure of polyhedron collapse to calculate $\Delta\mathbf{x}$. For brevity, we will give only high-level account of the method.

Algorithm 2 DESCENTDIR to find descent direction.

```
Require: Input data $\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i, d_i\}_{i=1}^N$, an estimate $\hat{\mathbf{x}}$.
1: $\hat{r} \leftarrow \mathrm{med}_i\, r_i(\hat{\mathbf{x}})$.
2: $\mathcal{A} \leftarrow \{p \mid r_p(\hat{\mathbf{x}}) = \hat{r}\}$.
3: $\mathcal{J} \leftarrow \{(p,q) \mid p \in \mathcal{A},\ r_{p,q}(\hat{\mathbf{x}}) = \hat{r}\}$.
4: $\mathcal{N} = \{\mathbf{n}_1, \mathbf{n}_2, \dots\} \leftarrow$ set of normals for the constraints
   indexed by $\mathcal{J}$; see (11) and surrounding text.
5: $\Delta\mathbf{x} \leftarrow$ null.
6: if $|\mathcal{N}| = 1$ then
7:     $\Delta\mathbf{x} \leftarrow \mathbf{n}_1$.
8: else if $|\mathcal{N}| = 2$ then
9:     $\Delta\mathbf{x} \leftarrow \mathbf{n}_1 + \mathbf{n}_2$.
10: else
11:     for each triplet $(\mathbf{n}_u, \mathbf{n}_v, \mathbf{n}_w)$ of $\mathcal{N}$ do
12:         $\mathbf{y} \leftarrow \mathbf{n}_u \times \mathbf{n}_v + \mathbf{n}_v \times \mathbf{n}_w + \mathbf{n}_w \times \mathbf{n}_u$.
13:         $s \leftarrow \langle \mathbf{n}_u, \mathbf{y}\rangle / |\langle \mathbf{n}_u, \mathbf{y}\rangle|$.
14:         $\mathbf{y} \leftarrow s\mathbf{y}$.
15:         if $\langle \mathbf{y}, \mathbf{n}\rangle > 0,\ \forall \mathbf{n} \in \mathcal{N}$ then
16:             $\Delta\mathbf{x} \leftarrow \mathbf{y}$.
17:             Break.
18:         end if
19:     end for
20: end if
21: return $\Delta\mathbf{x}$.
```

Each active constraint $r_{p,q}(\mathbf{x})$ indexed by $\mathcal{J}$ defines a 2D plane in 3D space, i.e.,

$$\begin{aligned} r_{p,q}(\mathbf{x}) &= \frac{\mathbf{a}_{p,q}^T\mathbf{x} + b_{p,q}}{\mathbf{c}_p^T\mathbf{x} + d_p} = \hat{r}, \\ \implies & (\mathbf{a}_{p,q}^T - \hat{r}\mathbf{c}_p^T)\mathbf{x} + b_{p,q} - \hat{r}d_p = 0. \end{aligned} \tag{10}$$

The normal of the plane pointing towards the negative direction is given by

$$\mathbf{n} = -(\mathbf{a}_{p,q}^T - \hat{r}\mathbf{c}_p^T). \tag{11}$$

The normal $\mathbf{n}$ is also the direction where $r_{p,q}(\mathbf{x})$ will reduce in value, starting from the point $\hat{\mathbf{x}}$.

If $\mathcal{J}$ has only one element, then the normal of that constraint is installed as $\Delta\mathbf{x}$ (Step 7). If there are more than one active constraints, then the normals of the constraints are combined: by a simple addition if there are two active constraints (Step 9), or if there are more active constraints, triplets of normals are considered. For each triplet, the vector that gives the same scalar product on the normals are computed (Step 14). The first such vector that allows all active constraints to reduce is then taken as $\Delta\mathbf{x}$ (Step 16)—if no such vector is available, the overall algorithm terminates.

Donné et al. showed that finding $\Delta\mathbf{x}$ in the manner above guarantees that $\Delta\mathbf{x}$ represents a direction from $\hat{\mathbf{x}}$ along which the active residuals (which are the median residuals in our case) decrease in value. Sec. 3.3 will establish that DESCENTDIR always find a descent direction until the convergence of Q-sweep to a local minimum.

Number of active constraints The cost of Algorithm 2 depends on the number $|\mathcal{J}|$ of active constraints. Donné et al. cited empirical evidence to support that the number of active constraints is small (3 or 4). Actually, as we prove below, there is a theoretical limit on the number of active constraints—this is another contribution of our work.

Theorem 1. *For any feasible $\hat{\mathbf{x}}$, there are at most 8 constraints $r_{i,j}(\mathbf{x})$ such that $r_{i,j}(\hat{\mathbf{x}}) = \mathrm{median}_i\, r_i(\hat{\mathbf{x}})$.*

Proof. The combinatorial dimension of quasiconvex ℓ_∞ triangulation is 4, and assuming that the input data is non-degenerate[2], the number of active residuals (i.e., the cardinality of $\mathcal{A}$ in Algorithm 2) is at most 4 [26].

From (6), in each residual $r_i(\mathbf{x})$, the operands $r_{i,1}(\mathbf{x})$ and $r_{i,2}(\mathbf{x})$ are symmetric, such that for any $\hat{r} > 0$,

$$r_{i,1}(\hat{\mathbf{x}}) = \hat{r} \quad \text{and} \quad r_{i,2}(\hat{\mathbf{x}}) = \hat{r} \tag{12}$$

cannot be satisfied simultaneously. Likewise for $r_{i,3}(\mathbf{x})$ and $r_{i,4}(\mathbf{x})$. Thus, for each active residual $r_p(\mathbf{x})$, there are at most two operands that satisfy $r_{p,q}(\hat{\mathbf{x}}) = \hat{r}$. The total number of active constraints thus cannot be more than 8. □

[2]For the precise definition of degeneracy, see [20, Sec. 2.2]. In practical instances that are affected by noise, the data is usually non-degenerate.

3.2. Computing step size

Henceforth represents a significant departure from polyhedron collapse. From $\hat{\mathbf{x}}$, the new estimate is obtained as

$$\mathbf{x}' = \hat{\mathbf{x}} + \alpha\Delta\mathbf{x}. \tag{13}$$

Naturally, $\mathbf{x}'$ must remain feasible, but we would also like to find the α that reduces the median error the most.

Along $\Delta\mathbf{x}$ and starting from $\hat{\mathbf{x}}$, the constraints can be rewritten as a function of α:

$$r_{i,j}(\alpha) = \frac{\mathbf{a}_{i,j}^T(\hat{\mathbf{x}} + \alpha\Delta\mathbf{x}) + b_{i,j}}{\mathbf{c}_i^T(\hat{\mathbf{x}} + \alpha\Delta\mathbf{x}) + d_i} := \frac{u_{i,j}\alpha + v_{i,j}}{w_i\alpha + z_i}, \tag{14}$$

where $u_{i,j}$, $v_{i,j}$, w_i and z_i are constants calculated from the data; $r_{i,j}(\alpha)$ is again a linear fractional function, which is quasiconvex [2]. Trivially, the reprojection error $r_i(\mathbf{x})$ along direction $\Delta\mathbf{x}$ and starting from $\hat{\mathbf{x}}$ is

$$r_i(\alpha) = \max(r_{i,1}(\alpha), r_{i,2}(\alpha), r_{i,3}(\alpha), r_{i,4}(\alpha)), \tag{15}$$

with the usual condition $w_i\alpha + z_i > 0$ on the denominator. Since each of the max operands in (15) is quasiconvex, $r_i(\alpha)$ is quasiconvex; Fig. 1(a) illustrates.

The problem of determining α can be formulated as

$$\begin{aligned} \alpha^* = &\operatorname*{argmin}_{\alpha\in\mathbb{R}_+} \quad \operatorname*{median}_{i\in\{1,\dots,N\}} r_i(\alpha), \\ &\text{s.t.} \quad w_i\alpha + z_i > 0 \;\; \forall i \in \{1,\dots,N\}, \end{aligned} \tag{16}$$

i.e., a quasiconvex LMS problem defined over α. Solving (16) exactly to find α^* remains *theoretically* intractable. Nonetheless, since we are dealing with only one dimension, there are "tricks" to do this efficiently.

Characterization of the solution Where can we expect α^* to lie? We first define several geometrical concepts.

Definition 1 (Extremity). *The extremity m_i of $r_i(\alpha)$ is the point at which $r_i(\alpha)$ attains the minimum. Since $r_i(\alpha)$ is a linear fractional function, it is actually pseudoconvex (a stronger condition than quasiconvexity) [2], implying that m_i is unique; see Fig. 1(a). The extremity can be obtained analytically by intersecting all the constraints*

$$\frac{u_{i,j}\alpha + v_{i,j}}{w_i\alpha + z_i} = \frac{u_{i,j'}\alpha + v_{i,j'}}{w_i\alpha + z_i}, \quad j, j' \in \{1,\dots,4\} \tag{17}$$

from $r_i(\alpha)$, and finding the roots. The smallest root that is not below $r_i(\alpha)$ is then installed as m_i.

Definition 2 (Intersection). *An intersection between $r_i(\alpha)$ and $r_{i'}(\alpha)$ is a point where the two error functions intersect. Note that for quasiconvex functions, there are in general more than one intersection. We let $I_{i,i'}^k$ denote the k-th intersection between $r_i(\alpha)$ and $r_{i'}(\alpha)$. The intersections can also be found analytically, by pairing the constraints from $r_i(\alpha)$ and $r_{i'}(\alpha)$, and solving the quadratic equations.*

Definition 3 (Boundary). *The boundary $\alpha_{\max}$ is the largest α such that $w_i\alpha + z_i > 0$ for all i.*

Definition 4 (Events). *The events are a set that consists of*

- *all extremities m_i in the range $[0, \alpha_{\max}]$.*
- *all intersections $I_{i,i'}^k$ in the range $[0, \alpha_{\max}]$.*
- *the boundary point $(\alpha_{\max}, \operatorname{median}_i\; r_i(\alpha_{\max}))$.*

We call an item of $\mathcal{E}$ an event point.

The following identifies the possible locations of α^*.

Theorem 2. *The minimizer α^* of problem* (16) *is an event point of the problem.*

Proof. In the feasible range $[0, \alpha_{\max}]$, let

$$g : [0, \alpha_{\max}] \mapsto \{1,\dots,N\} \tag{18}$$

give the index of the error corresponding to the median, i.e.,

$$r_{g(\alpha)}(\alpha) = \operatorname*{median}_i \; r_i(\alpha). \tag{19}$$

The function g partitions the feasible range $[0, \alpha_{\max}]$ into segments $(s_1, s_2, \dots)$, where α's from the same segment s_t yield the same index, i.e.,

$$g(\alpha_1) = g(\alpha_2) \;\; \text{for} \;\; \alpha_1, \alpha_2 \in s_t; \tag{20}$$

see Fig. 1(b). In turn, the segments give rise to a sequence of error functions $(r_{g_1}, r_{g_2}, \dots)$ that correspond to the median. It is thus sufficient to examine this sequence.

If an error function $r_{g_t}(\alpha)$ in the sequence achieves its minimum (or extremity) m_{g_t} in the segment s_t, then m_{g_t} is a local minimum of the median error; see Fig. 1(b).

Any two successive error functions $r_{g_t}(\alpha)$ and $r_{g_u}(\alpha)$ in the sequence give rise to an intersection I_{g_t,g_u}^k. If the gradient of $r_{g_t}(\alpha)$ and $r_{g_u}(\alpha)$ have opposing signs at I_{g_t,g_u}^k, then I_{g_t,g_u}^k is a local minimum; see Fig. 1(b).

Lastly, the median error may achieve a local minimum at the boundary point; see Fig. 1(b).

The above are all the possible local minima of (16), and one of them is the global minimum. □

Based on Theorem 2, a simple approach to solve (16) would be to visit all event points, and calculate the median error at each event point. In the following, a more efficient technique that avoids recomputing the median is described.

Quasiconvex plane sweep Plane sweep is a basic technique for many geometric problems, such as Delaunay triangulation [8]. Souvaine and Steele [27] developed an LMS *line fitting* algorithm based on plane sweep. However, their method is not directly applicable to quasiconvex LMS (16), due to several critical differences:

- A pair of quasiconvex curves may have multiple intersections (see the curves in Fig. 1(b)), while two lines have at most one intersection;

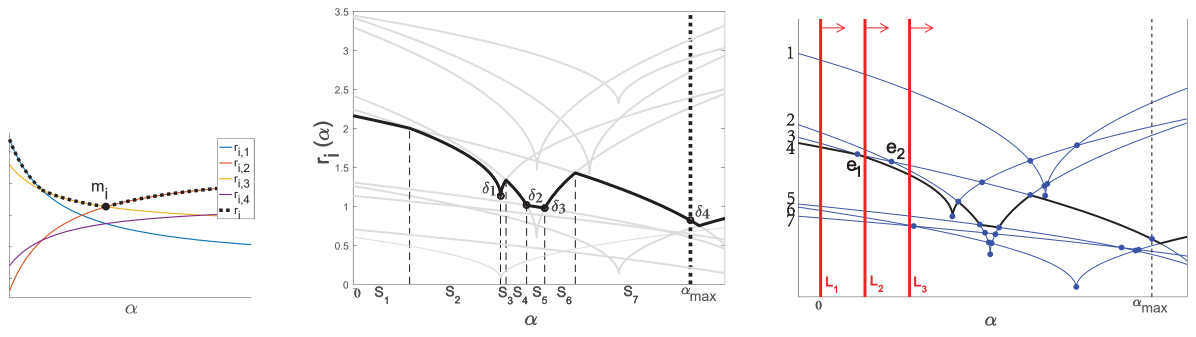


Figure 1. (Panel a) Reprojection error as a function of α. The black dashed curve is $r_i(\alpha)$ and the other four curves are the four constraints $r_{i,j}(\alpha)$ corresponding to $r_i(\alpha)$; m_i is the extremity of $r_i(\alpha)$. (Panel b) The black solid curve is the median over 7 reprojection errors $r_i(\alpha)$. Within each segment s_t, the median is defined by one of the error functions: $r_{g_t}(\alpha)$. Here, δ_1, δ_3 and δ_4 are local minima corresponding respectively to an extremity, an intersection, and the boundary. δ_2 is an intersection, but not a local minimum. (Panel c) Demonstrating plane sweep: the events are shown as dots. The sweep line is initialized at L_1, with the ordering **List** $= [1, 2, 3, 4, 5, 6, 7]$; the center item is 4, thus $r_4(\alpha)$ is the median. As the sweep line passes through event point e_1, indices 3 and 4 swap places; at L_2, the ordering becomes **List** $= [1, 2, 4, 3, 5, 6, 7]$, and $r_3(\alpha)$ is the median. After passing through e_2, **List** $= [1, 4, 2, 3, 5, 6, 7]$, and $r_3(\alpha)$ remains the median.

- At an intersection, two quasiconvex curves may not cross (i.e., they are tangent to each other), while two lines necessarily cross at their intersection
- A quasiconvex function may achieve a minimum in the feasible range, while a line is unbounded.

Here, we develop a novel plane sweep algorithm for (16), to be used as routine `STEPSIZE` in Q-sweep.

The idea is as follows: imagine a vertical line L in the plane $[0, \alpha_{\max}] \times \mathbb{R}_{\geq 0}$ that is "swept" from $\alpha = 0$ to $\alpha = \alpha_{\max}$; see Fig. 1(c). At each position of L, the error functions $r_i(\alpha)$ can be ordered based on their height along L; the ordering is called the **List**. The median error is exactly the height of the median point in L. In plane sweep, we visit the event points incrementally and query **List**.

A crucial observation is that **List** changes only when L passes through an event point that is a non-tangential intersection. In fact, when sweeping past an non-tangential intersection $I^k_{i,i'}$, only $r_i(\alpha)$ and $r_{i'}(\alpha)$ swap orders along L; see Fig. 1(c). Thus, **List** can be maintained and updated efficiently as the sweep line passes through the event points.

Algorithm 3 describes the proposed quasiconvex plane sweep algorithm in detail. Programmatically, an event point $e \in \mathcal{E}$ is endowed with attributes α, i and i', where

- $e.\alpha$ is the α value of the event point e.
- if e is an extremity, $e.i$ returns the index of the error $r_i(\alpha)$ that gives rise to e (here, $e.i'$ is null);
- if e is an intersection, $e.i$ and $e.i'$ are the indices of the errors $r_i(\alpha)$ and $r_{i'}(\alpha)$ whose intersection forms e.

Note that, as in most plane sweep-type algorithms [8], the sweep line L is not explicitly realized.

Complexity analysis The runtime of Algorithm 3 depends on the size of $\mathcal{E}$. Since there are at most N extremities, $\mathcal{E}$ is dominated by the intersections. The following establishes a bound on the number of intersections.

Algorithm 3 `STEPSIZE` to optimize step size.

```
Require: Input data $\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i, d_i\}_{i=1}^N$, current estimate
    $\hat{\mathbf{x}}$, and current descent direction $\Delta\mathbf{x}$.
1:  Convert reprojection errors to 1D version (15).
2:  $\alpha^* \leftarrow 0$.    /* Current estimate of $\alpha$ */
3:  $\mathcal{E} \leftarrow$ set of event points sorted ascendingly by $\{e.\alpha\}$.
4:  List $\leftarrow$ indices of error functions $r_i(\alpha)$, $i = 1, \ldots, N$,
    in descending order of error value $r_i(\alpha^*)$ at $\alpha^*$.
5:  $K \leftarrow \lceil N/2 \rceil$.
6:  $r^* \leftarrow r_{\mathbf{List}(K)}(\alpha^*)$.    /* Current median error */
7:  for each event point $e \in \mathcal{E}$ do
8:      if $e$ is an intersection then
9:          if $r_{e.i}(\alpha)$ and $r_{e.i'}(\alpha)$ are not tangent at $e$ then
10:             Swap the order of $e.i$ and $e.i'$ in List.
11:         end if
12:     end if
13:     if $e.i$ = List$(K)$ or $e.i'$ = List$(K)$ then
14:         if $r^* > r_{e.i}(e.\alpha)$ then
15:             $\alpha^* \leftarrow e.\alpha$,   $r^* \leftarrow r_{e.i}(e.\alpha)$.    /* Update */
16:         end if
17:     end if
18: end for
19: return $\alpha^*$.
```

Lemma 1. *The number of intersections of all errors $r_i(\alpha)$, $i = 1, \ldots, N$, is bounded above by $16N^2 - 16N$.*

Proof. Let $r_i(\alpha)$ and $r_{i'}(\alpha)$ be two reprojection errors. The intersection of 2 constraints, respectively from $r_i(\alpha)$ and $r_{i'}(\alpha)$, gives rise to a quadratic function with at most 2 intersections. Thus the number of intersections between $r_i(\alpha)$ and $r_{i'}(\alpha)$ is limited to 32. For all $\binom{N}{2}$ pairs of reprojection errors, the total number of intersections is bounded above by $32N(N-1)/2 = 16N^2 - 16N$. $\square$

By maintaining **List** in a binary heap equipped with an auxiliary pointer array [27], looking up the median error (Step 14) and conducting swapping (Step 10) can be done in constant time at each event point—the sweep thus consumes $\mathcal{O}(N^2)$ time. The cost of Algorithm 3 is thus dominated by the sorting of $\mathcal{E}$ (Step 3), which needs $\mathcal{O}(N^2 \log N)$ time.

3.3. Convergence of Q-sweep to local minimum

In Secs. 3.1 and 3.2 we have described the details of subroutines `DESCENTDIR` and `STEPSIZE` in Q-sweep (Algorithm 1). Here, we establish the convergence of Q-sweep to a local minimum of LMS triangulation (9).

Theorem 3. *Q-sweep (Algorithm 1) converges to a local minimum of problem* (9).

Proof. Without loss of generality, define $K = \lceil N/2 \rceil$ as the median index. Given the current estimate $\hat{\mathbf{x}}$, let

$$r_{(1)}(\hat{\mathbf{x}}), \dots, r_{(K)}(\hat{\mathbf{x}}), \dots, r_{(N)}(\hat{\mathbf{x}}), \tag{21}$$

be the ordered residuals, where $r_{(p)}(\hat{\mathbf{x}}) \le r_{(q)}(\hat{\mathbf{x}})\ \forall p < q$.

Algorithm 1 terminates when the $\Delta\mathbf{x}$ returned by Algorithm 2 is null. By [9, Supp. material], $\hat{\mathbf{x}}$ is thus the global minimizer to the ℓ_∞ triangulation problem defined by the subset of data indexed by $\{(1), \dots, (K)\}$, i.e.,

$$\begin{aligned} \hat{\mathbf{x}} = & \arg\min_{\mathbf{x}\in\mathbb{R}^3} \ \underset{i\in\{(1),\dots,(K)\}}{\text{maximum}} \ r_i(\mathbf{x}) \\ & \text{s.t.} \quad \mathbf{c}_i^T\mathbf{x} + d_i > 0 \ \ \forall i \in \{(1),\dots,(K)\}. \end{aligned} \tag{22}$$

By assuming non-degeneracy (see proof of Theorem 1), there is an open subset $\mathcal{X}$ of $\mathbb{R}^3$ containing $\hat{\mathbf{x}}$ such that

$$r_{(K+1)}(\mathbf{x}) > r_{(i)}(\mathbf{x}), \ \ i = 1\dots, K, \ \ \forall \mathbf{x} \in \mathcal{X}, \tag{23}$$

i.e., $r_{(K+1)}(\mathbf{x})$ is always the $(K+1)$-th largest residual in $\mathcal{X}$. Thus, when the stopping criterion is achieved, Algorithm 1 terminates at a local minimum of the median error.

For $\mathbf{x}$ such that $\mathbf{c}_i^T\mathbf{x} + d_i > 0$, $r_i(\mathbf{x})$ is bounded below by 0. Thus, $\text{median}_i \ r_i(\mathbf{x})$ is also bounded below by 0 in the feasible region. Since each iteration of Q-sweep follows a descent direction and guarantees reduction in the median error, the algorithm converges to a local minimum. $\square$

4. Results

We compared Q-sweep against the following methods for the triangulation problem:

- Polyhedron collapse [9], which solves (1) and is thus non-robust—we regard it as the **control method**;
- Random sampling heuristic [1] with confidence 0.99 and outlier rate 0.5 for the stopping criterion;
- Ke & Kanade's approximate algorithm for LMS [16];
- Li's globally optimal method [17];
- The proposed Q-sweep method (Algorithm 1); and
- Q-sweep method with brute force search to solve (16) for the step size, in place of plane sweep.

Since the originators' codes were not publicly available, we implemented polyhedron collapse ourselves in Matlab—this is sufficient since it is is a very efficient algorithm. For random sampling, DLT was used as the minimal solver. For Ke & Kanade, the feasibility test was solved using Matlab's LP solver. For Li's method, polyhedron collapse was used for basis computations. For Q-sweep, the plane sweep routine (Algorithm 3) was implemented in C-mex.

All experiments were conducted on a standard machine with a 3.6GHz Intel i7 CPU and 16GB RAM. Unavoidably, differences in implementation and programming languages will affect the relative runtimes of the above methods—in Sec. 4.1, we will factor out the effects of these differences by examining asymptotic runtime on synthetic data.

Details on initialization To initialize Ke & Kanade and Q-sweep, we used the mid-point method (a closed form solver) [13] on two randomly selected measurements to find the initial $\hat{\mathbf{x}}$, which was then tested for feasibility.

4.1. Synthetic data experiments

Synthetic datasets for triangulation were generated as follows: a dataset contained 20 random scene points in $\mathbb{R}^3$, and N cameras $\{\mathbf{P}_i\}_{i=1}^N$ created with random poses with the condition that the scene points lay in front of the cameras; see Fig. 2(a). A triangulation instance was formed by projecting a scene point onto the cameras, and adding Gaussian noise of $\sigma = 3$ pixels to the image points. To create outliers, 30% of the image points were randomly selected, and Gaussian noise of $\sigma = 9$ pixels was added to them.

Fig. 2(b) shows the runtime of all methods plotted against the size N of the outlier-contaminated triangulation instances (the runtime of each N was averaged over the 20 instances in the dataset). Expectedly, the runtime of the global method increased very rapidly. The cost of random sampling and polyhedron collapse (non-robust) remained more or less constant. Ke & Kanade and Q-sweep gave similar asymptotic behaviour—note, however, that Ke & Kanade does not guarantee local optimality, unlike Q-sweep. Finally, the runtime of Q-sweep with brute force step size search also grew rapidly, illustrating the significant computational savings due to plane sweep (Algorithm 3).

Figs. 2(c) and 2(d) show the converged reprojection error of all the triangulation methods. All LMS algorithms recover the noise level for inliers (3 pixels) while polyhedron collapse is affected by outliers (errors over 10 pixels). As is evident in Fig. 2(d), Q-sweep (and brute force variant) gave

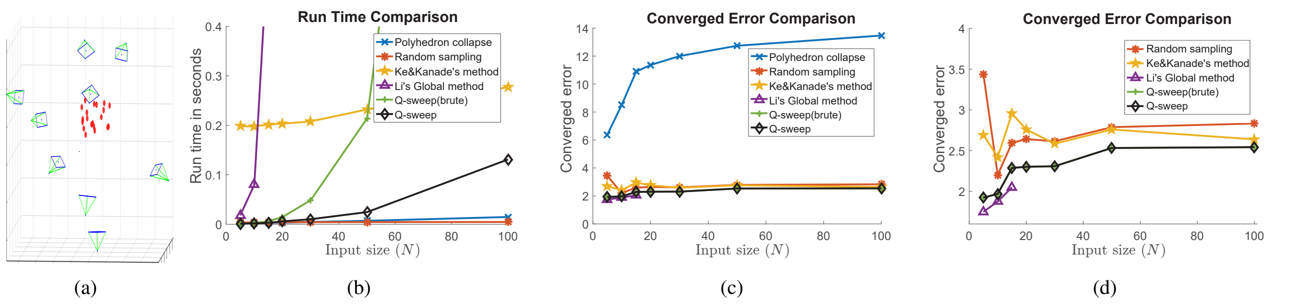


Figure 2. (a) A synthetic dataset with 20 scene points and $N = 9$ cameras. (b) Average runtime of all algorithms on synthetic triangulation instances plotted against input size N. (c) Average converged error for all methods. (d) Same as (c) but without polyhedron collapse.

	Temple ($\#p$=**6927**)		**Courtyard** ($\#p$=**59562**)		**University** ($\#p$=**19476**)		**Water Tower** ($\#p$=**58556**)	
Algorithm	Time (s)	OptErr	Time (s)	OptErr	Time (s)	OptErr	Time (s)	OptErr
Polyhedron collapse [9] (non-robust)	3.931	3.604	103.406	12.248	18.443	12.166	96.197	11.414
Random sampling (approx.)	8.927	0.935	109.660	1.906	26.797	4.564	101.905	3.151
Ke & Kanade [16] (approx.)	426.425	1.565	5257.441	4.072	1167.412	5.217	5084.582	4.131
Li [17] (global)	459.258	0.397	N/A	N/A	N/A	N/A	N/A	N/A
Q-sweep (locally optimal)	2.638	0.734	176.917	1.337	13.998	2.109	223.158	1.775
Q-sweep (brute force, locally optimal)	2.214	0.734	1338.820	1.337	24.586	2.109	2980.607	1.775

Table 1. Results on real datasets. $\#p$: total number of triangulation instances in the dataset. For information regarding the size N of the instances, see Panel (a) in Figs. 3 to 6. Time: *total* runtime for solving all triangulation instances (N/A if not finished by 2 hours); OptErr: the converged reprojection error, averaged over all instances, in pixels.

the lowest error among all approximate LMS algorithms, due to the ability of Q-sweep to converge to local minima.

4.2. Real data experiments

We used data from [10] (University of Washington, Alcatraz Courtyard, Alcatraz Water Tower) and [7] (Temple Ring). Olsson's SfM implementation [22] was used to estimate the camera poses and initial 3D structure (the input images were first resized to a factor 0.3). Then, SIFT [19, 29] was invoked on the original images to produce more feature correspondences, which were associated to form triangulation instances—these instances were contaminated by outliers, since they were not put through the SfM pipeline. The number of instances generated in this manner is shown as $\#p$ in Table 1, whereas Panel (a) in Figs. 3 to 6 plots the histogram of the sizes N of the instances—though most of the instances were small, a non-negligible number of them were of moderate to large sizes.

Table 1 summarizes the total runtime and average converged reprojection error for all methods.

Accuracy comparison On the smallest dataset (Temple), the global method expectedly gave the lowest error, followed by Q-sweep. However, the global method was not feasible on the other larger datasets; it was terminated after reaching the time limit of 2 hours.

On the other datasets, Q-sweep gave the lowest error, due to its ability to converge to local minima. This was followed by random sampling, Ke & Kanade, and polyhedron collapse. In Courtyard, University and Water Tower, the average converged error of polyhedron collapse was around 10 pixels, indicating the presence of outliers.

Panel (b) in Figs. 3 to 6 plots the histogram of converged errors for Q-sweep and polyhedron collapse. Evidently a large number of instances contain outliers, and the benefit of LMS triangulation with Q-sweep is clearly exhibited. Panels (c) and (d) in the figures show the triangulated points from both methods. Observe that there are much fewer spurious points in the results of Q-sweep.

Runtime comparison The recorded runtimes comply with the trends observed in the synthetic data experiments. We note again that step size search in Q-sweep with brute force was much more expensive than plane sweep.

5. Conclusions

Robust triangulation is a vital computer vision problem that has not been satisfactorily solved. The proposed Q-sweep algorithm fills a gap in currently available techniques for LMS triangulation. Unlike random sampling, it guarantees convergence to local minima, thus giving higher quality outcomes. Unlike global methods, Q-sweep is much more efficient and practical. At a higher level, our work illustrates useful adaptation of geometric algorithms to vision.

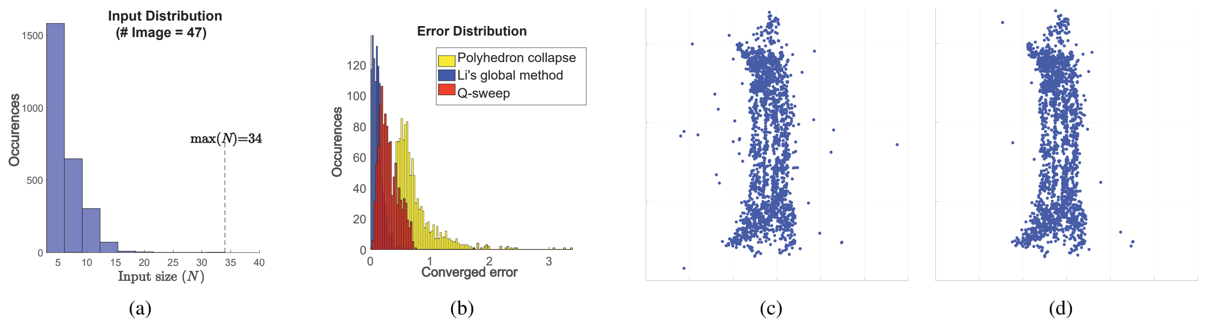


Figure 3. Results for **Temple Ring** dataset. (a) Distribution of instance sizes N. (b) Histogram of converged reprojection errors for polyhedron collapse, global method, and Q-sweep. (c)(d) 3D structure reconstructed respectively by polyhedron collapse and Q-sweep.

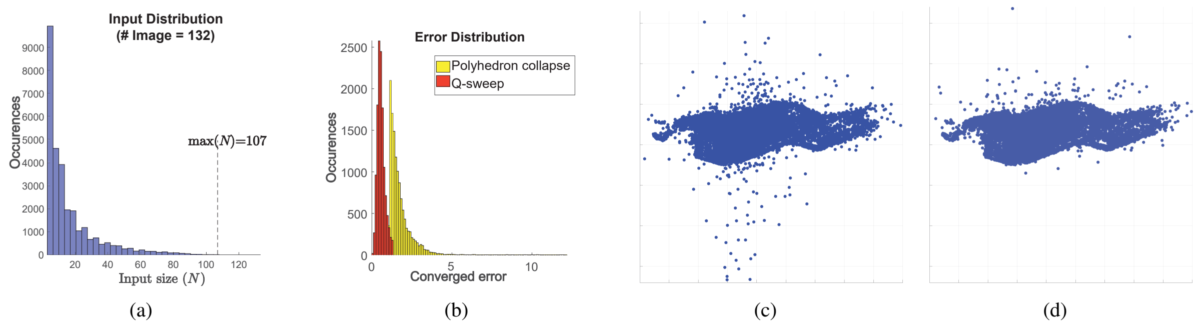


Figure 4. Results for **Alcatraz Courtyard** dataset. (a) Distribution of instance sizes N. (b) Histogram of converged reprojection errors for polyhedron collapse and Q-sweep. (c)(d) 3D structure reconstructed respectively by polyhedron collapse and Q-sweep.

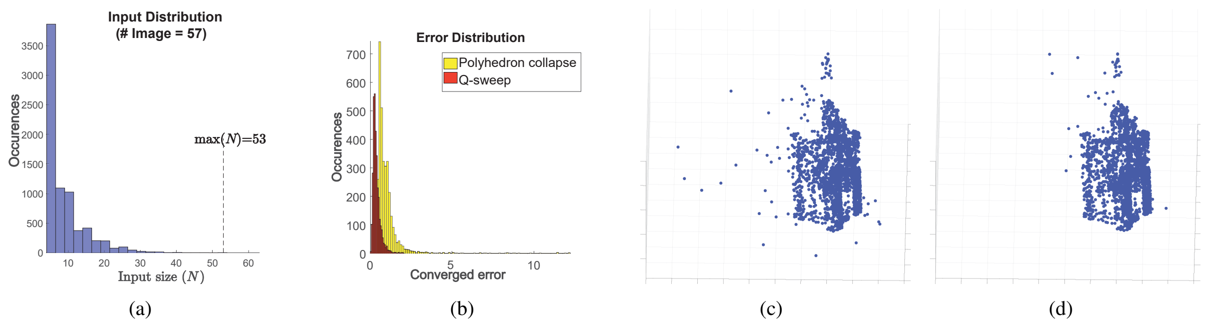


Figure 5. Results for **University of Washington** dataset. (a) Distribution of instance sizes N. (b) Histogram of converged reprojection errors for polyhedron collapse and Q-sweep. (c)(d) 3D structure reconstructed respectively by polyhedron collapse and Q-sweep.

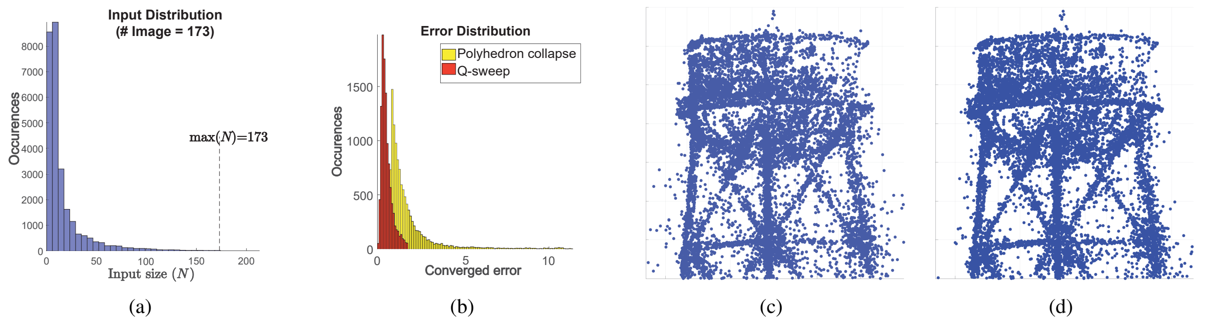


Figure 6. Results for **Alcatraz Water Tower** dataset. (a) Distribution of instance sizes N. (b) Histogram of converged reprojection errors for polyhedron collapse and Q-sweep. (c)(d) 3D structure reconstructed respectively by polyhedron collapse and Q-sweep.

References

[1] https://en.wikipedia.org/wiki/Random_sample_consensus.

[2] https://en.wikipedia.org/wiki/Linear-fractional_programming.

[3] S. Agarwal, N. Snavely, and S. M. Seitz. Fast algorithms for L_∞ problems in multiview geometry. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[4] J. Agulló. Exact algorithms for computing the least median of squares estimate in multiple linear regression. *Lecture Notes-Monograph Series*, pages 133–146, 1997.

[5] T.-J. Chin, P. Purkait, A. Eriksson, and D. Suter. Efficient globally optimal consensus maximisation with tree search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2413–2421, 2015.

[6] S. Choi, T. Kim, and W. Yu. Performance evaluation of ransac family. *Journal of Computer Vision*, 24(3):271–300, 1997.

[7] T. M. C. computer vision. Multi-view stereo dataset. http://vision.middlebury.edu/mview/data/.

[8] M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf. Computational geometry. In *Computational geometry*, pages 1–17. Springer, 2000.

[9] 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.

[10] 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.

[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.

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

[13] R. Hartley and F. Schaffalitzky. L_∞ minimization in geometric reconstruction problems. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2004.

[14] R. I. Hartley and P. Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

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

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

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

[18] H. Li. Consensus set maximization with guaranteed global optimality for robust geometry estimation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1074–1080. IEEE, 2009.

[19] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

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

[21] J. Nocedal and S. J. Wright. *Numerical optimization, second edition*. Springer, 2006.

[22] C. Olsson. Stable structure from motion for unordered image collections. http://www.maths.lth.se/matematiklth/personal/calle/sys_paper/sys_paper.html.

[23] C. Olsson, A. P. Eriksson, and F. Kahl. Efficient optimization for L_∞-problems using pseudoconvexity. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[24] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*, volume 589. John wiley & sons, 2005.

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

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

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

[28] A. J. Stromberg. Computing the exact least median of squares estimate and stability diagnostics in multiple linear regression. *SIAM Journal on Scientific Computing*, 14(6):1289–1299, 1993.

[29] A. Vedaldi. An open implementation of the sift detector and descriptor. *UCLA CSD*, 2007.

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

Chapter 5

Coresets for Triangulation

The work contained in this chapter has been published as the following paper

Qianggong Zhang and Tat-Jun Chin: Coresets for Triangulation. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 2017.

The published paper is available at
`http://ieeexplore.ieee.org/document/8031058/`

Statement of Authorship

Title of Paper	Coresets for Triangulation		
Publication Status	[✓ Published]	Accepted for Publication	
	Submitted for Publication	Unpublished and Unsubmitted work written in manuscript style	
Publication Details	Zhang, Q., & Chin, T. J. Coresets for Triangulation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11 September 2017.		

Principal Author

Name of Principal Author (Candidate)	Qianggong Zhang		
Contribution to the Paper	Developed the theoretical proof of key theorems, performed experiments and wrote manuscript.		
Overall percentage (%)	60%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	22/11/2017

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Tat-Jun Chin		
Contribution to the Paper	Supervised development of work and revised manuscript.		
Signature		Date	22/11/2017

Coresets for Triangulation

Qianggong Zhang and Tat-Jun Chin

Abstract—Multiple-view triangulation by ℓ_∞ minimisation has become established in computer vision. State-of-the-art ℓ_∞ triangulation algorithms exploit the quasiconvexity of the cost function to derive iterative update rules that deliver the global minimum. Such algorithms, however, can be computationally costly for large problem instances that contain many image measurements, e.g., from web-based photo sharing sites or long-term video recordings. In this paper, we prove that ℓ_∞ triangulation admits a *coreset approximation* scheme, which seeks small representative subsets of the input data called *coresets*. A coreset possesses the special property that the error of the ℓ_∞ solution on the coreset is within known bounds from the global minimum. We establish the necessary mathematical underpinnings of the coreset algorithm, specifically, by enacting the stopping criterion of the algorithm and proving that the resulting coreset gives the desired approximation accuracy. On large-scale triangulation problems, our method provides theoretically sound approximate solutions. Iterated until convergence, our coreset algorithm is also guaranteed to reach the true optimum. On practical datasets, we show that our technique can in fact attain the global minimiser much faster than current methods.

Index Terms—Coresets, approximation, generalised linear programming, multiple view geometry, triangulation.

1 Introduction

With the basic principles and algorithms of structure-from-motion well established, researchers have begun to consider large-scale reconstruction problems involving millions of input images. Arguably such large-scale problems, which arise from, e.g., photo sharing websites or long-term video observations in robotic exploration, are more common and practical. The significant problem sizes involved in such settings, however, compel practitioners to either use distributed computational architectures (e.g., GPU) to perform the required optimisation, or accept approximate solutions for the reconstruction.

This paper contains a *theoretical contribution* under the second paradigm. We introduce a *coreset approximation* scheme (more below) and prove its validity for multiple view 3D reconstruction, specifically for triangulation.

Triangulation is the task of estimating the 3D coordinates of a scene point from multiple 2D image observations of the point, given that the pose of the cameras are known [1]. The task is of fundamental importance to 3D vision, since it enables the recovery of the 3D structure of a scene. Whilst in theory structure and motion must be obtained simultaneously, there are many settings, such as large-scale reconstruction [2], [3] and SLAM [4], where the camera poses are first estimated with a sparse set of 3D points, before a denser scene structure is produced by triangulating other points using the estimated camera poses.

An established approach for triangulation is by ℓ_∞ minimisation [5]. Specifically, we seek the 3D coordinates that minimise the maximum reprojection error across all views. Unlike the sum of squared error function which contains multiple local minima, the maximum reprojection error function is quasiconvex and thus contains a single global minimum. Algorithms that take advantage of this property have been developed to solve such quasiconvex problems exactly [6], [7], [8], [9], [10], [11], [12], [13]. In particular, Agarwal et al. [10] showed that some of the most effective algorithms belong to the class of generalised fractional programming (GFP) methods [14], [15].

Although algorithms for ℓ_∞ triangulation have steadily improved, there is still room for improvement. In particular, on large-scale reconstruction problems or SLAM where there are usually a significant number of views per point (recall that the size of a triangulation problem is the number of 2D observations of a scene point), the computational cost of many of the algorithms can be considerable; we will demonstrate this in Section 5. A major reason is that the algorithms need to repeatedly solve convex programs to determine the update direction, which is of cubic complexity in worst case. It is thus of interest to investigate effective approximate algorithms.

1.1 Contributions

As alluded above, our main contribution in this paper is theoretical. Specifically, we prove that the ℓ_∞ triangulation problem admits a *coreset* approximation scheme [16], [17]. A coreset is a small representative subset of the data that approximates the overall distribution of the data. In the context of ℓ_∞ triangulation from N views, our algorithm iteratively accumulates a coreset, such that the error from solving the problem on the coreset is bounded within a factor of $(1+\epsilon)$ from the theoretically achievable minimum. Given a desired ϵ, we establish a stopping criterion for the algorithm such that the output coreset gives the required approximation accuracy. This provides a mathematically justified way to deal with large-scale problems where considering all available data may not be desirable or worthwhile.

Iterated until convergence, the coreset algorithm is guaranteed to attain the globally optimal solution. We experimentally demonstrate that the algorithm can in fact find the global minimiser much faster than many state-of-the-art ℓ_∞ triangulation methods. This superior performance was established on publicly available large scale 3D reconstruction datasets. From a practical standpoint, our algorithm thus

• *The authors are with the School of Computer Science, The University of Adelaide, Adelaide, SA, 5000, Australia.*
E-mail: {qianggong.zhang, tat-jun.chin}@adelaide.edu.au

provides a useful *anytime* behaviour, i.e., the algorithm can simply be run until convergence, or until the time budget is exhausted. In the latter case, we have a guaranteed bound of the approximation error w.r.t. the optimum.

The existence of coresets for quasiconvex vision problems was speculated by Li [18]. However, little progress has been made on this subject since. We provide a positive answer on one such problem. Our work is also one of the first to extend the idea of coresets in computational geometry [16], [17] to computer vision.

2 Background

Let $\{\mathbf{P}_i, \mathbf{u}_i\}_{i=1}^N$ be a set of data for triangulation, consisting of camera matrices $\mathbf{P}_i \in \mathbb{R}^{3\times 4}$ and observed image positions $\mathbf{u}_i \in \mathbb{R}^2$ of the same scene point $\mathbf{x} \in \mathbb{R}^3$. In this paper, by a "datum" we mean a specific camera and image point $\{\mathbf{P}_i, \mathbf{u}_i\}$. Let $\mathcal{X} = \{1, \dots, N\}$ index the set of data. The ℓ_∞ technique estimates $\mathbf{x}$ by minimising the maximum reprojection error

$$\begin{aligned} &\min_{\mathbf{x}} \max_{i\in\mathcal{X}} \;\; r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i), \\ &\text{subject to } \;\mathbf{P}_i^3\tilde{\mathbf{x}} > 0 \;\; \forall\, i \in \mathcal{X}. \end{aligned} \tag{1}$$

where

$$r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i) = \left\| \mathbf{u}_i - \frac{\mathbf{P}_i^{1:2}\tilde{\mathbf{x}}}{\mathbf{P}_i^3\tilde{\mathbf{x}}} \right\|_2 \tag{2}$$

is the reprojection error. Here, $\mathbf{P}_i^{1:2}$ and $\mathbf{P}_i^3$ respectively denote the first-two rows and third row of $\mathbf{P}_i$, and $\tilde{\mathbf{x}}$ is $\mathbf{x}$ in homogeneous coordinates. The reprojection error is basically the Euclidean distance between the observed point $\mathbf{u}_i$ and the projection of $\mathbf{x}$ onto the i-th image plane. The cheirality constraints $\mathbf{P}_i^3\tilde{\mathbf{x}} > 0 \; \forall i \in \mathcal{X}$ ensure that the estimated point lies in front of all the cameras.

Problem (1) belongs to a broader class of problems called generalised linear programs (GLP) [19]. Two properties of GLPs that will be useful later in this paper, are stated in the context of (1) as follows.

Property 1 (Monotonicity). *For any $\mathcal{C} \subseteq \mathcal{X}$,*

$$\min_{\mathbf{x}} \max_{i\in\mathcal{C}} \; r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i) \le \min_{\mathbf{x}} \max_{i\in\mathcal{X}} \; r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i) \tag{3}$$

given the appropriate cheirality contraints on both sides. □

Property 2 (Support set). *Let $\mathbf{x}^*$ and δ^* respectively be the minimiser and minimised objective value of (1). There exists a subset $\mathcal{B} \subseteq \mathcal{X}$ with $|\mathcal{B}| \le 4$, such that for any $\mathcal{C}$ that satisfies $\mathcal{B} \subseteq \mathcal{C} \subseteq \mathcal{X}$, the following holds*

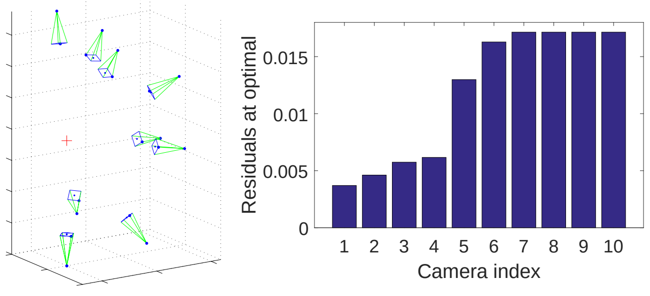
$$\begin{aligned} \delta^* &= \min_{\mathbf{x}} \max_{i\in\mathcal{B}} \; r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i) \\ &= \min_{\mathbf{x}} \max_{i\in\mathcal{C}} \; r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i) = \min_{\mathbf{x}} \max_{i\in\mathcal{X}} \; r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i) \end{aligned} \tag{4}$$

given the appropriate cheirality contraints. In fact, the three problems in (4) have the same minimiser $\mathbf{x}^*$. Further,

$$r(\mathbf{x}^* \mid \mathbf{P}_i, \mathbf{u}_i) = \delta^* \quad \textit{for any } i \in \mathcal{B}. \tag{5}$$

The subset $\mathcal{B}$ is called the "support set" of the problem. □

See [18], [19], [20] for details and proofs related to the above properties. Intuitively, (5) states that, at the solution of (1), the minimised maximum error occurs at the support set $\mathcal{B}$. Fig. 1 illustrates this property. Further, (4) states that solving (1) amounts to solving the same problem on $\mathcal{B}$. Many classical algorithms in computational geometry [21], [22], [23] exploit this property to solve GLPs.



(a) (b)

Fig. 1. Triangulating a point $\mathbf{x}$ observed in 10 views. The red '+' is the ℓ_∞ solution $\mathbf{x}^*$. Observe that there are four views/measurements with the same residual at $\mathbf{x}^*$. The index of the support set is thus $\mathcal{B} = \{7, 8, 9, 10\}$.

3 Coreset Algorithm

We first describe the coreset algorithm and focus on its operational behaviour, before embarking on a discussion of its convergence properties in Sec. 3.2 and the derivation of the coreset approximation bound in Sec. 3.3.

3.1 Main Operation

The coreset algorithm for ℓ_∞ triangulation is listed in Algorithm 1. The primary objective is to seek a representative subset $\mathcal{C}_s \subseteq \mathcal{X}$ of the data. This is accomplished by iteratively accumulating the data that should appear in the subset, where the datum that is selected for inclusion at each iteration is the most violating datum; see Step 6. The size of the subset, and equivalently the runtime of the algorithm, is controlled by the desired approximation error ϵ. To achieve, for e.g., a 1% approximation error, set $\epsilon = 0.01$.

Observe that Algorithm 1 is a *meta-algorithm*, since it requires executing a solver for (1) on the data subset indexed by the current subset $\mathcal{C}_t$ (see Steps 3 and 14). Any of the previous ℓ_∞ triangulation algorithms [6], [7], [8], [9], [10], [11] can be applied as the solver.

There are two terminating conditions for Algorithm 1:

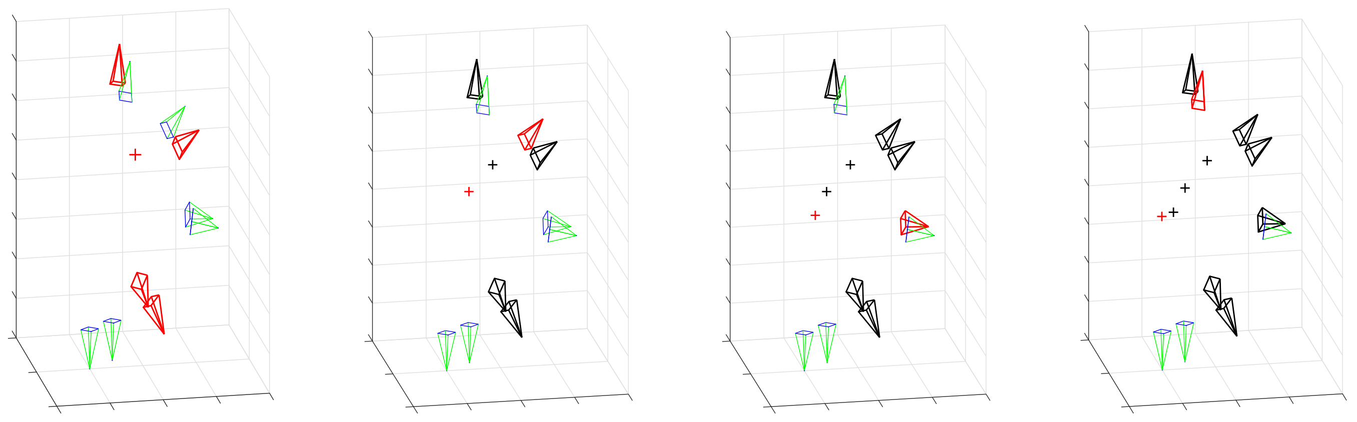
1) Iteration counter t reaches $\lceil 2/\epsilon \rceil$.

   In this case, the output $\mathcal{C}_s$ indexes a coreset with the desired approximation accuracy ϵ. Section 3.3 will establish the error bound for approximating (1) using the data indexed by $\mathcal{C}_s$.

2) The global minimiser has been found (Step 8).

   The satisfaction of the condition in Step 7 implies that $\mathcal{C}_{t-1}$ already contains the support set $\mathcal{B}$, since the largest error across all $\mathcal{X}$ is not larger than the value of (1) on the data indexed by $\mathcal{C}_{t-1}$; see Property 2.

To aid intuition, a sample partial run of Algorithm 1 is shown in Fig. 2.



(a) $t = 1$ (initialisation) (b) $t = 2$ (c) $t = 3$ (d) $t = 4$

Fig. 2. A sample run of Algorithm 1 on the data displayed in Fig. 1. (a) Four image measurements/camera viewpoints (in red) were selected to form the initial coreset $\mathcal{C}_1$. The current solution $\mathbf{x}_1$ is shown as a red cross. (b)–(d) Algorithm 1 progressively inserts new data into the coreset. Data in the current coreset is shown in black, and the newly inserted datum (chosen according to Step 6) is shown in red. Similary, the previous solutions $\mathbf{x}_s$ are shown in black, and the current estimate is shown in red. If terminated at $t = \lceil 2/\epsilon \rceil$, the estimate is a ϵ-approximation of the true optimum. Iterated until convergence, the global optimum is achieved. For anytime behaviour, the error bound can be backtracked (see Sec. 3.5) to obtain the approximation error of the last estimate at termination.

Algorithm 1 Coreset algorithm for ℓ_∞ triangulation (1).

```
Require: Input data $\{\mathbf{P}_i, \mathbf{u}_i\}_{i=1}^N$, approximation error $\epsilon$.
 1: Randomly permute the order of $\{\mathbf{P}_i, \mathbf{u}_i\}_{i=1}^N$, and define $\mathcal{X} = \{1, \dots, N\}$.
 2: $s \leftarrow 0,\ \gamma \leftarrow \infty,\ g \leftarrow 0,\ \mathcal{C}_1 \leftarrow \{1, 2, 3, 4\}$.
 3: $(\mathbf{x}_1, \delta_1) \leftarrow$ Minimiser and minimised value of (1) on data indexed by $\mathcal{C}_1$
 4: $t \leftarrow 2$
 5: while $t \leq \lceil 2/\epsilon \rceil$ do
 6:     $q \leftarrow \operatorname{argmax}_{i \in \mathcal{X}} r(\mathbf{x}_{t-1} \mid \mathbf{P}_i, \mathbf{u}_i)$.
 7:     if $r(\mathbf{x}_{t-1} \mid \mathbf{P}_q, \mathbf{u}_q) \leq \delta_{t-1}$ then
 8:         /* Found global minimum */
            $s \leftarrow t - 1,\ g \leftarrow 1,$ exit while loop.
 9:     end if
10:     if $r(\mathbf{x}_{t-1} \mid \mathbf{P}_q, \mathbf{u}_q) < \gamma$ then
11:         /* Found a better coreset */
            $s \leftarrow t - 1,\ \gamma \leftarrow r(\mathbf{x}_{t-1} \mid \mathbf{P}_q, \mathbf{u}_q)$.
12:     end if
13:     $\mathcal{C}_t \leftarrow \mathcal{C}_{t-1} \cup \{q\}$.
14:     $(\mathbf{x}_t, \delta_t) \leftarrow$ Minimiser and minimised value of (1) on data indexed by $\mathcal{C}_t$.
15:     $t \leftarrow t + 1$.
16: end while
17: if $g = 0$ then
18:     $q \leftarrow \operatorname{argmax}_{i \in \mathcal{X}} r(\mathbf{x}_{\lceil 2/\epsilon \rceil} \mid \mathbf{P}_i, \mathbf{u}_i)$.
19:     if $r(\mathbf{x}_{\lceil 2/\epsilon \rceil} \mid \mathbf{P}_q, \mathbf{u}_q) < \gamma$ then
20:         $s \leftarrow \lceil 2/\epsilon \rceil$.
21:     end if
22: end if
23: return $\mathcal{C}_s$, $\mathbf{x}_s$ and $\delta_s$.
```

3.2 Convergence to Global Minimum

If we are only interested in the global minimiser $\mathbf{x}^*$, then ϵ should be set to 0 (or a value small enough such that $\lceil 2/\epsilon \rceil \geq N - 3$). We prove that with this setting Algorithm 1 will always find $\mathbf{x}^*$ in a finite number of steps.

Theorem 1. *If $\lceil 2/\epsilon \rceil \geq N - 3$, then Algorithm 1 finds $\mathbf{x}^*$ in finite time.*

Proof. Let q be obtained according to Step 6.

- If $q \in \mathcal{C}_{t-1}$, then, by how $\mathbf{x}_{t-1}$ and δ_{t-1} were calculated in Step 14, the condition in Step 7 must be satisfied and $\mathbf{x}_{t-1}$ is the global minimiser.
- If $q \notin \mathcal{C}_{t-1}$ and the condition in Step 7 is satisfied, then equation (4) is implied and $\mathbf{x}_{t-1}$ is the global minimiser.
- If $q \notin \mathcal{C}_{t-1}$ and the condition in Step 7 is not satisfied, then Algorithm 1 will insert q into $\mathcal{C}_{t-1}$. There are at most N of such insertions (including the initial four insertions into $\mathcal{C}_1$). If $\lceil 2/\epsilon \rceil \geq N - 3$, in the worst case all of $\mathcal{X}$ will finally be inserted, and $\mathcal{C}_{\lceil 2/\epsilon \rceil} = \mathcal{X}$ and $\mathbf{x}_{\lceil 2/\epsilon \rceil} = \mathbf{x}^*$.

$\square$

Note that, whilst Algorithm 1 needs to repeatedly call an ℓ_∞ solver, it only invokes the solver on a small subset $\mathcal{C}_t$ of the data. Second, the way a new datum is selected (Step 6) to be inserted into $\mathcal{C}_{t-1}$—basically by choosing the most violating datum w.r.t. the current solution—enables $\mathcal{B}$ to be found quickly. Section 5 demonstrates that Algorithm 1 can in fact find the global minimiser much more efficiently than invoking an ℓ_∞ solver [6], [7], [8], [9], [10], [11] in "batch mode" on the whole input data $\mathcal{X}$.

Utilised as a global optimiser (i.e., set $\epsilon = 0$), Algorithm 1 can be viewed as a *Las Vegas* style randomised algorithm, since it always finds the correct result but a non-deterministic runtime. In addition, as indicated in the proof of Theorem 1, in the worst case Algorithm 1 takes N iterations since it considers each measurement at most once.

3.3 Coreset Approximation

Our primary contribution is to show that the subset $\mathcal{C}_s$ output by Algorithm 1 is a coreset of the ℓ_∞ triangulation problem (1). This is conveyed by the following theorem, which bounds the error of approximating (1) using $\mathcal{C}_s$.

Theorem 2. *Let $\mathcal{C}_s$, $\mathbf{x}_s$ and δ_s be the output of Algorithm 1. Then*

$$\max_{i\in\mathcal{X}} r(\mathbf{x}_s \mid \mathbf{P}_i, \mathbf{u}_i) \le (1+\epsilon)\delta^*, \tag{6}$$

where δ^* is the minimised objective value for problem (1) on the full data $\mathcal{X}$.

Intuitively, the above theorem states that the error of approximating $\mathbf{x}^*$ with $\mathbf{x}_s$ (the latter was computed using the $\mathcal{C}_s$ output by Algorithm 1) is at most $(1+\epsilon)$-times of the smallest possible error. This provides a mathematically justified way of dealing with large scale problems. The rest of this subsection is devoted to proving Theorem 2.

First, we define the set of geometrical quantities in Fig. 3. For an arbitrary camera matrix $\mathbf{P}$ with measurement $\mathbf{u}$, the reprojection error of a given $\mathbf{x}$ is

$$r(\mathbf{x} \mid \mathbf{P}, \mathbf{u}) = \left\| \mathbf{u} - \frac{\mathbf{P}^{1:2}\tilde{\mathbf{x}}}{\mathbf{P}^3\tilde{\mathbf{x}}} \right\|_2 = \|\mathbf{u} - f_{\mathbf{P}}(\mathbf{x})\|_2\,, \tag{7}$$

where $f_{\mathbf{P}}(\mathbf{x})$ is the projection of $\mathbf{x}$ onto the image. Given a set of data $\{\mathbf{P}_i, \mathbf{u}_i\}_{i=1}^N$, let $\mathbf{x}^*$ be the global minimiser of (1). Define a disc on the image plane with centre $\mathbf{u}$ and radius $r(\mathbf{x}^* \mid \mathbf{P}, \mathbf{u})$; backprojecting this disc creates a solid elliptic cone $c_{\mathbf{P}}(\mathbf{x}^*)$. Define $h_{\mathbf{P}}(\mathbf{x}^*)$ as the tangent plane on the surface of $c_{\mathbf{P}}(\mathbf{x}^*)$ that contains $\mathbf{x}^*$.

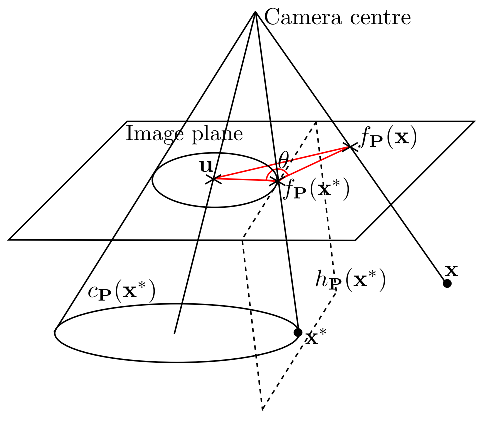

Camera centre
Image plane
$\mathbf{u}$
θ
$f_{\mathbf{P}}(\mathbf{x})$
$f_{\mathbf{P}}(\mathbf{x}^*)$
$c_{\mathbf{P}}(\mathbf{x}^*)$
$h_{\mathbf{P}}(\mathbf{x}^*)$
$\mathbf{x}$
$\mathbf{x}^*$

Fig. 3. Definition of several geometrical quantities for ℓ_∞ triangulation.

We now establish several intermediate results. In the following, we consider only $\mathbf{x} \in \mathbb{R}^3$ that lies in front of the camera, i.e., $\mathbf{x}$ is never on the same side of the image plane as the camera centre.

Lemma 1. *$\mathbf{x}$ is inside $c_{\mathbf{P}}(\mathbf{x}^*)$ iff $r(\mathbf{x} \mid \mathbf{P}, \mathbf{u}) < r(\mathbf{x}^* \mid \mathbf{P}, \mathbf{u})$.*

Lemma 2. *$\mathbf{x}$ is on the same side of $h_{\mathbf{P}}(\mathbf{x}^*)$ as $\mathbf{u}$ iff the angle θ formed by the three points $\mathbf{u} : f_{\mathbf{P}}(\mathbf{x}^*) : f_{\mathbf{P}}(\mathbf{x})$ is acute, i.e., $\theta < 90°$.*

Lemma 3. *$\mathbf{x}$ is on the opposite side of $h_{\mathbf{P}}(\mathbf{x}^*)$ as $\mathbf{u}$ iff the angle θ formed by the three points $\mathbf{u} : f_{\mathbf{P}}(\mathbf{x}^*) : f_{\mathbf{P}}(\mathbf{x})$ is obtuse, i.e., $\theta > 90°$.*

The above three lemmata can be proven easily by inspecting Fig. 3. As an extension of Lemma 2, the following statement can be made.

Lemma 4. *The angle $\angle(\mathbf{u} : f_{\mathbf{P}}(\mathbf{x}^*) : f_{\mathbf{P}}(\mathbf{x}))$ is acute iff there is a line segment*

$$S = \{\mathbf{x}' \mid \mathbf{x}' = \mathbf{x}^* + \alpha(\mathbf{x} - \mathbf{x}^*), 0 \le \alpha < 1\} \tag{8}$$

(i.e., S has a start point at $\mathbf{x}^*$ and lies along vector $\mathbf{x} - \mathbf{x}^*$) such that any point $\mathbf{x}'$ on S will give a strictly smaller reprojection error than $\mathbf{x}^*$, i.e.,

$$r(\mathbf{x}' \mid \mathbf{P}, \mathbf{u}) < r(\mathbf{x}^* \mid \mathbf{P}, \mathbf{u}) \quad \forall\, \mathbf{x}' \in S. \tag{9}$$

Proof. If $\angle(\mathbf{u} : f_{\mathbf{P}}(\mathbf{x}^*) : f_{\mathbf{P}}(\mathbf{x}))$ is acute, then from Lemma 2, $\mathbf{x}$ must be on the same side of $h_{\mathbf{P}}(\mathbf{x}^*)$ as $\mathbf{u}$. The line segment joining $\mathbf{x}$ and $\mathbf{x}^*$ must thus intersect the inside of $c_{\mathbf{P}}(\mathbf{x}^*)$; this intersection gives S. Since S is inside $c_{\mathbf{P}}(\mathbf{x}^*)$, from Lemma 1 any $\mathbf{x}' \in S$ must give a strictly smaller reprojection error than $\mathbf{x}^*$.

The reverse direction can be proven by realising that any $\mathbf{x}'$ which gives a strictly smaller reprojection error than $\mathbf{x}^*$ must lie in $c_{\mathbf{p}}(\mathbf{x}^*)$. Any line segment that joins $\mathbf{x}^*$ and $\mathbf{x}$ with $\mathbf{x}'$ in the middle must lie on the same side of $h_{\mathbf{P}}(\mathbf{x}^*)$ as $\mathbf{u}$. From Lemma 2, $\angle(\mathbf{u} : f_{\mathbf{P}}(\mathbf{x}^*) : f_{\mathbf{P}}(\mathbf{x}))$ must be acute. □

Of central importance is the following result.

Lemma 5. *Let $\{\mathbf{P}_i, \mathbf{u}_i\}_{i=1}^N$ be a set of data, $\mathbf{x}^*$ be the global minimiser of (1) on the data, and δ^* be the minimised value of (1). For an arbitrary $\mathbf{x} \in \mathbb{R}^3$ in front of the camera, there exists a datum $\{\mathbf{P}_j, \mathbf{u}_j\}$ such that*

$$\angle(\mathbf{u}_j : f_{\mathbf{P}_j}(\mathbf{x}^*) : f_{\mathbf{P}_j}(\mathbf{x})) > 90°. \tag{10}$$

Via the cosine rule, the above inequality can be re-expressed as

$$r(\mathbf{x} \mid \mathbf{P}_j, \mathbf{u}_j)^2 \ge \|f_{\mathbf{P}_j}(\mathbf{x}) - f_{\mathbf{P}_j}(\mathbf{x}^*)\|_2^2 + r(\mathbf{x}^* \mid \mathbf{P}_j, \mathbf{u}_j)^2. \tag{11}$$

Proof. From (5), at the solution $\mathbf{x}^*$ there must exist a support set $\mathcal{B}$ such that the data indexed by $\mathcal{B}$ attain the minimised maximum residual δ^*.

It is sufficient to consider $\mathcal{B}$. We aim to contradict the following assumption:

$$\exists \mathbf{x} \quad \text{s.t.} \quad \angle(\mathbf{u}_i : f_{\mathbf{P}_i}(\mathbf{x}^*) : f_{\mathbf{P}_i}(\mathbf{x})) < 90° \quad \forall\ i \in \mathcal{B}. \tag{12}$$

Given an $\mathbf{x}$ that satisfies (12), Lemma 4 states that for each $i \in \mathcal{B}$, there is a line segment S_i that lies completely inside $c_{\mathbf{P}_i}(\mathbf{x}^*)$.

Amongst all the segments S_i, $i \in \mathcal{B}$, pick the shortest one and call it $\bar{S}$. The segment $\bar{S}$ must lie simultaneously in all of the cones $c_{\mathbf{P}_i}(\mathbf{x}^*)$, $i \in \mathcal{B}$ (recall from (8) that all S_i's begin at $\mathbf{x}^*$ and lie along vector $\mathbf{x} - \mathbf{x}^*$). Any $\mathbf{x}' \in \bar{S}$ must thus yield a strictly smaller reprojection error than $\mathbf{x}^*$ for all $\{\mathbf{P}_i, \mathbf{u}_i\}$, $i \in \mathcal{B}$. This contradicts that $\mathcal{B}$ is a support set, thus falsifying (12).

The falsity of (12) implies that for an arbitrary $\mathbf{x}$, there must be an $i \in \mathcal{B}$ such that $\angle(\mathbf{u}_i : f_{\mathbf{P}_i}(\mathbf{x}^*) : f_{\mathbf{P}_i}(\mathbf{x})) > 90°$—set j as that i. □

Given the above results, we adapt Bădoiu and Clarkson's derivation [17] to yield the inequality (6) for triangulation. Define

$$\begin{aligned} \bar{\delta} &:= (1+\epsilon)\delta^*, \\ \lambda_t &:= \delta_t/\bar{\delta}, \\ k^{\mathbf{P}} &:= \|f_{\mathbf{P}}(\mathbf{x}_1) - f_{\mathbf{P}}(\mathbf{x}_2)\|_2\,. \end{aligned} \tag{13}$$

Note that $0 \leq \lambda_t \leq 1$. Further, since $\mathcal{C}_t \subseteq \mathcal{X}$, from Property 1, $\delta_t \leq \delta^*$, thus

$$\lambda_t \leq \delta^*/\bar{\delta} = 1/(1+\epsilon). \tag{14}$$

We aim to disprove the following assumption:

$$\nexists t \geq 2 \text{ such that } \max_{i \in \mathcal{X}} r(\mathbf{x}_t \mid \mathbf{P}_i, \mathbf{u}_i) \leq (1+\epsilon)\delta^*. \tag{15}$$

In words, (15) effectively states that none of the $\mathcal{C}_t$ accumulated throughout the iterations in Algorithm 1 gives a $(1+\epsilon)$ approximation to (1).

For any $t \geq 2$, Lemma 5 states that there exists a $j \in \mathcal{C}_{t-1}$ such that

$$\begin{aligned} & r(\mathbf{x}_t \mid \mathbf{P}_j, \mathbf{u}_j)^2 \\ & \quad \geq \left\| f_{\mathbf{P}_j}(\mathbf{x}_t) - f_{\mathbf{P}_j}(\mathbf{x}_{t-1}) \right\|_2^2 + r(\mathbf{x}_{t-1} \mid \mathbf{P}_j, \mathbf{u}_j)^2 \\ & \quad = (k_t^{\mathbf{P}_j})^2 + \delta_{t-1}^2 \end{aligned} \tag{16}$$

(recall that j indexes a datum in the support set of the data indexed by $\mathcal{C}_{t-1}$, thus $r(\mathbf{x}_{t-1} \mid \mathbf{P}_j, \mathbf{u}_j) = \delta_{t-1}$). Then

$$r(\mathbf{x}_t \mid \mathbf{P}_j, \mathbf{u}_j) \geq \sqrt{\lambda_{t-1}^2 \bar{\delta}^2 + (k_t^{\mathbf{P}_j})^2}. \tag{17}$$

For the q chosen in iteration t (Step 6 in Algorithm 1), via triangle inequality

$$\begin{aligned} & r(\mathbf{x}_{t-1} \mid \mathbf{P}_q, \mathbf{u}_q) \\ & \quad \leq r(\mathbf{x}_t \mid \mathbf{P}_q, \mathbf{u}_q) + \left\| f_{\mathbf{P}_q}(\mathbf{x}_t) - f_{\mathbf{P}_q}(\mathbf{x}_{t-1}) \right\|_2, \end{aligned} \tag{18}$$

implying that

$$\begin{aligned} & r(\mathbf{x}_t \mid \mathbf{P}_q, \mathbf{u}_q) \\ & \quad \geq r(\mathbf{x}_{t-1} \mid \mathbf{P}_q, \mathbf{u}_q) - \left\| f_{\mathbf{P}_q}(\mathbf{x}_t) - f_{\mathbf{P}_q}(\mathbf{x}_{t-1}) \right\|_2 \\ & \quad = r(\mathbf{x}_{t-1} \mid \mathbf{P}_q, \mathbf{u}_q) - k_t^{\mathbf{P}_q} > \bar{\delta} - k_t^{\mathbf{P}_q}. \end{aligned} \tag{19}$$

The last inequality follows from the assumption in (15) which states that none of the $\mathcal{C}_t$ for $t \geq 2$ gives a $(1+\epsilon)$ approximation of (1).

Since both j and q are in $\mathcal{C}_t$, by combining (17) and (19) we obtain

$$\begin{aligned} \lambda_t \bar{\delta} = \delta_t & \geq \max\left(r(\mathbf{x}_t \mid \mathbf{P}_j, \mathbf{u}_j),\ r(\mathbf{x}_t \mid \mathbf{P}_q, \mathbf{u}_q)\right) \\ & \geq \max\left(\sqrt{\lambda_{t-1}^2 \bar{\delta}^2 + (k_t^{\mathbf{P}_j})^2},\ \bar{\delta} - k_t^{\mathbf{P}_q}\right). \end{aligned} \tag{20}$$

Recall the definition of $k_t^{\mathbf{P}_j}$ and $k_t^{\mathbf{P}_q}$:

$$k_t^{\mathbf{P}_j} = \left\| f_{\mathbf{P}_j}(\mathbf{x}_t) - f_{\mathbf{P}_j}(\mathbf{x}_{t-1}) \right\|_2, \tag{21}$$

$$k_t^{\mathbf{P}_q} = \left\| f_{\mathbf{P}_q}(\mathbf{x}_t) - f_{\mathbf{P}_q}(\mathbf{x}_{t-1}) \right\|_2. \tag{22}$$

Geometrically, these quantities represent the 2D projection, respectively on cameras j and q, of the 3D shift $\|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2$ between the current and previous estimates.

At this juncture, the rest of the proof diverges based on the following conditions:

$$k_t^{\mathbf{P}_j} \geq k_t^{\mathbf{P}_q} \quad \text{or} \quad k_t^{\mathbf{P}_j} < k_t^{\mathbf{P}_q}. \tag{23}$$

Condition 1. $k_t^{\mathbf{P}_j} \geq k_t^{\mathbf{P}_q}$.

Under Condition 1, and following from (20),

$$\begin{aligned} \lambda_t \bar{\delta} & \geq \max\left(\sqrt{\lambda_{t-1}^2 \bar{\delta}^2 + (k_t^{\mathbf{P}_j})^2},\ \bar{\delta} - k_t^{\mathbf{P}_q}\right) \\ & \geq \max\left(\sqrt{\lambda_{t-1}^2 \bar{\delta}^2 + (k_t^{\mathbf{P}_j})^2},\ \bar{\delta} - k_t^{\mathbf{P}_j}\right), \end{aligned} \tag{24}$$

where the second inequality follows since $k_t^{\mathbf{P}_j}$ and $k_t^{\mathbf{P}_q}$ are both non-negative quantities. Interpreting the arguments in the second max

$$\sqrt{\lambda_{t-1}^2 \bar{\delta}^2 + (k_t^{\mathbf{P}_j})^2} \quad \text{and} \quad \bar{\delta} - k_t^{\mathbf{P}_j} \tag{25}$$

as two functions of $k_t^{\mathbf{P}_j}$, observe that the first function increases with $k_t^{\mathbf{P}_j}$ whilst the second decreases with $k_t^{\mathbf{P}_j}$. Therefore, the RHS of (24) achieves its minimum when

$$\sqrt{\lambda_{t-1}^2 \bar{\delta}^2 + (k_t^{\mathbf{P}_j})^2} = \bar{\delta} - k_t^{\mathbf{P}_j}. \tag{26}$$

Solving (26) for $k_t^{\mathbf{P}_j}$ and replacing it in (24), we arrive at

$$\lambda_t \bar{\delta} \geq \frac{1 + \lambda_{t-1}^2}{2} \bar{\delta} \implies 1 - \lambda_t \leq \frac{1 - \lambda_{t-1}^2}{2}. \tag{27}$$

The second inequality in (27) can "inverted" as

$$\frac{1}{1-\lambda_t} \geq \frac{2}{(1-\lambda_{t-1})(1+\lambda_{t-1})} \tag{28}$$

$$= \frac{1}{1-\lambda_{t-1}} + \frac{1}{1+\lambda_{t+1}} > \frac{1}{1-\lambda_{t-1}} + \frac{1}{2}, \tag{29}$$

where the last step is due to $\lambda_{t-1} < 1$. By recursively expanding the above from $t, t-1, \ldots, 2$, and recalling that $0 \leq \lambda_1 \leq 1$, we obtain

$$\frac{1}{1-\lambda_t} > \frac{1}{1-\lambda_1} + \frac{t-1}{2} > 1 + \frac{t-1}{2}, \tag{30}$$

which implies

$$\lambda_t > 1 - \frac{2}{1+t}. \tag{31}$$

For $t = \lceil 2/\epsilon \rceil + 1$ the last inequality reduces to

$$\lambda_{\lceil 2/\epsilon \rceil + 1} > 1 - \frac{2}{1 + (2/\epsilon + 1)} = \frac{1}{1+\epsilon} \tag{32}$$

which contradicts (14). Thus, (15) cannot be true. Whilst it may be disconcerting that we have chosen an iteration count $t = \lceil 2/\epsilon \rceil + 1$ that does not exist in Algorithm 1, for the purpose of a theoretical argument we can always arbitrarily extend the algorithm by one iteration.

Since (15) is false, there must be a $2 \leq t \leq \lceil 2/\epsilon \rceil$ (say t^*) that yields a $(1+\epsilon)$ approximation. The set index by $\mathcal{C}_s$, which satisfies

$$\max_{i \in \mathcal{X}} r(\mathbf{x}_s \mid \mathbf{P}_i, \mathbf{u}_i) \leq \max_{i \in \mathcal{X}} r(\mathbf{x}_{t^*} \mid \mathbf{P}_i, \mathbf{u}_i) \leq (1+\epsilon)\delta^*, \tag{33}$$

is thus a coreset. Assuming Condition 1 is always satisfied, therefore, the proof for Theorem 2 is complete.

Condition 2. $k_t^{\mathbf{P}_j} < k_t^{\mathbf{P}_q}$.

The above derivations for Condition 1 unfortunately do not cover Condition 2. In other words, if Condition 2 occurs during the iterations in Algorithm 1, we cannot guarantee that the output coreset $\mathcal{C}_s$ satisfies Theorem 2.

Fortunately this deficiency can be rectified by a small tweak to Algorithm 1. Specifically, we replace Steps 13 to 15 in the main algorithm with the slightly more elaborate steps in Algorithm 2. Note that index j in Algorithm 2 is appropriately chosen from $\mathcal{C}_{t-1}$ according to Lemma 5. Intuitively, the modification causes Algorithm 1 to "skip a step" whenever Condition 2 presents itself; namely, the most

Algorithm 2 Pseudo-code to replace Steps 13 to 15 in Algorithm 1 to accommodate Condition 2.

```
1: if $k_t^{\mathbf{P}_j} \geq k_t^{\mathbf{P}_q}$ then
2:     $\mathcal{C}_t \leftarrow \mathcal{C}_{t-1} \cup \{q\}$.
3:     $(\mathbf{x}_t, \delta_t) \leftarrow$ Minimiser and minimised value of (1) on data indexed by $\mathcal{C}_t$.
4:     $t \leftarrow t + 1$.
5: else
6:     $\mathcal{C}_{t-1} \leftarrow \mathcal{C}_{t-1} \cup \{q\}$.
7:     $(\mathbf{x}_{t-1}, \delta_{t-1}) \leftarrow$ Minimiser and minimised value of (1) on data indexed by $\mathcal{C}_{t-1}$.
8: end if
```

violating datum q is still inserted into the coreset, but the iteration counter is not incremented.

With the modification above and by Property 1, we have that two successive coresets $\mathcal{C}_{t-1}$ and $\mathcal{C}_t$ give

$$\delta_{t-1} \leq \delta_t, \tag{34}$$

since $\mathcal{C}_{t-1} \subseteq \mathcal{C}_t$. Thus, all the derivations starting from (20) will hold, and the output coreset $\mathcal{C}_s$ from Algorithm 1 with the above modification will always satisfy Theorem 2.

3.4 Size of Output Coreset and Runtime Analysis

With the modification to account for Condition 2, we must anticipate that in general

$$|\mathcal{C}_t| - |\mathcal{C}_{t-1}| \geq 1, \tag{35}$$

i.e., there could be occurrences of Condition 2 between any two successive increments to the iteration counter. This also implies that the size of the output coreset $\mathcal{C}_s$ is

$$\lceil 2/\epsilon \rceil + 3 + V, \tag{36}$$

where V is the total number of occurrences of Condition 2 throughout Algorithm 1 (recall that $\mathcal{C}_1$ is initialised already with four of the available measurements).

To facilitate the analysis of the runtime, define α as the probability of Condition 2 occurring at a particular iteration of the main loop (thus, $V = \alpha\lceil 2/\epsilon \rceil/(1-\alpha)$). Therefore, the number of times the main loop is traversed is

$$\lceil 2/\epsilon \rceil/(1-\alpha). \tag{37}$$

We argue that the value of α is dependent mainly on the distribution of the cameras and the structure of the scene, rather than on the number of measurements N itself (see evidence in Sec. 4). Under this assumption, the number of effective iterations of Algorithm 1, and hence the size of the output coreset, depends only on ϵ.

Of course, the actual runtime of Algorithm 1 depends closely on the routine used to solve (1) at each iteration. Many previous studies have shown that (1) can be solved efficiently [6], [9], [10], more so since the solver need only be invoked on a small subset $\mathcal{C}_t$ at each iteration in Algorithm 1. Sec. 5 will investigate the actual runtimes of Algorithm 1 on real image datasets for 3D reconstruction.

3.5 Error Backtracking for Anytime Operation

By anytime mode, we mean the allowance to stop Algorithm 1 prematurely (i.e., before any of the terminating conditions are met) with the ability to bound the approximation error of the current best estimate $\mathbf{x}_s$ w.r.t. $\mathbf{x}^*$.

If Algorithm 1 is run up to $t = \lceil 2/\epsilon \rceil$ (which implies that global convergence has not occurred before that), then the bound (6) holds. To facilitate analysis with non-integral t, we can equivalently state that if Algorithm 1 is run until *at least* $t = 2/\epsilon$, then (6) holds. Inverting the relationship between t and ϵ, we can restate the bound as

$$\max_{i \in \mathcal{X}} r(\mathbf{x}_s \mid \mathbf{P}_i, \mathbf{u}_i) \leq (1 + 2/t)\delta^*, \tag{38}$$

i.e., if Algorithm 1 is run up until an arbitrary t, we can expect a $(2/t)$-factor approximation to (1) (note that $\mathbf{x}_s$ in this case is the best estimate up to iteration t).

Care must be taken to spell out ”running up until an arbitrary t”. By this, we mean running the main loop (Steps 5 to 16) in its entirely (including the modification summarised in Algorithm 2) under that particular t value, *and then* conducting the post-hoc refinement (Steps 17 to 22) such that the estimate $\mathbf{x}_t$ from the last coreset $\mathcal{C}_t$ has a chance to be used to update the incumbent $\mathbf{x}_s$. If this last update is not attempted, then (38) is not guaranteed to hold.

4 Validation on Synthetic Data

Here we validate our theoretical results above on synthetically generated data for triangulation.

4.1 Data Generation

We synthesised four types of camera pose distributions:

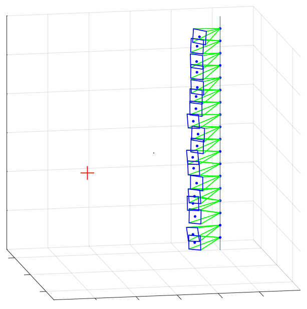
- Type A: Camera positions were on a straight line.
- Type B: Camera positions were randomly distributed.
- Type C: Camera positions were on a circle.
- Type D: Stereo cameras with fixed baselines and positions randomly distributed.

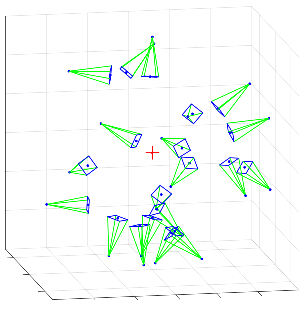
For Types B and D, the camera orientations were randomly generated. For Types A and C, angular noise was added to the orientation/rotation matrices. In all cases, the cameras could observe the 3D point to respect cheirality. See Fig. 4 for sample data instances generated. Type A simulates a robotic exploration scenario where the robot views a scene from a directed trajectory [4], Type B simulates large-scale 3D reconstruction from crowd-sourced images [2], Type C simulates the usage of a rotating platform for 3D modelling [24], and Type D simulates large-scale 3D reconstruction using stereo cameras [25].

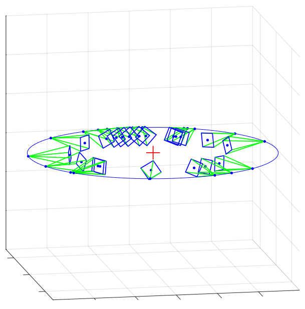
For each camera distribution, N image measurement/camera matrix pairs $\{\mathbf{P}_i, \mathbf{u}_i\}_{i=1}^N$ are generated by projecting the 3D point onto each camera and corrupting the projected point with Gaussian noise σ = 10 pixels.

4.2 Validation of Approximation Accuracy

To experimentally validate Theorem 2, one instance of each of the camera distribution type in Fig. 4 with $N = 100$ views were generated. For each camera distribution, 200 3D scene points were created (in a way that the 3D points are observable in all N cameras) and projected onto the N cameras. This created a total of 200 triangulation instances (1).


(a) Type A


(b) Type B


(c) Type C

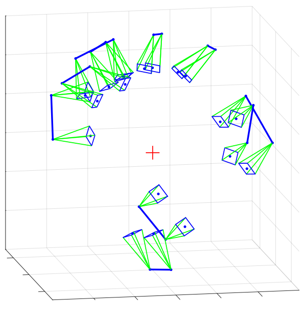

(d) Type D

Fig. 4. The four types of synthetically generated triangulation instances. Type A: Camera positions are on a straight line; Type B: Camera positions are randomly distributed; Type C: Camera positions are on a circle; Type D: Stereo cameras with fixed baselines and positions randomly distributed. In all cases, the cameras are roughly oriented towards the 3D point to respect cheirality.

On each triangulation instance, we executed Algorithm 1. The approximation error ratio

$$\frac{\max_{i \in \mathcal{X}} \; r(\mathbf{x}_s \mid \mathbf{P}_i, \mathbf{u}_i)}{\delta^*} \tag{39}$$

achieved by the current estimate $\mathbf{x}_s$ at each t is plotted; these are shown as red curves in Fig. 5 (one curve for each triangulation instance). Note that the horizontal axis begins at $t = 2$, since the bound (38) is not guaranteed to hold for the initial coreset $\mathcal{C}_1$. Note also that due to the allowance of "skipping" by inserting Algorithm 2 into Algorithm 1, each t can involve several updates to $\mathbf{x}_s$; in Fig. 5 we plotted the error ratio pertaining to final $\mathbf{x}_s$ in each t.

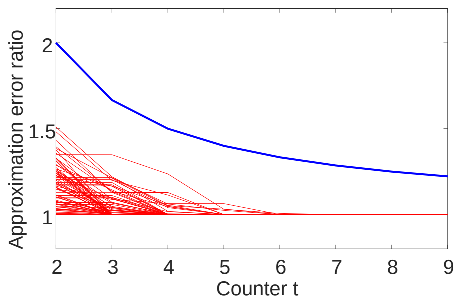


Fig. 5. Approximation error ratios (39) across counter t (red curves) for 200 synthetically generated triangulation instances. The blue curve is the upper bound on the error ratio $(1 + 2/t)$ as predicted by Theorem 2.

By the backtracking formula (38), the approximation error ratios should lie below the curve

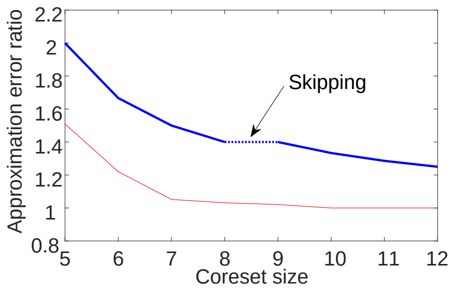
$$1 + 2/t; \tag{40}$$

this curve is plotted in blue in Fig. 5. As predicted, the bound is respected across all t. In Sec. 5, we will further validate Theorem 2 using real image data.

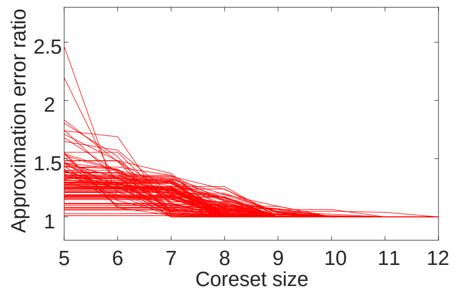
4.2.1 Illustrating Effect of Condition 2

The effect of Condition 2 on Algorithm 1 is demonstrated in Fig. 6. On one of the synthetic triangulation instances, Fig. 6(a) plots the approximation error ratio (39) against the coreset size, as the coreset is being accumulated in Algorithm 1 (the horizontal axis thus begins at 5 since the initial coreset $\mathcal{C}_1$ has size 4). The effects of skipping on the ratio bound (40) to account for Condition 2 is also shown; specifically, as the coreset is increased from size 8 to 9, Condition 2 occured and t was not incremented. Hence the bound (40) does not decrease between these two steps.

Fig. 6(b) plots the approximation error ratios for all the synthetic triangulation instances from Fig. 5 against coreset size. In this figure, since Condition 2 occurred at different iterations for the respective problem instances, the bounding curve is not plotted. Observe that all the problem instances converged to a coreset size that is not very much larger than the value of t at the time of convergence, cf. Fig. 5.



(a)



(b)

Fig. 6. (a) Approximation error ratio (39) plotted against coreset size for one of the synthetic data instances. In this instance, Condition 2 occurred as the coreset was increased from size 8 to 9, thus the bound (40) does not decrease between these two iterations. (b) Approximation error ratio plotted against coreset size for all the synthetic data instances. Since Condition 2 occurred at different iterations for the respective problem instances, the bounding curve is not plotted.

4.2.2 *Probability of Condition 2*

We demonstrate that the probability α of the occurrence of Condition 2 in any problem instance is mainly affected by the way the camera poses are distributed, and is not a factor of problem size N. For a complete execution of Algorithm 1 on the following particular problem instances, we obtain α empirically as the ratio of the number of occurrence of Condition 2 over the effective number of iterations in the algorithm.

The experimental settings were as follows: for each type of camera distribution, 200 3D points were randomly generated and projected onto N views, where N was varied from 100 to $10,000$. On each problem instance, Algorithm 1 was executed 20 times (with random initialisations) and the α values were recorded. Fig. 7 shows the average α over all instances as a function of N. Evidently α is almost constant across N, and the biggest factor in the difference in α is the type of camera pose distribution (the curves of Types B and D are similar since they are essentially randomly distributed camera poses). This supports the analysis in Sec. 3.4 that the total runtime of Algorithm 1, and hence the size of the output coreset, is mainly dependent on the desired approximation factor ϵ.

The next section will further investigate the size of the coreset output by Algorithm 1.

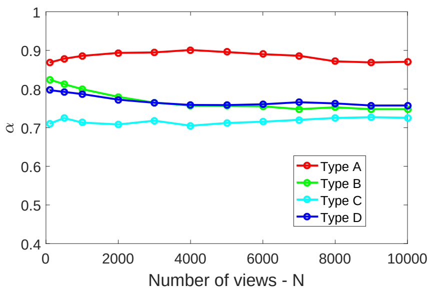


Fig. 7. Average α (probability of occurrence of Condition 2) as the problem size N increases, separated according to the type of camera pose distribution (see Fig. 4).

4.3 Size of Output Coreset

To investigate the size of the output coreset produced by Algorithm 1 as a function of the approximation error ϵ, we generated triangulation instances for the four types of camera distribution, with 200 3D points in each instance but with varying problem size (number of views) $N \in \{100, 500, 1000, 5000\}$. On each instance, the setting of ϵ for Algorithm 1 was varied decreasingly from 1 and the size of the output coreset was recorded.

Fig. 8 plots the maximum coreset size as a function of ϵ, averaged across all instances, but separated according to type of camera pose distributions and problem size N. The results show that the coreset size depends mainly on ϵ and is independent of the problem size N. Further, for all the data settings/parameters, the coreset size did not exceed 12. For some of the types of distribution, the coreset size also converged earlier due to earlier global convergence.

5 Experiments on Real Data

We conducted experiments on real data to validate Theorem 2 and investigate the performance of Algorithm 1 for ℓ_∞ triangulation. We used a standard machine with 3.2 GHz processor and 16 GB main memory.

5.1 Datasets and Initialisation

We tested on publicly available datasets for large scale 3D reconstruction, namely, Vercingetorix Statue, Stockholm City Hall, Arc of Triumph, Alcatraz, Örebro Castle [26], [27], and Notre Dame [2]. The *a priori* estimated camera poses and intrinsics supplied with these datasets were used to derive camera matrices. For triangulation, the size of an instance is the number of observations of the target 3D point. To avoid excessive runtimes, we randomly sampled 10% of the scene points in each dataset - this reduces the number of problem instances, but not the size of each of the selected instances. A histogram of the problem sizes for each of the above datasets are shown in the top left panel of Figs. 10 to 15.

Our coreset method was initialised as shown in the first few steps in Algorithm 1, which amounts to randomly choosing four data to instantiate $\mathbf{x}_1$ by solving (1). For any other algorithm that requires initialisation, the same $\mathbf{x}_1$ or its current maximum reprojection error were provided as the initial estimates.

5.2 Validation of Approximation Accuracy

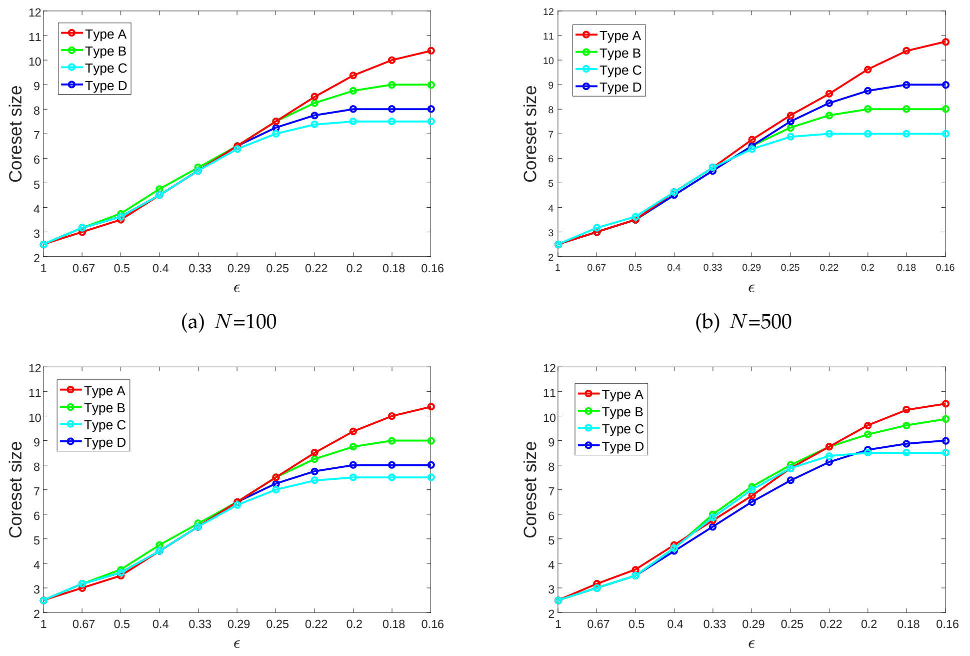
The top right panel in Figs. 10 to 15 show the actual ratio of errors versus the predicted ratio (using the backtracking formula in Sec. 3.5) across the iterations of Algorithm 1 for all problem instances in the datasets. Again, the results confirm the validity of Theorem 2.

5.3 Relative Speed-up of Coreset over Batch

Here we investigate the practicality of Algorithm 1 as a *global optimiser* for ℓ_∞ triangulation. As described in Section 3, Algorithm 1 is a meta-algorithm which requires a sub-routine to solve (1) on the subset indexed by $\mathcal{C}_t$. We thus compared running Algorithm 1 with a specific solver as a sub-routine, and the direct execution of the same solver in "batch mode" on the whole data. Since the runtime of Algorithm 1 depends on the efficiency of the embedded solver, the key performance indicator here is the *relative speed-up* achieved by coreset over batch.

Based on the investigations in [10], we have chosen to use bisection [6] and Dinkelbach's method [14] (equivalent to [9]) to embed into Algorithm 1. Although the best performing technique in [10] was Gugat's algorithm [15], our experiments suggested that it did not outperform Dinkelbach's method on the triangulation problem. SeDuMi [28] was used to solve the SOCP sub-problems in [6], [14].

The bottom diagrams of Figs. 10 to 15 show the average runtime of coreset and batch as a function of problem size (number of views), for each respective ℓ_∞ solver. Observe that the runtime of batch increased linearly and then exponentially, whilst coreset exhibited almost constant runtime—the latter observation is not surprising, since Algorithm 1 usually terminated at $\leq$ 10 iterations regardless of the problem size at shown in the top right panel.



(a) N=100 (b) N=500

(c) N=1000 (d) N=5000

Fig. 8. Average size of coreset plotted against decreasing ϵ, separated according to the four types of camera distribution as in Fig. 4. Panel (a),(b),(c), and (d) are the results respectively for N = 100, 500, 1000, and 5000.

Of course, on all of the datasets, most of the triangulation instances are small, as shown in the histogram at the upper left panel of Figs. 10 to 15. However, in these datasets, there are sufficient numbers of moderate to large problem instances, such that the total runtime of coreset is still much smaller than then total runtime of batch. Table 1 shows the total and average runtime of the variants considered. Evidently, coreset outperformed batch in all the datasets.

Although in principle any ℓ_∞ solver for ℓ_2 norm reprojection error [6], [7], [9], [10], [11] can be used in Algorithm 1, we emphasise again that the primary performance indicator here is the relative speed-up of coreset over batch. If a faster solver is used, it would likely improve both coreset and batch by the same factor.

5.4 Extensions to ℓ_1 and ℓ_∞ Reprojection Error

In the literature, apart from the more "traditional" ℓ_2 norm used in the reprojection error (2), different p-norms have been considered [10], i.e.,

$$r(\mathbf{x} \mid \mathbf{P}_i, \mathbf{u}_i) = \left\| \mathbf{u}_i - \frac{\mathbf{P}_i^{1:2}\tilde{\mathbf{x}}}{\mathbf{P}_i^{3}\tilde{\mathbf{x}}} \right\|_p, \quad p \in \{1, 2, \infty\}. \tag{41}$$

Our coreset theory was developed based on $p = 2$, thus the approximation bound (Theorem 2) will not hold for other p.

It is nonetheless feasible to apply Algorithm 1 as a meta-algorithm for solving ℓ_∞ triangulation (1) under different reprojection errors. Since problem (1) remains a GLP (Sec. 2) for $p = 1$ and $p = \infty$ in (41), global convergence is guaranteed. It is thus of interest to compare the relative speed-up given by the coreset method over a batch method in finding the globally optimal solution.

Table 2 shows the results of repeating the experiment in Sec. 5.3 with $p = 1$ and $p = \infty$. For $p = 1$, the Dinkelbach method was used as the embedded solver for Algorithm 1. For $p = \infty$, the state-of-the-art polyhedron collapsed method [13] was used as the embedded solver. Evidently the results show that the coreset method is able to significantly speed up global convergence.

6 Dealing with Outliers

While the existence of outliers in the measurements (e.g., from incorrect feature associations) is transparent to Algorithm 1 (i.e., the error bound in Theorem 2 will still hold), the result will of course be biased by the outliers—after all, the ℓ_∞ framework is not inherently robust [29].

Nonetheless, Sim and Hartley [20] showed how an effective outlier removal scheme can be constructed based on ℓ_∞ estimation. Basically, their scheme recursively conducts ℓ_∞ estimation and removes the support set (see Property 2) from the input data until the maximum residual is below a pre-determined inlier threshold. The remaining data then forms an inlier set.

Here, in the context of triangulation with outliers, we showed how the efficiency of Sim and Hartley's scheme can be improved by using Algorithm 1 (with a high ϵ) as a fast ℓ_∞ solver. Due to the approximation by Algorithm 1, instead of removing the support set, we remove the 4 measurements with the largest residuals.

Fig. 9 compares the runtime and number of remaining inliers produced by Sim and Hartley's original scheme (with Dinkelbach's method as the ℓ_∞ solver) and our coreset-enabled scheme (with $\epsilon = 0.4$). The results are based on 100 3D scene points projected onto N views ($10 \le N \le 500$), and where 90% of the 2D measurements in each problem instance were corrupted with Gaussian noise of $\sigma = 5$

			Total runtime (seconds)			
			Bisection solver		Dinkelbach solver	
Dataset	Scene points (number of triang. instances)	Number of views (max. triang. size)	Batch	Coreset	Batch	Coreset
Vercingetorix	594	68	57	**44** (23%)	23	**17** (26%)
Stockholm	2176	43	258	**177** (31%)	109	**74** (32%)
Arc of Triumph	2744	173	607	**243** (59%)	204	**89** (56%)
Alcatraz	4431	419	1231	**520** (57%)	452	**239** (47%)
Örebro Castle	5943	761	4052	**800** (80%)	1440	**351** (75%)
Notre Dame	7149	715	4148	**696** (83%)	2399	**582** (75%)

TABLE 1
Comparisons between coreset and batch in terms of total runtime in seconds. For coreset, the number in parentheses indicates the percentage of reduction in runtime by using the coreset method over the batch counterpart.

			Total runtime (seconds)			
			Dinkelbach ($p = 1$)		Polyhedron ($p = \infty$)	
Dataset	Scene points (number of triang. instances)	Number of views (max. triang. size)	Batch	Coreset	Batch	Coreset
Vercingetorix	594	68	1.61	**1.58** (13%)	1.82	**1.83** (-1%)
Stockholm	2176	43	10.4	**9.87** (19%)	12.2	**11.7** (4%)
Arc of Triumph	2744	173	20.5	**12** (49%)	23.5	**14.1** (40%)
Alcatraz	4431	419	166	**40** (73%)	146	**28.9** (80%)
Örebro Castle	5943	761	1011	**56** (94%)	887	**52.2** (94%)
Notre Dame	7149	715	1535	**116** (87%)	865	**31.7** (96%)

TABLE 2
Comparisons between coreset and batch in terms of total runtime in seconds, under the ℓ_1 and ℓ_∞ reprojection error (41). For ℓ_1 reprojection error, the Dinkelbach method was used as the embedded solver in Algorithm 1. For ℓ_∞ reprojection error, the state-of-art polyhedron collapse method [13] ("Polyhedron" above) was used as the embedded solver in Algorithm 1. For coreset, the number in parentheses indicates the percentage of reduction in runtime by using the coreset method over the batch counterpart.

pixels (the inliers), while the remaining 10% were corrupted with larger noise ($\sigma = 30$ pixels) to create outliers. The inlier threshold was set to 10 pixels. Evidently our coreset modification significantly improved the efficiency of the original scheme, without significantly affecting the quality of the result (number of remaining inliers).

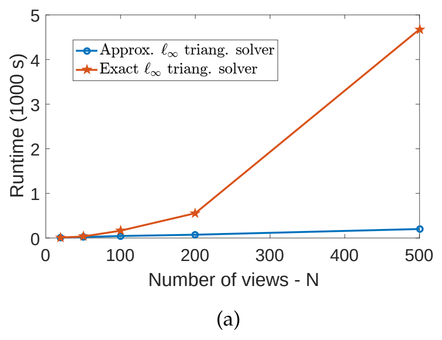
7 Conclusions and Open Questions

In this paper, we show that ℓ_∞ triangulation admits coreset approximation. We also provided comprehensive experimental results that establish the practical value of the coreset algorithm on large scale 3D reconstruction datasets.

There are several open questions:

- A deeper analysis of Condition 2 to hopefully remove it from consideration in the coreset bound, or at least to better characterise and predict its occurrence.
- The proof in Sec. 3.3 was inspired by the work of [17] on minimum enclosing ball (MEB) problems. There, the coreset size bound $\lceil 2/\epsilon \rceil$ was proven to be tight if the dimensionality d is comparable to $1/\epsilon$. For lower dimensional MEBs, tighter bounds have been proposed. It would be of interest to construct such tighter bounds for ℓ_∞ triangulation ($d = 3$).

Last but not least, we hope that our work encourages more effort to seek theoretically justifiable approximate algorithms for large scale geometric computer vision problems, especially the class of problems surveyed in [6], which can be seen as GLPs [19].



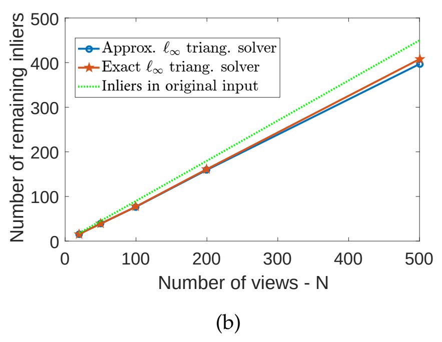


Fig. 9. Comparing Sim and Hartley's outlier removal scheme [20] with an exact solver (Dinkelbach's method) and with Algorithm 1. (a) Average runtime; (b) Number of inliers remaining in the final inlier set (the dashed green line to indicate the number of inliers in original input).

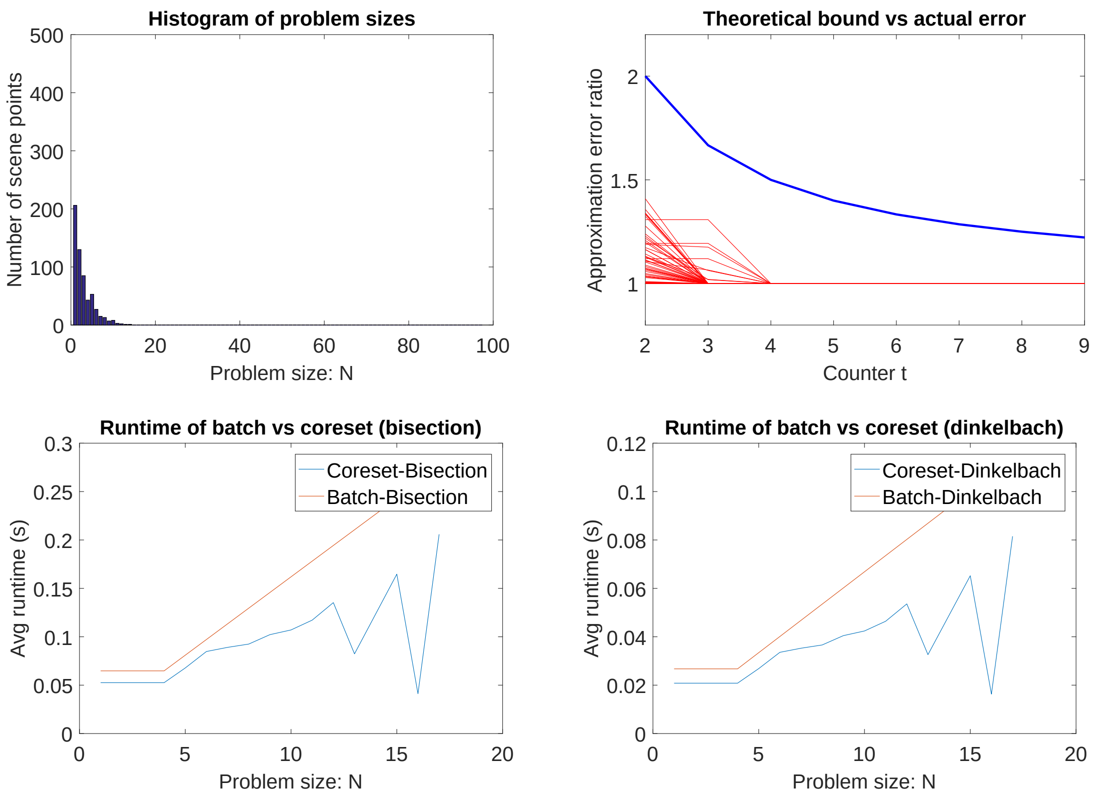


Fig. 10. Results for **Vercingetorix Statue**. (top left) Histogram of problem sizes. (top right) Approximation error ratio versus error ratio bound. (bottom left) Runtime of coreset vs batch, for bisection solver. (bottom right) Runtime of coreset vs batch, for Dinkelbach solver.

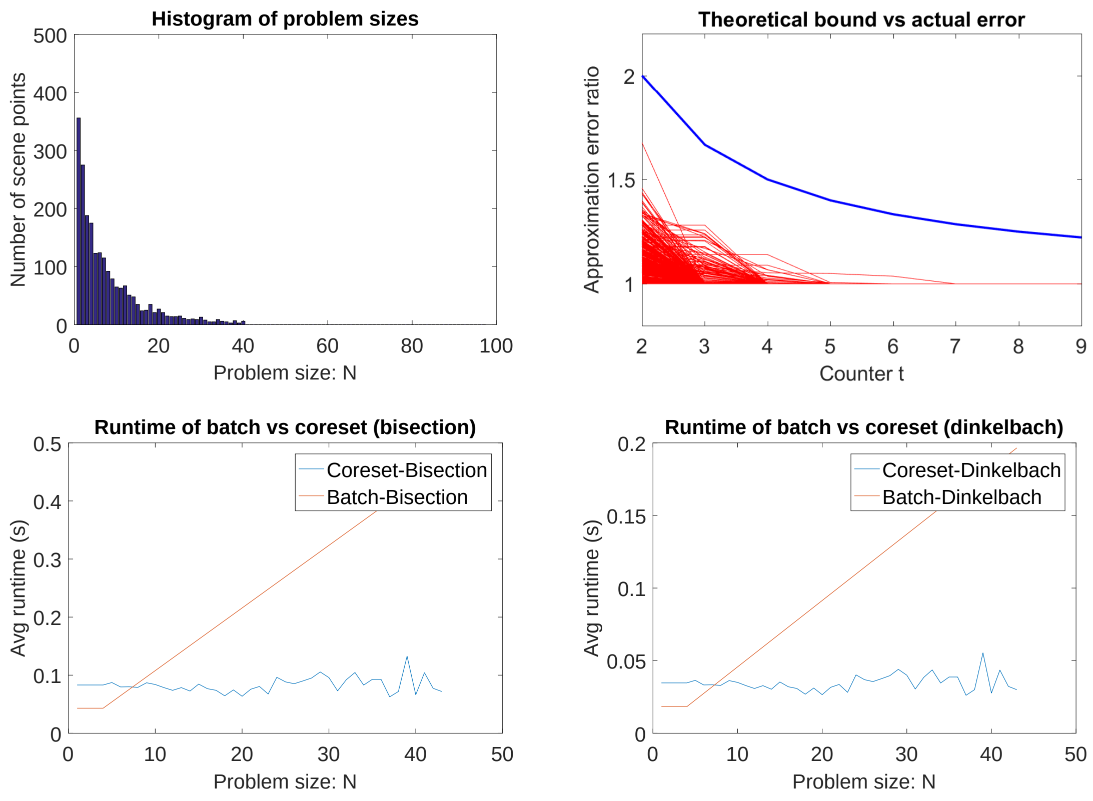


Fig. 11. Results for **Stockholm City Hall**. (top left) Histogram of problem sizes. (top right) Approximation error ratio versus error ratio bound. (bottom left) Runtime of coreset vs batch, for bisection solver. (bottom right) Runtime of coreset vs batch, for Dinkelbach solver.

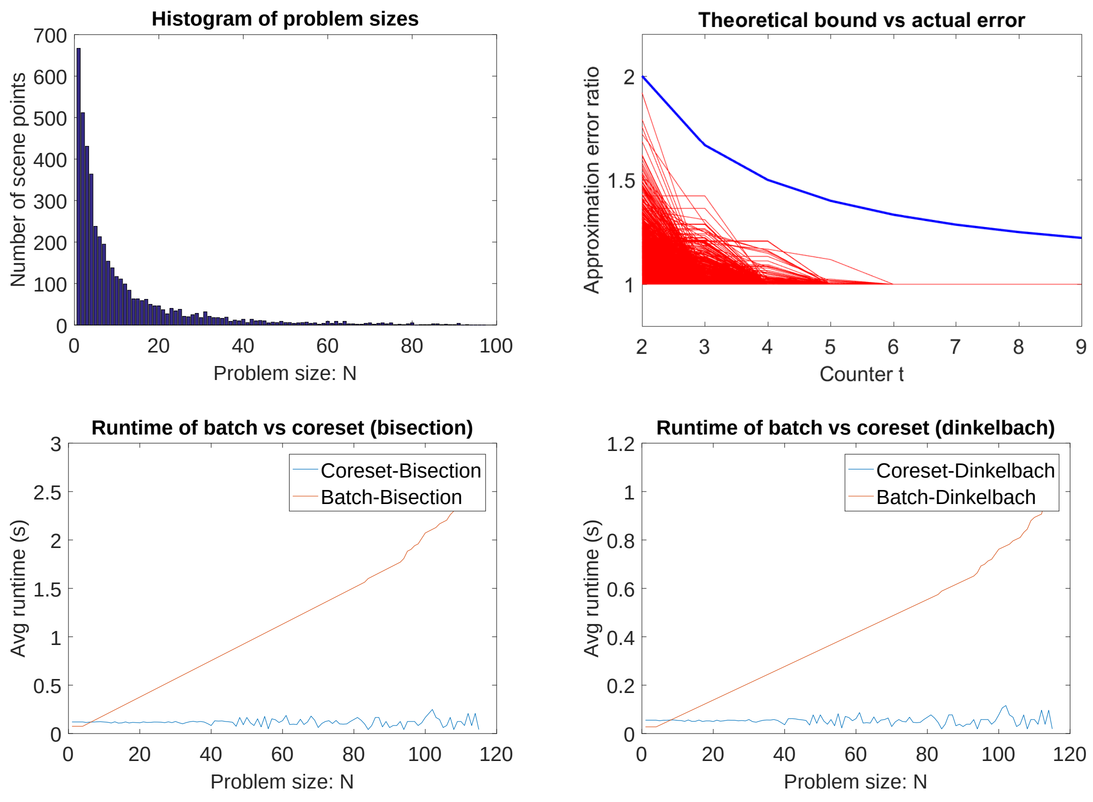


Fig. 12. Results for **Alcatraz**. (top left) Histogram of problem sizes. (top right) Approximation error ratio versus error ratio bound. (bottom left) Runtime of coreset vs batch, for bisection solver. (bottom right) Runtime of coreset vs batch, for Dinkelbach solver.

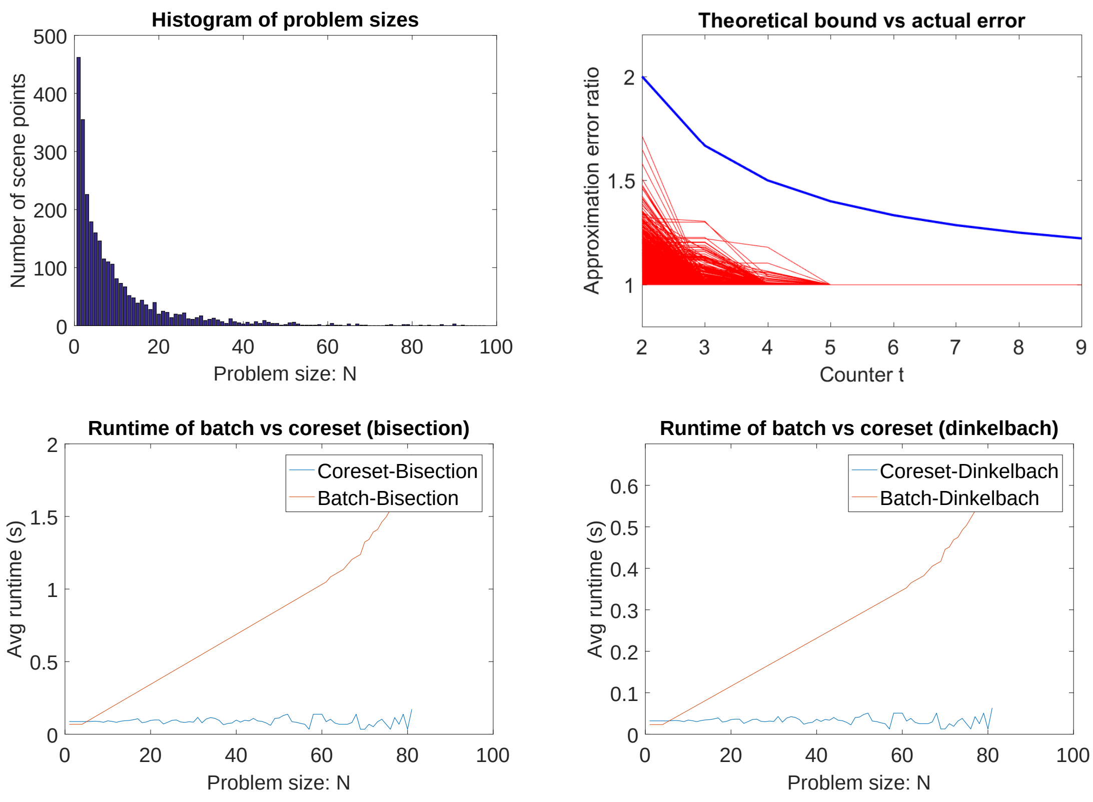


Fig. 13. Results for **Arc of Triumph**. (top left) Histogram of problem sizes. (top right) Approximation error ratio versus error ratio bound. (bottom left) Runtime of coreset vs batch, for bisection solver. (bottom right) Runtime of coreset vs batch, for Dinkelbach solver.

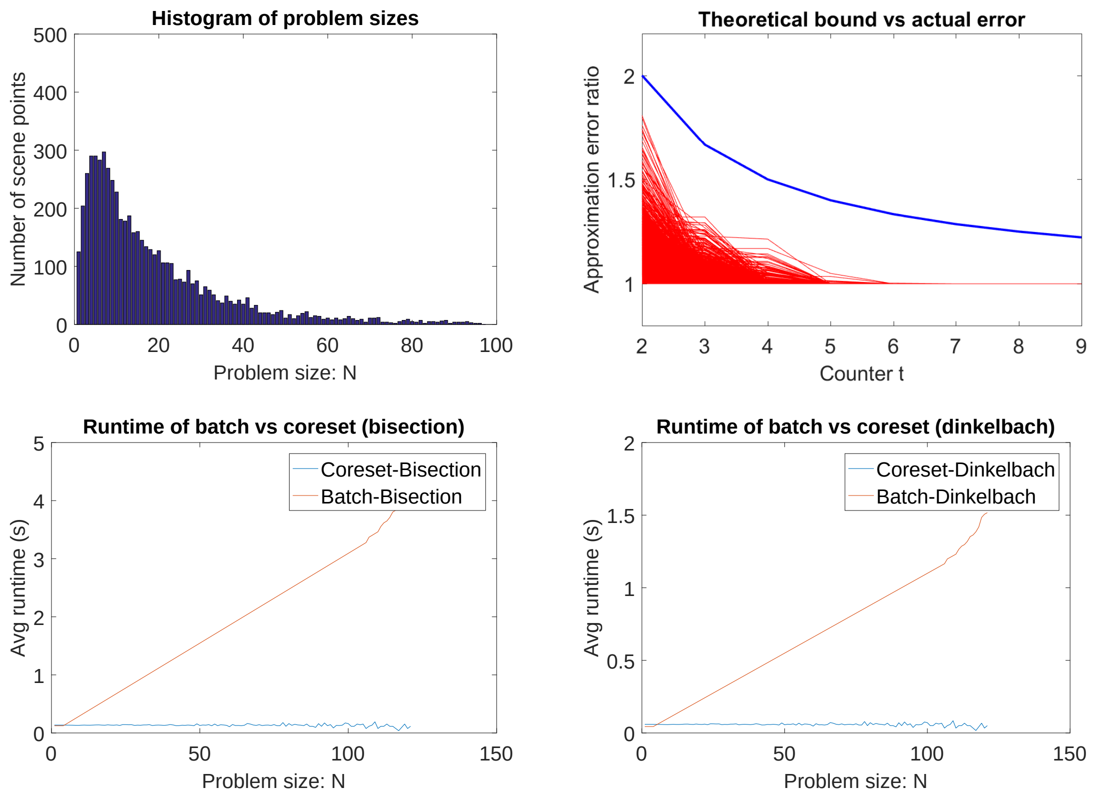


Fig. 14. Results for **Örebro Castle**. (top left) Histogram of problem sizes. (top right) Approximation error ratio versus error ratio bound. (bottom left) Runtime of coreset vs batch, for bisection solver. (bottom right) Runtime of coreset vs batch, for Dinkelbach solver.

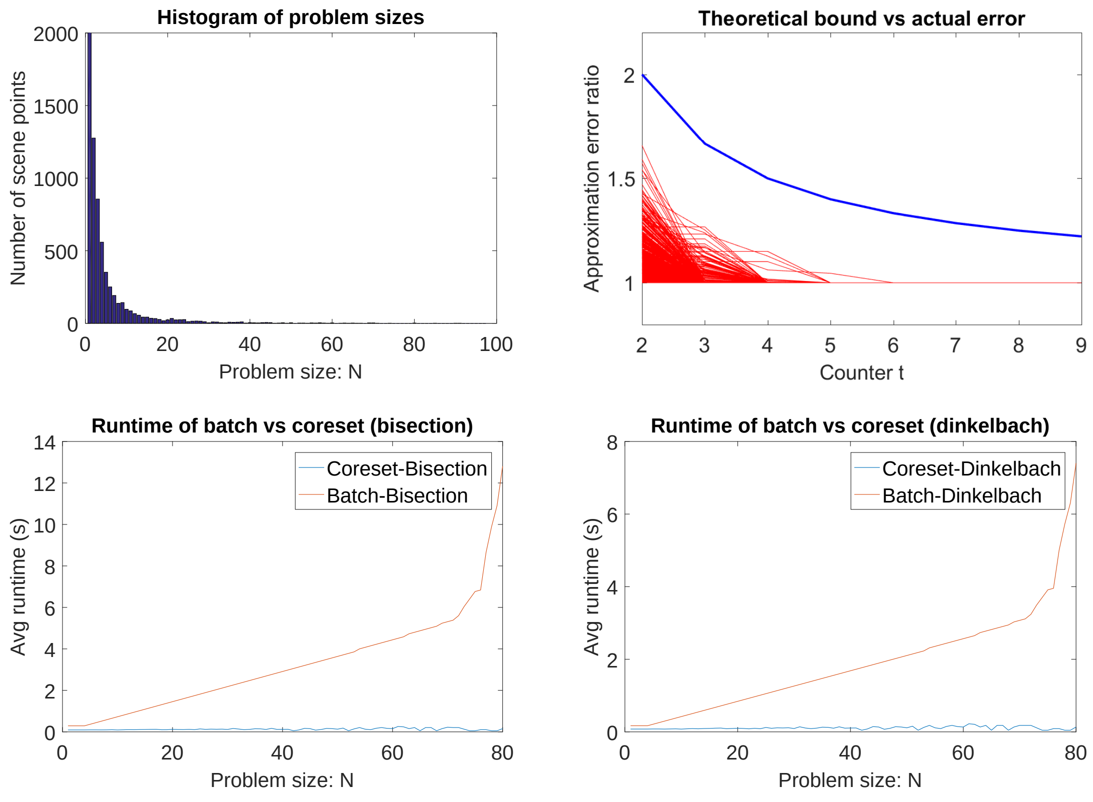


Fig. 15. Results for **Notre Dame**. (top left) Histogram of problem sizes. (top right) Approximation error ratio versus error ratio bound. (bottom left) Runtime of coreset vs batch, for bisection solver. (bottom right) Runtime of coreset vs batch, for Dinkelbach solver.

REFERENCES

[1] R. I. Hartley and P. Sturm, "Triangulation," *Computer vision and image understanding*, vol. 68, no. 2, pp. 146–157, 1997. 1

[2] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *International Journal of Computer Vision*, vol. 80, no. 2, pp. 189–210, 2008. 1, 6, 8

[3] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multi-view stereopsis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 8, pp. 1362–1376, 2010. 1

[4] R. Mur-Artal and J. D. Tardós, "Probabilistic semi-dense mapping from highly accurate feature-based monocular slam," *Proceedings of Robotics: Science and Systems, Rome, Italy*, vol. 1, 2015. 1, 6

[5] R. Hartley and F. Schaffalitzky, "L_∞ minimization in geometric reconstruction problems," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1. IEEE, 2004, pp. I–504. 1

[6] F. Kahl, "Multiple view geometry and the L_∞-norm," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2. IEEE, 2005, pp. 1002–1009. 1, 2, 3, 6, 8, 9, 10

[7] Q. Ke and T. Kanade, "Quasiconvex optimization for robust geometric reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 10, pp. 1834–1847, 2007. 1, 2, 3, 9

[8] Y. Seo and R. Hartley, "A fast method to minimize L_∞ error norm for geometric vision problems," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007, pp. 1–8. 1, 2, 3

[9] C. Olsson, A. P. Eriksson, and F. Kahl, "Efficient optimization for L_∞-problems using pseudoconvexity," in *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007, pp. 1–8. 1, 2, 3, 6, 8, 9

[10] S. Agarwal, N. Snavely, and S. M. Seitz, "Fast algorithms for L_∞ problems in multiview geometry," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8. 1, 2, 3, 6, 8, 9

[11] Z. Dai, Y. Wu, F. Zhang, and H. Wang, "A novel fast method for L_∞ problems in multiview geometry," in *European Conference on Computer Vision*. Springer, 2012, pp. 116–129. 1, 2, 3, 9

[12] 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*. IEEE, 2014, pp. 4066–4073. 1

[13] S. Donné, B. Goossens, and W. Philips, "Point triangulation through polyhedron collapse using the L_∞ norm," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 792–800. 1, 9, 10

[14] W. Dinkelbach, "On nonlinear fractional programming," *Management Science*, vol. 13, no. 7, pp. 492–498, 1967. 1, 8

[15] M. Gugat, "A fast algorithm for a class of generalized fractional programs," *Management Science*, vol. 42, no. 10, pp. 1493–1499, 1996. 1, 8

[16] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan, "Geometric approximation via coresets," *Combinatorial and computational geometry*, vol. 52, pp. 1–30, 2005. 1, 2

[17] M. Bădoiu and K. L. Clarkson, "Optimal core-sets for balls," *Computational Geometry*, vol. 40, no. 1, pp. 14–22, 2008. 1, 2, 4, 10

[18] H. Li, "Efficient reduction of l-infinity geometry problems," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 2695–2702. 2

[19] N. Amenta, "Helly-type theorems and generalized linear programming," *Discrete & Computational Geometry*, vol. 12, no. 3, pp. 241–261, 1994. 2, 10

[20] K. Sim and R. Hartley, "Removing outliers using the l\ infty norm," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 485–494. 2, 9, 10

[21] R. Seidel, "Small-dimensional linear programming and convex hulls made easy," *Discrete & Computational Geometry*, vol. 6, no. 3, pp. 423–434, 1991. 2

[22] K. L. Clarkson, "Las vegas algorithms for linear and integer programming when the dimension is small," *Journal of the ACM (JACM)*, vol. 42, no. 2, pp. 488–499, 1995. 2

[23] J. Matoušek, M. Sharir, and E. Welzl, "A subexponential bound for linear programming," *Algorithmica*, vol. 16, no. 4-5, pp. 498–516, 1996. 2

[24] D. P. Systems, "3d scanners," http://3dprintingsystems.com/products/3d-scanners/. 6

[25] P. F. Alcantarilla, C. Beall, and F. Dellaert, "Large-scale dense 3d reconstruction from stereo imagery." Georgia Institute of Technology, 2013. 6

[26] O. Enqvist, C. Olsson, and F. Kahl, "Stable structure from motion using rotational consistency," Citeseer, Tech. Rep., 2011. 8

[27] C. Olsson and O. Enqvist, "Stable structure from motion for unordered image collections," in *Scandinavian Conference on Image Analysis*. Springer, 2011, pp. 524–535. 8

[28] J. F. Sturm, "Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones," *Optimization methods and software*, vol. 11, no. 1-4, pp. 625–653, 1999. 8

[29] F. Kahl and R. Hartley, "Multiple-view geometry under the L_∞-norm," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 30, no. 9, pp. 1603–1617, 2008. 9

Qianggong Zhang received the BEng degree in computer science and techonology in 2004 and the MEng degree in computer science and techonology in 2007. Since 2015, he has been a PhD candidate at The University of Adelaide, South Australia. His primary research areas include approximation algorithms for geometric computer vision problems.

Tat-Jun Chin received the BEng degree in mechatronics engineering from Universiti Teknologi Malaysia (UTM) in 2003 and the PhD degree in computer systems engineering from Monash University, Victoria, Australia, in 2007. He was a research fellow at the Institute for Infocomm Research (I2R) in Singapore from 2007 to 2008. Since 2008, he has been at The University of Adelaide, South Australia, and is now an Associate Professor. He is an Associate Editor of IPSJ Transactions on Computer Vision and Applications (CVA). His research interests include robust estimation and geometric optimisation.

Chapter 6

A Fast Resection-Intersection Method for the Known Rotation Problem

The work contained in this chapter has been accepted for publication as the following paper

Qianggong Zhang, Tat-Jun Chin and Huu Minh Le: A Fast Resection-Intersection Method for the Known Rotation Problem. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018.

Statement of Authorship

Title of Paper	A Fast Resection-Intersection Method for the Known Rotation Problem		
Publication Status	☐ Published	[✓ Accepted for Publication]	
	☐ Submitted for Publication	☐ Unpublished and Unsubmitted work written in manuscript style	
Publication Details	Accepted to IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018		

Principal Author

Name of Principal Author (Candidate)	Qianggong Zhang		
Contribution to the Paper	Developed the theoretical proof of key theorems, performed experiments and wrote manuscript.		
Overall percentage (%)	70%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	22/11/2017

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);

ii. permission is granted for the candidate in include the publication in the thesis; and

iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Tat-Jun Chin		
Contribution to the Paper	Supervised development of work and revised manuscript.		
Signature		Date	22/11/2017

Name of Co-Author	Huu Minh Le		
Contribution to the Paper	Helped with experiments.		
Signature		Date	22/11/2017

Please cut and paste additional co-author panels here as required.

A Fast Resection-Intersection Method for the Known Rotation Problem

Qianggong Zhang Tat-Jun Chin Huu Minh Le
The University of Adelaide
Adelaide, South Australia, 5005, Australia
qianggong.zhang@adelaide.edu.au, tat-jun.chin@adelaide.edu.au, huu.le@adelaide.edu.au

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 (ℓ_∞) 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) [32] 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 [13, 28, 33, 16, 21] 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 two-fold: first, multiple-rotation averaging can often be solved much more easily (in certain cases, up to global optimality [24, 26, 12]); second, when formulated as a minimax (ℓ_∞) problem, KRot becomes an instance of pseudo-convex programming, which is also amenable to exact global solutions [18].

In this paper, we focus on KRot. Although the problem is tractable, as we will demonstrate later, most existing algorithms [17, 19, 27, 25, 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., [30, 20], 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 [22]—akin to the resection-intersection approach for BA [32].

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

[1] See the supplementary material for demo program.

show in Sec. 4, our algorithm can scale up to input sizes that are beyond the reach of existing methods.

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 [18], 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 [25], 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 [32], 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 ℓ_∞ triangulation [15], i.e., intersecting back-projected image points. Conversely, if $\{\mathbf{s}_k\}$ are fixed, the translations can also be optimised separately, viz.:

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

Problem (Res$_j$) is a special case of the ℓ_∞ camera resectioning problem [18]. The above properties motivate the resection-intersection method summarised in Algorithm 1.

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 $k = 1, \dots, M$, update $\mathbf{s}_k$ via (Int$_k$).
4:     For each $j = 1, \dots, L$, update $\mathbf{t}_j$ via (Res$_j$).
5: until convergence
6: return $\{\mathbf{t}_j\}_{j=1}^{L}$ and $\{\mathbf{s}_k\}_{k=1}^{M}$.
```

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 [32], it is effective for KRot since each sub-problem is pseudo-convex [25]. 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., [18, 25, 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 [22], allows to partition KRot into small sub-problems without affecting global optimality guarantees.

However, a more fundamental weakness of many previous methods [18, 19, 27, 25, 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 ℓ_∞ triangulation (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 (Int$_k$) and (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

$$\mathbf{A}_i = \begin{bmatrix}\mathbf{a}_{i,1}^T \\ \mathbf{a}_{i,2}^T\end{bmatrix} \in \mathbb{R}^{2\times3}, \quad \mathbf{b}_i = \begin{bmatrix} b_{i,1} \\ b_{i,2}\end{bmatrix} \in \mathbb{R}^2, \tag{3}$$

$\mathbf{c}_i \in \mathbb{R}^3$ and $d_i \in \mathbb{R}$ are constants derived from the data (see the supp. material for details of converting (Int$_k$) and (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 α 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 $\hat{\mathbf{x}}$.
2: while $\boldsymbol{\lambda}$ is not null do
3:     $\alpha \leftarrow$ Find step size using data, $\hat{\mathbf{x}}$ and $\boldsymbol{\lambda}$.
4:     $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}} + \alpha\boldsymbol{\lambda}$.
5:     $\boldsymbol{\lambda} \leftarrow$ 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 $\|\mathbf{m}^*\|_2 \le \epsilon_0$ then
4:     $\boldsymbol{\lambda} \leftarrow$ null.
5: else
6:     $\boldsymbol{\lambda} \leftarrow \mathbf{m}^* / \|\mathbf{m}^*\|_2$.
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.,

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

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 $p \ge 1$).

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

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

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 [23] 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 $\langle \boldsymbol{\lambda}, -\nabla f(\hat{\mathbf{x}})\rangle = -\nabla f(\mathbf{x})^T \boldsymbol{\lambda} > 0$.*

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 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. $\square$

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 $\overline{\mathbf{om}^*}$:

$$\overline{\mathbf{om}^*} \perp \mathcal{H}_0 \tag{15}$$

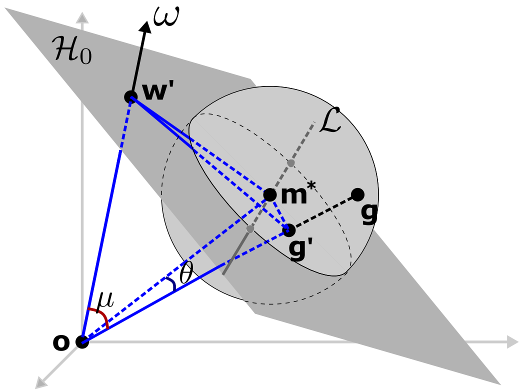


Figure 1. Diagram to support the proof of Theorem 2.

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

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

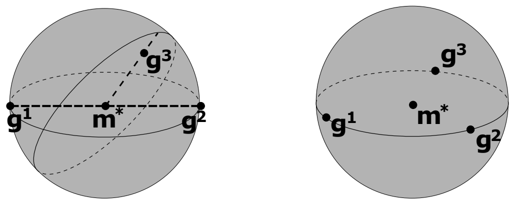
thus (14) is validated. □

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, 29]. 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 [34, 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:
  - **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 $f^* = 1$.



(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.

Degeneracies If the size of $\mathcal{G}$ is greater than four, we arbitrarily choose four items in $\mathcal{G}$ to calculate the MEB. If we obtain an $\mathbf{m}^*$ that is a valid descent direction, $\hat{\mathbf{x}}$ will be updated and the degeneracy will be resolved. If we obtain $\mathbf{m}^* = \mathbf{o}$, then, by the monotonicity of the MEB problem [5], the centre of the MEB on all of $\mathcal{G}$ will also coincide with $\mathbf{o}$, implying that there are no more descent directions.

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 α along $\boldsymbol{\lambda}$ to update $\hat{\mathbf{x}}$. The residual function parametrised by α 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 α 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., [25, 4, 18]) 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 α to solve (29).

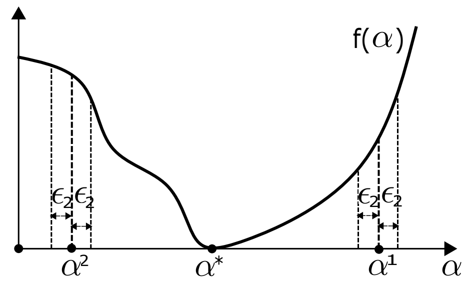


Figure 3. Illustration of Algorithm 4. If the current setting of $[lb, ub]$ yields $\alpha^1 = (lb + ub)/2$, then by $f(\alpha^1 - \epsilon_2) < f(\alpha^1) < f(\alpha^1 + \epsilon_2)$, α^1 is known to be on the monotonously increasing part of $f(\alpha)$, thus the optimal α^* is smaller than α^1 and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

Algorithm 4 describes our approach; see also Fig. 3 for an illustration. The lower bound for α is initialised to 0, while the initial upper bound is obtained as the largest α 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 α. Two convergence thresholds ϵ_1 and ϵ_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 $\epsilon_2$.
 1: $[lb, ub] \leftarrow$ lower and upper bound of $\alpha$ (see Sec. 3.2).
 2: Define $f(\alpha) = \max_i\ r_i(\alpha)$.
 3: while $ub - lb > \epsilon_1$ do
 4:     $\hat{\alpha} = (ub + lb)/2$.
 5:     $r_l \leftarrow f(\hat{\alpha} - \epsilon_2)$.
 6:     $r \leftarrow f(\hat{\alpha})$.
 7:     $r_r \leftarrow f(\hat{\alpha} + \epsilon_2)$.
 8:     if $r_l > r > r_r$ then
 9:         /* $\hat{\alpha}$ is on the decreasing part of $f(\alpha)$.*/
10:         $lb \leftarrow \alpha$.
11:     else if $r_l < r < r_r$ then
12:         /* $\hat{\alpha}$ is on the increasing part of $f(\alpha)$.*/
13:         $ub \leftarrow \alpha$.
14:     else
15:         /* $\hat{\alpha}$ is at a stationary point up to precision $\epsilon_2$.*/
16:         Break.
17:     end if
18: end while
19: return $\alpha = (ub + lb)/2$.
```

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

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* ϵ *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

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

Therefore, by Lemma 3, there exists a non-zero ϵ such that

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

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 ϵ_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 (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 Washington (Large). It is worth noting that for KRot, the

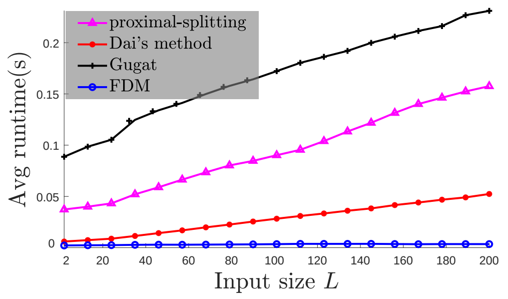


Figure 4. Runtime of competing methods on synthetic triangulation instances. For each input size L, the runtime was averaged over $M = 20$ random instances.

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.

4.1. Triangulation

We compared FDM with (1) Gugat's algorithm [14], which was shown in [4] to outperform the other methods that solve convex sub-problems (bisection [18], Dinkelbach's method [7, 25]), (2) Dai's method [6], and (3) proximal splitting [11]. All methods (including FDM) were executed in Matlab[3]. We used SeDuMi [31] 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) [15] 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 $\approx$1 ms) was observed in real data. Note that, although practical triangulation instances are small ($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 [20] 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 ration 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.

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**
UoW (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**
UoW (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.

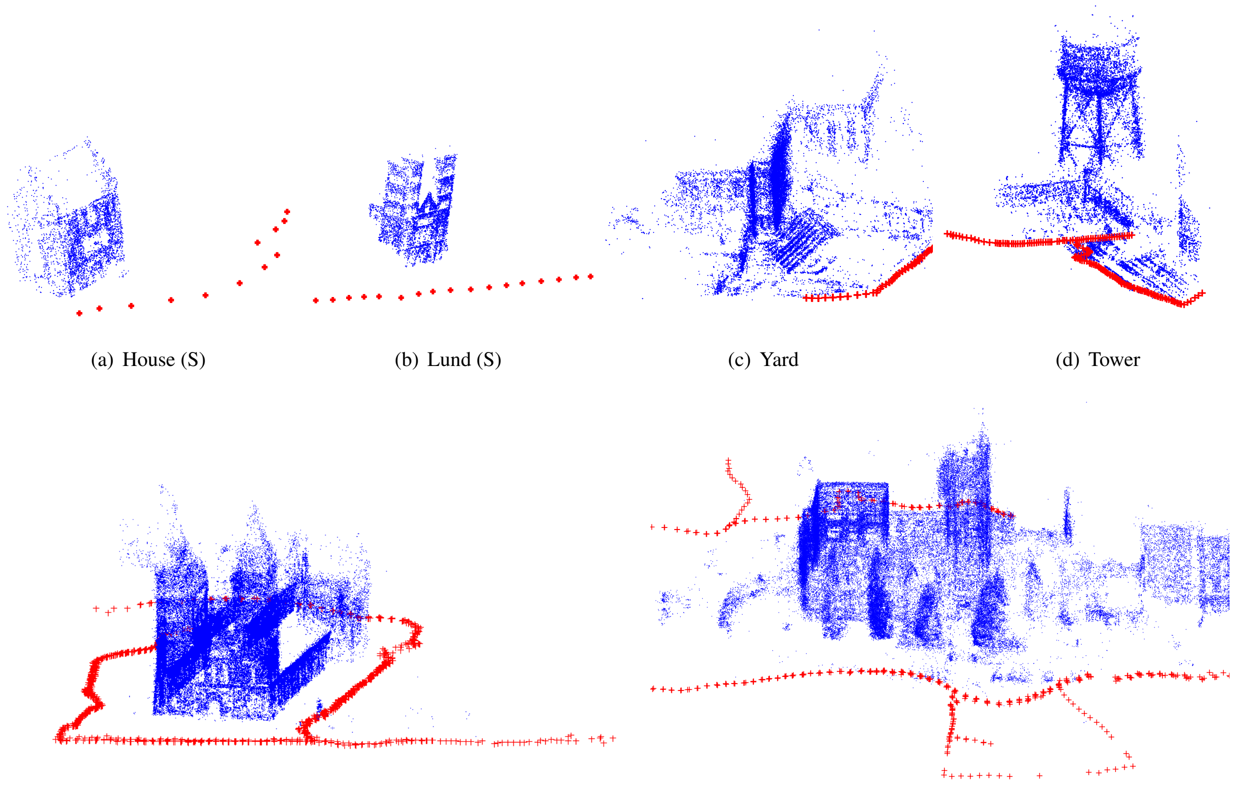

(a) House (S) (b) Lund (S) (c) Yard (d) Tower (e) Lund (L) (f) UoW (L)

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_∞ problems in multiview geometry. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008. 1, 2, 5, 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, 5

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

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

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

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

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

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

[19] 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

[20] 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

[21] 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

[22] 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

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

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

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

[26] 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

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

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

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

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

[31] 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

[32] 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

[33] 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

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

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis has led to significant progress in quasiconvex optimisation and SfM in general. On a practical level, state-of-the-art algorithms have been proposed to solve some of fundamental SfM problems, for example, the meta-algorithm and the coreset algorithm for large-scale triangulation, the Res-Int algorithm for KRot, and the Q-sweep algorithm for LMS triangulation. On a theoretical level, this thesis has successfully adopted some interdisciplinary achievements (e.g., concepts and algorithms from computational geometry) to SfM and advanced the scientific frontier of SfM.

7.2 Future work

7.2.1 ϵ_2 in coresets for triangulation

In Chapter 5, in addition to the dependence on the specified approximation accuracy ϵ in the coreset paradigm, the size of coresets for ℓ_∞ triangulation also depends on another factor ϵ_2 — the probability of a certain geometric condition (Condition. 2 in the published paper) being satisfied. Even though the empirical results suggest that the value of ϵ_2 is determined by the distribution of input data, no analytical results have been concluded yet. A deeper analysis of ϵ_2 (ideally removing it from consideration in the coreset bound, or at least better characterising and predicting its occurrence) is desirable.

7.2.2 The fast descent method

Although this thesis is based in the context of two particular SfM problems — triangulation and KRot, the conceptual and practical outcomes therein transfer readily to general quasiconvex problems. For example, the fast descent method (FDM) proposed in Chapter 6 can be generalised to broader optimisation problems. The bottleneck of FDM is the underlying MEB solver, which becomes less efficient as the problem size (e.g., the number of parameters to be estimated, and the size of input) increases. Nevertheless, FDM could still be a viable solver for applications with moderate number of to-be-estimated parameters, e.g., camera resectioning (11 parameters) and two-dimensional homographies (8 parameters) [29].

7.2.3 The Q-sweep method

The Q-sweep method introduced in Chapter 4 enforces robustness to the ℓ_∞ aggregation of multiple quasiconvex functions; potentially, it can be generalised to other multiple-view applications with similar formulation as surveyed in [32].

7.2.4 The Res-Int method

The BA procedure (see Section 1.1.1) can either be directly solved as a non-linear least squares problem, or be solved by the resection-intersection method [54], which alternates between *resection* (i.e., camera resectioning, which amounts to estimating camera poses [29]) and *intersection* (i.e., triangulation). With respect to the resection-intersection BA method, a possibly more efficient strategy would be to alternate between estimating the absolute rotations of cameras (via the relatively easy rotation averaging procedure) and estimating the camera translations and the structure (via the proposed Res-Int KRot method).

Bibliography

[1] http://www.maths.lth.se/matematiklth/personal/calle/dataset/dataset.html.

[2] https://en.wikipedia.org/wiki/Norm_(mathematics).

[3] https://en.wikipedia.org/wiki/Quasiconvex_function.

[4] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. “Geometric approximation via coresets”. In: *Combinatorial and computational geometry* 52 (2005), pp. 1–30.

[5] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. “Building Rome in A Day”. In: *Communications of the ACM* 54.10 (2011), pp. 105–112.

[6] S. Agarwal, N. Snavely, and S. M. Seitz. “Fast algorithms for L_∞ problems in multiview geometry”. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on.* IEEE. 2008, pp. 1–8.

[7] J. Agulló. “Exact algorithms for computing the least median of squares estimate in multiple linear regression”. In: *Lecture Notes-Monograph Series* (1997), pp. 133–146.

[8] N. Amenta. “Helly-type theorems and generalized linear programming”. In: *Discrete & Computational Geometry* 12.3 (1994), pp. 241–261.

[9] M. Bădoiu and K. L. Clarkson. “Smaller core-sets for balls”. In: *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms.* Society for Industrial and Applied Mathematics. 2003, pp. 801–802.

[10] C. Beder and R. Steffen. “Determining an initial image pair for fixing the scale of a 3d reconstruction from an image sequence”. In: *Joint Pattern Recognition Symposium.* Springer. 2006, pp. 657–666.

[11] J. L. Bentley and T. A. Ottmann. “Algorithms for reporting and counting geometric intersections”. In: *IEEE Transactions on computers* 9 (1979), pp. 643–647.

[12] D. P. Bertsekas and A. Scientific. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

[13] A. Biniaz and D. Gh. "A comparison of plane sweep Delaunay triangulation algorithms". In: *Proceedings of CSICC* (2007).

[14] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[15] K. Cornelis, F. Verbiest, and L. Van Gool. "Drift detection and removal for sequential structure from motion algorithms". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.10 (2004), pp. 1249–1259.

[16] Z. Cui and P. Tan. "Global structure-from-motion by similarity averaging". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 864–872.

[17] Z. Dai, Y. Wu, F. Zhang, and H. Wang. "A novel fast method for L_∞ problems in multiview geometry". In: *Computer Vision–ECCV 2012* (2012), pp. 116–129.

[18] A. Delaunoy and M. Pollefeys. "Photometric bundle adjustment for dense multiview 3d modeling". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1486–1493.

[19] W. Dinkelbach. "On nonlinear fractional programming". In: *Management science* 13.7 (1967), pp. 492–498.

[20] 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*. 2015, pp. 792–800.

[21] 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*. IEEE. 2014, pp. 4066–4073.

[22] A. Eriksson, C. Olsson, F. Kahl, O. Enqvist, and T.-J. Chin. "Why Rotation Averaging is Easy". In: *arXiv preprint arXiv:1705.01362* (2017).

[23] K. Fischer, B. Gärtner, and M. Kutz. "Fast smallest-enclosing-ball computation in high dimensions". In: *ESA*. Vol. 2832. Springer. 2003, pp. 630–641.

[24] M. A. Fischler and R. C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Readings in computer vision*. Elsevier, 1987, pp. 726–740.

[25] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. "Towards internet-scale multi-view stereo". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE. 2010, pp. 1434–1441.

[26] M. Gugat. “A fast algorithm for a class of generalized fractional programs”. In: *Management Science* 42.10 (1996), pp. 1493–1499.

[27] R. Hartley and F. Schaffalitzky. “L_∞ minimization in geometric reconstruction problems”. In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on.* Vol. 1. IEEE. 2004, pp. I–I.

[28] R. Hartley, J. Trumpf, Y. Dai, and H. Li. “Rotation averaging”. In: *International journal of computer vision* 103.3 (2013), pp. 267–305.

[29] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision.* Cambridge university press, 2003.

[30] P. J. Huber et al. “Robust estimation of a location parameter”. In: *The Annals of Mathematical Statistics* 35.1 (1964), pp. 73–101.

[31] F. Kahl, S. Agarwal, M. K. Chandraker, D. Kriegman, and S. Belongie. “Practical global optimization for multiview geometry”. In: *International Journal of Computer Vision* 79.3 (2008), pp. 271–284.

[32] F. Kahl and R. Hartley. “Multiple-View Geometry Under the L_∞-Norm”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.9 (2008), pp. 1603–1617.

[33] Q. Ke and T. Kanade. “Quasiconvex optimization for robust geometric reconstruction”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.10 (2007), pp. 1834–1847.

[34] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun. “Tanks and temples: Benchmarking large-scale scene reconstruction”. In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), p. 78.

[35] H. Li. “A practical algorithm for L_∞ triangulation with outliers”. In: *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on.* IEEE. 2007, pp. 1–8.

[36] H. Li. “Efficient reduction of L-infinity geometry problems”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE. 2009, pp. 2695–2702.

[37] J. Matoušek, M. Sharir, and E. Welzl. “A subexponential bound for linear programming”. In: *Algorithmica* 16 (1996), pp. 498–516.

[38] N. Megiddo. “Linear-time algorithms for linear programming in R^3 and related problems”. In: *SIAM journal on computing* 12.4 (1983), pp. 759–776.

[39] P. Moulon, P. Monasse, and R. Marlet. “Global fusion of relative motions for robust, accurate and scalable structure from motion”. In: *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE. 2013, pp. 3248–3255.

[40] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. “DTAM: Dense tracking and mapping in real-time”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2320–2327.

[41] J. Nocedal and S. J. Wright. *Numerical optimization, second edition*. Springer, 2006.

[42] C. Olsson and O. Enqvist. “Stable structure from motion for unordered image collections”. In: *Image Analysis* (2011), pp. 524–535.

[43] C. Olsson, A. P. Eriksson, and F. Kahl. “Efficient optimization for L_∞-problems using pseudoconvexity”. In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8.

[44] S. Recker, M. Hess-Flores, M. A. Duchaineau, and K. I. Joy. “Visualization of Scene Structure Uncertainty in a Multi-View Reconstruction Pipeline.” In: *VMV*. 2012, pp. 183–190.

[45] S. Recker, M. Hess-Flores, and K. I. Joy. “Statistical angular error-based triangulation for efficient and accurate multi-view scene reconstruction”. In: *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*. IEEE. 2013, pp. 68–75.

[46] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*. Vol. 589. John wiley & sons, 2005.

[47] J. L. Schonberger and J.-M. Frahm. “Structure-from-motion revisited”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4104–4113.

[48] Y. Seo and R. Hartley. “A fast method to minimize L_∞ error norm for geometric vision problems”. In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE. 2007, pp. 1–8.

[49] M. Sharir and E. Welzl. “A combinatorial bound for linear programming and related problems”. In: *Annual Symposium on Theoretical Aspects of Computer Science*. Springer. 1992, pp. 567–579.

[50] K. Sim and R. Hartley. “Removing Outliers Using The L_∞ Norm”. In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2006, pp. 485–494.

[51] N. Snavely, S. M. Seitz, and R. Szeliski. “Photo tourism: exploring photo collections in 3D”. In: *ACM transactions on graphics (TOG)*. Vol. 25. 3. ACM. 2006, pp. 835–846.

[52] D. L. Souvaine and J. M. Steele. “Time-and space-efficient algorithms for least median of squares regression”. In: *Journal of the American Statistical Association* 82.399 (1987), pp. 794–801.

[53] A. J. Stromberg. “Computing the exact least median of squares estimate and stability diagnostics in multiple linear regression”. In: *SIAM Journal on Scientific Computing* 14.6 (1993), pp. 1289–1299.

[54] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. “Bundle adjustment—a modern synthesis”. In: *International workshop on vision algorithms.* Springer. 1999, pp. 298–372.

[55] R. J. Vanderbei and D. F. Shanno. “An interior-point algorithm for nonconvex nonlinear programming”. In: *Computational Optimization and Applications* 13.1-3 (1999), pp. 231–252.

[56] E. Welzl. “Smallest enclosing disks (balls and ellipsoids)”. In: *New results and new trends in computer science* (1991), pp. 359–370.