THE UNIVERSITY of ADELAIDE

Doctoral Thesis

Robust Rotation Search in Computer Vision

Author:
Álvaro Joaquín Parra Bustos

Supervisors:
Dr. Tat-Jun CHIN
Prof. David SUTER

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

in the

Faculty of Engineering, Computer and Mathematical Sciences
School of Computer Science

August 2016

Declaration

Signed: ______________________________

Date: ______________________________

THE UNIVERSITY OF ADELAIDE

Abstract

School of Computer Science

Doctor of Philosophy

Robust Rotation Search in Computer Vision

by Álvaro Joaquín PARRA BUSTOS

Rotation search is a fundamental problem which has significant importance in geometric computer vision. In many practical settings, data or measurements for rotation search are usually contaminated with large errors, leading to the existence of *outliers* in the data. As a consequence, traditional least-squares rotation estimation methods are not suitable in many practical applications. A more appropriate approach is to search for the rotation based on a robust criterion. However, optimisation problems involving robust criteria are hard to solve since the objective functions are usually non-differentiable and non-convex.

This thesis makes several fundamental contributions in robust rotation search. In contrast to approximations or local methods that are typically used by current practitioners, the presented methods in this thesis guarantee global optimality. The main challenge for robust rotation search algorithms is to find an optimal result in reasonable time (to be practical in out-of-lab applications). The work in this thesis is a contribution in this direction.

To efficiently solve robust rotation search, several strategies are presented based on new insights into the geometry of rotations, from the perspective of global optimisation. Firstly, for point set registration on horizontally levelled data, the presented algorithms make it possible to globally find the best rotation in *real-time*. Secondly, for the fully unconstrained 3D rotation search problem, the presented algorithms outperform previous methods by an order of magnitude. The final contribution of this thesis is an algorithm to safely remove true outliers when rotation is computed on outlier contaminated point correspondences. Substantial speed-up can be obtained when the proposed outlier removal is used as a preprocessor to globally optimal algorithms. Since no inliers are discarded, global optimality is guaranteed.

The contributions in this thesis can impact on computer vision problems where rotation search is invoked as a subroutine. This thesis presents examples from 3D point cloud registration and image stitching.

Acknowledgements

I am pleased to thank to my supervisor, Dr. Tat-Jun Chin, for all his advice, comments, and critical revisions throughout this thesis and the articles we published during my PhD. I really appreciate his interest in this research. Working with him has certainly been an enriching life experience. I would also like to thank to my co-supervisor Prof. David Suter for his advice, revisions and comments during research meetings. I extend this thanks to Anders Eriksson for his valuable comments on ideas and works that are part of this thesis.

I would also like to thank my Master's supervisor Dr. Julián Ortiz for his constant advice.

I would like to express my sincere appreciation to my parents for all they support and encouragement. I extend this thanks to my tata and tía Maruja for all the unconditional support. I want to thanks to Carola Cardoso for her help, company and encouragement.

I am also grateful to family and friends that visited me and shared amazing moments. A big thank to Pamela Cordero and Nicolás Seitz.

During these years I have shared with incredible people. I am very grateful to Exequiel Sepúlveda and his family for their constant support. A big thank also to Andrés Figueroa and Paula Núñez for the shared moments, trips and help.

I also would like to extend thanks to Quoc-Huy Tran, Trung T. Pham, and Roberto Shinmoto for the general advice and encouragement, and to Russell Disher for his hospitality. Many thanks to Sergio Palacio, Carl Vail, Greg Rowlands and Kingsley Denton for their help.

Last I would like to thank CONICYT Becas-Chile for founding my PhD.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

BnB	**B**ranch a**n**d **B**ound
CSM	**C**onsensus **S**et **M**aximisation
DoF	**D**egrees **o**f **F**reedom
GM	**G**eometric **M**atching
LS	**L**east **S**quares
SLAM	**S**imultaneous **L**ocalisation **A**nd **M**apping

Publications

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

- Álvaro Parra Bustos, Tat-Jun Chin, Anders Eriksson, Hongdong Li and David Suter: Fast rotation search with stereographic projections for 3D registration. IEEE Transactions on Pattern Analysis and Machine Intelligence.
  Accepted on 23 Dec 2015.
  (DOI: 10.1109/TPAMI.2016.2517636)
- Álvaro Parra Bustos and Tat-Jun Chin: Guaranteed Outlier Removal for Rotation Search. In International Conference on Computer Vision (ICCV) 2015: 2165-2173
- Álvaro Parra Bustos, Tat-Jun Chin and David Suter: Fast rotation search with stereographic projections for 3D registration. In Computer Vision and Pattern Recognition (CVPR) 2014: 3930-3937
  (DOI: 10.1109/CVPR.2014.502)
- Tat-Jun Chin, Álvaro Parra Bustos, Michael Brown and David Suter: Fast rotation search for real-time interactive point cloud registration. In ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D) 2014: 55-62
  (DOI: 10.1145/2556700.2556712)

For my parents.

Chapter 1

Introduction

The overall objective of computer vision is to construct computer algorithms that can understand visual information, similar to the ability of the human visual system. For example, recognising objects such as cars or birds in a picture. Such tasks can be performed effortlessly by the human vision system. However, it is very challenging to mimic such abilities on an artificial system. On the other hand, although the human visual system can achieve very sophisticated tasks, it is affected by limitations that do not strongly affect computer vision systems. For instance, our visual perception suffers from limited visible spectrum, fallible memory and optical illusions. Moreover, since our brain perceives the 3D world from 2D visual information acquired using the retina, the resulting interpretations can be ambiguous and inaccurate. On the other hand, computer vision systems are not limited to using the visible spectrum, they can "remember" a huge amount of visual data, and they can directly perceive in 3D.

At present, there are different mature technologies that enable computers to sense in 3D. The primary example is light detection and ranging (LIDAR) devices, which make point-wise depth measurements based on the time of flight of light pulses. For example, the 3D models of an underground mine in Figure 1.1a were captured with a LIDAR scanner. Another notable example is depth cameras such as *Microsoft Kinect*, which captures depth measurements based on the principle of active stereo with structured infrared light. There are also technologies that recover the 3D structure of an environment from 2D image sequences directly based on the principle of multiple view geometry and simultaneous localisation and mapping (SLAM). An example of such a technology is Google's *Project Tango* that operates in real-time on a smartphone device.

Many useful applications can be implemented on 3D computer vision systems. Examples are as follows:

- Recognising a known 3D object within a scene captured in 3D.
- Given a 3D shape, search for a similar shape amongst a database of 3D shapes.
- Stitching several sets of 3D scans together to obtain a more complete 3D model.
- Track changes of a scene over time.

In all of the applications above, a recurrent and basic task is the *registration* of different sets of 3D measurements. Registration is defined as the process of transforming or aligning several sets of measurements into a common coordinate system. An example is presented in Figure 1.1. The alignment in Figure 1.1c was obtained after finding the optimal rigid transform to align the point clouds in Figure 1.1a. This thesis focuses on the optimisation of the rotational parameters in the registration problem, i.e., the *rotation search* problem. Although one must estimate all 6 parameters of a 3D rigid transform to accomplish registration, usually registration problems are decoupled into rotation and translation search, with 3 parameters in each subproblem. For example, in [50], least squares estimation of 3D rigid transform, rotation and translation are treated independently. A more recent approach consists of a nested design. For example, in [22], an inner rotation kernel is embedded in an outer translation search loop. Rotation search algorithms can be embedded as a kernel in such a kind of registration methods. On the other hand, there exist cases where only rotation is required, for instance if an object is scanned on a turntable [17].

Rotation search is useful not only to computer vision problems that operate on 3D measurements. Many problems in geometric vision, which take as input 2D images, also require the estimation of rotational parameters. For example, stitching images when building a 360° panorama or stitching photos of a far away scene. In these cases, the problem of estimating the homography that relates two images can be accomplished as a pure rotation search problem. The input of the rotation search problem is obtained by applying the inverse camera calibration matrix on 2D image keypoints. Another problem taking 2D images as input, is localising a photo on a given 3D model reconstructed from images. The rotational parameters can be estimated by registering backprojected rays from 2D image keypoints with the 3D model. More applications with 2D images as input are presented Section 1.1.

Despite the prevalence of applications that require rotation search, the problem has not been satisfactorily solved. For example, the least squares algorithms are not robust towards outliers. Hence, the rotation obtained can be heavily biased away from the desired result. More recent robust rotation search algorithms are designed to handle outliers. However, the existing algorithms are rather inefficient. This thesis makes several fundamental contributions toward efficient and robust rotation search.

The rest of this chapter further illustrates the usefulness and difficulty of rotation search. A summary of recent progress in this area of computer vision is also given.

1.1 Applications involving rotation search

As alluded above, a vital component of 3D point cloud registration [6, 16, 60, 59] is rotation search. Further, if translation is known, rotation search and rigid registration are the same problem. It is emphasised that this thesis focusses on *rigid* point cloud registration. For comparisons between different registration problems and methods, the reader can refer to Tam et al. [70] and Salvi et al. [67]. Applications for point cloud registration are extensive. Some examples are heritage recording [63], registration of underground mine sites [16], and shape acquisition [28, 1]. In general, any application that requires integrating data captured from different views has a registration component.

Apart from point cloud registration, the necessity of estimating rotations is present on many other geometric computer vision problems.

Arguably, camera pose estimation is the more well-known fundamental problem that has benefited from recent advances in globally optimal rotation search [51, 22, 34, 6]. This problem consists of estimating the position and orientation of a camera from point matches between a 2D image and a 3D model. Camera pose estimation is an important problem in geometric computer vision since it is a core problem in many applications such as motion segmentation and object recognition.

Previously in this thesis, image stitching was presented as an example application for rotation search that takes 2D images as input. Again, the homography relating two images can be obtained as a rotational search problem if the input images are of a far away scene. There exist many panoramic image stitching works [6, 19, 20, 59] that have benefited from advances in rotation search. An interesting related application is the creation of a 360° panorama by using a smartphone. An application guides the user holding the smartphone to move the smartphone's camera to obtain the needed images to be stitched. The Google Photo Sphere project [30] is an example for such an application.

Although the focus of this thesis is on computer vision application, rotation search algorithms are important in other fields, such as robotics.

Hand-eye calibration [36, 68, 64, 35] is an important problem in robotics. Briefly, hand-eye calibration is defined as the problem of estimating the transformation between a camera and the robot hand where the camera is mounted. Typically, rotational parameters need to be estimated as part of an affine transformation. However, for the problem of a rotational sensor with a camera attached on it, calibration is solved by rotation search only [68].

Location recognition [69] is another interesting problem involving rotation search. Location recognition is about estimating the position in a 3D model from where an input 2D image was captured. This problem can be formulated as a registration problem concerning rotation and translation. For a 2D image point, the points in the 3D model with angular error lower than a threshold lie in a cone. Then, the registration problem consists of aligning the 3D points in the model to the cones relating points in the 2D image.

Although rotation search has many applications, in the rest of this thesis, aspects of the problem and of the algorithms will be illustrated by focussing on the applications of point cloud registration (Chapters 3 and 4) and image stitching (Chapter 5). The reader should be reminded, however, that the novel algorithms presented in this thesis can be easily transported to other applications such as the examples presented above.

1.2 Why is rotation search difficult?

In the ideal case of data without outliers, rotation search can be solved very efficiently in closed form [37, 2] in the least squares sense. However, least squares algorithms fail in outlier contaminated data, which is a common scenario in out-of-lab applications. The methods presented in this thesis are directed at solving rotation search on such "difficult" input data. These are the cases when only a small portion of the input point clouds can be rotationally aligned, i.e., the *inlier ratio* or the percentage of overlap are low. Consider the example presented in Figure 1.1 in which a small portion of points genuinely overlap (Figure 1.1c). There is also the case where rotation is estimated from pre-determined point correspondences, and outliers occur when point correspondences are incorrect (e.g., red lines in Figure 1.3b). A discussion on robust estimation is presented in Section 1.4.

Run-time is another important aspect in practical applications. Typically, real-life problems require a solution in a "short" period of time, and, in some cases, in real-time. An example of an application that needs to solve rotation search in real-time is the user-assisted registration system presented in Chapter 3.

At a fundamental level, what makes rotation search difficult is that most of the useful objective functions are nonconvex. For example, observe the maximisation problem plotted in Figure 1.2a which is used to robustly register the point clouds in Figure 1.1a.

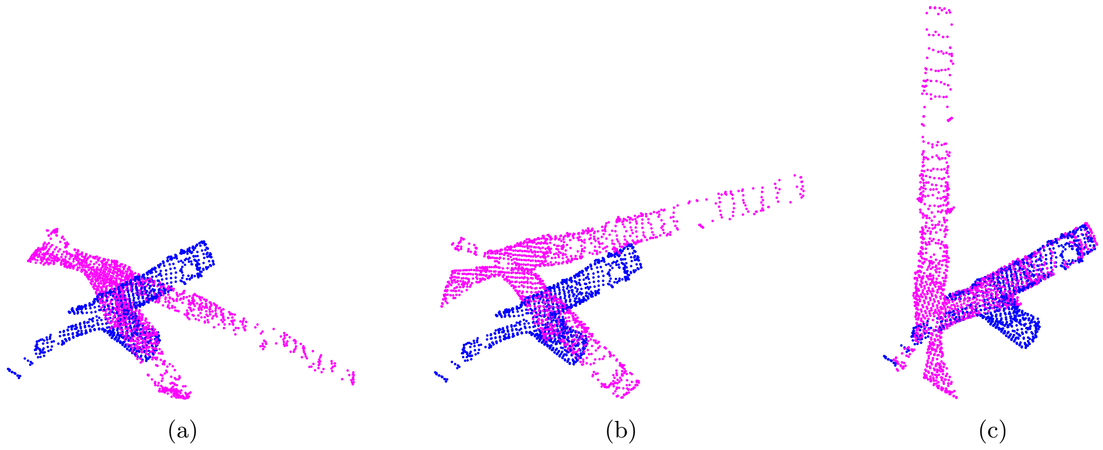


FIGURE 1.1: Example of rotation search in point cloud registration. (a) Two input point clouds that are related by a rotation. (b) Wrong result produced by a local method (ICP). (c) The globally optimal alignment according to the geometric matching criterion.

The non-convexity in rotation search is not only due to the non-convexity of robust objective functions. Hartley and Kahl [32] presented several L_∞ and Euclidean norm multiple view geometric problems with quasi-convex objective functions. In many cases, such problems can be solved efficiently. However, if rotational parameters form part of the unknowns, the problems become nonconvex due to the additional constraints arising from the space of rotations.

To address nonconvex optimisation, usually practical applications employ heuristics, approximate methods or local algorithms, which do not guarantee optimality. In order to globally solve rotation search, recent advances use the branch and bound (BnB) optimisation framework to efficiently explore the problem's parametric space. The work presented in this thesis contributes to the usage of BnB to robustly solve rotation search in real situations.

1.3 Current rotation search methods

Recent methods for rotation search can be distinguished into two groups. The first group [28, 52, 3, 59, 12] takes as input *a priori* established point correspondences. The second group [11, 41, 21, 73, 60, 16, 12] conducts rotation search on the *raw data*, where point correspondences are not determined beforehand. Such challenging data occurs frequently in real-life applications, e.g., surveying, mining, construction, and robotics. Figures 1.1 and 1.3 are examples of rotation search problems on raw data and data with point correspondences for 3D scans obtained from an underground mine. Note that the point clouds only partially overlap.

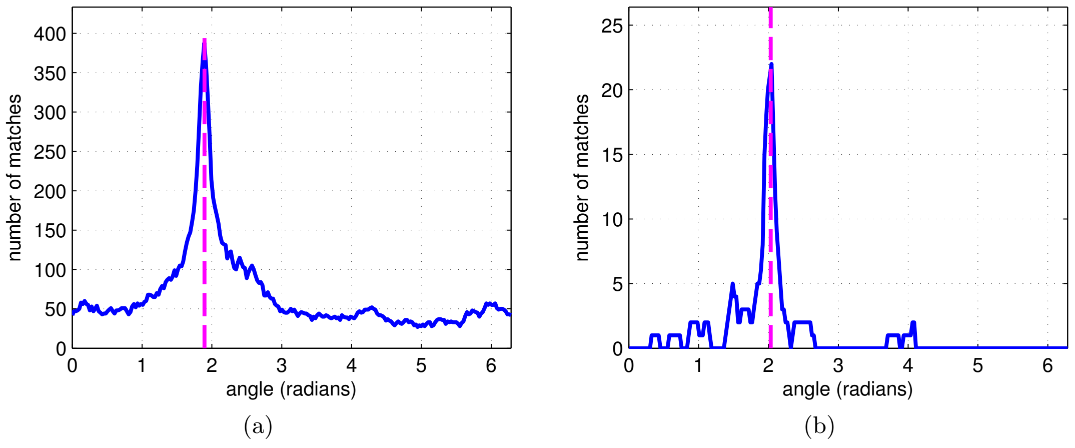


FIGURE 1.2: Plot of robust objective functions used to solve rotation search problems in Figures 1.1 and 1.3. (a) The geometric matching objective function is plotted for 250 rotation angles in $[0, 2\pi]$ about the optimal axis of rotation. (b) The consensus set maximisation objective function is plotted for 250 rotation angles in $[0, 2\pi]$ about the optimal axis of rotation. In both plots, the optimum rotation angle is at the vertical line.

Recently, BnB has gained popularity in the computer vision community as an optimisation framework suitable for nonconvex global optimisation. As discussed before, geometric computer vision problems are generally nonconvex, if rotational parameters need to be estimated. Breuel [11] pioneered the use of BnB in geometric computer vision by registering 2D points to a known 2D model. In Breuel's formulation, registration is solved on the raw data, i.e., without the need for a pre-established set of point correspondences. However, a naive extension to estimate the six parameters of a 3D rigid registration problem is unwieldy, where the primary difficulty is the estimation of the 3D rotational parameters.

More recently, several BnB methods have been presented to globally solve registration problems on point correspondences. These include Li and Hartley's algorithm [41] to rotationally align 3D point clouds, and the algorithms for 3D point cloud registration by Olsson et al. [53] and Enqvist and Kahl [22]. Another notable method that uses BnB to globally optimise camera pose with known point correspondences was presented by Hartley and Kahl [34].

Methods that operate on point correspondences are susceptible to the uncertainty of keypoint detection and matching techniques. In applications such as the registration of mine sites in Figure 1.1, it is difficult to find discriminative keypoints. Moreover, matching points between two point clouds is especially difficult when the genuine point correspondences lie in a small region. In the example in Figure 1.3, more than 99% of identified point correspondences are incorrect, i.e., they are outliers. Algorithms presented in [41, 34, 53] will fail in this type of data since they optimise objective functions that are not robust to outliers.

Global registration methods have been presented to solve problems without a pre-established set of point correspondences. Li and Hartley [41] presented a BnB algorithm to solve a "Lipschitzised" objective function. However, they assume point clouds of same size and a one-to-one valid matching for all points, i.e., no outliers. Go-ICP [73] encapsulates ICP [9] in a nested BnB scheme to find the optimal solution. Go-ICP does not assume one-to-one point matches, however its objective function is not robust to outliers. Moreover, optimality can be affected by approximations used in conducted nearest neighbour searches.

In addition to BnB, other methodologies have been proposed to globally solve geometric computer vision problems [52, 3, 19, 20]. They are based on exploring stationary points of subproblems of a fixed size. While this class of algorithm can robustly solve rotation search in polynomial time, they are too slow to be practical on typical data sizes.

The two main characteristics of algorithms pursued in this thesis are global optimality and robustness. The main contributions of the thesis are novel and *provably more efficient* algorithms for rotation search that satisfy the two characteristics (more details are in Section 1.5).

1.4 Robust estimation

Contamination in data is one of the common difficulties associated with many geometric computer vision problems. Contamination can be categorised into two types: noise and outliers. Noise is understood as low scale error; for instance, dispersion in measures produced during the data acquisition process. Usually, noise is modelled as a probability distribution function. Under the typical assumption of Gaussian noise, the *maximum likelihood* estimate corresponds to the least squares solution [31]. However, in many problems, contamination in data cannot be explained as noise only. Data often contains outliers, which usually manifest as large-scale errors. A typical example is incorrect assignments in data association processes. Since a noise model does not explain outliers, algorithms that only model contamination as noise will break down in outlier prone data.

Non-robust criteria such as least squares and L_∞ norm are inadequate in problems with outlier contamination. As a consequence, intrinsically robust criteria are required for *robust estimation*, i.e., the estimation of a parametric model under the presence of noise and outliers.

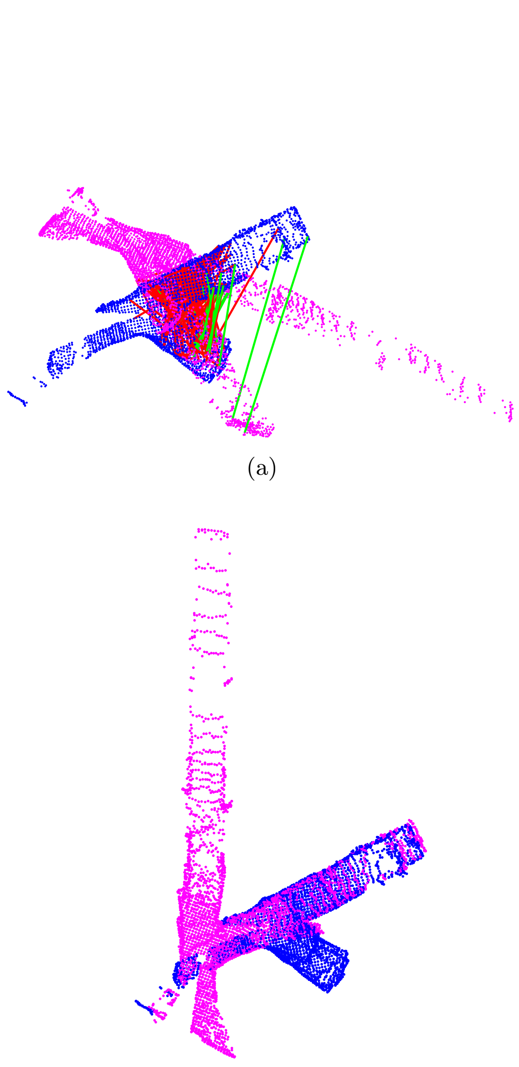

(a) (b) (c) (d)

Figure 1.3: Example of rotation search on point correspondences. The green lines represent inliers and the red lines represent outliers. There are only 24 inliers in a total of 250 point correspondences. (a) Initial alignment. (b) The initial alignment in (a) is translated for a better visualisation of point correspondences. (c) Optimal alignment found by BnB. (d) The approximate alignment produced for a RANSAC instance.

An intrinsically robust criterion is to find the solution that maximises number of points into alignment. This formulation is commonly referred to as *consensus set maximisation* when the input is a set of point correspondences, and as *geometric matching* [11] when the problem is solved with unknown point correspondences. Usually, this type of problems are much harder to solve, since the corresponding objective function is discrete and non-differentiable (see Figure 1.2).

Arguably the most popular algorithm for consensus set maximisation is RANSAC (RANdom SAmple Consensus) [24]. Briefly, RANSAC stochastically evaluates hypothesised models obtained by sampling minimal sets of the input data. Even though RANSAC is an approximate method, it has shown success in many computer vision applications. The main weakness of RANSAC is its stochastic nature, which results in unpredictable accuracy. Also, the runtime of RANSAC exponentially increases with respect to the outlier ratio: as such, it is impractical for highly contaminated data.

For example, in Figure 1.3, RANSAC found 21 matches, however the optimal solution has 24 matches as reported by BnB.

For registration on raw point clouds, a popular algorithm is ICP [9], however it is non-robust to outliers (see the incorrect alignment produced by ICP in Figure 1.1b). To increase robustness, Trimmed ICP [14] considers only the k closest matching point. In general, it is hard to estimate k, i.e., the number of points with a genuine match. Another variant is LM-ICP [26] which replaces the Euclidean norm of the ICP criterion with a robust norm that truncates errors above a threshold, and then solves the resulting problem using iterative optimisation. Whilst more robust than ICP, both variants remain locally convergent methods.

1.5 Research contributions

The contributions of this thesis are novel and efficient algorithms for robust and globally optimal rotation search.

Specifically, the major contributions are:

1. A globally optimal and real-time planar (1D) rotation search method for user-assisted point cloud registration [16]. The method conducts BnB optimisation with a novel bounding function whose evaluation amounts to simple sorted array operations. The presented algorithm greatly outperforms conventional registration methods on planar rotational motions (see Chapter 3).

2. A fast, globally optimal 3D rotation search method [60, 59]. For problems where it is hard to obtain good quality correspondences, this is a fast alternative that outperforms existing algorithms. Based on BnB, bound evaluations are accelerated by using stereographic projections to precompute and index all possible point matches in spatial R-trees (see Chapter 4).

3. A novel, guaranteed outlier removal for rotation search (GORE) [58]. When point correspondences (potentially with outliers) are given, the presented method remove outliers without compromising global optimality. Based on simple geometric operations, GORE is deterministic and fast. Used as a preprocessor to prune a large

portion of the outliers from the input data, GORE enables substantial speed-up of rotation search algorithms (see Chapter 5).

1.6 Thesis outline

The rest of this thesis is structured as follows. Chapter 2 presents a review of rotation search methods in geometric computer vision. The next two chapters propose geometric matching algorithms to globally solve for rotations directly on raw data. The special case for 1D rotations is presented in Chapter 3, while a method to handle 3D rotations is addressed in Chapter 4. Chapter 5 presents a practical guaranteed outlier removal method for rotation search which can be used to accelerate rotation search methods without compromising optimality. Finally, Chapter 6 provides a conclusion and a discussion on future research directions.

Chapter 2

The Rotation Search Problem

2.1 Introduction

Rotation search is a core sub-routine in many geometric vision applications such as 3D point cloud registration [6, 16, 60, 59], essential matrix and camera pose estimation [22, 34, 6], hand-eye calibration [68, 35, 64], camera localisation [69] and image stitching [6, 19, 20, 59]. This chapter presents the necessary background theory for rotation search, and reviews existing rotation search formulations and algorithms. In most practical cases, the input data for rotation search is contaminated with outliers, thus the focus is on *robust* rotation search.

The rotation search problems in the literature can be divided into two main types.

Type 1 - estimate rotation from a set of point correspondences
Given a set of one-to-one point correspondences $\mathcal{H} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, where each $(\mathbf{x}_i, \mathbf{y}_i)$ is a pair of corresponding points in $\mathbb{R}^3$, find the rotation $\mathbf{R} \in \mathrm{SO}(3)$ that best aligns the corresponding points. This problem type was examined in [34, 68, 64, 7, 58].

Type 2 - estimate rotation from "raw" input point sets
Let $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^M$ and $\mathcal{Y} = \{\mathbf{y}_j\}_{j=1}^N$ be two point sets in $\mathbb{R}^3$, where $\mathcal{X}$ and $\mathcal{Y}$ can have different cardinalities in general. Find a rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$ that best aligns $\mathcal{X}$ and $\mathcal{Y}$. This type of problem was examined in [41, 16, 60, 59].

In general, problem Type 2 is much more difficult, since algorithms need to simultaneously solve for the rotation and point correspondences (the latter can be estimated explicitly or implicitly). One of the purposes of this chapter is to survey the specific rotation search formulations proposed in the literature and the associated algorithms.

The rest of this chapter is organised as follows: Sections 2.2 presents common rotation representations. Section 2.3 presents rotation distances. Section 2.4 surveys rotation search algorithms. Section 2.5 elaborates on the BnB optimisation framework, since this methodology forms the basis of the novel algorithms to be presented in Chapters 3 and 4. Finally, a summary of this chapter is presented in Section 2.6.

2.2 Rotation representations

This section presents common rotation representations and their relationships. A large part of the description is based on [33].

2.2.1 Matrix representation

The most common way to represent a rotation is by a 3×3 orthogonal matrix with determinant equal to 1. Thus, the set of rotations is given by

$$\mathrm{SO}(3) = \left\{ \mathbf{R} \in \mathbb{R}^{3\times 3} \mid \mathbf{R}^\top \mathbf{R} = \mathbf{I},\ \det(\mathbf{R}) = 1 \right\}, \tag{2.1}$$

where $\mathbf{I}$ is the 3×3 identity matrix and $\det(\cdot)$ calculates the matrix determinant.

Results from Lie theory are used to connect the rotation matrix representation to alternative representations. SO(3) is a *Lie group* with associated *Lie algebra* $\mathfrak{so}(3)$ consisting of all 3×3 skew-symmetric matrices. $\mathfrak{so}(3)$ is connected with SO(3) by means of the exponential map $\exp : \mathfrak{so}(3) \to \mathrm{SO}(3)$. Moreover, the exponential map allows to connect $\mathbb{R}^3$ to SO(3) since any Lie algebra member in $\mathfrak{so}(3)$ can be represented by a vector $\mathbf{v} = [x, y, z]^\top$ in $\mathbb{R}^3$. The corresponding skew-symmetric matrix can be obtained by the map $\mathbf{v} \to [\mathbf{v}]_\times$, where

$$[\mathbf{v}]_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}. \tag{2.2}$$

2.2.2 Axis-angle representation

A more intuitive rotation representation is the *axis-angle* representation. *Euler's rotation theorem* states that a rotation (or a sequence of rotations) is equivalent to a single rotation about a fixed axis. Thus, every rotation in SO(3) can be represented as a *rotation angle* θ about a unitary *rotation axis* $\hat{\mathbf{r}}$ in $\mathbb{R}^3$. This means each rotation can be

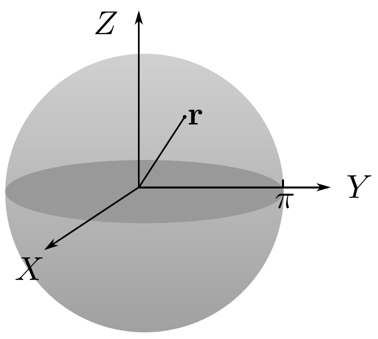


FIGURE 2.1: Illustrating the π-ball.

encoded as a vector

$$\mathbf{r} = \theta\hat{\mathbf{r}} \tag{2.3}$$

in $\mathbb{R}^3$.

Without restricting θ, the axis-angle representation is not unique. For instance, $\theta\hat{\mathbf{r}}$ and $(2\pi-\theta)(-\hat{\mathbf{r}})$ are the same rotation. To define a unique representation, the rotation angle is restricted to take values in $[0,\pi]$. Then, the corresponding rotation domain is a closed ball $B_\pi \subset \mathbb{R}^3$ of radius π and centred at the the origin as illustrated in Figure 2.1—this is usually also referred to as the π-*ball*. However, the B_π representation is not entirely devoid of redundancies, since oppositely oriented points on the boundary of B_π represent the same rotation. Intuitively, when rotating π radians, the orientation of the rotation axis is irrelevant.

The connection between the axis-angle and the matrix representation is given by the exponential map. Specifically, the rotation matrix $\mathbf{R}$ in SO(3) corresponding with the axis-angle representation vector $\mathbf{r}$ in B_π is given by the exponential map over the Lie algebra matrix $[\mathbf{r}]_\times$.

The exponential map can be obtained in closed form using *Rodrigues' formula*

$$\exp\left([\theta\hat{\mathbf{r}}]_\times\right) = \mathbf{I} + \sin(\theta)[\hat{\mathbf{r}}]_\times + (1-\cos(\theta))([\hat{\mathbf{r}}]_\times)^2. \tag{2.4}$$

The vector in $\mathbb{R}^3$ equivalent to $\mathbf{R}$ can be obtained by the logarithm map $\log(\cdot) : \mathrm{SO}(3) \to \mathbb{R}^3$ defined as

$$\log(\mathbf{R}) = \begin{cases} \arcsin(\|\mathbf{y}\|_2)\dfrac{\mathbf{y}}{\|\mathbf{y}\|_2} & \text{if } \mathbf{y} \neq \mathbf{0} \\ \mathbf{0} & \text{if } \mathbf{y} = \mathbf{0}, \end{cases} \tag{2.5}$$

where $\mathbf{y} = [y_1, y_2, y_3]^\top$ is such that $[\mathbf{y}]_\times = \frac{1}{2}\left(\mathbf{R} - \mathbf{R}^\top\right)$.

2.2.3 Quaternion representation

Quaternions can be thought of as an extension of complex numbers consisting of 3 imaginary components $(v1, v3, v3)$ and a real part w. Analogously to complex numbers, a quaternion can be represented by a vector $\mathbf{q} = [w, v_1, v_2, v_3]^\top$ in $\mathbb{R}^4$. Rotations can be represented by *unit-quaternions*, i.e., quaternions with an Euclidean norm equal to one.

The quaternion representation is important from a theoretical point of view, but is also a convenient representation in practice. Firstly, it is a more compact representation than the matrix representation. Also, the quaternion representation is practical for composing rotations since it requires of fewer operations than the matrix representation[1]. The quaternion representation facilitates addressing numerical errors in rotations. The computed results may drift away from the correct rotation as a consequence of rounding errors in floating-point calculations. By normalising the resulting quaternion vector, a valid rotation is obtained, i.e., with an equivalent matrix representation as in (2.1).

There are many geometric vision algorithms that use the quaternion representation for rotations [37, 54, 51, 53].

Unit-quaternions form a group with the non-commutative quaternion multiplication: the *Hamilton* product. The unit-quaternion group is a smooth Lie 3-manifold embedded in $\mathbb{R}^4$ referred to as the *unit-quaternion sphere* and denoted as S^3. By writing a quaternion as a pair $(w, \mathbf{v})$ where w is the real part and $\mathbf{v} = [v_1, v_2, v_3]^\top$ is the imaginary vector, the Hamilton product for two quaternions $\mathbf{q}_1 = (w_1, \mathbf{v}_1)$ and $\mathbf{q}_2 = (w_2, \mathbf{v}_2)$ is

$$\mathbf{q}_1 \cdot \mathbf{q}_2 = (w_1 w_2 - \langle \mathbf{v}_1, \mathbf{v}_2 \rangle,\ w_1 \mathbf{v}_1 + w_2 \mathbf{v}_2 + \mathbf{v}_1 \times \mathbf{v}_2) \tag{2.6}$$

where $\langle \cdot, \cdot \rangle$ is the standard inner product and $\times$ the cross product in $\mathbb{R}^3$.

Unit-quaternions and the axis-angle representation are closely related. The quaternion representation for a rotation encoded in the axis-angle vector $\theta \hat{\mathbf{r}}$ is

$$\mathbf{q} = (\cos(\theta/2),\, \sin(\theta/2)\hat{\mathbf{r}}). \tag{2.7}$$

The rotation matrix $\mathbf{R}$ equivalent to $\mathbf{q}$ can be obtained by using the exponential map over its axis-angle representation, i.e.,

$$\mathbf{R} = \exp\left([\theta \hat{\mathbf{r}}]_\times\right). \tag{2.8}$$

[1]Matrix multiplication requires 27 multiplications and 18 additions, whereas 16 multiplications and 12 additions are needed if using quaternions.

The quaternion representation helps in visualising SO(3). S^3 is a two-fold covering space of SO(3) in which opposite points encode the same rotation. In S^3, rotations with rotation angle less than π are mapped one-to-one to the "Northern Hemisphere" with the "North Pole" $[1,0,0,0]^\top$ encoding the identity rotation, i.e., with rotation angle equal to 0.

2.3 Rotation distances

This section provides intuition relating to measuring rotations, and presents results to bound SO(3) that are important in BnB rotation search formulations. A large part of the description is based on [33].

2.3.1 Angular distance

An intuitive metric to compare rotations is the *angular distance* which takes values in the $[0,\pi]$ interval. The angular distance of a rotation $\mathbf{R}$ with respect to the identity rotation $\mathbf{I}$, is given by the angle of rotation about the rotation axis of $\mathbf{R}$, and forced to lie in $[0,\pi]$ by inverting the rotation axis of $\mathbf{R}$ if needed.

For two rotations $\mathbf{R}$ and $\mathbf{S}$, the angular distance $d_\angle(\mathbf{R},\mathbf{S})$ is defined as the rotation angle of $\mathbf{R}\mathbf{S}^\top$. When rotations are expressed as matrices, the angular distance can be computed by using of the logarithm map. More specifically,

$$d_\angle(\mathbf{R},\mathbf{S}) = d_\angle(\mathbf{R}\mathbf{S}^\top,\mathbf{I}) = \|\log(\mathbf{R}\mathbf{S}^\top)\|_2. \tag{2.9}$$

The angular distance between two unit-quaternions $\mathbf{r}$ and $\mathbf{s}$ (with equivalent rotation matrices $\mathbf{R}$ and $\mathbf{S}$) is

$$d_\angle(\mathbf{R},\mathbf{S}) = 2\arccos(|w|), \text{ where } (w,\mathbf{v}) = \mathbf{r}^{-1}\mathbf{s}. \tag{2.10}$$

2.3.2 Chordal distance

The distance between rotation matrices can be computed directly on the matrix representation by means of the Frobenius norm $\|\cdot\|_\mathrm{F}$ on $\mathbb{R}^{3\times3}$. The Frobenius norm is the natural extension of the Euclidean norm to matrices. The *chordal distance* between two matrices $\mathbf{R}$ and $\mathbf{S}$ is defined as

$$d_\mathrm{chord}(\mathbf{R},\mathbf{S}) = \|\mathbf{R}-\mathbf{S}\|_\mathrm{F}. \tag{2.11}$$

The chordal distance is related to the angular distance by

$$d_{\text{chord}}(\mathbf{R}, \mathbf{S}) = 2\sqrt{2} \sin\left(\frac{d_\angle(\mathbf{R}, \mathbf{S})}{2}\right). \tag{2.12}$$

2.3.3 Quaternion distance

The quaternion distance is the Euclidean distance of unit-quaternions embedded in $\mathbb{R}^4$. Since opposite points in S^3 correspond to the same rotation, some care is needed to properly define a distance. This ambiguity can be fixed by defining the quaternion distance for two unit-quaternions $\mathbf{r}$, $\mathbf{s}$ with equivalent rotation matrices $\mathbf{R}$, $\mathbf{S}$ as

$$d_{\text{quat}}(\mathbf{R}, \mathbf{S}) = \min\left\{\|\mathbf{r} - \mathbf{s}\|_2,\ \|\mathbf{r} + \mathbf{s}\|_2\right\}. \tag{2.13}$$

The quaternion distance is related to the angular distance by

$$d_{\text{quat}}(\mathbf{R}, \mathbf{S}) = 2 \sin\left(\frac{d_\angle(\mathbf{R}, \mathbf{S})}{4}\right). \tag{2.14}$$

2.3.4 Axis-angle distance

In a similar fashion to the quaternion distance, the *axis-angle distance* is defined by means of the Euclidean norm in $\mathbb{R}^3$. However, such a definition leads to discontinuities for rotations near to the boundary of B_π. For example, If $\mathbf{r}$ and $\mathbf{s}$ are restricted to lie in B_π, its Euclidean distance could be larger than the Euclidean distance of two vectors encoding the same rotations but one of them is outside of B_π. This discontinuity issue can be fixed by redefining the logarithm map to consider equivalent axis-angle vectors outside of B_π

$$d_{\log}(\mathbf{R}, \mathbf{S}) = \min_{\substack{\mathbf{r},\mathbf{s} \\ \exp[\mathbf{r}]_\times = \mathbf{R} \\ \exp[\mathbf{s}]_\times = \mathbf{S}}} \|\mathbf{r} - \mathbf{s}\|_2. \tag{2.15}$$

The axis-angle distance is very useful for bounding the angular distance. The following bounds are valid for the axis-angle representation [4]

$$d_\angle(\mathbf{R}, \mathbf{S}) \le d_{\log}(\mathbf{R}, \mathbf{S}) \le \frac{\pi}{2} d_\angle(\mathbf{R}, \mathbf{S}). \tag{2.16}$$

Many recent BnB algorithms for rotation search use the above result to derive bounds for their objective functions [41, 34, 60, 7, 73, 64].

2.4 Rotation search methods

This section formally defines rotation search problems, and surveys existing variants of the problem and associated algorithms.

2.4.1 Minimal cases

Given point correspondences $\mathcal{H} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ that are related by a rotation $\mathbf{R}$, such that $\mathbf{y}_i = \mathbf{R}\mathbf{x}_i$, to find $\mathbf{R}$ is equivalent to finding the rotation between the underlying coordinate systems. Consequently, determining the minimal case corresponds to establishing the minimum number of points needed to uniquely define a coordinate system. Two points only are sufficient to define a coordinate system if they are non collinear with the origin. The rotation that relates the defined coordinate systems is the same that relates the underlying coordinate systems.

The minimum number of correspondences $(\mathbf{x}_i, \mathbf{y}_i)$ to align two point clouds related by rotation is two [37]. Given two correspondences $(\mathbf{x}_1, \mathbf{y}_1)$ and $(\mathbf{x}_2, \mathbf{y}_2)$, the coordinate system $(\hat{\mathbf{i}}_x, \hat{\mathbf{j}}_x, \hat{\mathbf{j}}_x)$ associated to $\mathcal{X}$ is uniquely defined by

$$\hat{\mathbf{i}}_x = \frac{\mathbf{x}_1}{\|\mathbf{x}_1\|} \tag{2.17}$$

$$\hat{\mathbf{j}}_x = \frac{\mathbf{x}_2 - [\mathbf{x}_2 \cdot \hat{\mathbf{i}}_x]\hat{\mathbf{i}}_x}{\|\mathbf{x}_2 - [\mathbf{x}_2 \cdot \hat{\mathbf{i}}_x]\hat{\mathbf{i}}_x\|} \tag{2.18}$$

$$\hat{\mathbf{j}}_x = \hat{\mathbf{i}}_x \times \hat{\mathbf{j}}_x. \tag{2.19}$$

Analogously, the coordinate system $(\hat{\mathbf{i}}_y, \hat{\mathbf{j}}_y, \hat{\mathbf{j}}_y)$ associated to $\mathcal{Y}$ can be defined by using the two associated points from $\mathcal{Y}$. Then, the problem is reduced to finding the rotation that aligns both coordinate systems.

Given two point sets $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$ and $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^N$, that are not in correspondence, the minimal case is given by 2 subsets with 2 points each: $\{\mathbf{x}_1, \mathbf{x}_2\} \subseteq \mathcal{X}$ and $\{\mathbf{y}_1, \mathbf{y}_2\} \subseteq \mathcal{Y}$. These 2 subsets lead to 2 possible point assignments $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2)\}$ and $\{(\mathbf{x}_1, \mathbf{y}_2), (\mathbf{x}_2, \mathbf{y}_1)\}$ for which a rotation can be estimated on each combination.

2.4.2 Least squares methods

A classical optimality criterion is to minimise the sum of squared residuals. Although least squares (LS) methods are non-robust to outliers, LS algorithms can be useful as sub-routines in robust methods for Type 1 problems (see page 11). For problem Type 2 (see page page 11), LS algorithms are useful in refining rough alignments produced by approximate algorithms.

2.4.2.1 Least squares methods for Type 1 problems

For problem Type 1, the LS method minimises the sum of squared residuals

$$\min_{\mathbf{R}\in\mathrm{SO}(3)} \sum_{i=1}^{N} r_i^2, \qquad r_i = d(\mathbf{R}\mathbf{x}_i, \mathbf{y}_i), \tag{2.20}$$

where $d(\cdot,\cdot)$ is some distance function which usually corresponds to the Euclidean distance or the angular distance (see Section 2.3.1). The LS rotation can be efficiently obtained in closed form, e.g., by using Horn's [37] method and SVD [2].

It can be easily shown that the solution of (2.20) is the solution of

$$\max_{\mathbf{R}\in\mathrm{SO}(3)} \sum_{i=1}^{N} \langle \mathbf{y}_i, \mathbf{R}\mathbf{x}_i \rangle \tag{2.21}$$

when choosing $d(\cdot)$ as the Euclidean distance, where $\langle\cdot,\cdot\rangle$ is the dot product.

The information needed in the following two closed-form methods is contained in the *cross-covariance* matrix

$$\mathbf{S} = \sum_{i=1}^{N} \mathbf{x}_i \mathbf{y}_i^\top. \tag{2.22}$$

Horn's method The method proposed by Horn [37] uses the quaternion representation. An equivalent quaternion based algorithm was addressed by Faugeras [23].

The quaternion form of the LS problem (2.21) is

$$\max_{\mathbf{r}\in S^3} \sum_{i=1}^{N} \langle \mathbf{r}\mathbf{x}_i\mathbf{r}^{-1}, \mathbf{y}_i \rangle, \tag{2.23}$$

where S^3 is the unit-quaternion sphere (see Section 2.2.3), and vectors $\mathbf{x}_i$ and $\mathbf{y}_i$ are now understood as the purely imaginary equivalent quaternions. The objective function of the quaternion formulation is equivalent to

$$\sum_{i=1}^{N} \langle \mathbf{r}\mathbf{x}_i, \mathbf{y}_i\mathbf{r} \rangle. \tag{2.24}$$

Equation (2.24) can be rewritten as $\mathbf{r}^\top \mathbf{Q}\mathbf{r}$, for $\mathbf{Q}$ being a 4×4 symmetric matrix derived uniquely from elements of $\mathbf{S}$. The quaternion $\mathbf{r}$ that maximises the objective function of (2.23) corresponds to the eigenvector associated with the largest eigenvalue of $\mathbf{Q}$. See [37] for a proof, and how to build $\mathbf{Q}$.

SVD method The LS solution can also be calculated by SVD on $\mathbf{S}$. This method, presented in [2], computes an orthonormal matrix $\mathbf{X}$ ($\det(\mathbf{X}) = \pm 1$) that maximises the objective function of (2.21).

More precisely, the orthonormal matrix $\mathbf{X}$ is obtained as

$$\mathbf{X} = \mathbf{V}\mathbf{U}^\top, \tag{2.25}$$

where $\mathbf{V}$ and $\mathbf{U}$ are from the SVD of $\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$.

Proof that $\mathbf{X}$ is the LS solution when it corresponds to a rotation ($\det(\mathbf{X}) = 1$) is presented in [2]. In the case that $\mathbf{X}$ is a reflection matrix ($|\mathbf{X}| = -1$), it is also shown in [2] that by replacing $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$ by $\mathbf{V}' = [\mathbf{v}_1, \mathbf{v}_2, -\mathbf{v}_3]$, the resulting $\mathbf{X}$ is a rotation matrix and it maximises the objective function of (2.21).

2.4.2.2 Least squares methods for Type 2 problems

The global LS solution for a problem of Type 2 (see page 11) was addressed in [41]. Assuming that $\mathcal{X}$ and $\mathcal{Y}$ have the same number of points, and each point in $\mathcal{X}$ has a valid match in $\mathcal{Y}$, the point assignment for $\mathcal{X}$ and $\mathcal{Y}$ can be formulated as an *permutation matrix*. A permutation matrix $\mathbf{P} \in \mathbb{P}^N$ is defined as a $N \times N$ matrix such that elements $p_{ij} = 1$ if $\mathbf{x}_i$ is in correspondence with $\mathbf{y}_j$, and 0 otherwise. Thus, the LS problem is as follows

$$\min_{\mathbf{R}\in SO(3),\, \mathbf{P}\in\mathbb{P}^N} \sum_{i=1}^{N} r_i^2, \quad r_i = d(\mathbf{R}\mathbf{P}\mathbf{x}_i, \mathbf{y}_i). \tag{2.26}$$

Li and Harley [41] used a BnB formulation to globally solve the above problem. The experiments in [41] took more than 1000 seconds on $N = 200$ points, whereas only ≈ 0.005 seconds were needed for the LS closed-form methods of Type 1 tested in [18] for problems of 200 point correspondences.

Another suitable methodology is iterative closest point (ICP) [9]. ICP is a *local method* that operates by alternating between two simple steps: point assignments and updates to the transform parameters.

The ICP algorithm is presented in Algorithm 2.1. Briefly, ICP iteratively rotates one point cloud (*the source*), such that the sum of squared residuals with respect to the other point cloud (*the target*) is minimised. At each iteration, residuals—associated to points in the source point cloud—are defined to be the distance to the closest point in the target point cloud.

To increase robustness, trimmed ICP [14] considers only the k closest matching points. However, *a priori* knowing the number of points $k < M$ from $\mathcal{X}$, that has a valid match in $\mathcal{Y}$, can be nontrivial in practice. Another variant is LM-ICP [26] which replaces the L_2-norm used in the ICP criterion with a robust norm that truncates errors above a threshold, then solves the resulting problem using iterative optimisation. Whilst more robust than ICP, both variants remain locally convergent methods.

Algorithm 2.1 ICP for rotation search.

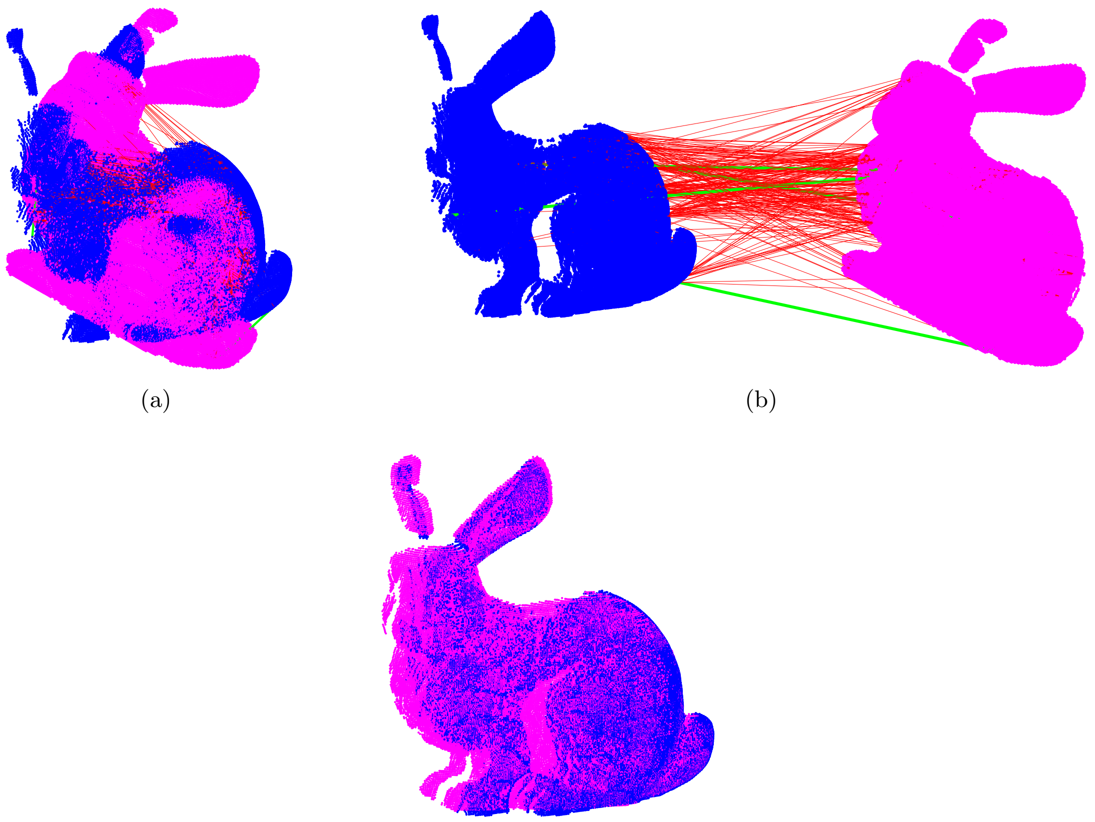
```
Require: Point clouds $\mathcal{X}$ and $\mathcal{Y}$, error $\epsilon$.
1: For each point in $\mathcal{X}$, finds the closest point in $\mathcal{Y}$.
2: Estimate rotation in closed-form (see Section 2.4.2.1) on the point correspondences
   obtained in the previous step.
3: Apply estimated rotation to points in $\mathcal{X}$.
4: Iterate until the squared residual error is below $\epsilon$.
```

2.4.3 Outliers in rotation search

Outliers are differently characterised in Type 1 and Type 2 problems. In the context of Type 1 problems, outliers are incorrect point correspondences $(\mathbf{x}_i, \mathbf{y}_j)$ between two point clouds. Figure 2.2 is an example of a Type 1 problem highly contaminated with outliers. A total of 250 point correspondences were obtained and plotted as green and red lines to distinguish between inliers and outliers. More than 98% of point correspondences are incorrect, and, only 4 of the total of 250 point correspondences are inliers. Usually, the high number of outliers makes it hard to solve for rotation in real 3D data, for which 3D keypoint detection and matching techniques [72, 66, 74] are much less accurate than their 2D counterparts (such as SIFT [43] and SURF [5]). Moreover, the accuracy of keypoint detection and matching techniques can be especially compromised in point clouds with a small overlapping region or insufficient sampling resolution.

In Type 2 problems, outliers are non-overlapping points. More precisely, for 2 point clouds $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{M}$ and $\mathcal{Y} = \{\mathbf{y}_j\}_{j=1}^{N}$, outliers are such points that under the optimal rotation will not match any point in the other point cloud.

Non-overlapping regions may be caused by several factors such as self-occlusion, dust scattering and non-reflective surfaces. In many cases, outliers are simply the consequence



(a) (b) (c)

FIGURE 2.2: Example of a Type 1 problem with a significant amount of outliers. (a) Initial pose of point clouds. Red lines indicate outliers (wrong matches) and green lines are inliers. (b) For a better visualisation of the massive amount of outliers, a point cloud was translated. (c) Optimal alignment obtained by BnB.

of scanning a scene from different viewpoints. An example is presented in Figure 2.3 for the problem of registering an underground mine site. Note the small overlapping region in the acceptable alignment in Figure 2.3b. As a consequence of the massive large number of outliers, the robust criterion of geometric matching failed to register point clouds in Figure 2.3. The acceptable alignment in Figure 2.3a was obtained after manually fixing translational parameters to search for the optimal rotation in accordance to the geometric matching criterion.

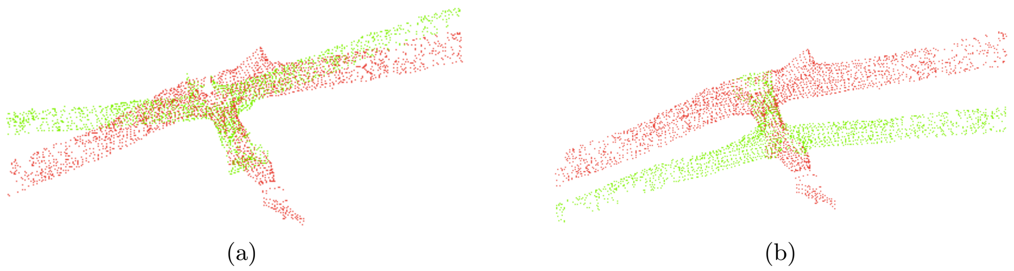


FIGURE 2.3: Example of a Type 2 problem with a significant amount of outliers. (a) Global optimal for full 3D registration under the geometric matching criterion. (b) Acceptable alignment produced by a robust rotation search method after manually fixing translation parameters.

2.4.4 Consensus set maximisation methods

A popular robust criterion to solve Type 1 problems (see page 11) is *consensus set maximisation* (CSM)

$$\begin{aligned} \max_{\mathbf{R}\in SO(3),\, \mathcal{I}\subseteq\mathcal{H}} \quad & |\mathcal{I}| \\ \text{subject to} \quad & d(\mathbf{R}\mathbf{x}_i, \mathbf{y}_i) \leq \epsilon,\ \forall i \in \mathcal{I}, \end{aligned} \tag{2.27}$$

where ϵ is an error threshold and $d(\cdot,\cdot)$ a distance function. In words, CSM seeks the rotation $\mathbf{R}$ that agrees with as many of point correspondences as possible, where agreement is up to threshold ϵ.

2.4.4.1 RANSAC

Introduced by Fischler and Bolles [24], RANSAC (random sample consensus) is one of the most popular methods in computer vision for robust estimation. The goal of RANSAC is to maximise the consensus set by stochastically exploring hypothesised solutions from minimal sets—Only two point correspondences are needed to solve for rotations (see Section 2.4.1). The optimal value of RANSAC, in general, does not equal to the CSM solution.

Algorithm 2.2 presents RANSAC applied to rotation search problem for a given set of input correspondences between point clouds $\mathcal{X}$ and $\mathcal{Y}$. The number of RANSAC iterations K can be estimated from the inlier ratio w, i.e., the number of inliers over the number of points correspondences. The probability p of at least one minimal set of size n containing only inliers, and therefore being likely to produce a good estimation, is given by

$$1 - p = (1 - w^n)^K. \tag{2.28}$$

After taking logarithm,

$$K = \frac{\log(1-p)}{\log(1-w^n)}. \tag{2.29}$$

For example, if seeking rotation (Algorithm 2.2) with an inlier ratio of 50%, then $K \approx 24$ iterations are required to find at least one minimal set containing only true inliers with probability $p = 0.999$. Following the same example but choosing the inlier ratio $w = 5\%$, the required number of iterations is $K \approx 2760$. In effect, it can be shown that K increases exponentially with respect to the outlier ratio $(1 - w)$.

In practice, it is difficult to estimate the number of iterations of RANSAC since w is usually unknown and its estimation is generally nontrivial.

Algorithm 2.2 RANSAC for rotation search.

```
Require: Point cloud correspondences $\mathcal{H} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, maximum number of itera-
    tions $K$, and inlier threshold $\epsilon$.
 1: $\mathbf{R}^* \leftarrow \mathbf{I}$
 2: for $k = 1, \ldots, K$ do
 3:   Randomly sample two correspondences $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$ from $\mathcal{H}$.
 4:   Hypothesise a rotation $\mathbf{R}$ using a LS closed form method (see Section 2.4.2.1)
      from the minimal set $\{(\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}_j, \mathbf{y}_j)\}$.
 5:   Find the consensus set $\mathcal{I}_\mathbf{R} = \{(\mathbf{x}_i, \mathbf{y}_i), \text{such that } d(\mathbf{R}x_i, \mathbf{y}_i) < \epsilon, \forall i = 1 \ldots N\}$,
      where $d(\cdot)$ is a distance function (see Section 2.3).
 6:   if $|\mathcal{I}_\mathbf{R}| > |\mathcal{I}_{\mathbf{R}^*}|$ then
 7:     $\mathbf{R}^* \leftarrow \mathbf{R}$
 8:   end if
 9: end for
10: return The best found rotation $\mathbf{R}^*$.
```

2.4.4.2 Polynomial-time methods

Polynomial-time methods [52, 19, 20, 3] rely on the fact that the solution to (2.27) is equal to the solution of the same problem on a subset of $\mathcal{H}$ of size at most d, where d is 3 for rotation search. The rotation that realises the CSM objective function is found by enumerating all $\binom{N}{p}$ subsets of $\mathcal{H}$ for all $p \leq d$ and examining only rotations at KKT-points [19] or FJ-points [3, 20] related to the visited subset.

Despite polynomial time complexity in the problem size N, a general weakness for this type of algorithms is their high computational cost. In the case of [19], the number of unique subsets to test is enormous even for moderate N (e.g., for $N = 500$ there are ≥ 20 million 3-subsets).

2.4.4.3 Branch and bound

BnB has been successfully applied to CSM for rotation estimation. Bazin et al. [7] extended Hartley and Kahl's BnB rotation search algorithm [34] to solve the CSM criterion. The BnB method reported runtimes of a few seconds in rotational registration problems[2]. Section 2.5 is dedicated to BnB due to the importance of BnB to robustly solve Type 1 and Type 2 problems.

2.4.5 Geometric matching methods

The *geometric matching* (GM) criterion is intrinsically robust for Type 2 problems (see page 11). The GM method is given by

$$\max_{\mathbf{R}\in \mathrm{SO}(3)} \sum_{i=1}^{M} \max_{j=1\ldots N} \lfloor d(\mathbf{R}\mathbf{x}_i, \mathbf{y}_j) \leq \epsilon \rfloor , \tag{2.30}$$

where $d(\cdot,\cdot)$ is a distance function, ϵ is an error threshold, and $\lfloor\cdot\rfloor$ is the indicator function that returns 1 if its predicate is true, and 0 if not. Intuitively, the GM criterion is a discrete function that counts the number of matches when evaluating a certain rotation. Compared to the CSM formulation, GM is certainly a much harder problem since rotation and data assignment must be jointly solved, i.e., for each $\mathbf{x}_i$ is solved if there is a $\mathbf{y}_j$ that agrees up to the threshold ϵ at the optimal rotation. Therefore, GM is not limited by the inaccuracy of feature matching techniques as in Type 1 problems.

Notice that the GM criterion is nonconcave. Also, it is nonsymmetric, in the sense that the solution would change if point clouds $\mathcal{X}$ and $\mathcal{Y}$ were swapped. Many BnB algorithms have been proposed for solving Type 2 problems. Breuel [11] was one of the earliest to apply BnB for point cloud registration. Globally optimal rotation search methods for 3D points have been achieved by pre-computing and indexing all possible point matches for 1D rotation search [16] and 3D rotation search [60].

Note that ICP cannot solve (2.30) since its assignment step assumes that all points in $\mathcal{X}$ have a valid match in $\mathcal{Y}$.

2.5 Branch and bound

BnB is a general optimisation technique that ensures global optimality, up to the desired level of accuracy. What makes BnB relevant is its ability to solve difficult optimisation

[2]The number of keypoint correspondences were not explicitly reported, but from [7, Fig. 3] there seems to be approximately 100 keypoint correspondences.

problems, including nonconvex problems. BnB systematically decomposes and prunes the problem search space, where pruning is achieved using a bounding function. A detailed presentation of BnB can be found in [38]. In this section, BnB is presented in the context of rotation search. Also, a BnB algorithm prototype (Algorithm 2.3) is given as the general structure for BnB algorithms presented in this thesis. Finally, important techniques for exploring and bounding SO(3) are presented at the end of this section, as they might be useful for designing BnB algorithms for Type 1 and Type 2 problems (see page 11).

2.5.1 Branching step

BnB hierarchically partitions the parametric space. Without loss of generality, consider a maximisation problem for a function $f : \mathbb{T} \to \mathbb{R}$, where $\mathbb{T}$ is some parametric domain. BnB splits a region $\mathbb{T}$ into n disjoint regions to evaluate bounds of the objective function. n is called the *branch factor*. Hence, the problem can be structured into a tree with nodes associated to search space partitions.

2.5.2 Bounding step

The key of BnB relies on being able to define upper and lower bounds to perform feasibility tests when visiting partitions. Let assume that lower and upper bounds $l_{\mathbb{P}}$, $u_{\mathbb{P}}$ can be established for each partition $\mathbb{P} \subseteq \mathbb{T}$ such that

$$l_{\mathbb{P}} \leq \max_{x \in \mathbb{P}} f(x) \leq u_{\mathbb{P}}. \tag{2.31}$$

In the bounding step, upper and lower bounds are computed so that a complete branch of the tree can be safely discarded if its associated upper bound is lower than the largest found so far lower bound across all nodes. BnB terminates when the difference between the highest upper bound of all nodes and the best (highest) found so far lower bound is up to the desired accuracy.

A trivial lower bound can be obtained by simply evaluating f at the “centre of $\mathbb{P}$”. In fact, for all x in $\mathbb{P}$, $f(x)$ is a valid lower bound. However, upper bound functions are usually problem dependant and require more effort to define. Usually, there exist a trade-off between evaluation efficiency and bound tightness. Though a tight bounding function is likely to conduct more aggressive pruning of the search tree—and hence fewer bound evaluations—its evaluation time could negatively impact the time of the whole algorithm.

2.5.3 Search strategies

Different search strategies can be conducted. *Depth-first* and *breath-first* are common choices. Breath-first first examines all nodes at the same level in the tree before expanding. By contrast, depth-first examines deeper nodes first until reaching a pre-defined maximum level before to continuing to explore a lower level node. Hartley and Kahl [34] noticed that both strategies perform well for 3D rotation search. Although depth-first would fully traverse paths that originated from unpromising nodes, it uses less memory than the breath-first strategy. A third strategy is *best-first* which expands the tree by visiting first more *promising* nodes, e.g., nodes with the largest associated upper bound.

2.5.4 BnB algorithm prototype

Implementation of a best-first algorithm can be carried out by a priority queue where nodes are sorted according to their associated upper bound. A general rotation search algorithm conducting best-first search is presented in Algorithm 2.3. Visiting nodes according to their upper bound priority allows to check optimality at the visited node and safely terminate BnB without the need to visit all nodes in the tree (line 5 in Algorithm 2.3).

Similarly, depth-first search can also be implemented using a priority queue where the priority is given by the node's depth. Unlike best-first, all nodes in the queue have to be explored to ensure global optimality.

Algorithm 2.3 can be easily adapted for minimisation problems by interchanging the roles of upper and lower bound functions. Also notice that Algorithm 2.3 is completely defined if a splitting procedure (line 12) is defined and the upper (lower) bound function (line 14) is given in the case of a maximisation (minimisation) problem. A splitting procedure and angular distance bounds for SO(3) partitions are presented below.

2.5.5 Rotation space decomposition

As shown in Section 2.2.2, SO(3) can be represented for a π-ball, where each point in the π-ball corresponds to a rotation in the axis-angle representation.

Following [41], the π-ball can be conveniently divided into blocks. The cube $C_\pi = [-\pi, \pi]^3$ in $\mathbb{R}^3$ can be systematically divided; Figure 2.4 illustrates the decomposition for a block $\mathbb{B}$. To avoid over-exploring SO(3), a simple test can be carried out to discard cubes that do not intersect the π-ball.

Algorithm 2.3 BnB best-first algorithm prototype for maximising $f(x)$.

```
1: Initialise priority queue $q$, $y^* \leftarrow 0$, $T^* \leftarrow null$.
2: Insert $\mathbb{T}$ into $q$.
3: while $q$ is not empty do
4:    Obtain the highest priority partition $\mathbb{P}$ and its upper bound $u_{\mathbb{P}}$ from $q$.
5:    if $u_{\mathbb{P}} - y^* \leq \epsilon$ then
6:       Terminate
7:    end if
8:    Compute a lower bound $l_{\mathbb{P}}$ for $\mathbb{P}$ by evaluating $f(T)$ for some $T \in \mathbb{P}$.
9:    if $l_{\mathbb{P}} > y^*$ then
10:      $y^* \leftarrow l_{\mathbb{P}}$, $T^* \leftarrow T$.
11:   end if
12:   Split $\mathbb{P}$ into $n$ disjoint partitions $\{\mathbb{P}_i\}_{i=1}^n$.
13:   for all partitions $\mathbb{P}_i$, $i = 1 \ldots n$ do
14:      Compute an upper bound $u_{\mathbb{P}_i}$ for $\mathbb{P}_i$.
15:      if $u_{\mathbb{P}_i} > y^*$ then
16:         Insert $\mathbb{P}_i$ with priority $u_{\mathbb{P}_i}$ into $q$.
17:      end if
18:   end for
19: end while
20: return Optimal transformation $T^*$ with objective value $y^*$.
```

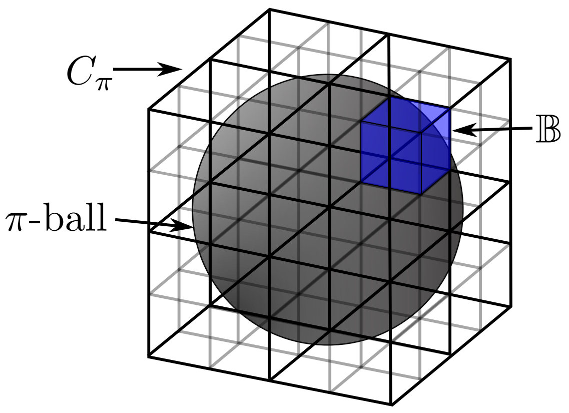


FIGURE 2.4: Illustrating the decomposition of the π-ball into blocks.

2.5.6 Bounding the rotation space

Following [41, 34], bounding functions can be derived from (2.16). The left hand side of inequality (2.16) can be rewritten as the following lemma.

Lemma 2.1. *If* $\mathbf{R}, \mathbf{S}$ *represent two rotations in* $SO(3)$ *and* $\mathbf{r}, \mathbf{s}$ *are the corresponding axis-angle vectors, then*

$$d_\angle(\mathbf{R}, \mathbf{S}) \leq \|\mathbf{r} - \mathbf{s}\|. \tag{2.32}$$

In words, the angular distance of two rotations is upper bounded by the Euclidean distance of their axis-angle representations.

To further derive a bounding function for f for a partition $\mathbb{P}$ of the π-ball, it is convenient to establish a bound $\delta_{\mathbb{P}}$ for the angular distance of rotations in $\mathbb{P}$ with respect to the "central" rotation, i.e.,

$$\max_{\mathbf{R}\in\mathbb{P}} \mathrm{d}_{\angle}(\mathbf{R}_c, \mathbf{R}) \leq \delta_{\mathbb{P}}, \tag{2.33}$$

where $\mathbf{r}_c$ is the centre of a partition $\mathbb{P}$ and $\mathbf{R}_c$ its equivalent rotation matrix.

It follows from Lemma 2.32 that $\delta_{\mathbb{P}}$ can be defined as

$$\delta_{\mathbb{P}} := \max_{\mathbf{r}\in\mathbb{P}} \left(\|\mathbf{r} - \mathbf{r}_c\|\right). \tag{2.34}$$

In the case of a block decomposition of the π-ball, the centre rotation and the angular bound $\delta_{\mathbb{B}}$ are derived from the block coordinates. Let $\mathbf{p}$ and $\mathbf{q}$ be the vectors at two opposite corners of $\mathbb{P}$. Then,

$$\mathbf{r}_c = \frac{(\mathbf{p}+\mathbf{q})}{2} \tag{2.35}$$

and

$$\delta_{\mathbb{P}} = \frac{\|\mathbf{p}-\mathbf{q}\|}{2}. \tag{2.36}$$

2.6 Summary

This chapter formulated rotation search problems and presented a review of main algorithms for rotation search used in geometric computer vision. Two problem categories were identified (see page 11). The first class of methods estimates rotations from a set of point correspondences (Type 1 problems) whereas the second estimates rotations from "raw" input point sets (Type 2 problems).

Algorithms were also classified according to the optimality criterion. Since robustness to outliers is one of the pursued objectives in this thesis, this chapter presented intrinsically robust methods for both type of problem. For Type 1 problems, an intrinsically robust criterion is CSM (see Section 2.4.4), whereas for Type 2 problems the criterion is GM (see Section 2.4.5). Also LS methods were presented (see Section 2.4.2). Although LS methods are non-robust to outliers, LS closed-form algorithms are invoked as a subroutine in RANSAC, which is a stochastic method for CSM.

In relation to finding the global solution, a BnB prototype (Algorithm 2.3) was presented as the general structure for BnB algorithms presented in this thesis.

Type 2 problems are challenging since point correspondences have to be estimated along with the optimal rotation. Since optimality is not affected by the uncertainty of point assignment procedures (point assignment is solved optimally), this class of problems is important for data with insufficiently distinctive local features. Chapters 3 and 4 present globally optimal methods under the GM criterion.

Chapter 3

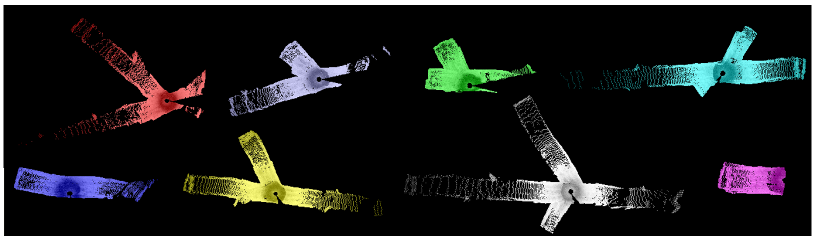
Robust 1D Rotation Search

3.1 Introduction

In surveying applications in construction, mining, safety inspection, and documentation of heritage structures, the motions or transformations that need to be estimated are usually horizontally constrained. This is because when sensing for terrestrial information using LIDAR scanning (see Figure 3.1), usually a calibration procedure is conducted in such a way that the captured range image or point cloud is horizontally levelled. Thus, the registration of two scans is reduced to a 4 DoF problem, where 3 DoF relate to translation and 1 DoF to azimuth.

Although registering horizontally levelled point clouds involves fewer parameters, it is still difficult to globally solve the reduced 4 DoF formulation in data contaminated with outliers, as is typically encountered in real-life surveying applications. In such cases,

FIGURE 3.1: Example of terrestrial LIDAR scanner. Source: Wikimedia Commons.


(a)

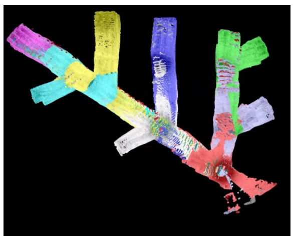

(b)

FIGURE 3.2: Registering mine sites. (a) A set of partially overlapping LIDAR scans (viewed from the top) from an underground mine and (b) their correct registration.

registration methodologies based on keypoint detection and matching are usually inadequate, especially on data with "organic" or "self-similar" structure such as underground mine sites. See Figure 3.2 for examples.

This chapter presents an *user-assisted system* for the registration problems in such a setting. The system generates instant guidance to the user to facilitate finding a correct alignment. The key component of the system is a novel *real-time* and *globally optimal* 1D rotation search algorithm. This algorithm is aimed at problem Type 2 (see page 11 in Section 2.1) to solve the GM criterion (see Section 2.4.5).

This chapter is organized as follows: Section 3.2 discuses the significance of point cloud registration and existing algorithms. Section 3.3 proposes the user-assisted point cloud registration system. Section 3.4 presents the 1D rotation search problem. Section 3.5 gives a detailed description of the proposed 1D rotation search algorithm. Section 3.6 experimentally compares the proposed rotation search algorithm against other techniques. Section 3.7 shows the usage of the user-assisted system in commercial software. Section 3.8 summarises the chapter.

3.2 Point cloud registration for LIDAR scans

LIDAR scans are useful in applications involving large-scale 3D modelling, surveying and mapping. In such applications, a recurrent task is to align multiple scans. Registration is difficult due to the small partial overlaps between the point clouds. See Figure 3.2. This occurs frequently in practice, since a surveyor will economise the work by spreading the scans over the scene uniformly, thus reducing scan overlaps.

Despite significant efforts, fully automatic registration of multiple point clouds remains challenging [70]. Globally optimal methods do not require human input [11, 41], but they can be very slow. Due to the difficulty of globally solving this kind of problems, practical systems usually use approximations or local methods. Approximate methods are fast,

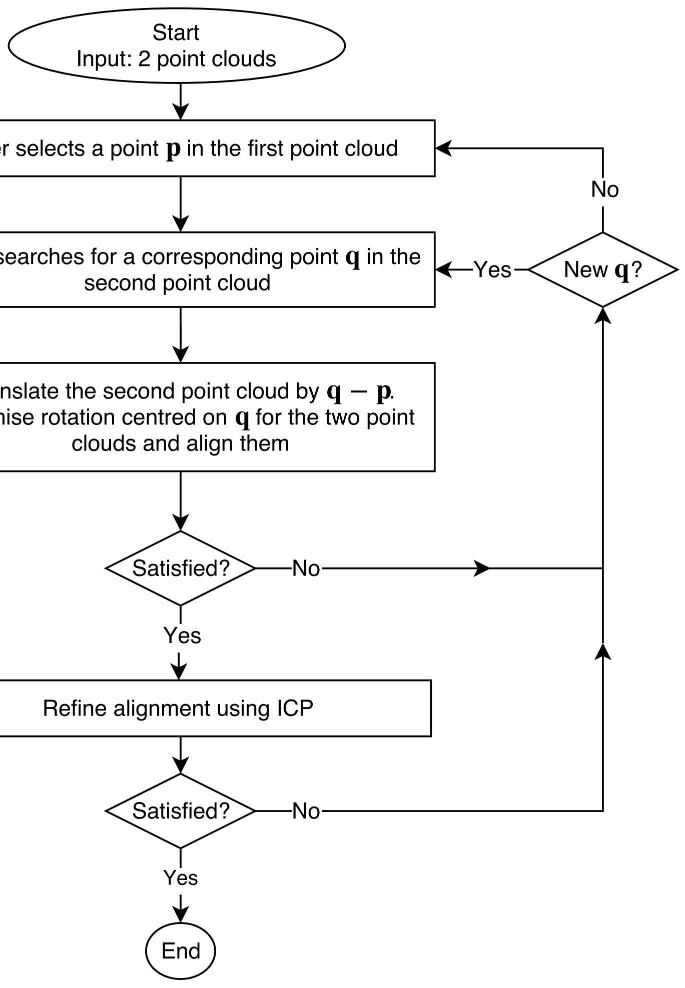

Start
Input: 2 point clouds
User selects a point $\mathbf{p}$ in the first point cloud
User searches for a corresponding point $\mathbf{q}$ in the second point cloud
Translate the second point cloud by $\mathbf{q} - \mathbf{p}$. Optimise rotation centred on $\mathbf{q}$ for the two point clouds and align them
Satisfied?
Yes
No
New $\mathbf{q}$?
Yes
No
Refine alignment using ICP
Satisfied?
Yes
No
End

FIGURE 3.3: Diagram of the user-assisted point cloud registration system.

such as those based on randomised heuristics [13, 1, 56, 71] or feature matching [28, 42, 27, 62]. However, approximate methods do not guarantee good results. Additionally, local methods such as ICP [9] and its variants [14, 25], only succeed on roughly aligned point clouds, i.e., they depend on good initialisations. Unfortunately, it is tough to initialise problems such as the example in Figure 3.2. In the case of underground scans, GPS localisation of the LIDAR devices cannot be used to initialise the registration. Manual initialisation can be tricky and laborious in "self-similar" point clouds such as in the given example.

3.3 User-assisted point cloud registration system

This section presents a novel *user-assisted* system with *real-time* interaction for point cloud registration. The proposed system, summarised in Figure 3.3, is less cumbersome and time-consuming than the typical process of alignment using only ICP. Guided by the system, the user's role is to identify overlapping regions across the point clouds. More specifically, given two input point clouds, the user first selects a point $\mathbf{p}$ in the first point cloud, then hovers the mouse over the second point cloud to look for a matching point. Each mouse position gives rise to a potential corresponding point $\mathbf{q}$, and the first point cloud is translated by vector $\mathbf{q} - \mathbf{p}$ such that $\mathbf{p}$ and $\mathbf{q}$ coincide. In real-time, the system calculates the rotation centred on $\mathbf{q}$ that best aligns the local point clouds around $\mathbf{p}$ and $\mathbf{q}$. This provides instant verification of the match $\mathbf{p} \leftrightarrow \mathbf{q}$. If the result is not satisfactory, the user can simply move the mouse to a different location, or restart by reselecting point $\mathbf{p}$. Once a satisfactory rotational alignment based on $\mathbf{p} \leftrightarrow \mathbf{q}$ is obtained, the user can refine the overall alignment by using ICP to estimate a full 3D rigid transform using all the points. This system allows the user to register multiple overlapping point clouds by simply repeating the steps above. Please watch a demonstration video in [15] or Figures 3.4, 3.5 and 3.6 for screenshots of the system in action.

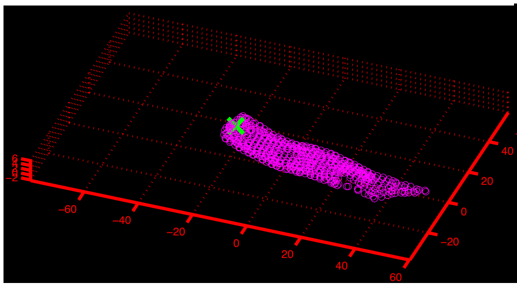
From a user's standpoint, identifying corresponding regions across point clouds is much easier than initialising their alignment as required in ICP. Arguably, manually initialising alignment involves first finding where the point clouds overlap. Our system thus requires much less effort on the part of the user; for example, the point sets in Figure 3.2 can be aligned in mere minutes. Note that $\mathbf{p}, \mathbf{q}$ need not be salient points or points with high curvature [27, 62]. A human observer can draw upon structural characteristics of the scene (e.g., end points or intersections of tunnels; see Figures 3.4 and 3.5) to quickly find overlaps.

Finally, although this section concentrates on LIDAR scans of underground mines, the system is certainly applicable to other kinds of objects or structures, as long as each point cloud is level with respect to the ground plane. Examples include the Stanford range data [17] where the objects were rotated on a turntable; see the demonstration video [15] or Figure 3.6.

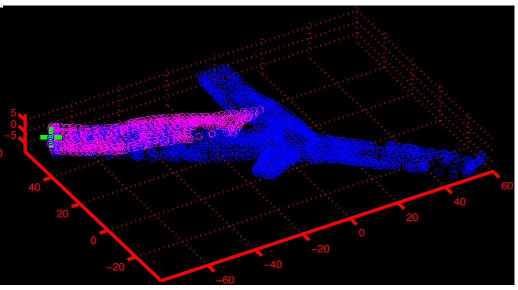
3.4 Rotation search for point cloud registration

Within the context of problem Type 2 (see page 11), here rotation search is solved for the GM criterion (see Section 2.4.5). Let $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{M}$ and $\mathcal{Y} = \{\mathbf{y}_j\}_{j=1}^{N}$ be two 3D points clouds. Assume that $\mathcal{X}$ and $\mathcal{Y}$ have been translated such that the corresponding points $\mathbf{p} \in \mathcal{X}$ and $\mathbf{q} \in \mathcal{Y}$ are at the origin in $\mathbb{R}^3$. The GM criterion seeks the rotation matrix $\mathbf{R} \in \mathrm{SO}(3)$ that maximises the function

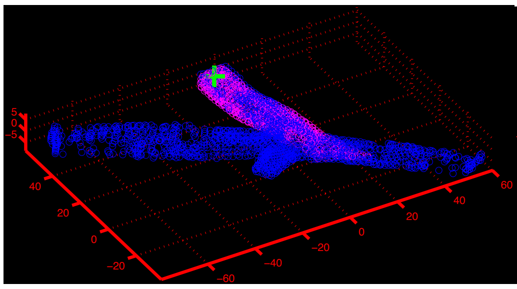
$$Q(\mathbf{R}) = \sum_{i=1}^{M} \max_{j=1\ldots N} \lfloor \|\mathbf{R}\mathbf{x}_i - \mathbf{y}_j\| \leq \epsilon \rfloor, \tag{3.1}$$



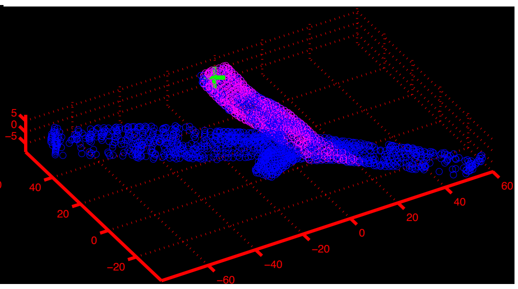
(a) Point **p** (green cross) selected by the user in the first point cloud.



(b) Point **q** (green cross) selected by the user in the second point cloud and the optimised rotational alignment.

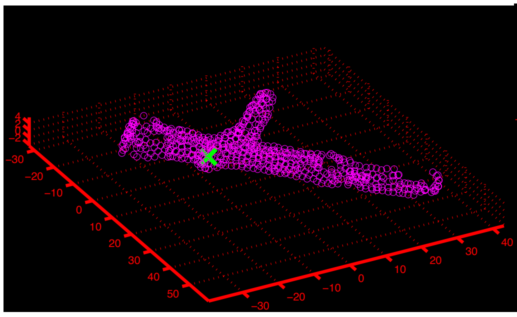


(c) A better point **q** and the rotational alignment.

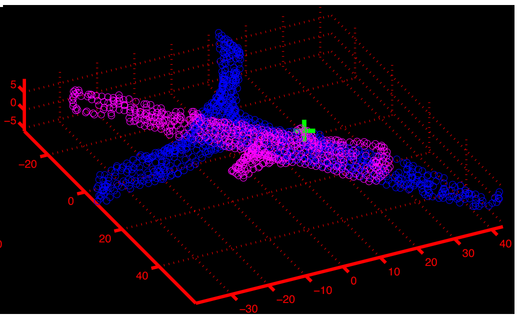


(d) ICP refinement of the result in (c).

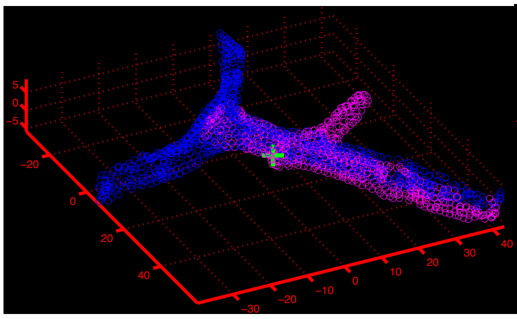
FIGURE 3.4: Sample result of the interactive 3D registration system on underground mine scans.



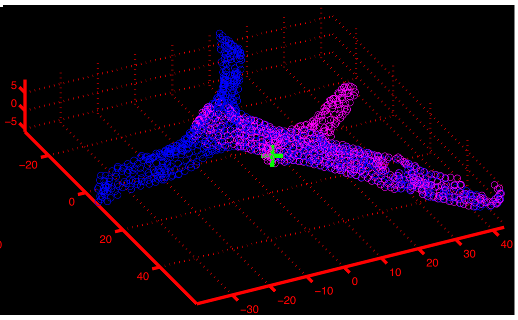
(a) Point **p** selected by the user in the first point cloud.



(b) Point **q** (green cross) selected by the user in the second point cloud and the optimised rotational alignment.

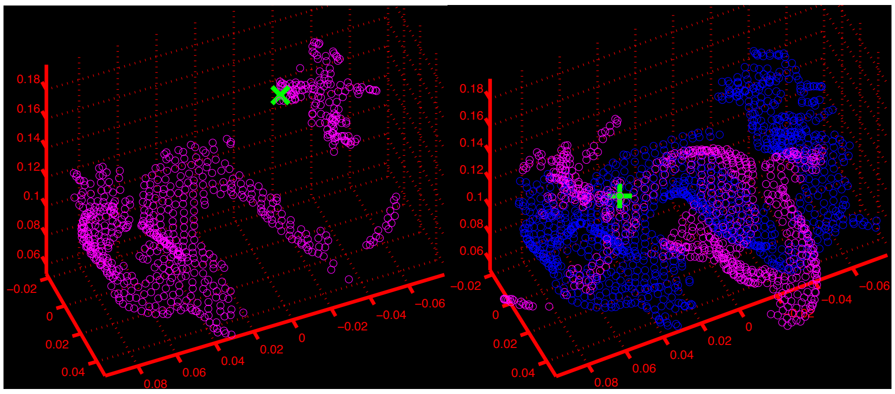


(c) A better point **q** and the rotational alignment.



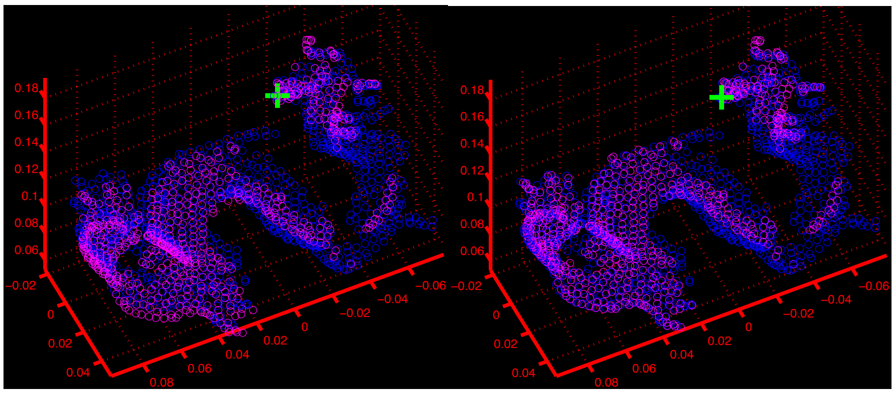
(d) ICP refinement of the result in (c).

FIGURE 3.5: Sample result of the interactive 3D registration system on underground mine scans.



(a) Point **p** selected by the user in the first point cloud.

(b) Point **q** (green cross) selected by the user in the second point cloud and the optimised rotational alignment.



(c) A better point **q** and the rotational alignment.

(d) ICP refinement of the result in (c).

FIGURE 3.6: Sample result of the interactive 3D registration system on the Stanford Dragon (views 96° and 120°).

where $\lfloor \cdot \rceil$ equals 1 if the condition $\cdot$ is true and 0 otherwise, and ϵ is the error threshold. Intuitively, $Q(\mathbf{R})$ counts the number of points in $\{\mathbf{R}\mathbf{x}_i\}_{i=1}^{M}$ that are matched (within distance ϵ) to a point in $\mathcal{Y}$.

Optimising over the space of 3D rotations $\mathbf{R}$ still presents a major problem, as observed in other works. For example, the box-and-ball algorithm presented by Li and Hartley requires more than 150 seconds to rotationally align two point clouds of size of just 100 points each [41]. Moreover, they considered an easier problem with no outliers, i.e., each point in $\mathcal{X}$ must have a matching point in $\mathcal{Y}$. Rotation search has also been investigated for other applications, e.g., for estimating camera poses [34, 68, 7, 6]. Most of these methods are not amenable to real-time performance.

As mentioned earlier, the presented methodology relies on the level compensator existing on many LIDAR devices to remove two angular DoF from $\mathbf{R}$, leaving only the azimuth

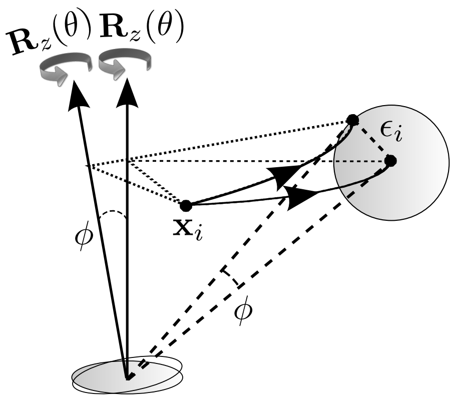


FIGURE 3.7: Plumb-line uncertainty compensation. If the angular deviation of the z-axis is within ϕ, then all possible variations of $\mathbf{R}_z(\theta)\mathbf{x}_i$ due to the uncertainty lie within a ball of radius ϵ_i, where ϵ_i can be obtained based on the cosine rule.

θ unknown. The criterion (3.1) now takes the form

$$Q(\theta) = \sum_{i=1}^{M} \max_{j=1\ldots N} \lfloor \|\mathbf{R}_z(\theta)\mathbf{x}_i - \mathbf{y}_j\| \leq \epsilon \rfloor, \tag{3.2}$$

where $\mathbf{R}_z(\theta)$ is a rotation matrix of θ radians about the z-axis. It is assumed in (3.2) that each $\mathcal{X}$ and $\mathcal{Y}$ is levelled with respect to the ground plane. Secondly, given a correct correspondence $\mathbf{p} \leftrightarrow \mathbf{q}$, it is unnecessary to try to rotationally align all the points, since the point clouds only partially overlap. Thus, it is excluded from (3.2) points in $\mathcal{X}$ and $\mathcal{Y}$ whose norm is greater than a threshold ϵ_n. While this reduces the size of the problem, on dense point clouds such as those in Figure 3.2, the size of the remaining point sets can still be large.

To account for potential inaccuracies in level compensation, an extra tolerance

$$\epsilon_i := \sqrt{2\|\mathbf{x}_i\|^2 - 2\|\mathbf{x}_i\|^2 \cos\phi} \tag{3.3}$$

is optionally added to ϵ for each $\mathbf{x}_i$, where ϕ is the angular uncertainty of the plumb-line (the z-axis); see Figure 3.7. To illustrate the resulting $Q(\theta)$, Figure 3.8 plots the function for two of the point clouds from Figure 3.2.

It is clear that $Q(\theta)$ has multiple local maxima, and a simple strategy to maximise the function by sampling at discrete intervals is unlikely to be successful and efficient. Recall that each evaluation of (3.2) requires checking whether each $\mathbf{R}_z(\theta)\mathbf{x}_i$ matches a point in $\mathcal{Y}$, which can be an expensive nearest neighbour problem if $\mathcal{X}$ and $\mathcal{Y}$ are large. Moreover, as exemplified in Figure 3.8, many of the evaluations will be wasted on totally unpromising values of θ. The existence of multiple local maxima in (3.2) also rules out the usage of iterative local improvement methods to find the best θ, since these can only converge to a local maximum.

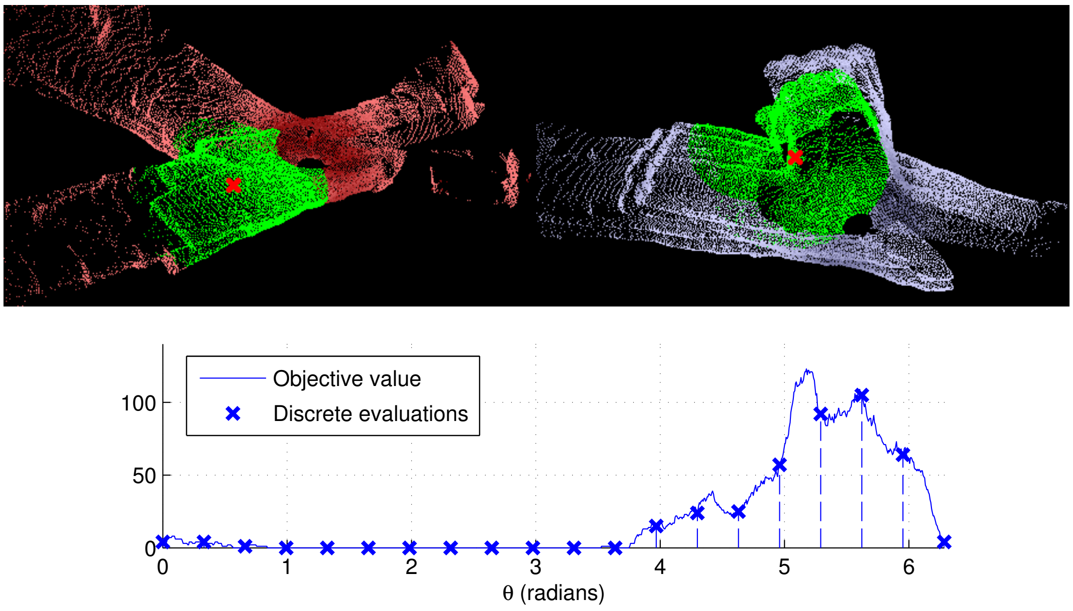


FIGURE 3.8: Plot of objective function for point cloud rotational registration. The top figures show $\mathcal{X}, \mathcal{Y}$ with $\mathbf{p}, \mathbf{q}$ indicated by red crosses. Only the points (coloured green) with distance $\leq \epsilon_n$ to $\mathbf{p}, \mathbf{q}$ are used. The objective function (3.2) is plotted at the bottom, where the sampling of (3.2) at 20 discrete positions is also shown.

3.5 BnB rotation search

The 1D rotation search problem (3.2) can be globally solved by using BnB (see Section 2.5). The method recursively explores and prunes the search space $[0, 2\pi]$ by maintaining a list of intervals. Given an interval, it is tested whether a better rotation than the best found so far exists in the interval. If the test fails then the whole interval is discarded; else the interval is divided into smaller subintervals. The process is repeated until the global maximum is found. Algorithm 3.1 summarises the procedure.

A key component of Algorithm 3.1 is the bounding function $\hat{Q}(\mathbb{T})$, which gives the upper bound of $Q(\theta)$ in the interval $\mathbb{T} \subseteq [0, 2\pi]$:

$$\hat{Q}(\mathbb{T}) \geq \max_{\theta \in \mathbb{T}} Q(\theta). \tag{3.4}$$

As shown in Algorithm 3.1, $\hat{Q}(\mathbb{T})$ plays the crucial role of pruning unpromising search intervals. Following the original work of Breuel [11] on geometric matching, the rotational bounding function can be defined as

$$\hat{Q}_{\text{gm}}(\mathbb{T}) = \sum_{i=1}^{M} \max_{j=1\dots N} \lfloor \|\mathbf{R}_z(\theta_c)\mathbf{x}_i - \mathbf{y}_j\| \leq \epsilon + \delta_i \rfloor, \tag{3.5}$$

Algorithm 3.1 BnB rotation search to solve (3.2).

```
Require: Point sets $\mathcal{X}$ and $\mathcal{Y}$, threshold $\epsilon$.
 1: Set $Q_{\max} = 0$, $\mathbb{T}_0 = [0, 2\pi]$, and $list = (\mathbb{T}_0, \hat{Q}(\mathbb{T}_0))$.
 2: while $list$ is not empty do
 3:   Remove from $list$ the interval $\mathbb{T}$ with the highest $\hat{Q}(\mathbb{T})$.
 4:   if $\hat{Q}(\mathbb{T}) > Q_{\max}$ then
 5:     $\theta_c \leftarrow$ centre of $\mathbb{T}$.
 6:     if $Q(\theta_c) = \hat{Q}(\mathbb{T})$ then
 7:       $\theta^* \leftarrow \theta_c$ and terminate algorithm.
 8:     else if $Q(\theta_c) > Q_{\max}$ then
 9:       $Q_{\max} \leftarrow Q(\theta_c)$ and $\theta^* \leftarrow \theta_c$.
10:     end if
11:     $\mathbb{T}_l \leftarrow [\min(\mathbb{T}), \theta_c]$ and $\mathbb{T}_r \leftarrow [\theta_c, \max(\mathbb{T})]$.
12:     $list \leftarrow list \cup (\mathbb{T}_l, \hat{Q}(\mathbb{T}_l)) \cup (\mathbb{T}_r, \hat{Q}(\mathbb{T}_r))$.
13:   end if
14: end while
```

where θ_c is the centre value of $\mathbb{T}$, and

$$\delta_i := \sqrt{2\|\mathbf{x}_i\|^2 - 2\|\mathbf{x}_i\|^2 \cos(\max(\mathbb{T}) - \theta_c)}, \tag{3.6}$$

is an additional tolerance that arose due to the actions of all possible rotations in $\mathbb{T}$:

$$\delta_i \geq \max_{\theta \in \mathbb{T}} \|\mathbf{R}_z(\theta)\mathbf{x}_i - \mathbf{x}_i\|. \tag{3.7}$$

In order for (3.5) to satisfy condition (3.4), it is necessary to show that any match in $\max_{\theta \in \mathbb{T}} Q(\theta)$ will be accounted in (3.5). The terms of the sum in (3.2) can be bounded by using the triangle inequality:

$$\begin{aligned} \|\mathbf{R}_z(\theta)\mathbf{x}_i - \mathbf{y}_j\| &\leq \|\mathbf{R}_z(\theta)\mathbf{x}_i - \mathbf{R}_z(\theta_c)\mathbf{x}_i\| + \|\mathbf{R}_z(\theta_c)\mathbf{x}_i - \mathbf{y}_j\| \\ &\leq \|\mathbf{R}_z(\theta_c)\mathbf{x}_i - \mathbf{y}_j\| + \delta_i. \end{aligned} \tag{3.8}$$

This result shows that for any θ in $\mathbb{T}$, a match $(\mathbf{R}_z(\theta)\mathbf{x}_i, \mathbf{y}_j)$ up to ϵ is accounted by the match $(\mathbf{R}_z(\theta_c)\mathbf{x}_i, \mathbf{y}_j)$ up to ϵ plus the extra tolerance δ_i, where θ_c is a fixed angle (the centre of value of $\mathbb{T}$). See [11] for more details.

It will be shown experimentally in Section 3.6 that applying (3.2) and (3.5) in Algorithm 3.1 does not produce real-time performance.

3.5.1 A novel rotational bounding function

Clearly, the runtime of Algorithm 3.1 depends on the total number of angular intervals that are tested before the optimal θ is found. A tighter bounding function will reduce

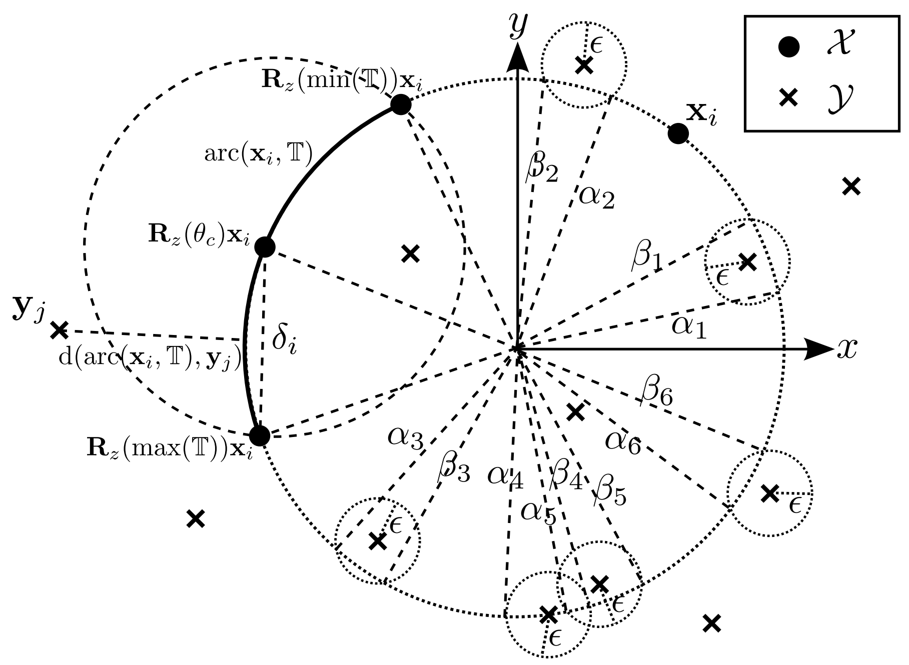

y
x
● $\mathcal{X}$
× $\mathcal{Y}$
$\mathbf{x}_i$
$\mathbf{R}_z(\min(\mathbb{T}))\mathbf{x}_i$
$\text{arc}(\mathbf{x}_i, \mathbb{T})$
$\mathbf{R}_z(\theta_c)\mathbf{x}_i$
$\mathbf{y}_j$
$\text{d}(\text{arc}(\mathbf{x}_i, \mathbb{T}), \mathbf{y}_j)$
δ_i
$\mathbf{R}_z(\max(\mathbb{T}))\mathbf{x}_i$
β_1 α_1 β_2 α_2 α_3 β_3 α_4 β_4 α_5 β_5 α_6 β_6
ϵ

FIGURE 3.9: Top view illustration of the rotation search problem. For simplicity, only a point $\mathbf{x}_i$ from $\mathcal{X}$ is shown. Under all possible rotation angles in a range $\mathbb{T}$, a direct application (3.5) of Breuel's bounding function conservatively assumes $\mathbf{x}_i$ may lie anywhere within a δ_i-sphere centred at $\mathbf{R}_z(\theta_c)\mathbf{x}_i$. In reality, $\mathbf{x}_i$ can only lie on an arc. The bounding function (3.9) exploits this knowledge. Further, if $\mathbf{y} \in \mathcal{Y}$ can possibly match with $\mathbf{x}_i$, then an ϵ-ball centred at $\mathbf{y}$ intersects with the circular trajectory of $\mathbf{x}_i$ at an arc. Section 3.5.2 describes a method that indexes the set of possible matches with $\mathbf{x}_i$ in a simple array for rapid evaluations of (3.2) and (3.9).

the number of intervals that needs to be tested, since it can conduct more aggressive pruning. The core insight is that a direct application in (3.5) of Breuel's bounding function ignores the *limited range* of rotational transforms, i.e., a point can only lie in a circular arc under all possible $\mathbf{R}_z(\theta)$ for $\theta \in \mathbb{T}$. Thus, it is unnecessary to expect that $\mathbf{x}_i$ may lie anywhere within the δ_i-ball centred at $\mathbf{R}_z(\theta_c)\mathbf{x}_i$, as assumed in (3.5). See an illustration of this in Figure 3.9.

This observation is crystallised by the following proposed bounding function

$$\hat{Q}_{\text{arc}}(\mathbb{T}) = \sum_{i=1}^{M} \max_{j=1\dots N} \lfloor d(\text{arc}(\mathbf{x}_i, \mathbb{T}), \mathbf{y}_j) \leq \epsilon \rfloor, \tag{3.9}$$

where $\text{arc}(\mathbf{x}_i, \mathbb{T})$ is defined as the circular arc formed when $\mathbf{x}_i$ is rotated about the z-axis by all possible angles in $\mathbb{T}$,

$$\text{arc}(\mathbf{x}_i, \mathbb{T}) = \{ \ \mathbf{R}_z(\theta)\mathbf{x}_i \mid \theta \in \mathbb{T} \ \}, \tag{3.10}$$

while d(arc($\mathbf{x}_i, \mathbb{T}), \mathbf{y}_j$) is the shortest distance of $\mathbf{y}_j$ to the arc,

$$d(\text{arc}(\mathbf{x}_i, \mathbb{T}), \mathbf{y}_j) = \min_{\mathbf{x} \in \text{arc}(\mathbf{x}_i, \mathbb{T})} \|\mathbf{x} - \mathbf{y}_j\|. \tag{3.11}$$

Figure 3.9 illustrates this idea. Next, it is proven that $\hat{Q}_{\text{arc}}(\mathbb{T})$ is a valid bounding function that is also tighter than $\hat{Q}_{gm}(\mathbb{T})$.

Lemma 3.1. *The condition*

$$\hat{Q}_{arc}(\mathbb{T}) \geq \max_{\theta \in \mathbb{T}} Q(\theta) \tag{3.12}$$

is satisfied for all intervals $\mathbb{T}$ *within* $[0, 2\pi]$.

Proof. To prove the lemma, it is sufficient to show that, if it is possible for a pair $\mathbf{x}_i$ and $\mathbf{y}_j$ to be matched under angle $\theta \in \mathbb{T}$, then the same pair of points must contribute 1 to the function $\hat{Q}_{\text{arc}}(\mathbb{T})$. This can be trivially demonstrated, since if the condition

$$\|\mathbf{R}_z(\theta)\mathbf{x}_i - \mathbf{y}_j\| \leq \epsilon \tag{3.13}$$

is true, i.e., $\mathbf{x}_i$ and $\mathbf{y}_j$ are matched under a $\theta \in \mathbb{T}$, then

$$d(\text{arc}(\mathbf{x}_i, \mathbb{T}), \mathbf{y}_j) \leq \epsilon \tag{3.14}$$

must be true, since it is known that there exists an $\mathbf{x} \in \text{arc}(\mathbf{x}_i, \mathbb{T})$ such that $\|\mathbf{x} - \mathbf{y}_j\| \leq \epsilon$ is satisfied, i.e., set $\mathbf{x} = \mathbf{R}_z(\theta)\mathbf{x}_i$. □

Another important condition for a valid bounding function is that $\hat{Q}(\mathbb{T})$ approaches $Q(\theta)$ as the volume of $\mathbb{T}$ approaches zero [38]. This can be easily seen in $\hat{Q}_{\text{arc}}(\mathbb{T})$, since as $\mathbb{T}$ collapses to a single point θ_c (volume of $\mathbb{T}$ becomes 0), then $d(\text{arc}(\mathbf{x}_i, \theta_c), \mathbf{y}_j) = \|\mathbf{R}_z(\theta_c)\mathbf{x}_i - \mathbf{y}_j\| \implies \hat{Q}_{\text{arc}}(\theta_c) = Q(\theta_c)$.

Lemma 3.2. *The condition*

$$\hat{Q}_{gm}(\mathbb{T}) \geq \hat{Q}_{arc}(\mathbb{T}) \tag{3.15}$$

is satisfied for all intervals $\mathbb{T}$ *within* $[0, 2\pi]$*, i.e.,* $\hat{Q}_{arc}(\mathbb{T})$ *is a tighter bounding function than* $\hat{Q}_{gm}(\mathbb{T})$.

Proof. Showing that $\hat{Q}_{\text{gm}}(\mathbb{T})$ at least equals $\hat{Q}_{\text{arc}}(\mathbb{T})$ can be done by simply appealing to be fact that arc($\mathbf{x}_i, \mathbb{T}$) lies in δ_i-sphere with centre $\mathbf{R}_z(\theta_c)\mathbf{x}_i$. Then, for any $\mathbf{y}_j$ such that the point-to-arc distance $d(\text{arc}(\mathbf{x}_i, \mathbb{T}), \mathbf{y}_j) \leq \epsilon$, the condition $\|\mathbf{R}_z(\theta_c)\mathbf{x}_i - \mathbf{y}_j\| \leq \epsilon + \delta_i$ is also satisfied.

To show that $\hat{Q}_{\text{gm}}(\mathbb{T})$ can be greater than $\hat{Q}_{\text{arc}}$, it is sufficient to show that there exist hypothetical points $\mathbf{x}_i$ and $\mathbf{y}_j$ that contribute 1 to $\hat{Q}_{\text{gm}}(\mathbb{T})$, but may not contribute 1 to $\hat{Q}_{\text{arc}}$. An example is

$$\mathbf{x}_i \quad \text{and} \quad \mathbf{y}_j = \mathbf{R}_z(\theta_c)\mathbf{x}_i \left(1 + \frac{\epsilon + \delta_i}{\|\mathbf{x}_i\|}\right) \tag{3.16}$$

which clearly satisfy $\|\mathbf{R}_z(\theta_c)\mathbf{x}_i - \mathbf{y}_j\| \leq \epsilon + \delta_i$. However, the point-to-arc distance for this pair is

$$d(\text{arc}(\mathbf{x}_i, \mathbb{T}), \mathbf{y}_j) = \epsilon + \delta_i \tag{3.17}$$

which violates the required condition $d(\text{arc}(\mathbf{x}_i, \mathbb{T}), \mathbf{y}_j) \leq \epsilon$. Of course, this assumes that $\mathbb{T}$ is not a point θ_c, else $\delta_i = 0$ and all $\hat{Q}_{\text{gm}}(\theta_c) = \hat{Q}_{\text{arc}}(\theta_c) = Q(\theta_c)$ by design. □

3.5.2 Efficient evaluations for real-time search

The efficiency of evaluating the objective and bounding functions is also crucial to the speed of Algorithm 3.1, since these functions are called repeatedly. Kd-trees are the workhorse in Breuel's method [11] for speeding up the function evaluations. As Section 3.6 will show, this is still insufficient for real-time performance.

Here, this section proposes a much faster method for evaluating (3.2) and (3.9). Based on the same insight in Section 3.5.1, as $\mathbf{x}_i$ is subjected to all allowable rotations $\mathbf{R}_z(\theta)$, only a subset $\mathcal{Y}_{\mathbf{x}_i}$ of $\mathcal{Y}$ can possibly match with $\mathbf{x}_i$, since $\mathbf{R}_z(\theta)\mathbf{x}_i$ is limited to lie on a circular arc. Define $circ_{\mathbf{x}_i}$ as the trajectory resulting from rotating $\mathbf{x}_i$ about the z-axis by 2π radians. An ϵ-ball centred at a point $\mathbf{y} \in \mathcal{Y}_{\mathbf{x}_i}$ intersects with $circ_{\mathbf{x}_i}$ at a circular arc; see Figure 3.9. Intuitively, solving the query

$$\max_{j=1\ldots N} \lfloor \|\mathbf{R}_z(\theta)\mathbf{x}_i - \mathbf{y}_j\| \leq \epsilon \rfloor \tag{3.18}$$

amounts to testing if $\mathbf{R}_z(\theta)\mathbf{x}_i$ lies within any of the circular arcs due to the intersections between ϵ-balls centred on $\mathcal{Y}_{\mathbf{x}_i}$ and $circ_{\mathbf{x}_i}$. Note that this test does not involve nearest neighbour searches.

To implement the above idea, let $[\alpha, \beta]$ be the angular interval subtended by the intersection between an ϵ-ball and $circ_{\mathbf{x}_i}$; see Figure 3.9. The intervals relating to all $\mathbf{y} \in \mathcal{Y}_{\mathbf{x}_i}$

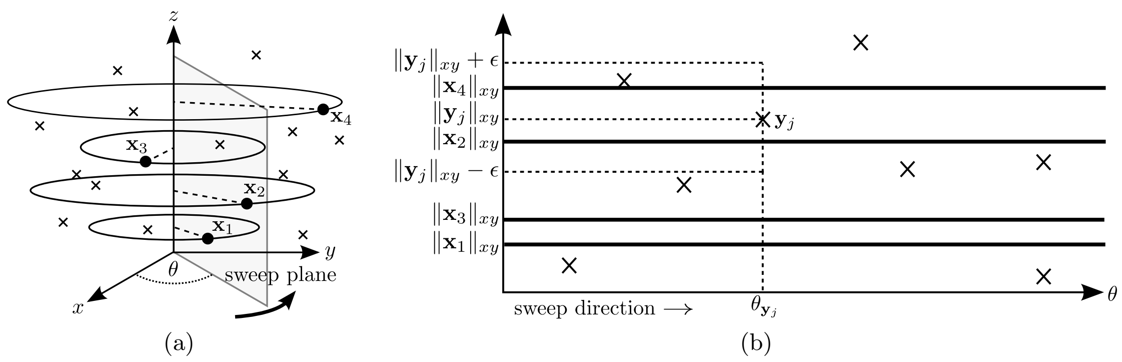


FIGURE 3.10: (a) An illustration of the modified sweep plane algorithm. The sweep plane is rotated about the z-axis for 2π radians. (b) This is the top view of (a) which is unrolled to lie flat on the plane. In this example, "*status*" in Algorithm 3.2 has the content $[\|\mathbf{x}_1\|_{xy}, \|\mathbf{x}_3\|_{xy}, \|\mathbf{x}_2\|_{xy}], \|\mathbf{x}_4\|_{xy}]$. Like in the classical sweep plane algorithm, the proposed algorithm visits a finite set of "events", corresponding to the points $\mathcal{Y}$, in the order of their azimuth. In this example, $\mathbf{y}_j$ has the potential to match $\mathbf{x}_2$ and $\mathbf{x}_4$.

are stored in the array

$$int_{\mathbf{x}_i} = [\alpha_1, \beta_1, \alpha_2, \beta_2, \dots] \tag{3.19}$$

which is maintained to be *sorted*. If an ϵ-ball intersects $circ_{\mathbf{x}_i}$ at one point, then α equals β (there may exist repeated values in $int_{\mathbf{x}_i}$). Also, overlapping intervals in $int_{\mathbf{x}_i}$ are merged into a single interval; for example, in Figure 3.9 $[\alpha_4, \beta_4]$ and $[\alpha_5, \beta_5]$ will be merged. An efficient algorithm to compute $int_{\mathbf{x}_i}$ will be given later. Computing (3.18) amounts to simply finding the position of the angle $\theta + \theta_{\mathbf{x}_i}$ in $int_{\mathbf{x}_i}$, where the offset $\theta_{\mathbf{x}_i}$ is the azimuth of $\mathbf{x}_i$

$$\theta_{\mathbf{x}_i} = \arcsin\left(\frac{\mathbf{x}_i(2)}{\sqrt{\mathbf{x}_i(1)^2 + \mathbf{x}_i(2)^2}}\right) \in [0, 2\pi]. \tag{3.20}$$

If $\theta + \theta_{\mathbf{x}_i}$ lies within any neighbouring $[\alpha, \beta]$, then the solution to (3.18) is 1; else, it is 0. A similar idea can be used for the range query

$$\max_{j=1\dots N} \lfloor d(\text{arc}(\mathbf{x}_i, \mathbb{T}), \mathbf{y}_j) \le \epsilon \rfloor \tag{3.21}$$

contained in (3.9), where $\mathbb{T} = [\theta_1, \theta_2]$. If $\theta_1 + \theta_{\mathbf{x}_i}$ and $\theta_2 + \theta_{\mathbf{x}_i}$ both lie in the *same* position in $int_{\mathbf{x}_i}$, and that position is *not* within any neighbouring $[\alpha, \beta]$, then the solution to (3.21) is 0; else it is 1. Thus, (3.2) and (3.9) are reduced to a series of sorted array insertion operations, which are extremely simple and efficient.

The following is a discussion on how to build the arrays $\{int_{\mathbf{x}_i}\}_{i=1}^M$ given point clouds $\mathcal{X}$ and $\mathcal{Y}$. Since $\mathcal{X}$ and $\mathcal{Y}$ change rapidly in the presented real-time interactive system, it is vital to compute $\{int_{\mathbf{x}_i}\}_{i=1}^M$ very quickly. The proposed solution is inspired by the classical sweep plane algorithm [55, Section 7.7]. A sweep plane rotates about the z-axis, and each time the sweep plane "hits" a point $\mathbf{y}_j \in \mathcal{Y}$, the angular intervals corresponding to $\mathbf{y}_j$ are inserted into the relevant arrays; Figure 3.10 illustrates the idea, and Algorithm 3.2 summarises the method. Note that the subsets $\{\mathcal{Y}_{\mathbf{x}_i}\}_{i=1}^{M}$ are implicitly contained in $\{int_{\mathbf{x}_i}\}_{i=1}^{M}$, thus they are not output by Algorithm 3.2.

Similar to the original sweep plane algorithm, the key to the efficiency of Algorithm 3.2 is the maintenance of a *status* array, which allows to avoid exhaustively checking all $M \times N$ potential matches between $\mathcal{X}$ and $\mathcal{Y}$. Here, *status* contains the sorted norms $\{\|\mathbf{x}_i\|_{xy}\}_{i=1}^{M}$, where $\|\mathbf{x}_i\|_{xy}$ is the norm of $\mathbf{x}_i$ on the xy-plane

$$\|\mathbf{x}_i\|_{xy} = \sqrt{\mathbf{x}_i(1)^2 + \mathbf{x}_i(2)^2}. \tag{3.22}$$

Essentially $\|\mathbf{x}_i\|_{xy}$ is the radius of $circ_{\mathbf{x}_i}$, and this quantity is "perpendicular" to the sweep direction; see Figure 3.10b. In addition, analogous to the sorting of *intersection events* in the original algorithm, Algorithm 3.2 visits the points in $\mathcal{Y}$ in the order of their azimuth angle (3.20). This ensures that in Step 13 the intervals are inserted in sorted order into $int_{\mathbf{x}_i}$. Merging potentially overlapping intervals on-the-fly can be done efficiently by maintaining a stack along with each sorted interval array $int_{\mathbf{x}_i}$; since this is a very common problem, the details will not be discussed here.

Using similar analysis from computational geometry [55, Section 7.7], for each point in $\mathcal{Y}$ it is needed to query the sorted array *status* of length M twice. The complexity of Algorithm 3.2 is $\mathcal{O}(N \log M + k)$, where k is the total number of possible matches between $\mathcal{X}$ and $\mathcal{Y}$. Results in Section 3.6 demonstrate that Algorithm 3.2 consumes very little overhead. In turn, this enables very efficient evaluations of functions (3.2) and (3.9), leading to real-time optimal rotation search.

3.6 Results

The performance of the proposed rotation search algorithm is tested and benchmarked by evaluating the following methods:

- **bnb-M-circ**: the proposed algorithm.
- **bnb-1-tree**: Algorithm 3.1 using Breuel's original bounding function $\hat{Q}_{\text{gm}}$ (3.5). Given $\mathcal{X}$ and $\mathcal{Y}$, a kd-tree is used to index $\mathcal{Y}$ to speed-up the evaluation of (3.2) and (3.5).
- **bnb-M-tree**: same as **bnb-1-tree** but to further speed up (3.2) and (3.5), for each $\mathbf{x}_i \in \mathcal{X}$, a kd-tree indexes the set of possible matches $\mathcal{Y}_{\mathbf{x}_i}$. This yields M

Algorithm 3.2 Modified sweep plane algorithm.

```
Require: Point sets $\mathcal{X}$ and $\mathcal{Y}$, threshold $\epsilon$.
1: Set $int_{\mathbf{x}_i} = \emptyset$ for all $i = 1, \ldots, M$.
2: Reorder the points in $\mathcal{Y}$ based on their azimuth (3.20).
3: $status \leftarrow$ sort $\{\|\mathbf{x}_i\|_{xy}\}_{i=1}^{M}$ for all $\mathbf{x}_i \in \mathcal{X}$; see (3.22).
4: Reorder the points in $\mathcal{X}$ based on their position in $status$.
5: for $j = 1, \ldots, N$ do
6:   $i_l \leftarrow$ smallest $i$ such that $status(i) \geq \|\mathbf{y}_j\|_{xy} - \epsilon$.
7:   $i_u \leftarrow$ largest $i$ such that $status(i) \leq \|\mathbf{y}_j\|_{xy} + \epsilon$.
8:   if both $i_l$ and $i_u$ are not null then
9:     for $i = i_l, \ldots, i_u$ do
10:      if $(\|\mathbf{x}_i\|_{xy} - \|\mathbf{y}_j\|_{xy})^2 \leq \epsilon^2 - (\mathbf{x}_i(3) - \mathbf{y}_j(3))^2$ then
11:        $arc \leftarrow$ intersect $circ_{\mathbf{x}_i}$ and $\epsilon$-ball centred at $\mathbf{y}_j$.
12:        $[\alpha, \beta] \leftarrow$ angular interval subtended by $arc$.
13:        Insert (or, if required, merge) $[\alpha, \beta]$ into $int_{\mathbf{x}_i}$.
14:      end if
15:    end for
16:  end if
17: end for
18: return Set of arrays $\{int_{\mathbf{x}_i}\}_{i=1}^{M}$.
```

kd-trees in total. To check if $\mathbf{R}_z(\theta)\mathbf{x}_i$ matches a point in $\mathcal{Y}$, only the kd-tree associated with $\mathbf{x}_i$ needs to be queried.

- **icp-rotate**: a special case of ICP whereby given $\mathcal{X}$ and $\mathcal{Y}$, the optimised transformation is restricted to $\mathbf{R}_z(\theta)$ (see Algorithm 2.1).

Theoretically, since in **bnb-M-tree** only the possible matches $\mathcal{Y}_{\mathbf{x}_i}$ of $\mathbf{x}_i$ are searched for a match, the bound $\hat{Q}_{\text{gm}}$ is equal to $\hat{Q}_{\text{arc}}$, i.e., **bnb-M-circ** and **bnb-M-tree** will take the same number of iterations in Algorithm 3.1. The fundamental difference is thus the efficiency required to evaluate the objective and bounding functions. The performance metrics used to compare the above methods are:

- Total runtime, which includes all data structure preparation time (kd-trees, sorted arrays, etc.) and searching for θ until convergence (all methods compared can provably converge). All the methods were implemented in C and run on a machine with an Intel Core i7 3.40 GHz CPU.
- Number of matched points (3.2) at convergence. All the BnB methods will yield the same globally maximal value. For **icp-rotate**, the converged θ value is simply used to evaluate (3.2).

Further, to evaluate the metrics for **icp-rotate**, it was reported the average result for 10 different random initialisations.

To simulate the user-assisted search for matching regions, given two input point clouds, the sampling of local point sets $\mathcal{X}$ and $\mathcal{Y}$ is as follows:

1. On each point cloud, 3D keypoints are detected. While many methods are applicable, for convenience *ISS* [74] is used as implemented on Point Cloud Library (PCL) [61].

2. 100 keypoint matches are randomly sampled across the two point clouds. For each match, the local point cloud (within radius ϵ_n) around each keypoint are taken as $\mathcal{X}$ and $\mathcal{Y}$.

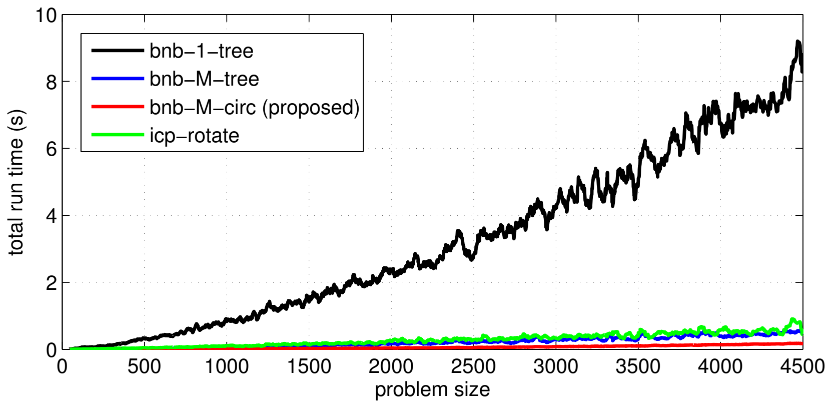
3. ISS also gives the surface normal at the keypoints. Each $\mathcal{X}$ and $\mathcal{Y}$ pair is oriented such that the normals are pointing up. The rotation search algorithm then optimises the rotation $\mathbf{R}_z(\theta)$ that best aligns $\mathcal{X}$ and $\mathcal{Y}$.

Note that the sampled keypoint matches here need not be genuine matches; the purpose is to examine the speed of various methods in rotation search, not their accuracy in matching keypoints. Also, ϵ_n is varied across ten different values for each keypoint match, thus yielding $\mathcal{X}$ and $\mathcal{Y}$ of different sizes. In total, for each pair of input point clouds, $100 \times 10 = 1000$ rotation search problems are obtained.

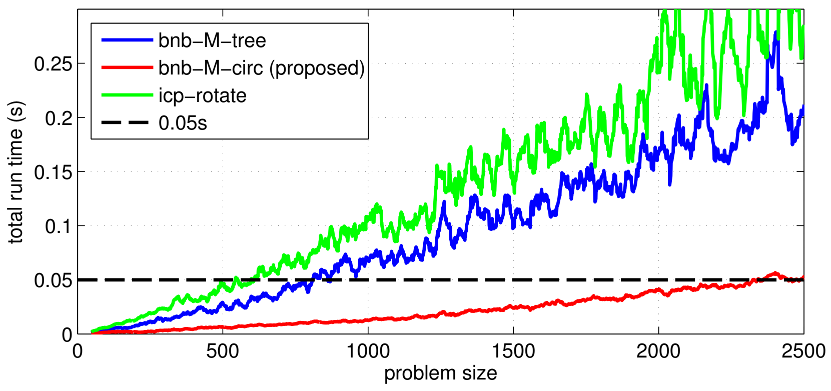
The experiment is conducted in the following pairs of point clouds from the underground mine data in Figure 3.2, namely Set 1 vs Set 2, Set 2 vs Set 3, and Set 1 vs Set 4 (the numbering is taken in the left-right order in Figure 3.2). As output by the LIDAR device, these point clouds are individually level with respect to the ground plane. For completeness, the range data from the Stanford 3D Scanning Repository [17] was also used, namely, *bunny*, *dragon* and *armadillo* (for each object, point clouds from two different views were chosen). The objects were rotated on a turntable, hence the data satisfies the assumption of being levelled; see Figure 3.6.

Figure 3.11 shows the overall result on all datasets, and plots both runtime and number of matches against problem size. To enable a consistent way of measuring problem size, *before* performing rotation search, for each $\mathcal{X}$ and $\mathcal{Y}$ it is ensured that $\mathcal{X}$ is smaller than $\mathcal{Y}$ by swapping the point sets if required. M is taken as the size of each rotation search problem. Doing this also ensures that the quality value is bounded by M, as defined in the objective function (3.2).

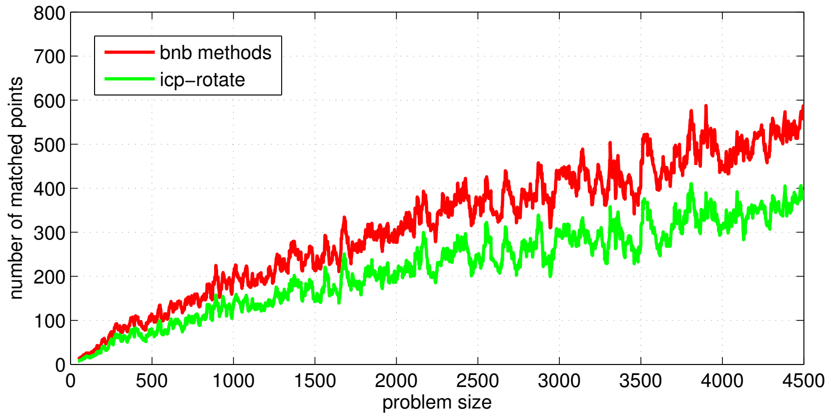
The runtime results in Figure 3.11a clearly show that **bnb-M-circ** has a much better computational complexity than **bnb-1-tree**. It is also evident that **bnb-1-tree** cannot provide real-time performance. Figure 3.11b gives a closer comparison between **bnb-M-circ**, **bnb-M-tree** and **icp-rotate**. The two methods **bnb-M-circ** and **bnb-M-tree**



(a) Total runtime versus problem size.



(b) Total runtime versus problem size (up to size 2500 only).



(c) Number of matched points versus problem size.

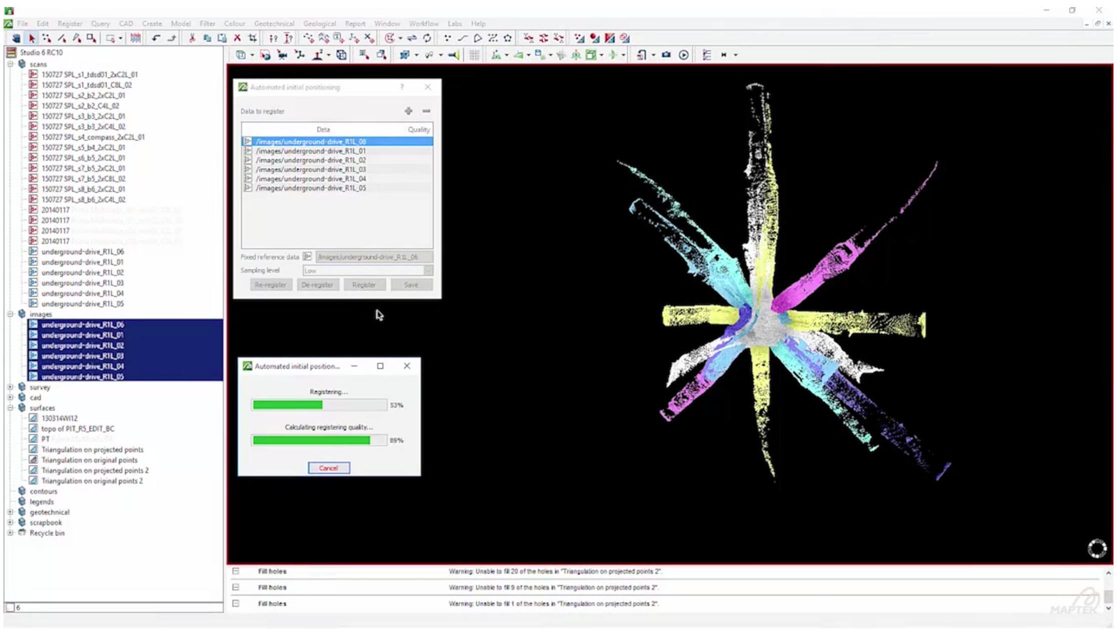
FIGURE 3.11: Comparing runtime and quality (number of matched points) of various rotation search methods.

show a clear divergence starting from small problem sizes. Taking 20 frames-per-second (0.05 seconds per rotation search) as real-time, **bnb-M-tree** can only handle up to 800 points, whereas **bnb-M-circ** is good up to 2500 points. This indicates the superior efficiency of the novel bound evaluation techniques in Section 3.5.2 (recall that since both methods theoretically evaluate the same bounds, their core difference is in the bound evaluation time). In problems involving high resolution LIDAR scans such as surveying the underground mines in Figure 3.2 (e.g., an individual scan can contain up to $500,000$ points), extremely fast rotation search is crucial for aligning the local point clouds $\mathcal{X}$ and $\mathcal{Y}$ with sizes up to the range of thousands. Somewhat surprisingly, the results show that **icp-rotate** is slower than **bnb-M-circ** (note that the plots in Figure 3.11 show the *average* time and quality for **icp-rotate** over 10 repetitions). This implies that the number of iterations required in **icp-rotate** is typically higher than the presented BnB method. In any case, as Figure 3.11c shows, the quality of **icp-rotate** is lower than BnB (note that all the BnB methods return the same globally optimal result), and its quality will vary more erratically without the averaging conducted here.

Other globally optimal methods. As surveyed in Section 3.2, there exist other methods that promise global optimality in point cloud registration. Most of them tackle full 3D rigid transforms, thus a direct comparison is only possible if significant changes are made to those methods. For an indication of how the other methods compare, Li and Hartley [41] used their method to optimise pure rotation in an experiment, where one of the *bunny* point clouds was downsampled and then rotated to yield two point sets $\mathcal{X}$ and $\mathcal{Y}$ (their method requires $\mathcal{X}$ and $\mathcal{Y}$ to have the same size and each point in $\mathcal{X}$ must have a matching point in $\mathcal{Y}$). For point clouds with 200 points their method requires more than 1000 seconds. A globally optimal rotation search algorithm was proposed by Bazin et al. [7], however, they require 3D point correspondences to be established between the point clouds $\mathcal{X}$ and $\mathcal{Y}$ to guide the search. Thus, they are tackling a Type 1 problem instead. Moreover, their method is only optimal with respect to the set of discrete correspondences between $\mathcal{X}$ and $\mathcal{Y}$, and not to all the available points.

3.7 Usage of Algorithm 3.1 in commercial software

The novel user-assisted registration system (Section 3.3) is part of the automated registration tool in Maptek I-Site Studio 6 [46]. Of vital importance to the system is the efficiency of Algorithm 3.1 for 1D rotation search. Watch the system in action in the demonstration video for automated registration in [46] or see Figure 3.12 for screenshots of the system.



Automated initial positioning

Data to register

Data	Quality
/images/underground-drive_R1L_06	
/images/underground-drive_R1L_01	
/images/underground-drive_R1L_02	
/images/underground-drive_R1L_03	
/images/underground-drive_R1L_04	
/images/underground-drive_R1L_05	

Fixed reference data: /images/underground-drive_R1L_06

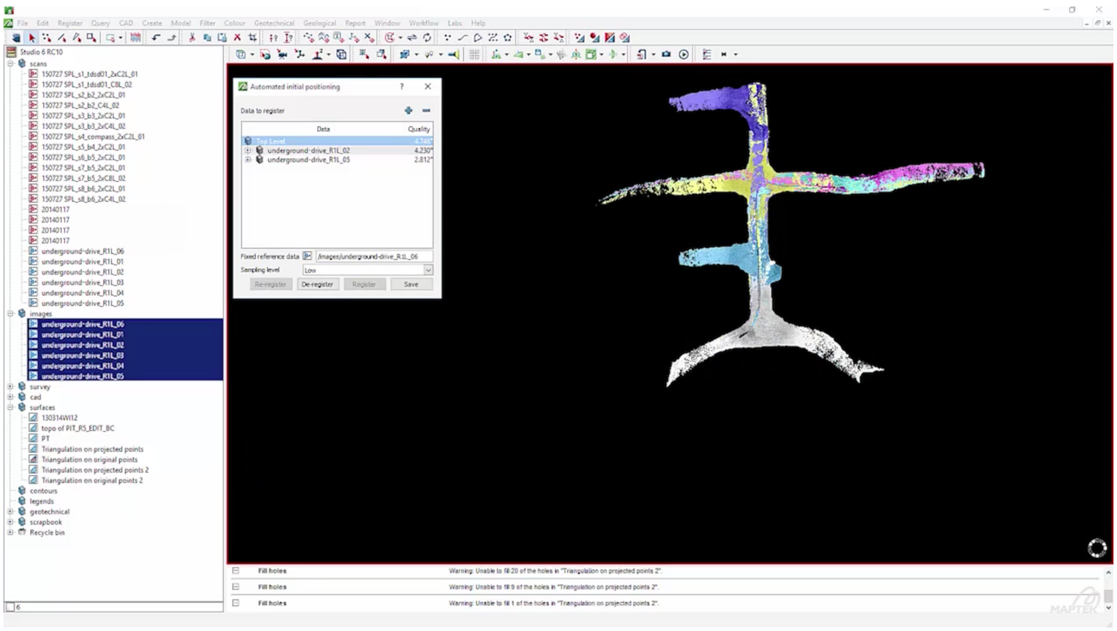
Sampling level: Low

Registering... 53%

Calculating registering quality... 89%

```
Fill holes    Warning: Unable to fill 20 of the holes in "Triangulation on projected points 2".
Fill holes    Warning: Unable to fill 9 of the holes in "Triangulation on projected points 2".
Fill holes    Warning: Unable to fill 1 of the holes in "Triangulation on projected points 2".
```

(a) Unregistered LIDAR scans.



Automated initial positioning

Data to register

Data	Quality
Top Level	4.745°
underground-drive_R1L_02	4.230°
underground-drive_R1L_05	2.812°

Fixed reference data: /images/underground-drive_R1L_06

Sampling level: Low

```
Fill holes    Warning: Unable to fill 20 of the holes in "Triangulation on projected points 2".
Fill holes    Warning: Unable to fill 9 of the holes in "Triangulation on projected points 2".
Fill holes    Warning: Unable to fill 1 of the holes in "Triangulation on projected points 2".
```

(b) Final registration.

Figure 3.12: Screenshots of the automated registration tool in Maptek I-Site Studio 6 [46]. (a) Unregistered point clouds. (b) Final registration.

3.8 Summary

This section presented a fast 1D rotation search algorithm for Type 2 problems (see Algorithm 3.1) in the context of a user-assisted point cloud registration system with real-time interaction; the diagram in Figure 3.3 summarises the system.

This system is useful for aligning multiple point clouds for large-scale 3D modelling and surveying. In particular, in settings where fully automatic registration may not be feasible due to large problem sizes or lack of information for initialisation, this system provides an efficient means for registering the point clouds with minimal user effort. At the core of this system is the novel presented rotation search algorithm that is able to globally solve rotation search in real-time. This algorithm is based on BnB and relies on a highly efficient bounding function. Experiments show that the system is orders of magnitude faster than previous approaches.

Chapter 4 extends the 1D rotation search algorithm to full 3D rotation search.

Chapter 4

Robust 3D Rotation Search

4.1 Introduction

Many existing point cloud registration systems still rely on the classical ICP method [9], which conducts an EM-like optimisation that alternates between point assignments and updates to the rigid transform parameters. While highly efficient, ICP requires careful initialisations since it is only locally convergent. A similar weakness afflicts the well-known SoftAssign method [29], which also performs alternating optimisation. In many applications, the required initialisation is not available or is too laborious to be acquired. Thus, there is the need to consider algorithms that are globally convergent.

This chapter presents a global 6 DoF point cloud registration algorithm that solves the GM criterion (2.30). The presented algorithm does not assume any constraints on the original alignment of the point clouds, unlike the user-assisted system presented in Chapter 3 which depend on the point clouds being horizontally level. The algorithm is based on nested BnB, whereby an outer BnB loop optimises the translation and the inner BnB loop conducts rotation search. The main contribution of this chapter is a rotation search kernel in such a nested design.

Since the rotation search algorithm is invoked repeatedly in the nested BnB algorithm, its efficiency has a large impact on the runtime of 6 DoF point cloud registration. Typically, optimising the rotational parameters is less efficient than the translational parameters, due to the special structure of SO(3). As discussed in Section 1.2, this is by no means a trivial problem, and significant efforts have been devoted purely to rotation search [34, 68, 64, 7]; see Section 2.4 for a review of rotation search algorithms.

The presented rotation search algorithm exploits the geometry of rotational transforms to derive a bounding function that is provably tighter than those proposed previously.

This allows pruning of unpromising rotations to be conducted more aggressively in BnB, thus speeding up convergence. Further, this chapter also proposes a fast algorithm to evaluate the proposed bounding function. Specifically, the algorithm precomputes all possible point matches using *stereographic projections* [49] and indexes them in *circular R-trees* [45]. This facilitates rapid bound evaluations and further increases the performance of BnB. The result is a rotation search algorithm that is robust, globally optimal *and* fast; the method can register up to 1000 points in 2 seconds.

This chapter is organised as follows: Section 4.2 surveys related works on point cloud registration. Section 4.3 describes the proposed rotation search algorithm and the novel bounding function for BnB. Section 4.4 describes the proposed method based on stereographic projections and R-trees for rapid bound evaluations. Section 4.5 shows how the rotation search kernel can be used in nested BnB to globally optimise 6 DoF rigid transforms. Section 4.6 provides experimental results and comparisons. Section 4.7 summarises this chapter.

4.2 Related work on 6 DoF point cloud registration

To facilitate discussion, the classification for rotation search problems presented in page 11 is extended to classify 6 DoF registration methods. Here, Type 1 and Type 2 problems, originally defined for rotation search, are interpreted as estimating a 6 DoF rigid transform on point correspondences, and on the "raw" data, respectively.

Significant effort has been devoted to 6 DoF point cloud registration. Available techniques include randomised heuristics [13, 1], feature detection and matching [27], and methods that construct alternative representations for the point sets such as mixture models [39, 48] or Fourier transforms [44].

6 DoF registration algorithms of Type 1 [28, 52, 3] are susceptible to the veracity of the given point matches. Since Type 2 registration algorithms [11, 41, 21, 73] use the original point clouds without prior matching, they are not exposed to the potential errors of "hard-coded" point matches. Such methods face a tougher problem, since they must also optimise the matching (either implicitly or explicitly) along with the rigid transform. This chapter focusses on problem Type 2.

Of closer relevance to this chapter is the class of methods that employ mathematical optimisation to deterministically find the best solution. One of the earliest globally optimal registration methods was proposed by Breuel [11] for GM, e.g., finding a previously seen configuration of 2D points in an input edge map. The method is based on BnB, which guarantees global optimality. While Breuel's original formulation is fast enough for optimising 2D rigid transforms with 3 DoF, a naive extension to estimate 3D rigid transforms with 6 DoF is unwieldy, as the volume of the search space is significantly increased by going from 2D to 3D.

Li and Hartley [41] formulated a "Lipschitzised" objective function that can be globally optimised using BnB. However, the method must assume one-to-one matching of the point clouds (i.e., no partial overlaps), which can be too restrictive for many practical applications. Further, the reported computational time is quite high.

More recently, Yang et al. [73] proposed a globally optimal method called Go-ICP, which does not assume one-to-one point correspondences. The algorithm consists of two nested BnB loops, where the outer loop optimises the rotation while the inner loop searches for the translation given candidate rotations. Unlike Go-ICP, the registration method presented in this chapter (Section 4.5), conducts rotation search in the inner loop. However, with respect to rotation BnB, stemming from the fact that the presented algorithm conducts GM [11] instead of LS [9], a much tighter bounding function can be constructed. Further, the bounding function can be rapidly evaluated with a method based on stereographic projections, R-trees and matchlists, all of which cannot be easily exploited under LS; see Section 4.4 for details. By virtue of a much faster rotation search kernel, the presented 6 DoF algorithm outperforms Go-ICP significantly.

A distinct class of methods conduct 3D registration as a graph matching problem, which involves pairwise matching constraints. Graph matching is well known to be fundamentally difficult. Spectral approximations have been proposed [40], but only very small point sets can be handled. Enqvist et al. [21] posed graph matching as a vertex cover problem, and a BnB algorithm was proposed. In practice, solving vertex cover is computationally exorbitant. Thus, only relatively small point sets with large overlaps have been tested.

4.3 Fast BnB rotation search

This section presents a fast BnB rotation search algorithm for problem Type 2.

4.3.1 Objective function and BnB algorithm

The GM criterion (see Section 2.4.5) is rewritten for the Euclidean norm. Let $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{M}$ and $\mathcal{Y} = \{\mathbf{y}_j\}_{j=1}^{N}$ be two 3D point clouds which are assumed to be potentially related by a 3D rotation. The GM criterion seeks the rotation $\mathbf{R} \in \mathrm{SO}(3)$ that maximises

$$Q(\mathbf{R}) = \sum_{i=1}^{M} \max_{j=1\ldots N} \lfloor \|\mathbf{R}\mathbf{x}_i - \mathbf{y}_j\| \leq \epsilon \rfloor, \tag{4.1}$$

where $\lfloor \cdot \rfloor$ is the indicator function which returns 1 if the condition $\cdot$ is true and 0 otherwise, and $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^3$. The criterion (4.1) is robust since two points are matched only if their distance is less than the inlier threshold ϵ.

It is instructive to compare (4.1) against the LS criterion in ICP, which minimises

$$E(\mathbf{R}) = \sum_{i=1}^{M} \min_{j=1\ldots N} \|\mathbf{R}\mathbf{x}_i - \mathbf{y}_j\|^2. \tag{4.2}$$

For each point $\mathbf{R}\mathbf{x}_i$, its nearest neighbour in $\mathcal{Y}$ must be found. Contrast this to (4.1) where as long as a sufficiently close point in $\mathcal{Y}$ exists, the distance of $\mathbf{R}\mathbf{x}_i$ to its nearest neighbour is irrelevant.

Algorithm 4.1 summarises the presented BnB algorithm for finding the globally optimal 3D rotation with respect to maximising (4.1). The basic idea is to recursively subdivide and prune the rotation space, until the global optimum is found; see Section 2.5 for a description of BnB. Here, rotations are represented using the axis-angle representation (see Section 2.2.2) in which all rotations are contained in a π-ball. The π-ball is initially enclosed with a cube $\mathbb{B}$ of side 2π, then it is successively decomposed into eight smaller cubes; see Section 2.5.5 for more details.

Crucially influencing the runtime of Algorithm 4.1 is the tightness of the bounding function $\hat{Q}$, which satisfies

$$\hat{Q}(\mathbb{B}) \geq \max_{\mathbf{r} \in \mathbb{B}} Q(\mathbf{R}_\mathbf{r}), \tag{4.3}$$

where $\mathbf{R}_\mathbf{r}$ is the matrix form of rotation $\mathbf{r}$. A tighter $\hat{Q}$ will prune more aggressively and yield fewer iterations. Equally important is the efficiency of evaluating Q and $\hat{Q}$, since they are called repeatedly. This chapter makes contributions in both aspects, as described in Sections 4.3.3 and 4.4.

4.3.2 Previous results

From Lemma 2.32, given two rotation vectors $\mathbf{u}$ and $\mathbf{v}$ in the π-ball

$$\angle(\mathbf{R}_\mathbf{u}\mathbf{x}, \mathbf{R}_\mathbf{v}\mathbf{x}) \leq \|\mathbf{u} - \mathbf{v}\|, \tag{4.4}$$

Algorithm 4.1 BnB algorithm to maximise (4.1).

```
Require: Point sets $\mathcal{X}$ and $\mathcal{Y}$, threshold $\epsilon$.
 1: Initialise priority queue $q$, $\mathbb{B} \leftarrow$ cube of side $2\pi$,
    $Q^* \leftarrow 0$, $\mathbf{R}^* \leftarrow \emptyset$.
 2: Insert $\mathbb{B}$ with priority $\hat{Q}(\mathbb{B})$ into $q$.
 3: while $q$ is not empty do
 4:   Obtain highest priority cube $\mathbb{B}$ from $q$.
 5:   If $\hat{Q}(\mathbb{B}) = Q^*$ then terminate.
 6:   $\mathbf{R}_\mathbf{c} \leftarrow$ centre rotation of $\mathbb{B}$.
 7:   If $Q(\mathbf{R}_\mathbf{c}) > Q^*$ then $Q^* \leftarrow Q(\mathbf{R}_\mathbf{c})$, $\mathbf{R}^* \leftarrow \mathbf{R}_\mathbf{c}$.
 8:   Subdivide $\mathbb{B}$ into eight cubes $\{\mathbb{B}_d\}_{d=1}^{8}$.
 9:   For each $\mathbb{B}_d$, if $\hat{Q}(\mathbb{B}_d) > Q^*$, insert $\mathbb{B}_d$ with priority $\hat{Q}(\mathbb{B}_d)$ into $q$.
10: end while
11: return Optimal rotation $\mathbf{R}^*$ with quality $Q^*$.
```

where $\mathbf{x}$ is a 3D point, and $\angle(\cdot,\cdot)$ gives the angular distance. Further, given a cube $\mathbb{B}$, let $\mathbf{p}$ and $\mathbf{q}$ be the points at two opposite corners of $\mathbb{B}$. Then,

$$\mathbf{c} := \frac{\mathbf{p}+\mathbf{q}}{2} \tag{4.5}$$

is the centre of $\mathbb{B}$ with rotation matrix $\mathbf{R}_\mathbf{c}$. For any rotation $\mathbf{u}$ situated in the box $\mathbb{B}$, it is obtained that

$$\begin{aligned}\angle(\mathbf{R}_\mathbf{c}\mathbf{x}, \mathbf{R}_\mathbf{u}\mathbf{x}) &\leq \max_{\mathbf{u}\in\mathbb{B}} \|\mathbf{c}-\mathbf{u}\| \\ &= \frac{\|\mathbf{p}-\mathbf{q}\|}{2} := \alpha_\mathbb{B}\end{aligned} \tag{4.6}$$

as a direct consequence of (4.4). It thus follows that

$$\|\mathbf{R}_\mathbf{c}\mathbf{x} - \mathbf{R}_\mathbf{u}\mathbf{x}\| \leq \delta, \tag{4.7}$$

where the bound δ is based on the cosine rule

$$\delta = \sqrt{2\|\mathbf{x}\|^2(1-\cos\alpha_\mathbb{B})}. \tag{4.8}$$

The result (4.7) immediately suggests the following bounding function for the objective function (4.1):

$$\hat{Q}_{br}(\mathbb{B}) = \sum_{i=1}^{M} \max_{j=1\ldots N} \lfloor \|\mathbf{R}_\mathbf{c}\mathbf{x}_i - \mathbf{y}_j\| \leq \epsilon + \delta_i \rfloor, \tag{4.9}$$

where δ_i is defined as (4.8) evaluated with $\mathbf{x}_i$. This bounding function was also originally proposed by Breuel; see [11] for proof that (4.9) is a valid bound for (4.1).

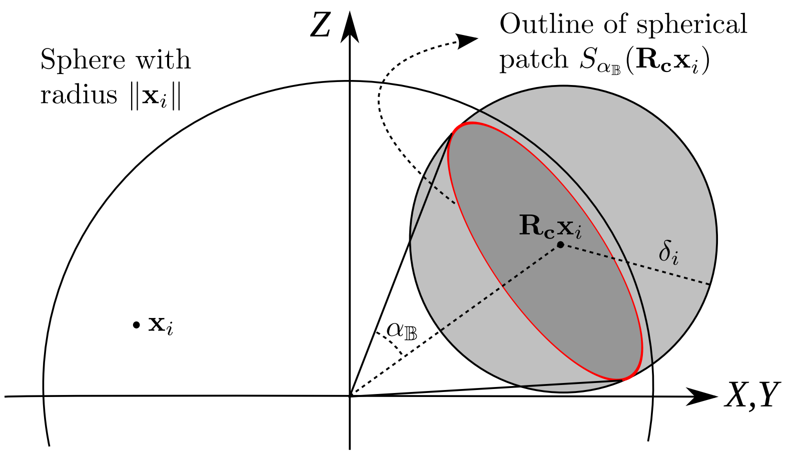

Z
Sphere with radius $\|\mathbf{x}_i\|$
Outline of spherical patch $S_{\alpha_{\mathbb{B}}}(\mathbf{R_c x}_i)$
$\mathbf{R_c x}_i$
δ_i
$\mathbf{x}_i$
$\alpha_{\mathbb{B}}$
X,Y

FIGURE 4.1: Under the action of all possible rotations in $\mathbb{B}$, $\mathbf{x}_i$ may lie only on a spherical patch centered at $\mathbf{R_c x}_i$. However the bounding function (4.9) assumes that $\mathbf{x}_i$ may lie in the δ_i-ball centred at $\mathbf{R_c x}_i$.

The bounding function (4.9) is unnecessarily conservative. Geometrically, the result (4.7) says that $\mathbf{R_u x}_i$ may lie anywhere within a ball of radius δ_i centred at $\mathbf{R_c x}_i$. Intuitively, this is inaccurate, since the actions of all possible rotations in $\mathbb{B}$ may only allow $\mathbf{x}_i$ to lie on a patch on the surface of the sphere with radius $\|\mathbf{x}_i\|$; see Figure 4.1. The proposed method in this chapter exploits this key insight.

4.3.3 Improving the tightness of the bound

Let $S_\theta(\mathbf{x})$ represent the *spherical patch* (see Figure 4.1) centred at $\mathbf{x}$ with angular radius θ, i.e.,

$$S_\theta(\mathbf{x}) = \{\mathbf{p} \in \mathbb{R}^3 \mid \|\mathbf{p}\| = \|\mathbf{x}\|,\ \angle(\mathbf{x},\mathbf{p}) \leq \theta\}. \tag{4.10}$$

$S_{2\pi}(\mathbf{x})$ is thus the sphere of radius $\|\mathbf{x}\|$ centred at the origin, and $S_\theta(\mathbf{x}) \subseteq S_{2\pi}(\mathbf{x})$. Further, the outline of $S_\theta(\mathbf{x})$ is a circle on the surface of $S_{2\pi}(\mathbf{x})$. Using the above notation, the result (4.6) can be reexpressed as

$$\mathbf{R_u x} \in S_{\alpha_{\mathbb{B}}}(\mathbf{R_c x}) \tag{4.11}$$

where $\mathbf{c}$, $\mathbf{u}$ and $\alpha_{\mathbb{B}}$ are as defined previously.

Let $l_\epsilon(\mathbf{y})$ denote the closed ball of radius ϵ centred at $\mathbf{y}$:

$$l_\epsilon(\mathbf{y}) = \{\mathbf{p} \mid \|\mathbf{p} - \mathbf{y}\| \leq \epsilon\}. \tag{4.12}$$

The objective function (4.1) can be rewritten as

$$Q(\mathbf{R}) = \sum_{i=1}^{M} \max_{j=1\ldots N} \lfloor \mathbf{R}\mathbf{x}_i \in l_\epsilon(\mathbf{y}_j) \rfloor . \tag{4.13}$$

From (4.11), since $\mathbf{x}_i$ can only lie in $S_{\alpha_\mathbb{B}}(\mathbf{R_c}\mathbf{x}_i)$ under all possible rotations in $\mathbb{B}$, determining if $\mathbf{x}_i$ can match with $\mathbf{y}_j$ under $\mathbb{B}$ amounts to checking if $S_{\alpha_\mathbb{B}}(\mathbf{R_c}\mathbf{x}_i)$ intersects with $l_\epsilon(\mathbf{y}_j)$. This leads to the upper bound

$$\hat{Q}_{sp}(\mathbb{B}) = \sum_{i=1}^{M} \max_{j=1\ldots N} \lfloor S_{\alpha_\mathbb{B}}(\mathbf{R_c}\mathbf{x}_i) \cap l_\epsilon(\mathbf{y}_j) \neq \emptyset \rfloor . \tag{4.14}$$

To qualify as a valid bounding function for BnB, $\hat{Q}_{sp}$ has to meet the following provided conditions.

Lemma 4.1. *For any cube* $\mathbb{B}$

$$\hat{Q}_{sp}(\mathbb{B}) \geq \max_{\mathbf{r}\in\mathbb{B}} Q(\mathbf{R_r}). \tag{4.15}$$

Also as $\mathbb{B}$ *collapses to a single point* $\mathbf{r}$*,*

$$\hat{Q}_{sp}(\mathbb{B}) = Q(\mathbf{R_r}). \tag{4.16}$$

Proof. To prove (4.15), it is sufficient to show that if the pair $\mathbf{x}_i$ and $\mathbf{y}_j$ contribute 1 to $Q(\mathbf{R_r})$ for any $\mathbf{r} \in \mathbb{B}$, they must also contribute 1 to $\hat{Q}_{sp}(\mathbb{B})$. If $\mathbf{x}_i$ and $\mathbf{y}_j$ contribute 1 to $Q(\mathbf{R_r})$, then $\|\mathbf{R_r}\mathbf{x}_i - \mathbf{y}_j\| \leq \epsilon$ and $\mathbf{R_r}\mathbf{x}_i \in l_\epsilon(\mathbf{y}_j)$. Since $\mathbf{r}$ is in $\mathbb{B}$ then $\mathbf{R_r}\mathbf{x}_i$ must lie in $S_{\alpha_\mathbb{B}}(\mathbf{R_c}\mathbf{x}_i)$; see (4.11). This proves that the intersection $S_{\alpha_\mathbb{B}}(\mathbf{R_c}\mathbf{x}_i) \cap l_\epsilon(\mathbf{y}_j)$ contains at least the item $\mathbf{R_r}\mathbf{x}_i$ and is thus nonempty.

To prove (4.16), based on (4.6) as $\mathbb{B}$ collapses to a single point, $\mathbf{p} = \mathbf{q} = \mathbf{c}$ and $\alpha_\mathbb{B} = 0$. Thus $S_{\alpha_\mathbb{B}}(\mathbf{R_c}\mathbf{x}_i)$ collapses to a single point $\mathbf{R_c}\mathbf{x}_i$, rendering (4.14) to equal (4.13). □

Intuitively, $\hat{Q}_{sp}$ imposes a tighter bound than $\hat{Q}_{br}$, since given $\mathbb{B}$, $\hat{Q}_{sp}$ allows $\mathbf{x}_i$ to vary within a spherical patch while $\hat{Q}_{br}$ allows $\mathbf{x}_i$ to vary within a ball that encloses the spherical patch. A formal proof is as follows.

Lemma 4.2. *For any cube* $\mathbb{B}$

$$\hat{Q}_{br}(\mathbb{B}) \geq \hat{Q}_{sp}(\mathbb{B}). \tag{4.17}$$

Proof. Since both functions are already lower-bounded by $\max_{\mathbf{r}\in\mathbb{B}} Q(\mathbf{R_r})$, it is sufficient to show that there are hypothetical pairs $\mathbf{x}_i$ and $\mathbf{y}_j$ that contribute 1 to $\hat{Q}_{br}$ but 0 to

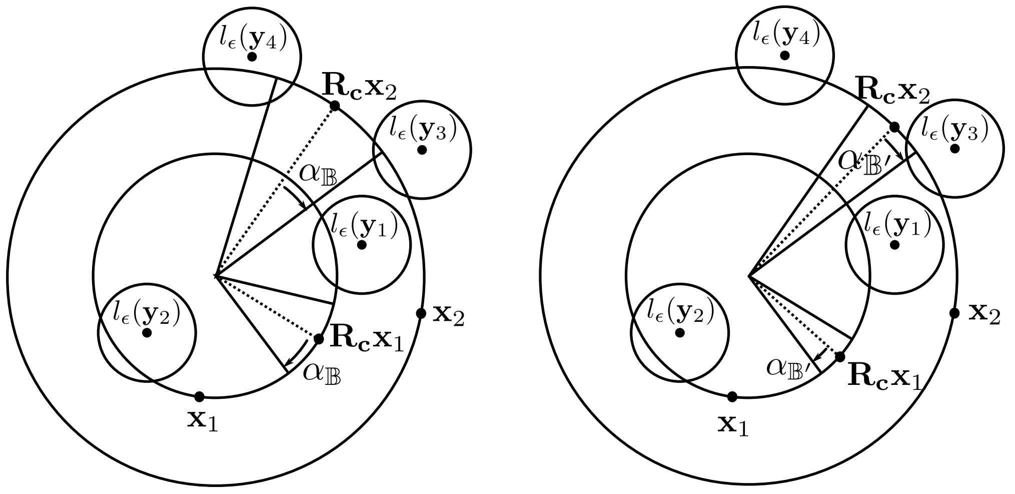


FIGURE 4.2: Illustrating the idea of matchlists on 1D rotation search—an analogous idea exists for 3D rotation search. Here, the origin is at the centre of the largest circle. (Left) Under the actions of all possible rotations in an interval $\mathbb{B} \subseteq [0, 2\pi]$, $\mathbf{x}_1$ cannot match with any of the $\mathbf{y}_j$'s, whilst $\mathbf{x}_2$ can match (up to ϵ) with $\mathbf{y}_3$ and $\mathbf{y}_4$. (Right) For a subinterval $\mathbb{B}'$ of $\mathbb{B}$, only $\mathbf{x}_2$ needs to be tested for potential intersections (i.e., the matchlist of $\mathbb{B}'$ is $\{\mathbf{x}_2\}$), since it is not possible for $\mathbf{x}_1$ to have a match under $\mathbb{B}'$.

$\hat{Q}_{sp}$. Set $\mathbf{y}_j = \mathbf{R_c x}_i(1 + \frac{\epsilon+\delta_i}{\|\mathbf{x}_i\|})$; clearly the condition $\|\mathbf{R_c x}_i - \mathbf{y}_j\| \leq \epsilon + \delta_i$ holds and $\mathbf{x}_i$ and $\mathbf{y}_j$ are matched under $\hat{Q}_{br}$. However, then $\|\mathbf{y}_j\| - \|\mathbf{x}_i\| > \epsilon$ and $l_\epsilon(\mathbf{y}_j)$ cannot intersect with $S_{\alpha_{\mathbb{B}}}(\mathbf{x}_i)$, thus giving 0 to $\hat{Q}_{sp}$. $\square$

Applying $\hat{Q}_{sp}$ in BnB instead of Breuel's original bound $\hat{Q}_{br}$ allows more aggressive pruning of unpromising rotations and thus leads to faster convergence.

4.3.4 Matchlists

Let $\mathcal{N}$ be the subset of points in $\mathcal{X}$ that can potentially have matches with $\mathcal{Y}$ under the rotations in $\mathbb{B}$, i.e.,

$$\mathcal{N} = \{\mathbf{x} \in \mathcal{X} \mid \exists\, \mathbf{y} \in \mathcal{Y},\ \ S_{\alpha_{\mathbb{B}}}(\mathbf{R_c x}) \cap l_\epsilon(\mathbf{y}) \neq \emptyset\}, \tag{4.18}$$

where $\mathbf{R_c}$ and $\alpha_{\mathbb{B}}$ are as defined in (4.6) for $\mathbb{B}$. For a subcube $\mathbb{B}' \subseteq \mathbb{B}$, it can be established that

$$\hat{Q}_{sp}(\mathbb{B}') \leq \hat{Q}_{sp}(\mathbb{B}) = |\mathcal{N}|. \tag{4.19}$$

Further, points not in $\mathcal{N}$ cannot possibly have matches under rotations in $\mathbb{B}'$. Thus, we need to sum over only $\mathcal{N}$ when evaluating $\hat{Q}_{sp}(\mathbb{B}')$. Figure 4.2 illustrates the idea.

Breuel called $\mathcal{N}$ the *matchlist* of $\mathbb{B}'$ [10]. Using matchlists avoids redundant intersection queries in (4.14), especially in the later stages of BnB. To apply the idea in Algorithm 4.1, when inserting a cube $\mathbb{B}$ into the queue, the algorithm also records the indices of points that are matched under $\mathbb{B}$, such that the subcubes of $\mathbb{B}$ can benefit from using matchlists. Note that the quality evaluation $Q(\mathbf{R_c})$ for the centre rotation of $\mathbb{B}$ (step 6 in Algorithm 4.1) can also be speeded up using the matchlist of $\mathbb{B}$.

It is worthwhile to note that matchlists are not applicable in a BnB algorithm for the LS criterion (4.2), since each $\mathbf{x}_i$ must always be matched to its nearest point in $\mathcal{Y}$ regardless of the distance.

4.4 Efficient bound evaluations

Using a tighter bound in BnB can be counterproductive if evaluating the bound itself takes significant time. The Kd-tree is the main workhorse in [11] for evaluating Q and $\hat{Q}_{br}$. Points in $\mathcal{Y}$ are indexed in a single kd-tree which is queried during BnB with rotated points from $\mathcal{X}$. This takes $\mathcal{O}(M \log N)$ effort per function evaluation.

To evaluate the proposed bound (4.14), it is necessary to solve multiple queries of the following kind:

$$\max_{j=1\dots N} \lfloor S_{\alpha_\mathbb{B}}(\mathbf{R_c}\mathbf{x}_i) \cap l_\epsilon(\mathbf{y}_j) \neq \emptyset \rfloor \,. \tag{4.20}$$

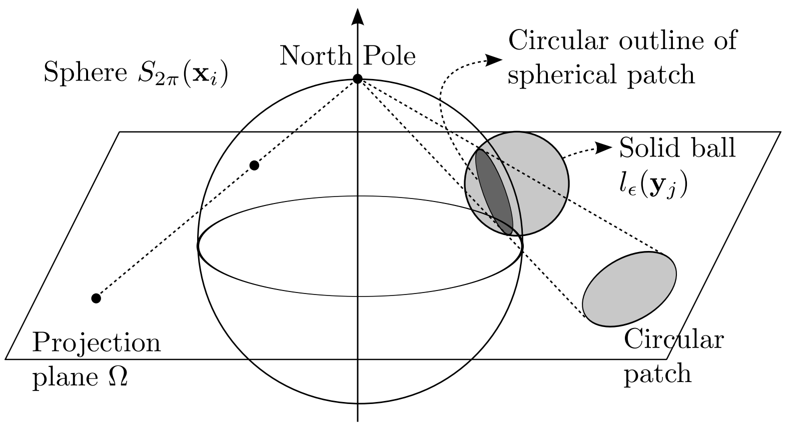
Intuitively, since $S_{\alpha_\mathbb{B}}(\mathbf{R_c}\mathbf{x}_i)$ must lie on the surface of the sphere $S_{2\pi}(\mathbf{x}_i)$, only the subset of $\mathcal{Y}$ whose $l_\epsilon(\mathbf{y}_j)$ intersect with $S_{2\pi}(\mathbf{x}_i)$ can possibly have a non-zero intersection with $S_{\alpha_\mathbb{B}}(\mathbf{R_c}\mathbf{x}_i)$. This subset is defined as

$$\mathcal{Y}_{\mathbf{x}_i} = \{\mathbf{y}_j \mid \mathbf{y}_j \in \mathcal{Y}, |\|\mathbf{x}_i\| - \|\mathbf{y}_j\|| \leq \epsilon\}\,, \tag{4.21}$$

and the maximisation in (4.20) can be taken over just $\mathcal{Y}_{\mathbf{x}_i}$. Section 4.4.3 will provide an efficient algorithm for finding $\mathcal{Y}_{\mathbf{x}_i}$ for all i, given two point clouds $\mathcal{X}$ and $\mathcal{Y}$.

4.4.1 Kd-tree approach

To evaluate (4.20) quickly, each $\mathcal{Y}_{\mathbf{x}_i}$ can be indexed with a kd-tree. A total of M kd-trees are thus constructed. Given $\mathbb{B}$, a *range query* can be performed on the i-th kd-tree with point $\mathbf{R_c}\mathbf{x}_i$ and range $(\delta_i + \epsilon)$ (recall that $S_{\alpha_\mathbb{B}}(\mathbf{R_c}\mathbf{x}_i)$ is enclosed by the $(\delta_i + \epsilon)$-ball centred at $\mathbf{R_c}\mathbf{x}_i$). This disregards points in $\mathcal{Y}$ that will never match with $\mathbf{x}_i$. Evaluating (4.14) thus takes $\mathcal{O}(M \log N_{av})$ effort, where $N_{av} \leq N$ is the average size of $\{\mathcal{Y}_{\mathbf{x}_i}\}_{i=1}^{M}$.



Sphere $S_{2\pi}(\mathbf{x}_i)$
North Pole
Circular outline of spherical patch
Solid ball $l_\epsilon(\mathbf{y}_j)$
Projection plane Ω
Circular patch

FIGURE 4.3: Stereographic projection of a spherical patch. A solid ball $l_\epsilon(\mathbf{y}_j)$ intersects the surface of the sphere $S_{2\pi}(\mathbf{x}_i)$ at a spherical patch, which has a circular outline on the sphere. Under stereographic projection, the spherical patch is projected to become a circular patch.

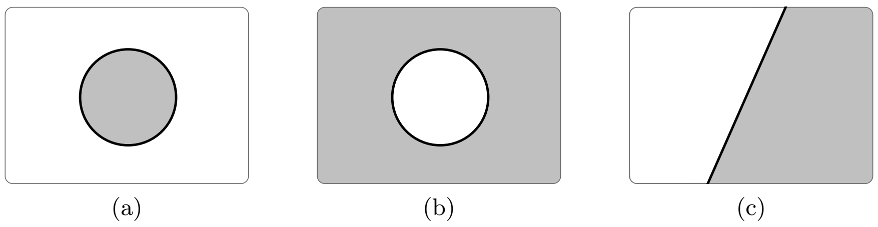


FIGURE 4.4: The three types of patches arising from projecting spherical patches. (a) Interior patch. This is the case shown in Figure 4.3. (b) Exterior patch. The spherical patch contains the North Pole, thus the "contents" of the spherical patch are projected outside the circle. (c) Half-plane. The North Pole lies exactly on the outline of the spherical patch.

4.4.2 Circular R-tree approach

While the M kd-tree approach permits faster bound evaluation than naive search, the technique to be proposed in this section gives bigger computational gains. Continuing the above observations, each $l_\epsilon(\mathbf{y}_j)$ for $\mathbf{y}_j \in \mathcal{Y}_{\mathbf{x}_i}$ intersects $S_{2\pi}(\mathbf{x}_i)$ at a spherical patch—recall that a sphere-to-sphere intersection yields a circle, i.e., the outline of the spherical patch, see Figure 4.3. The size of the patch depends on the distance of $\mathbf{y}_j$ to $S_{2\pi}(\mathbf{x}_i)$.

The idea is to *stereographically project* the spherical patches onto the xy-plane Ω. Assuming an unit-sphere and a projection pole at $[0,0,1]^\top$ (North Pole), a point $\mathbf{x}$ on the sphere and its projection $\mathbf{p} = [p_1,\ p_2]^\top$ are related by

$$\mathbf{x} = \left[\frac{2p_1}{1+\mathbf{p}^\top\mathbf{p}},\ \frac{2p_2}{1+\mathbf{p}^\top\mathbf{p}},\ \frac{\mathbf{p}^\top\mathbf{p}-1}{1+\mathbf{p}^\top\mathbf{p}}\right]^\top. \tag{4.22}$$

The crucial property is that circles are projected as circles; see Figure 4.3. In order to see this, recall that a circle arises from the intersection between a plane τ and the sphere. Let τ be $[a\ b\ c]\,\mathbf{x} = d$. Putting (4.22) into the plane equation yields

$$(c-d)(p_1^2+p_2^2)+2ap_1+2bp_2-(c+d)=0. \tag{4.23}$$

If $c \neq d$, (4.23) is a circle; else it is a line. In the latter case, the pole lies on the circle formed by the plane-sphere intersection. Circle intersections are also preserved, i.e., circles on the surface of the sphere that intersect will also intersect in the projection plane Ω. See [49] for details.

A spherical patch is thus projected to become a *circular patch* in Ω; see Figure 4.3. Given the circular patches from $\mathcal{Y}_{\mathbf{x}_i}$, to solve (4.20), first $S_{\alpha_{\mathbb{B}}}(\mathbf{R_c x}_i)$ is stereographically projected to obtain the query patch $\mathcal{L}_q$, then it is checked if $\mathcal{L}_q$ intersects any of the spherical patches from $\mathcal{Y}_{\mathbf{x}_i}$. What makes this technique more efficient than the M kd-tree approach is the usage of *spatial access* data structures [45] to query for intersections. Note that the stereographic projection of a circle can be computed in closed-form in constant time, thus it presents little overheads.

The following subsections explain stereographic projection of spherical patches and efficient indexing schemes for circular patches. Note that by replacing $\mathcal{L}_q$ with the stereographic projection of $\mathbf{Rx}_i$, the methods below can also be used to evaluate the quality (4.13).

4.4.2.1 Projection of spherical patches

Discussing the full details of stereographic projection of circles is beyond the scope of this chapter. This section provides only the essential details. The reader can refer to the text [49] for a more comprehensive description.

Typically the vast majority of spherical patches do not intersect with or contain the North Pole. These are projected to become *interior patches*, i.e., the interior of the spherical patch is projected to the interior of the circle in Ω. This is the case in Figure 4.3. If a spherical patch contains the North Pole in its interior, it is projected to become an *exterior patch*, i.e., its contents are projected outwards. If the North Pole lies exactly on the circular outline of the spherical patch, the projection gives rise to a *half-plane*. Figure 4.4 shows the three possibilities.

To stereographically project a spherical patch $S_\alpha(\mathbf{x})$, it is convenient to first normalise the patch such that it lies on the unit-sphere. This implies making $\|\mathbf{x}\| = 1$ while leaving α unchanged. A proportional scaling of the ϵ-ball that gave rise to $S_\alpha(\mathbf{x})$ is not required, since the angular deviation α does not change with this normalisation. Let $(\varphi_{\mathbf{x}}, \theta_{\mathbf{x}})$ be the spherical coordinates of $\mathbf{x}$, where $\varphi_{\mathbf{x}} \in [0,\pi]$ and $\theta_{\mathbf{x}} \in [0, 2\pi]$ are the inclination

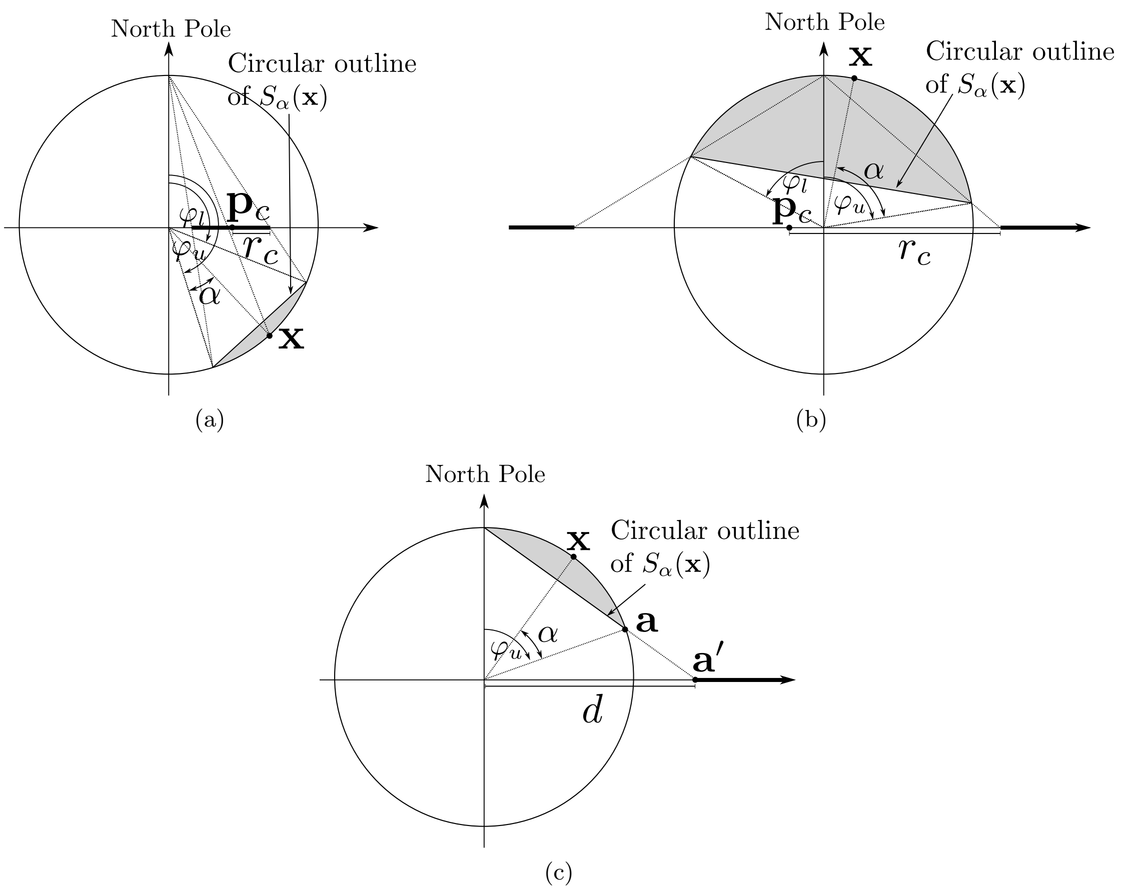


FIGURE 4.5: Projection of a spherical patch $S_\alpha(\mathbf{x})$. The diagrams show the side view of Figure 4.3, where the horizontal axis represents the xy-plane Ω. As explained in Figure 4.4, three cases can arise: (a) an interior patch, (b) an exterior patch, or (c) a half-plane. In the above diagrams, the bolded segments on the horizontal axes indicate the resulting patches.

and azimuth. If $\varphi_{\mathbf{x}} - \alpha > 0$, $S_\alpha(\mathbf{x})$ does not contain the North Pole; see Figure 4.5a. If $\varphi_{\mathbf{x}} - \alpha < 0$, the North Pole lies in the interior of $S_\alpha(\mathbf{x})$; see Figure 4.5b. Finally, if $\varphi_{\mathbf{x}} - \alpha = 0$, the North Pole lies exactly on the circular outline of $S_\alpha(\mathbf{x})$; see Figure 4.5c.

Consider first the spherical patches that are projected to yield interior and exterior patches. The aim is to project the circular outline of $S_\alpha(\mathbf{x})$ to a circle $(\mathbf{p}_c, r_c)$ on Ω.

Let points $\mathbf{a}'$ and $\mathbf{b}'$ on the circle be the closest and the farthest points to the origin (see Figure 4.6). The following lemma establishes that $\mathbf{a}'$, $\mathbf{b}'$, $\mathbf{p}_c$ and the origin are collinear.

Lemma 4.3. *The farthest and the closest point on a circle from* $\mathbf{p}$ *are collinear with* $\mathbf{p}$ *and the circle's centre* $\mathbf{c}$.

Proof. Without loss of generality, say the circle centre $\mathbf{c}$ coincides with the origin and $\mathbf{p}$ lies on the x-axis such that $\mathbf{p} = [p_x, 0]^\top$ and $p_x \geq 0$. The distance l from $\mathbf{p}$ to any point

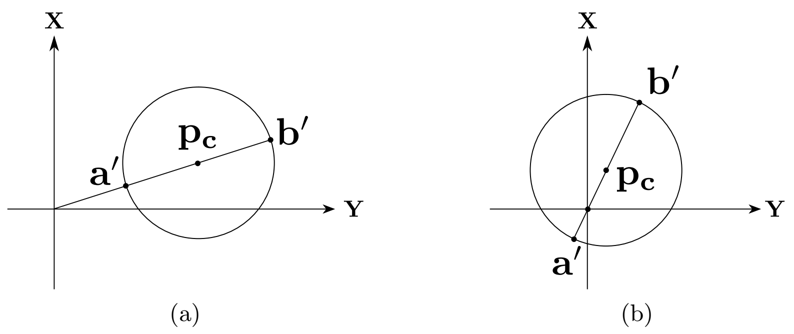


FIGURE 4.6: The closest point $\mathbf{a}'$ on a circle to the origin and the farthest one $\mathbf{b}'$ are in the line passing through the origin and the centre of the circle $\mathbf{p}_c$. (a) When the origin is in the exterior region of the circle, azimuth of $\mathbf{a}'$ and $\mathbf{b}'$ are equal. (b) When the origin is in the interior region of the circle, the absolute difference of azimuth of $\mathbf{a}'$ and azimuth of $\mathbf{b}'$ is π.

in a circle with polar coordinates (r, θ) is given by

$$l^2 = r^2 + p_x^2 - 2rp_x \cos(\theta). \tag{4.24}$$

Then, the minimum distance is reached for $\theta_{min} = 2k\pi$; and the maximum one for $\theta_{max} = (2k+1)\pi$, where $k \in \mathbb{N}$. As the polar coordinates (r, θ_{max}) and (r, θ_{min}) lie over the x-axis as well as $\mathbf{p}$ and $\mathbf{c}$, all of them are collinear. □

From Lemma 4.3, the points in the circle with minimum and maximum radial distance ($\mathbf{a}'$ and $\mathbf{b}'$) lie on a line passing through the circle's centre (see Figure 4.6). Then, the circle's parameters can be obtained as

$$\mathbf{p}_c = \frac{\mathbf{a}' + \mathbf{b}'}{2} \quad \text{and} \quad r_c = \frac{\|\mathbf{a}' - \mathbf{b}'\|}{2}. \tag{4.25}$$

The stereographic projection of spherical coordinates on the unit-sphere (inclination $\varphi \in [0, \pi]$ and azimuth $\theta \in [0, 2\pi]$) to polar coordinates (radial distance r' and azimuth θ') is given by

$$(r', \theta') = \left(\frac{\sin(\varphi)}{1 - \cos(\varphi)}, \ \theta \right). \tag{4.26}$$

Since $r'(\varphi)$ is decreasing with respect to φ, the points $\mathbf{a}$, $\mathbf{b}$ with the highest and the lowest inclination in the circular outline are projected to Ω as the points $\mathbf{a}'$, $\mathbf{b}'$ with the lowest and the highest radial distance. Let $(\varphi_{\mathbf{a}}, \theta_{\mathbf{a}})$ and $(\varphi_{\mathbf{b}}, \theta_{\mathbf{b}})$ be the spherical coordinates of $\mathbf{a}$ and $\mathbf{b}$, and $(r'_{\mathbf{a}}, \theta_{\mathbf{a}})$ and $(r'_{\mathbf{b}}, \theta_{\mathbf{b}})$ the polar coordinates of $\mathbf{a}'$ and $\mathbf{b}'$. From Lemma 4.3, $\mathbf{a}'$ and $\mathbf{b}'$ are collinear with the origin, therefore $\theta_{\mathbf{a}} = \theta_{\mathbf{b}}$ when $\mathbf{a}'$ and $\mathbf{b}'$ are at the same side with respect to the origin, and $|\theta_{\mathbf{a}} - \theta_{\mathbf{b}}| = \pi$ if the origin is between them (see Figure 4.6).

Following (4.26), the circle's parameters can be rewritten when $\theta_{\mathbf{a}} = \theta_{\mathbf{b}}$ as

$$\mathbf{p}_c = \frac{r'(\varphi_{\mathbf{a}}) + r'(\varphi_{\mathbf{b}})}{2} \begin{bmatrix} \cos(\theta_{\mathbf{a}}) \\ \sin(\theta_{\mathbf{a}}) \end{bmatrix} \quad \text{and} \quad r_c = \frac{r'(\varphi_{\mathbf{b}}) - r'(\varphi_{\mathbf{a}})}{2}, \tag{4.27}$$

and when $|\theta_{\mathbf{a}} - \theta_{\mathbf{b}}| = \pi$ as

$$\mathbf{p}_c = \frac{r'(\varphi_{\mathbf{a}}) - r'(\varphi_{\mathbf{b}})}{2} \begin{bmatrix} \cos(\theta_{\mathbf{a}}) \\ \sin(\theta_{\mathbf{a}}) \end{bmatrix} \quad \text{and} \quad r_c = \frac{r'(\varphi_{\mathbf{b}}) + r'(\varphi_{\mathbf{a}})}{2}. \tag{4.28}$$

By expressing the inclination of **a** and **b** as

$$\varphi_u = \varphi_{\mathbf{x}} + \alpha, \quad \varphi_u \in [0, 2\pi] \tag{4.29}$$

$$\varphi_l = \varphi_{\mathbf{x}} - \alpha, \quad \varphi_l \in [-\pi, \pi] \tag{4.30}$$

it can be easily verified that circle's parameters can be obtained as

$$\mathbf{p}_c = \frac{r(\varphi_l) + r(\varphi_u)}{2} \hat{\mathbf{x}}' \quad \text{and} \quad r_c = \frac{|r(\varphi_l) - r(\varphi_u)|}{2}, \tag{4.31}$$

where $r(\cdot)$ is defined as r' in (4.26) but allowed to take values in $[-\pi, 2\pi]$, and

$$\hat{\mathbf{x}}' := \begin{bmatrix} \cos(\theta_{\mathbf{x}}) \\ \sin(\theta_{\mathbf{x}}) \end{bmatrix} = \frac{[\mathbf{x}(1), \mathbf{x}(2)]^\top}{\|[\mathbf{x}(1), \mathbf{x}(2)]^\top\|} \tag{4.32}$$

is the unit vector of the *orthogonal* projection of $\mathbf{x}$ onto Ω.

Now consider the $S_\alpha(\mathbf{x})$ that project to a half-plane (see Figure 4.5c). That condition is satisfied when $\varphi_l = 0$. Following an analogous analysis that when the projection is a circle, let **a** be the point in $S_\alpha(\mathbf{x})$ such that its stereographic projection $\mathbf{a}'$ is the closest point to the origin on Ω. Then projection of remaining points in the circle must lie in a line passing through $\mathbf{a}'$ and normal to it. Since the side of the line where $S_\alpha(\mathbf{x})$ is mapped is flipped when $\varphi_u > \pi$, the half-plane is defined as

$$\hat{\mathbf{a}}'\mathbf{p} - d \geq 0 \quad \text{if } \varphi_u < \pi \tag{4.33}$$

$$\hat{\mathbf{a}}'\mathbf{p} - d < 0 \quad \text{if } \varphi_u \geq \pi \tag{4.34}$$

where $\mathbf{p}$ is an arbitrary point in Ω, $d = |r(2\alpha)|$ is the radial distance for the inclination $\varphi_u = 2\alpha$ (recall that $\varphi_l = 0$ in this case), and $\hat{\mathbf{a}}'$ the unit vector in $\mathbb{R}^2$ corresponding to the *orthogonal* projection of **a** onto Ω. See Figure 4.5c.

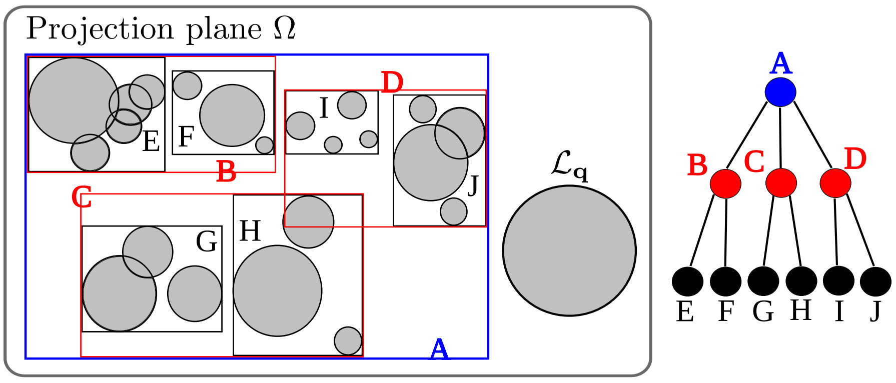


FIGURE 4.7: A set of interior patches in the projection plane is indexed in a circular R-tree. The MBR at each node is also drawn. The tree structure is shown on the right. A query patch $\mathcal{L}_q$ is also shown; in this example, $\mathcal{L}_q$ does not intersect with the largest MBR at the root node, hence the search need *not* proceed beyond the root.

4.4.2.2 Indexation for fast intersection queries

Once the spherical patches from $\mathcal{Y}_{\mathbf{x}_i}$ are projected onto the xy-plane Ω, they are indexed to facilitate efficient intersection queries. Here, indexing schemes are described for the three possible types of circular patches.

The indexing of exterior patches and half-planes is first explained. These resulting patches arise from spherical patches that are containing or intersecting the North Pole. As these patches are infrequent in practice (in fact, no half-planes existed in the experiments of Section 4.6), they are simply indexed in a list. Given a query patch $\mathcal{L}_q$, the list is scanned to see if $\mathcal{L}_q$ intersects with any of the entries; as soon as a hit is encountered, the search is stopped and 1 is returned to (4.20). In fact, if $\mathcal{L}_q$ is itself an exterior patch or a half-plane, it will always intersect with an entry in the list (since all the originating spherical patches contain and intersect at the North Pole) and the scan can be avoided.

Solving (4.20) is dominated by testing the interior patches for overlaps with $\mathcal{L}_q$. To facilitate efficient querying, the interior patches are indexed (specifically, their circular outlines) in a *circular R-tree*. R-trees are indexing structures designed for *spatial access* queries; see [45] for a general exposition. In the used circular R-tree, the circles are hierarchically indexed in a balanced tree. Circles in the same node are enclosed by a minimum bounding rectangle (MBR). For example, Figure 4.7 shows a circular R-tree of depth three. The main parameter for tree building is the branching factor and maximum depth.

Regardless of the type of circular patch $\mathcal{L}_q$, querying the circular R-tree is conducted similarly; the distinction lies in how the overlaps are defined. At each node, if $\mathcal{L}_q$ overlaps with the MBR of the node, the children of the node are traversed; at a leaf node, $\mathcal{L}_q$ is simply tested for overlaps with the interior patches contained therein, and if a hit is encountered the query is terminated instantly. If $\mathcal{L}_q$ does not overlap with the MBR of a node, the whole branch can be ignored; contrast this to kd-tree queries, where the full depth of the tree must be reached, such that candidate nearest distances are obtained to enable pruning of branches. In fact, in the circular R-tree, should (4.20) evaluate to 0, it is usually unnecessary to explore all tree levels. In many actual cases, only the first-few levels are descended; Figure 4.7 shows an example. This difference in behaviour is the source of massive improvements in runtime, as will be shown in Section 4.6.1.

4.4.3 Modified plane sweep algorithm

As defined earlier, $\mathcal{Y}_{\mathbf{x}_i} \subseteq \mathcal{Y}$ is the set of points for which the ϵ-ball $l_\epsilon(\mathbf{y}_j)$ intersects with the sphere $S_{2\pi}(\mathbf{x}_i)$. A naive method to compute all $\{\mathcal{Y}_{\mathbf{x}_i}\}_{i=1}^{M}$ is thus to test for each pair (i, j) whether $S_{2\pi}(\mathbf{x}_i)$ intersects with $l_\epsilon(\mathbf{y}_j)$. Testing all $M \times N$ pairs is wasteful, since not all the pairs intersect. This section proposes a more efficient method inspired by the plane sweep algorithm [55] used for calculating line segment intersections. See Algorithm 4.2.

The algorithm is described using the concepts of *status* and *events* used in plane sweep. Specifically, the plane sweep algorithm maintains a status containing the sorted values of the norms $\{\|\mathbf{x}_i\|\}_{i=1}^{M}$. The algorithm then iterates over events $\{\mathbf{y}_j\}_{j=1}^{N}$. For each event $\mathbf{y}_j$, it inserts the values $\|\mathbf{y}_j\| - \epsilon$ and $\|\mathbf{y}_j\| + \epsilon$ into the status; let i_l and i_u respectively be the position of $\|\mathbf{y}_j\| - \epsilon$ and $\|\mathbf{y}_j\| + \epsilon$ in the status. Due to the presorting of the norms $\{\|\mathbf{x}_i\|\}_{i=1}^{M}$, any $\mathbf{x}_i$ whose index in status is below i_l or above i_u cannot intersect with $l_\epsilon(\mathbf{y}_j)$. Thus, the algorithm does not exhaustively test all $M \times N$ pairs of data for intersections. More specifically, it takes $\mathcal{O}(N \log M)$ effort, since it inserts into a sorted array (the status) N times.

Algorithm 4.2 Modified plane sweep algorithm for finding the intersecting ϵ-ball set $\{\mathcal{Y}_{\mathbf{x}_i}\}_{i=1}^{M}$.

```
Require: Point sets $\mathcal{X}$ and $\mathcal{Y}$, threshold $\epsilon$.
 1: Set $\mathcal{Y}_{\mathbf{x}_i} = \emptyset$ for all $i = 1, \dots, M$.
 2: $status \leftarrow$ sort $\{\|\mathbf{x}_i\|\}_{i=1}^{M}$ for all $\mathbf{x}_i \in \mathcal{X}$.
 3: Reorder $\mathcal{X}$ based on their position in $status$.
 4: for $j = 1, \dots, N$ do
 5:   $i_l \leftarrow$ smallest $i$ such that $status(i) \geq \|\mathbf{y}_j\| - \epsilon$.
 6:   $i_u \leftarrow$ largest $i$ such that $status(i) \leq \|\mathbf{y}_j\| + \epsilon$.
 7:   if both $i_l$ and $i_u$ are not null then
 8:     for $i = i_l, \dots, i_u$ do
 9:       $\mathcal{Y}_{\mathbf{x}_i} \leftarrow \mathcal{Y}_{\mathbf{x}_i} \cup \{l_\epsilon(\mathbf{y}_j)\}$.
10:     end for
11:   end if
12: end for
```

4.4.4 Computational analysis

To evaluate the proposed bounding function (4.14), it will be necessary to build and query M circular R-trees. Similarly for the kd-tree approach, M kd-trees are required. Search efficiency is of greater interest since querying occurs multiple times during BnB. Theoretically, R-trees and kd-trees have similar search complexities, which is $\mathcal{O}(\log n)$. In the worst case the algorithm will need to traverse the full depth of the tree and visit other branches. In practice, however, significant speedups are observed when using circular R-trees. A reason behind this was given in Section 4.4.2.2.

Of secondary interest is the tree-building time, which occurs only once before the main loop of Algorithm 4.1. Given $\mathcal{Y}_{\mathbf{x}_i}$, constructing a balanced circular R-tree and kd-tree have similar complexities. In particular, there exists a linear time worst case algorithm for insertion in R-tree; see [45] for details. Both types of trees will require finding $\{\mathcal{Y}_{\mathbf{x}_i}\}_{i=1}^{M}$ using Algorithm 4.2.

4.5 6 DoF registration

This section shows how the proposed fast rotation search method can be used for full 3D (6 DoF) point cloud registration.

4.5.1 Locally optimal method (Loc-GM)

Firstly, a locally optimal method is presented. Whilst locally optimal methods for the LS criterion are abundant [9, 25, 65], there appear to be no such algorithms for the GM criterion.

Formally, the aim is to maximise

$$Q(\mathbf{R}, \mathbf{T}) = \sum_{i=1}^{M} \max_{j=1\ldots N} \lfloor \|\mathbf{R}\mathbf{x}_i + \mathbf{T} - \mathbf{y}_j\| \leq \epsilon \rfloor , \tag{4.35}$$

where $\mathbf{T} \in \mathbb{R}^3$ is a translation vector. The proposed algorithm is simple: given the current parameters $(\mathbf{R}^{(t)}, \mathbf{T}^{(t)})$, it repeatedly updates $\mathbf{R}$ and $\mathbf{T}$ by holding one of the components constant at each iteration. However, both subproblems are nonconvex. Each subproblem is *globally* solved by using BnB. Fixing $\mathbf{R}^{(t)}$, the new translation component $\mathbf{T}^{(t+1)}$ is obtained by BnB over the translation parameter space. Optimising over $\mathbb{R}^3$ is more efficient than SO(3), since the underlying "rectangular" structure of $\mathbb{R}^3$ enables the tightest possible bound. The translation search can be done easily by a 3D extension of Breuel's algorithm [11]. Given $\mathbf{T}^{(t+1)}$, $\mathbf{R}^{(t+1)}$ is obtained via Algorithm 4.1. The subproblems are solved repeatedly until $Q(\mathbf{R}, \mathbf{T})$ does not change with the updates.

4.5.2 Globally optimal method (Glob-GM)

To conduct globally optimal 6 DoF registration, this chapter is based on the *nested BnB* idea used in Go-ICP [73], where two BnB algorithms (one each for $\mathbf{R}$ and $\mathbf{T}$) are executed in a nested manner to reach globally optimal solutions. Different from Go-ICP, here, the inner BnB optimises rotation and the outer one optimises translation. See Algorithm 4.3.

The goal is to maximise the objective function

$$Q(\mathbf{R}, \mathbf{T}) = \sum_{i=1}^{M} \max_{j=1\ldots N} \lfloor \|\mathbf{R}(\mathbf{x}_i + \mathbf{T}) - \mathbf{y}_j\| \leq \epsilon \rfloor. \tag{4.36}$$

From the perspective of the outer BnB loop, the goal is to find the translation that maximises

$$V(\mathbf{T}) = \max_{\mathbf{R}} \sum_{i=1}^{M} \max_{j=1\ldots N} \lfloor \|\mathbf{R}(\mathbf{x}_i + \mathbf{T}) - \mathbf{y}_j\| \leq \epsilon \rfloor, \tag{4.37}$$

where $V(\mathbf{t})$ is "evaluated" given $\mathbf{T}$ by invoking Algorithm 4.1 to rotationally align points $\mathcal{X} + \mathbf{T}$ and $\mathcal{Y}$. Given a box of translations $\mathbb{T} \subset \mathbb{R}^3$ the upper bound

$$\hat{V}(\mathbb{T}) = \max_{\mathbf{R}} \sum_{i=1}^{M} \max_{j=1\ldots N} \lfloor \|\mathbf{R}(\mathbf{x}_i + \mathbf{T}_c) - \mathbf{y}_j\| \leq \epsilon + \delta \rfloor, \tag{4.38}$$

can again be evaluated by using Algorithm 4.1 to rotationally align points $\mathcal{X} + \mathbf{T}_c$ and $\mathcal{Y}$, where $\mathbf{T}_c$ is the centre of $\mathbb{T}$, and δ is half of the longest diagonal in $\mathbb{T}$. Effectively, both $\mathbf{R}$ and $\mathbf{T}$ are co-optimised and the final result is guaranteed to be 6 DoF globally optimal [73].

Another crucial feature of Go-ICP adopted here, is an auxiliary local method to improve the quality of the current best solution—see Line 10 in Algorithm 4.3. It has been shown in [73] that such an auxiliary routine (ICP was used in their case) helps to significantly speed up the overall 6 DoF algorithm. To this end, Algorithm 4.3 uses the locally convergent method Loc-GM described in Section 4.5.1.

Algorithm 4.3 Nested BnB algorithm to maximise (4.37).

```
Require: Point sets $\mathcal{X}$ and $\mathcal{Y}$, threshold $\epsilon$.
 1: Initialise priority queue $q$, $V^* \leftarrow 0$, $\mathbf{T}^* \leftarrow \emptyset$, $\mathbf{R}^* \leftarrow \emptyset$.
 2: Insert initial translation cube $\mathbb{T}$ into $q$.
 3: while $q$ is not empty do
 4:     Obtain highest priority cube $\mathbb{T}$ from $q$.
 5:     $\mathbf{T_c} \leftarrow$ centre of $\mathbb{T}$.
 6:     Calculate $V(\mathbf{t_c})$ by calling Algorithm 4.1 to align $\mathcal{X}+\mathbf{t_c}$ with $\mathcal{Y}$ based on threshold $\epsilon$.
 7:     If $V(\mathbf{t_c}) = V^*$, then terminate.
 8:     if $V(\mathbf{t_c}) > V^*$ then
 9:         $V^* \leftarrow V(\mathbf{t_c})$, $\mathbf{t}^* \leftarrow \mathbf{t_c}$, $\mathbf{R}^* \leftarrow \mathbf{R}$ from Line 6.
10:         Call Loc-GM (Sec 4.5.1) to refine $(\mathbf{R}^*, \mathbf{t}^*)$.
11:     end if
12:     Subdivide $\mathbb{T}$ into 8 sub-cubes $\{\mathbb{T}_d\}_{d=1}^{8}$.
13:     for each $\mathbb{T}_d$ do
14:         $\delta \leftarrow 1/2$ of the length of the diagonal of $\mathbb{T}_d$.
15:         $\mathbf{T_c} \leftarrow$ centre of $\mathbb{T}_d$.
16:         Calculate $\hat{V}(\mathbb{T}_d)$ by calling Algorithm 4.1 to align $\mathcal{X} + \mathbf{t_c}$ with $\mathcal{Y}$ based on threshold $\epsilon + \delta$.
17:         If $\hat{V}(\mathbb{T}_d) > V^*$, queue $\mathbb{T}_d$ with priority $\hat{V}(\mathbb{T}_d)$.
18:     end for
19: end while
20: return Optimal rotation $\mathbf{R}^*$ and translation $\mathbf{t}^*$.
```

4.6 Results

4.6.1 Rotation search

Firstly, the efficiency of the 3D rotation search method (Section 4.3) is examined. The efficiency of the 6 DoF point cloud registration algorithm (Section 4.5) will be analysed in Section 4.6.2.

The experiment was designed as follows. The methods were run on scans of objects from three different sources, namely, the Stanford 3D Scanning Repository [17] (specifically *bunny*, *armadillo*, *dragon*, and *buddha*), Mian's dataset [47] (specifically *parasaur*, *t-rex* and *chicken*) and a proprietary dataset of laser scans of underground mines[1] (specifically *mine-a*, *mine-b* and *mine-c*). For each object, two partially overlapping point clouds $\mathcal{V}_1$ and $\mathcal{V}_2$ were chosen. Figure 4.8 shows the point clouds used in this experiment, whilst Columns 2–3 in Table 4.1 list the sizes of $\mathcal{V}_1$ and $\mathcal{V}_2$.

Realistic point clouds $\mathcal{X}$ and $\mathcal{Y}$ were generated as input of rotation search for each object based on the following steps:

[1]The mining dataset was provided by Maptek.

Object	$\lvert\mathcal{V}_1\rvert$	$\lvert\mathcal{V}_2\rvert$	avg M	inliers (%)	1KDT time (s)	1KDT-ML time (s)	MKDT time (s)	MKDT-ML time (s)	MCIRC time (s)	MCIRC-ML time (s)
bunny	7055	6742	379.27	42	9.53	6.14	9.04	5.91	2.80	1.73
armadillo	5619	5483	356.94	14	17.81	10.30	16.17	9.11	4.54	2.38
dragon	6991	6200	349.28	29	9.26	5.95	8.50	5.69	2.00	1.22
buddha	5312	5109	374.71	20	14.09	8.04	12.44	6.85	3.43	1.74
mine-a	1285	917	362.33	10	233.84	65.48	172.74	46.96	48.96	12.39
mine-b	1445	1271	168.52	42	42.80	15.66	28.92	10.77	8.14	2.78
mine-c	1274	1102	183.10	81	45.38	16.33	30.34	11.10	8.29	2.83
parasaur	4495	3642	295.77	34	15.79	6.79	7.17	4.88	1.79	1.21
t-rex	6970	7636	226.72	13	11.83	7.97	8.96	5.77	2.48	1.49
chicken	7592	7829	294.51	17	11.53	8.51	8.56	6.10	2.18	1.53

TABLE 4.1: Comparing the performance of BnB rotation search methods using different bounds and bound evaluation methods. 1KDT: bound (4.9) using 1 kd-tree, MKDT: bound (4.14) using M kd-trees, MCIRC: bound (4.14) using M circular R-trees. 1KDT-ML, MKDT-ML and MCIRC-ML are variants of the above using match-lists (Section 4.3.4).

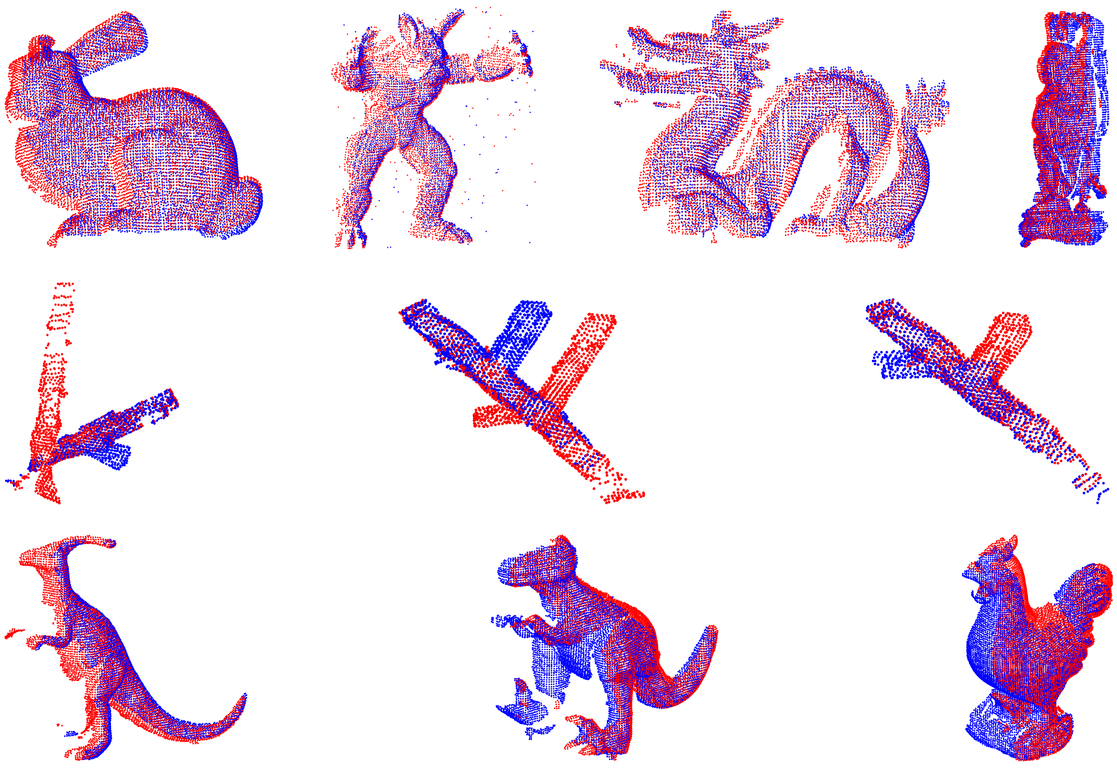


FIGURE 4.8: Point clouds used in the evaluation of rotation search: *bunny*, *armadillo*, *dragon*, *buddha*, *mine-a*, *mine-b*, *mine-c*, *parasaur*, *t-rex* and *chicken*.

1. Detect 3D keypoints in $\mathcal{V}_1$ and $\mathcal{V}_2$. It was used the ISS3D detector in Point Cloud Library (PCL) [61].

2. Calculate descriptors for the 3D keypoints. This was done by using the PFH method in PCL.

3. Match the keypoints between $\mathcal{V}_1$ and $\mathcal{V}_2$ by computing the L_2 distance between PFH descriptors. The matching threshold was chosen such that exactly 100 key-point matches were obtained. The matching precision varied across the objects (see Column 5 in Table 4.1), but it was verified that at least 3 true positive matches existed per $\mathcal{V}_1$ and $\mathcal{V}_2$ pair.

4. For each match $\mathbf{p} \leftrightarrow \mathbf{q}$, translate $\mathcal{V}_1$ by $-\mathbf{p}$ and $\mathcal{V}_2$ by $-\mathbf{q}$ such that they potentially differ by a rotation about the origin. Then take points from $\mathcal{V}_1$ and $\mathcal{V}_2$ within a radius δ_{loc} from the origin to produce the pair $\mathcal{X}$ and $\mathcal{Y}$ as input for rotation search. To "normalise" the sizes here, $M \leq N$ is ensured by swapping $\mathcal{X}$ and $\mathcal{Y}$ if needed (see Column 4 in Table 4.1 for the average size of $\mathcal{X}$ for each object). Since $\mathcal{Y}$ is indexed in efficient data structures, its actual size is of less concern. The radius δ_{loc} was chosen as a ratio of the point cloud extent from the origin and fixed ($\delta_{loc} = 5$, except for the underground mine scans where it was set to 20) for each $\mathcal{V}_1$ and $\mathcal{V}_2$ pair.

Note that the focus in this experiment is rotation search performance, thus there is less of a concern with actually registering the point clouds (again, this will be examined in Section 4.6.2). The following rotation search methods were benchmarked. All methods were implemented in C++ and executed on an Intel Core i7 3.40 GHz CPU.

- 1KDT: Breuel's original method [11], i.e., BnB with objective (4.1) and bound (4.9). The bound is evaluated using 1 kd-tree.

- MKDT: BnB with objective (4.13) and bound (4.14). The bound is evaluated using M kd-trees (see Section 4.4.1).

- MCIRC: BnB with objective (4.13) and bound (4.14). The bound is evaluated using steoreographic projection and M circular R-trees (see Section 4.4.2).

- 1KDT-ML, MKDT-ML and MCIRC-ML: Variants of the above with matchlists (see Section 4.3.4).

Note that all the BnB methods above optimise the same GM criterion for rotation search (in fact, they all achieve the same globally optimal quality), thus their runtime results are directly comparable.

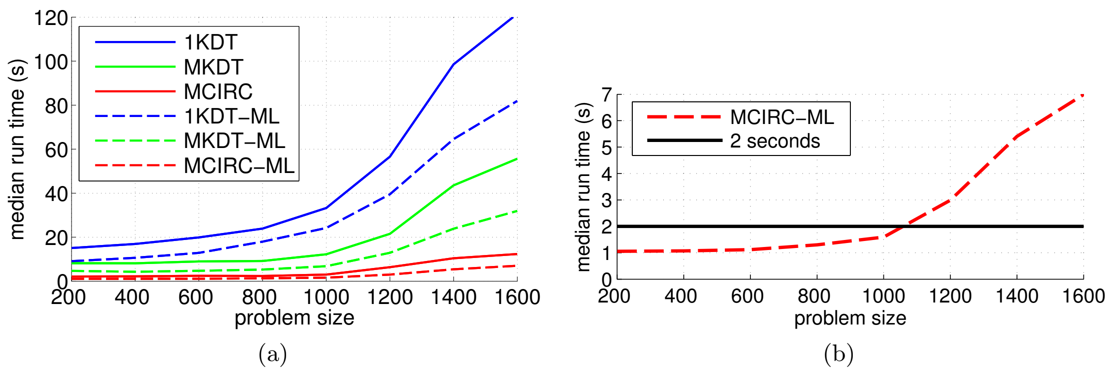


FIGURE 4.9: (a) Median runtime versus problem size M. (b) Median runtime versus problem size M for MCIRC-ML. In this experiment, it was ensured that the input point clouds $\mathcal{X}$ and $\mathcal{Y}$ are of equal size so as to compare against [7].

The average runtime of all methods are listed in Columns 6–11 in Table 4.1. Note that the recorded times include durations for all data structure preparations (e.g., building kd-trees or circular R-trees). It can be seen that the datasets differ significantly in difficulty. In general, the runtime is an order of magnitude larger on the underground mine dataset, possibly due to the more "organic" looking 3D structures. Also, using matchlists generally helps in substantially speeding up BnB convergence, and this effect is more pronounced in the harder point clouds. The results also point to the significant computational gains obtained via the proposed bounding function and bound evaluation algorithm. Specifically, the proposed approach requires an order of magnitude less processing time than Breuel's original method. Using matchlists also allows MCIRC-ML to be several times faster than MCIRC.

4.6.1.1 Scalability of rotation search algorithm

To investigate the scalability of the rotation search algorithms, the above experiment is repeated with only the *armadillo* scans to avoid excessive runtimes. Also, the neighbourhood size δ_{loc} was varied to test a wider range of sizes of $\mathcal{X}$ and $\mathcal{Y}$. Figure 4.9a plots the median runtime as a function of the size of $\mathcal{X}$. Note that for the BnB rotation search algorithms, the size of $\mathcal{X}$ is a good representation of the problem size, since $\mathcal{Y}$ is indexed in data structures and its size is not influential to runtime. These results verify the superior performance of the proposed algorithm on a large range of problem sizes.

4.6.1.2 Comparison with other BnB rotation search

Comparing with other formulations and techniques for rotation search is nontrivial, but an endeavour has been made to quantitatively benchmark against [7]. While the authors of [7] also use BnB, there are crucial differences. First, their algorithm takes a set of point matches as input, while Algorithm 4.1 does not require any *a priori* determined point matches between $\mathcal{X}$ and $\mathcal{Y}$. Using point matches obviates the need to search for matches during BnB optimisation. Second, the error used in [7] is the *angular error* between matching points, while Algorithm 4.1 uses the L_2 distance. In the experiment of [7], keypoint matches were first obtained from two partially overlapping scans of the *bunny* dataset. The scans differed purely by rotation. The number of keypoint matches were not explicitly reported, but from [7, Figure 3] there seems to be approximately 100 keypoint matches. It was further reported that their algorithm "converges in a couple of seconds".

The scalability experiment in Section 4.6.1.1 was rerun. To obtain results that are comparable, here it was ensured that the size of $\mathcal{Y}$ is similar to the size of $\mathcal{X}$, by using an appropriately selected radius on the subsampled point clouds (note that this does not mean that the resulting point clouds have one-to-one correspondence). Figure 4.9b shows the median runtime of MCIRC-ML. This algorithm is evidently superior, since it can align up to 1000 points within 2 seconds despite the need to conduct matching between $\mathcal{X}$ and $\mathcal{Y}$ during the optimisation.

4.6.2 Globally optimal 6 DoF registration

This section examines the performance of the globally optimal 3D registration method (Glob-GM, Algorithm 4.3) which encapsulates the proposed fast rotation search algorithm. The closest work to the proposed method is Go-ICP [73], which globally minimises the LS criterion (4.2). Glob-GM was inspired by Go-ICP's nested BnB scheme (although their outer BnB loop optimises the rotation and the inner BnB loop estimates the translation). In Go-ICP, the standard ICP algorithm [9] was also incorporated to locally refine intermediate solutions. An equivalent step also exists in Glob-GM, where the novel local method Loc-GM (Section 4.5) is used to speed up convergence.

In the experiments in [73], nearest neighbours (NN) distance calculations were speeded up using distance transforms (DT) [25]. A DT is basically a discrete lookup table for NN distance values. If the data does not lie on a uniform grid, DT can only approximate the true NN distances. Thus, the global optimality guarantee of Go-ICP can be compromised by the approximations. It is stressed, however, that the original ideas of nested BnB and local refinement for speedup remain valid. In the experiments, the DT in Go-ICP was replaced with a kd-tree, which gives exact NN distances (since the point sets lie in 3D, using kd-tree is ideal). Thus, this implementation of Go-ICP can achieve the true global minimum.

Also, Glob-GM was compared against K-4PCS [71], which is a state-of-the-art approximate algorithm for point cloud registration. K-4PCS randomly generates and evaluates rigid transforms to find the best alignment. Briefly, 4 approximately coplanar points in $\mathcal{Y}$ are sampled and tested/matched against points from $\mathcal{X}$. The number of samples or iterations is determined from the overlap ratio (equivalent to inlier rate), which must be known beforehand or estimated on-the-fly (note that this information is not required in Go-ICP and Glob-GM). To improve efficiency, K-4PCS first subsamples the dense input point clouds by 3D keypoint detections. 3D keypoint detection was not further conducted for data reduction, since $\mathcal{X}$ and $\mathcal{Y}$ were already subsampled in conducted experiments (see below for details of the subsampling),

Two different settings were tested in this experiment:

- Full overlap: $\mathcal{X}$ is a subsample of $\mathcal{Y}$, i.e., $\mathcal{X} \subset \mathcal{Y}$. This is the same setting as that used in [73].
- Partial overlap: $\mathcal{X}$ and $\mathcal{Y}$ are two different scans, thus not all the points in $\mathcal{X}$ have a match in $\mathcal{Y}$.

Based on the objects used in Section 4.6.1, the data for the above two settings was created as follows.

For the full overlap scenario, for each object, one of the scans was selected and downsampled to obtain $\mathcal{Y}$. $\mathcal{X}$ is created as a sampled region of 100 points from $\mathcal{Y}$. For the underground mine dataset, the above procedure was performed for each individual scan, thus yielding six pairs of $\mathcal{X}$ and $\mathcal{Y}$ which are named *mine-1* to *mine-6*. Figure 4.11 shows the data in their initial (unregistered) poses.

For the partial overlap scenario, the point cloud pairs $\mathcal{X}$ and $\mathcal{Y}$ were simply downsampled versions of the original $\mathcal{V}_1$ and $\mathcal{V}_2$ used in Section 4.6.1. See Figure 4.12 for the resulting data and their initial (unregistered) poses.

To avoid excessive runtimes, $\mathcal{X}$ was fixed between 400 and 1200 points for the full overlap case and between 180 and 410 points for the partial overlap scenario. The sizes of $\mathcal{X}$ and $\mathcal{Y}$ are listed in Columns 2 and 3 of Tables 4.2 and 4.3 for both settings. Each point cloud set was uniformly scaled to fit the cube $[-50,\ 50]^3$. Also, $\mathcal{X}$ and $\mathcal{Y}$ for each object were translated such that their respective centroids coincide with the origin. The chosen initial unregistered poses were ensured to be far from the globally optimal solution.

The following variants of the Algorithm 4.3 were compared against Go-ICP and K-4PCS:

- Glob-GM-N: Algorithm 4.3 with local refinment (Step 10) disabled.

- Glob-GM: Algorithm 4.3 with local refinement.
- Glob-GM-N-ML, Glob-GM-ML: Variants of the above methods with the usage of matchlists in the rotation search.

Note that matchlists cannot be easily applied in Go-ICP, since each point $\mathbf{x}_i$ in $\mathcal{X}$ must always be matched to a nearest point in $\mathcal{Y}$ (see Section 4.3.4).

For Glob-GM and variants, the matching threshold ϵ was chosen as half of the cell grid side used during the downsampling step; see Column 4 in Tables 4.2 and 4.3 for the actual values. For Go-ICP, following the experiment in [73], the algorithm was terminated when the difference between the upper and lower bounds is $\leq \sqrt{0.05}$.

The suggested setting for the 4 thresholds (δ_1, δ_2, δ_3, δ_4) of K-4CPS in [71] is

$$\delta_1 = \delta_3 = 4\tau; \qquad \delta_2 = \tau; \qquad \delta_4 = \rho^2, \tag{4.39}$$

where τ is related to the size of the cell grid used for downsampling, and ρ is the point cloud density.

Two variants of K-4PCS were designed by changing the method settings. The first variant K-4PCS-Quick was tuned to return fast approximate results, while the second variant K-4PCS-Quality was tuned to obtain high quality results. Specifically, parameters were chosen as follows:

- K-4PCS-Quick: K-4PCS with the proposed setting for $\tau = 2\epsilon$.
- K-4PCS-Quality: K-4PCS with $\tau = \epsilon$ and $\delta_4 = \epsilon$.

Note that under experiment settings ϵ is taken as the half of the cell grid side used for downsampling (see Column 4 in Tables 4.2 and 4.3 for used ϵ values). Since K-4PCS is randomised in nature, the median of 10 runs of each variant is reported.

Tables 4.2 and 4.3 report the runtimes and quality metrics for the optimised alignment. A timeout of 5 hours (18000 seconds) was imposed for all methods—if a method cannot terminate successfully within the time limit, the result is marked with a '-' in the tables. For comprehensive benchmarking, four quality metrics are used:

- geometric matching value (4.35).
- angular error (ang. err.) of the optimised rotation with respect to the ground truth rotation.

- translation error (tr. err.) with respect to the ground truth translation.
- RMS error (cost function minimized by ICP).

In the full overlap setting, Glob-GM and variants predictably obtained the alignment with the highest possible quality value ($Q^* = 100$, since $|\mathcal{X}| = 100$ for all data). Go-ICP also expectedly found the result with $\leq \sqrt{0.05}$ RMS error. As expected, the quality of K-4PCS-Quality is better than K-4PCS-Quick, at the cost of longer runtimes. In fact, in the case of full overlap, K-4PCS-Quality is as accurate as the globally optimal methods—it is stressed, however, that unlike the globally optimal algorithms, K-4PCS-Quality cannot certify optimality of its results. Among the Glob-GM variants, clearly the usage of local refinement and matchlists provide significant speedups. The results also confirm that the partial overlap setting is more challenging than the full overlap setting. See Figures 4.11 and 4.12 for the globally optimal registration by Glob-GM. In the partial overlap setting, although K-4PCS-Quality was more accurate than K-4PCS-Quick, it was not able to correctly align for all datasets, e.g., no acceptable alignments were obtained for the mine objects as shown in Figure 4.13.

Comparing Go-ICP and the Glob-GM variants, it is evident that the latter is much more efficient than the former. In fact, in the partial overlap setting, Go-ICP did not finish executing within the time limit for all data. A major factor behind the superior efficiency of Glob-GM is a faster rotation search kernel, which is enabled by the proposed tighter rotational bounding function and its efficient evaluation based on stereographic projection and circular R-trees. The usage of matchlists also contributes to more efficient optimisation.

4.6.2.1 Convergence of BnB algorithm

To illustrate the convergence of Algorithm 4.3, Figure 4.10 plots the evolution of the upper and lower bounds during the registration of the bunny dataset. The time instances where a result is updated and then refined by the local method (Loc-GM) are also plotted. The result clearly affirms the ability of the local refinement idea of Yang et al. [73] to speed up BnB convergence. Figure 4.10 also shows that, similar to all BnB methods, the bulk of the time in Algorithm 4.3 is spent on waiting for the gap between the bounds to reduce to zero, even if the globally optimal estimate has been obtained much earlier (at approximately the 1250-th iteration). Thus, for practical applications it is probably convenient to terminate Algorithm 4.3 much earlier by using a non-zero convergence gap (e.g., as mentioned earlier, Go-ICP is terminated when the lower and upper bounds differ by $\leq \sqrt{0.05}$).

	Full overlap (Glob-GM and variants)										
Dataset	$\|\mathcal{X}\|$	$\|\mathcal{Y}\|$	ϵ	Glob-GM-N time (s)	Glob-GM-N-ML time (s)	Glob-GM time (s)	Glob-GM-ML time (s)	Q^*	ang. err.	tr. err.	RMS
bunny	100	566	3.25	1600.42	1143.01	1721.08	995.46	100	3.06	2.75	2.07
armadillo	100	1162	2.00	921.80	730.51	589.19	496.77	100	4.04	3.35	1.27
dragon	100	691	2.75	902.85	532.11	823.00	498.59	100	7.31	4.13	1.82
buddha	100	843	2.00	5484.89	3778.08	327.48	265.01	100	4.93	2.73	1.09
parasaur	100	703	1.50	5516.18	4523.27	2372.94	1842.79	100	1.73	1.29	1.12
t-rex	100	743	2.00	11001.40	7557.27	57.43	44.96	100	6.50	2.46	1.38
chicken	100	714	2.00	4177.20	3131.83	4000.96	2960.39	100	3.06	1.71	1.33
mine-1	100	412	0.88	1515.23	777.15	1513.00	845.01	100	1.88	0.40	0.48
mine-2	100	649	1.25	305.21	166.64	148.53	83.97	100	0.77	1.14	1.04
mine-3	100	627	1.25	126.07	84.29	132.13	76.85	100	1.71	0.89	0.79
mine-4	100	678	1.00	1760.09	1219.15	1750.77	1240.67	100	3.06	0.98	0.61
mine-5	100	933	1.00	11777.50	7816.78	11440	7695.75	100	1.73	0.77	0.78
mine-6	100	496	1.00	4249.53	2091.67	4238.54	2079.69	100	1.68	0.76	0.75

	Full overlap (Go-ICP)							
Dataset	$\|\mathcal{X}\|$	$\|\mathcal{Y}\|$	ϵ	Go-ICP				
				time (s)	Q	ang. err.	tr. err.	RMS
bunny	100	566	3.25	9364.16	100	1.7E-4	6.8E-5	3.2E-5
armadillo	100	1162	2.00	6321.26	100	4.5E-4	2.4E-4	3.0E-5
dragon	100	691	2.75	4314.57	100	3.5E-4	1.8E-4	2.3E-5
buddha	100	843	2.00	2027.46	100	1.4E-4	8.9E-5	1.4E-5
parasaur	100	703	1.50	5171.75	100	1.8E-4	1.1E-4	1.2E-5
t-rex	100	743	2.00	14137.80	100	4.4E-4	8.9E-5	1.8E-5
chicken	100	714	2.00	9936.01	100	2.2E-4	1.1E-4	1.7E-5
mine-1	100	412	0.88	1270.40	100	1.3E-4	2.0E-5	1.0E-5
mine-2	100	649	1.25	-	-	-	-	-
mine-3	100	627	1.25	12496.60	100	1.1E-4	5.4E-5	1.4E-5
mine-4	100	678	1.00	14723.70	100	5.7E-5	7.8E-5	1.9E-5
mine-5	100	933	1.00	12189.10	100	7.7E-5	6.2E-6	6.0E-6
mine-6	100	496	1.00	-	-	-	-	-

	Full overlap (K-4PCS variants)													
Dataset	$\|\mathcal{X}\|$	$\|\mathcal{Y}\|$	ϵ	K-4PCS-Quick					K-4PCS-Quality				
				time (s)	Q	ang. err.	tr. err.	RMS	time (s)	Q	ang. err.	tr. err.	RMS
bunny	100	566	3.25	32.15	16	142.38	23.58	48.08	143.35	38	145.75	29.69	46.74
armadillo	100	1162	2.00	21.01	33	161.77	62.58	48.75	1277.53	100	1.4E-3	8.7E-4	5.0E-4
dragon	100	691	2.75	0.04	84	4.68	2.68	2.62	23.13	100	1.2E-3	6.8E-4	2.8E-4
buddha	100	843	2.00	0.43	58	9.62	4.40	3.45	33.82	100	1.9E-3	9.1E-4	4.9E-4
parasaur	100	703	1.50	1.03	66	11.81	3.43	2.91	202.19	100	1.2E-3	4.1E-4	2.7E-4
t-rex	100	743	2.00	0.35	69	9.18	3.91	2.21	11.25	100	3.2E-3	4.7E-4	5.8E-4
chicken	100	714	2.00	0.37	55	42.45	2.17	6.42	18.11	100	9.2E-4	5.0E-4	3.6E-4
mine-1	100	412	0.88	0.97	100	1.71	0.46	0.47	17.86	100	4.6E-4	2.4E-4	2.2E-4
mine-2	100	649	1.25	0.30	82	6.26	0.24	1.28	25.14	100	1.8E-4	2.6E-4	2.7E-4
mine-3	100	627	1.25	0.23	91	3.12	0.65	1.15	2.03	100	1.2E-3	5.9E-4	6.1E-4
mine-4	100	678	1.00	0.03	100	3.0E-3	1.8E-3	7.0E-4	1.47	100	1.6E-3	1.1E-3	4.3E-4
mine-5	100	933	1.00	2.42	100	1.8E-3	2.0E-4	3.6E-4	119.00	100	6.1E-4	2.2E-4	2.9E-4
mine-6	100	496	1.00	0.17	99	2.16	0.44	0.36	11.24	100	2.0E-3	4.5E-4	5.0E-4

TABLE 4.2: Comparing performance of 3D registration methods on point clouds with full overlap. Glob-GM: Algorithm 4.3, Glob-GM-N: Algorithm 4.3 w/o local refinement, Glob-GM-ML and Glob-GM-N-ML: variants of the above with matchlists, K-4PCS-Quick: K-4CPS optimised for fast approximate solutions, K-4PCS-Quality: K-4CPS optimised for quality.

	Partial overlap (Glob-GM and variants)											
Dataset	$\lvert\mathcal{X}\rvert$	$\lvert\mathcal{Y}\rvert$	ϵ	Glob-GM-N time (s)	Glob-GM-N-ML time (s)	Glob-GM time (s)	Glob-GM-ML time (s)	Q^*	ang. err	tr. err	RMS
bunny	359	340	2.75	-	16576.50	-	14545.80	176	1.22	0.76	1.89
armadillo	281	276	2.00	7082.31	3469.95	6926.02	3427.36	211	1.19	0.86	1.41
dragon	358	343	2.75	-	6358.54	-	5987.86	305	0.52	0.87	1.76
buddha	232	199	2.75	-	9462.10	-	9302.98	161	11.78	2.42	3.49
parasauro	371	311	1.50	6414.84	2670.91	5479.75	2271.37	249	0.32	0.37	1.12
t-rex	371	417	2.00	12438.00	5213.35	12121.80	4955.63	265	0.98	0.30	1.33
chicken	379	407	2.00	13807.70	5942.92	13813.40	6139.23	293	0.47	0.75	1.38
mine-a	305	195	1.00	-	9592.71	-	9206.10	129	5.89	0.35	0.78
mine-b	264	188	1.68	-	13647.90	-	12832.40	145	3.73	0.26	1.19
mine-c	235	218	1.50	11952.20	5638.28	11125.00	5199.41	164	1.59	0.23	0.99

	Partial overlap (Go-ICP)							
				Go-ICP				
Dataset	$\lvert\mathcal{X}\rvert$	$\lvert\mathcal{Y}\rvert$	ϵ	time (s)	Q	ang. err.	tr. err.	RMS
bunny	359	340	2.75	-	-	-	-	-
armadillo	281	276	2.00	-	-	-	-	-
dragon	358	343	2.75	-	-	-	-	-
buddha	232	199	2.75	-	-	-	-	-
parasauro	371	311	1.50	-	-	-	-	-
t-rex	371	417	2.00	-	-	-	-	-
chicken	379	407	2.00	-	-	-	-	-
mine-a	305	195	1.00	-	-	-	-	-
mine-b	264	188	1.68	-	-	-	-	-
mine-c	235	218	1.50	-	-	-	-	-

	Partial overlap (K-4PCS variants)													
				K-4PCS-Quick					K-4PCS-Quality				
Dataset	$\lvert\mathcal{X}\rvert$	$\lvert\mathcal{Y}\rvert$	ϵ	time (s)	Q	ang. err.	tr. err.	RMS	time (s)	Q	ang. err.	tr. err.	RMS
bunny	359	340	2.75	94.51	62	13.09	8.10	9.70	1212.46	162	0.91	1.80	2.22
armadillo	281	276	2.00	1.04	132	3.38	1.72	2.61	28.22	192	0.55	0.48	1.21
dragon	358	343	2.75	0.72	241	4.67	1.68	3.14	36.92	291	1.13	0.73	1.71
buddha	232	199	2.75	0.31	83	48.94	7.48	12.05	11.88	140	4.16	0.71	2.05
parasauro	371	311	1.50	1.17	155	2.42	1.58	1.91	12.36	227	0.88	0.45	1.07
t-rex	371	417	2.00	7.36	163	4.38	3.05	3.82	206.14	232	0.97	0.62	1.54
chicken	379	407	2.00	6.01	129	4.38	3.12	3.79	168.59	276	1.06	0.47	1.37
mine-a	305	195	1.00	35.58	85	59.24	9.70	16.26	416.48	86	59.21	10.68	18.05
mine-b	264	188	1.68	3.99	82	4.17	9.15	9.25	62.45	99	7.37	11.44	11.59
mine-c	235	218	1.50	5.08	80	176.06	7.82	9.44	109.83	145	3.82	9.73	9.79

Table 4.3: Comparing performance of 3D registration methods on point clouds with partial overlap. Glob-GM: Algorithm 4.3, Glob-GM-N: Algorithm 4.3 w/o local refinement, Glob-GM-ML and Glob-GM-N-ML: variants of the above with matchlists, K-4PCS-Quick: K-4CPS optimised for fast approximate solutions, K-4PCS-Quality: K-4CPS optimised for quality.

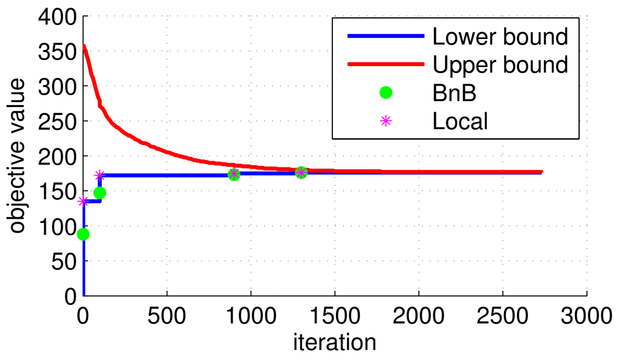


FIGURE 4.10: Evolution of upper and lower bounds as a function of iteration count in the Glob-GM method (Algorithm 4.3). The time instances where a result is updated and then refined by the local method (Loc-GM) are also plotted. The algorithm is terminated only when the upper bound equals the lower bound (at ≈ the 2750-th iteration).

4.7 Summary

This chapter presented an efficient BnB algorithm for 3D rotation search (Algorithm 4.1) that solves the GM criterion.

Of significant importance on runtime is the tightness of the bounding function. The proposed bounding function (4.11) is provably tighter than previously available bounds. A very efficient algorithm to evaluate the bound based on stereographic projections and R-trees was also presented. The proposed globally optimal rotation search algorithm was shown to be an order of magnitude faster than the original BnB algorithm of Breuel [11].

This chapter also presented a globally optimal 6 DoF point cloud registration algorithm (Algorithm 4.3) that encapsulates the presented rotation search method. Experimental results demonstrate the superior efficiency of this algorithm for point cloud registration, owing to a faster rotation search kernel.

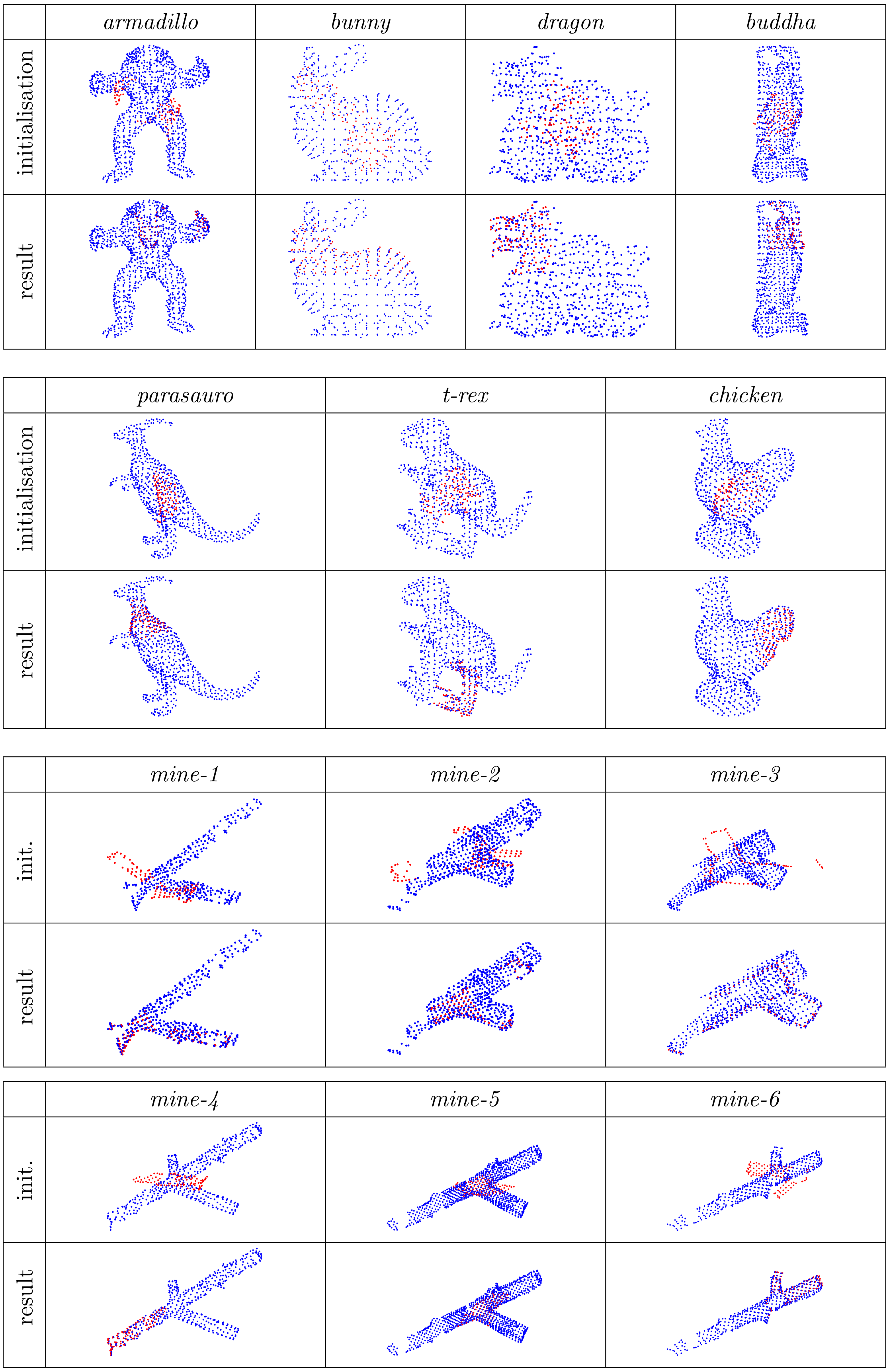
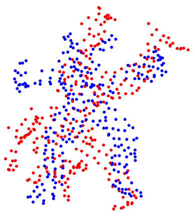
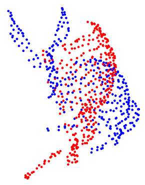
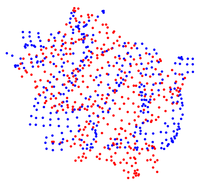
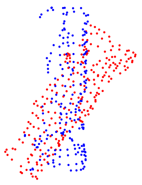
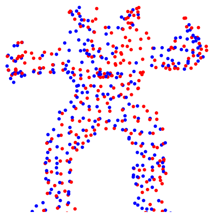
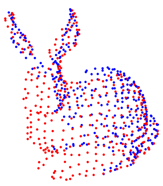
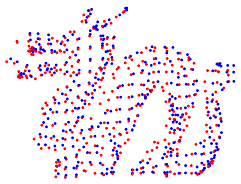
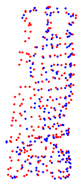
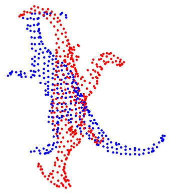
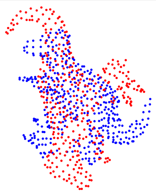
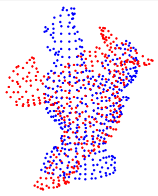
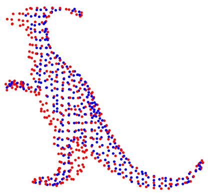
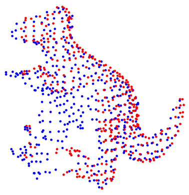
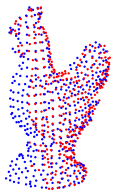


Figure 4.11: Initial poses of point clouds and globally optimal results by Glob-GM under the full overlap scenario.

	armadillo	*bunny*	*dragon*	*buddha*
initialisation				
result				

	parasauro	*t-rex*	*chicken*
initialisation			
result			

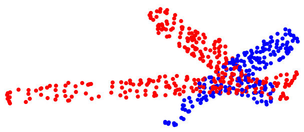
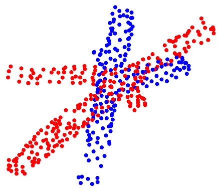
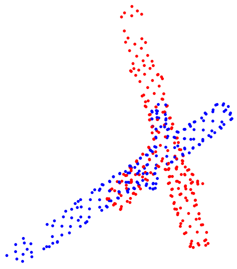
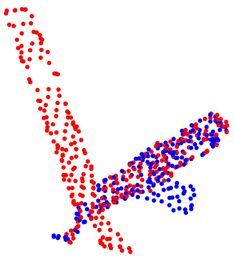
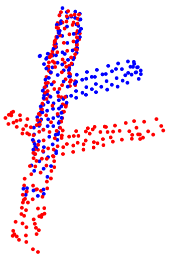
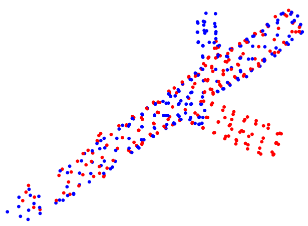
	mine-a	*mine-b*	*mine-c*
initialisation			
result			

FIGURE 4.12: Initial poses of point clouds and globally optimal results by Glob-GM under the partial overlap scenario.

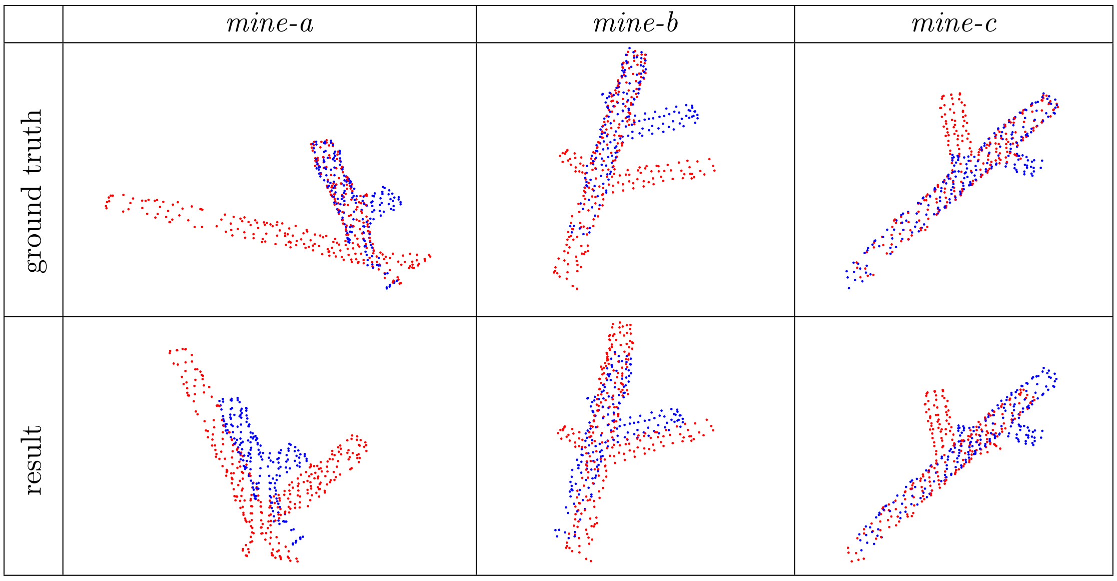


Figure 4.13: Ground truth and results of K-4PCS-Quality over the mining dataset under the partial overlap scenario.

Chapter 5

Guaranteed Outlier Removal for Rotation Search

5.1 Introduction

The previous two chapters solved rotation search problems of Type 2 under the GM criterion by using BnB, which guarantees global optimality. However, the runtime of BnB can be prohibitive on large point sets. Although real-time performance is achievable for 1D rotation search (see Chapter 3), longer runtimes were needed for 3D rotation search (see Chapter 4).

To reduce the problem's difficulty, rotations can be estimated from point correspondences, i.e., by solving a problem of Type 1 (see page 11). However, BnB can be inefficient when a large proportion of the point correspondences are outliers. This was observed in the experiments in Chapter 4.

To estimate rotations robustly in the presence of outlying correspondences, rotation search can be posed as a CSM problem (see Section 2.4.4). Using the angular distance $\angle(\cdot,\cdot)$ the CSM formulation for rotation search is

$$\begin{aligned} &\underset{\mathbf{R},\, \mathcal{I}\subseteq\mathcal{H}}{\text{maximize}} && |\mathcal{I}| \\ &\text{subject to} && \angle(\mathbf{R}\mathbf{x}_i, \mathbf{y}_i) \leq \epsilon,\ \forall i \in \mathcal{I}, \end{aligned} \tag{5.1}$$

where agreement is up to the inlier threshold ϵ. Here, $\mathcal{H} = \{1, \dots, N\}_{i=1}^{N}$ indexes the set of all point matches $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$. The optimal $\mathbf{R}^*$ is consistent with the largest possible subset $\mathcal{I}^* \subseteq \mathcal{H}$ of the data. Note that $\mathbf{R}^*$ can be easily found given $\mathcal{I}^*$, and vice versa. Thus, $\mathcal{I}^*$ or $\mathbf{R}^*$ can be referred as the solution without ambiguity.

RANSAC [24] (see Section 2.4.4.1) can be applied to approximately solve (5.1). Although RANSAC is very efficient, in general it does not provide the optimal solution $\mathcal{I}^*$. Formally, let $\tilde{\mathcal{I}} \subseteq \mathcal{H}$ be the result of RANSAC. Then $|\tilde{\mathcal{I}}| \leq |\mathcal{I}^*|$, and in general $\tilde{\mathcal{I}} \nsubseteq \mathcal{I}^*$, i.e., genuine inliers may be discarded.

Unlike RANSAC, global algorithms such as BnB and polynomial-time methods [52, 19] (see Section 2.4.4.2) are guaranteed to find the globally optimal result. A general weakness of global algorithms, however, is their high computational cost, especially for data with large sizes N and high outlier contamination rates; see Section 2.4.4.2 for a discussion on the cost of solving polynomial-time methods.

This chapter presents a novel ***g**uaranteed* **o**utlier **re**moval technique for rotation search (GORE). Specifically, GORE is able to reduce $\mathcal{H}$ to a subset $\mathcal{H}'$ of point matches, in a way that any $(\mathbf{x}_i, \mathbf{y}_i)$ discarded by reducing $\mathcal{H}$ to $\mathcal{H}'$ is a genuine outlier, i.e., any $(\mathbf{x}_i, \mathbf{y}_i)$ that is removed does not belong to $\mathcal{I}^*$. More formally, GORE ensures that $\mathcal{I}^* \subseteq \mathcal{H}' \subseteq \mathcal{H}$, which is a result RANSAC cannot guarantee.

GORE functions as an efficient *preprocessor* to the rotation search problem (5.1). Based on simple geometric operations, GORE is deterministic and fast. By aggressively reducing the population of true outliers (almost 90% can be eliminated), GORE significantly accelerates BnB. For example, using GORE before BnB reduces the overall runtime by an order of magnitude. Note that the global solution to the reduced data $\mathcal{H}'$ equals the global solution $\mathcal{I}^*$ to the original $\mathcal{H}$.

This chapter is organised as follows: Section 5.2 presents a guaranteed outlier removal method. Section 5.3 presents this chapter's main contribution; a tight lower bound that allows efficient removal of true outliers. Section 5.4 describes the main algorithm of GORE. Experiments are presented in Section 5.5 and Section 5.6 summarises this chapter.

5.2 Guaranteed outlier removal

Using the angular distance renders the norm of the points irrelevant. Henceforth, all the points are assumed to have unit norm. The rotation search problem (5.1) can be rewritten as

$$\underset{k \in \mathcal{H}}{\text{maximize}} \quad f_k, \tag{5.2}$$

where f_k is defined as the maximum objective value of the subproblem P_k, with $k = 1, \dots, N$:

$$\begin{aligned} \underset{\mathbf{R}_k,\, \mathcal{I}_k \subseteq \mathcal{H} \setminus \{k\}}{\text{maximize}} \quad & |\mathcal{I}_k| + 1 \\ \text{subject to} \quad & \angle(\mathbf{R}_k \mathbf{x}_i, \mathbf{y}_i) \le \epsilon,\ \forall i \in \mathcal{I}_k, \\ & \angle(\mathbf{R}_k \mathbf{x}_k, \mathbf{y}_k) \le \epsilon. \end{aligned} \tag{P_k}$$

In words, P_k seeks the rotation $\mathbf{R}_k$ that agrees with as many of the correspondences $(\mathbf{x}_k, \mathbf{y}_k)$ as possible, given that $\mathbf{R}_k$ *must align* $(\mathbf{x}_k, \mathbf{y}_k)$. The reformulation in (5.2) does not make the original problem (5.1) any easier—its utility derives from clarifying how an upper bound on f_k allows to identify outliers.

Let $l \le |\mathcal{I}^*|$ be a lower bound for the solution of the rotation search problem (5.1). The presented outlier removal technique depends on the ability to calculate an upper bound $\hat{f}_k$ for the result of each P_k, i.e., $\hat{f}_k \ge f_k$. Given the lower and upper bound values, the following result can be established.

Theorem 5.1. *If $\hat{f}_k < l$, then $(\mathbf{x}_k, \mathbf{y}_k)$ is a true outlier, i.e., k does not exist in the solution $\mathcal{I}^*$ to (5.1).*

Proof. The proof is by contradiction. If k is in $\mathcal{I}^*$, then $f_k = |\mathcal{I}^*|$. However, if $\hat{f}_k < l$, then $f_k < l \le |\mathcal{I}^*|$, which contradicts the previous condition. Hence, k cannot exist in $\mathcal{I}^*$. ☐

This chapter's main algorithm (Section 5.4) applies Proposition 5.1 iteratively for $k = 1, \dots, N$ to remove outliers. The main contribution in this chapter is an efficient algorithm to calculate a tight upper bound $\hat{f}_k$ for P_k (Section 5.3) for each k. As a by-product, the upper bound algorithm also computes a tight lower bound l for (5.1) to enable efficient removal of true outliers.

5.3 Efficient algorithm for upper bound

Recall that any candidate rotation $\mathbf{R}_k$ to solve P_k must bring $\mathbf{x}_k$ within angular distance ϵ from $\mathbf{y}_k$, i.e.,

$$\angle(\mathbf{R}_k \mathbf{x}_k, \mathbf{y}_k) \le \epsilon. \tag{5.3}$$

$\mathbf{R}_k$ can be interpreted by decomposing it into two rotations

$$\mathbf{R}_k = \mathbf{A}\mathbf{B}, \tag{5.4}$$

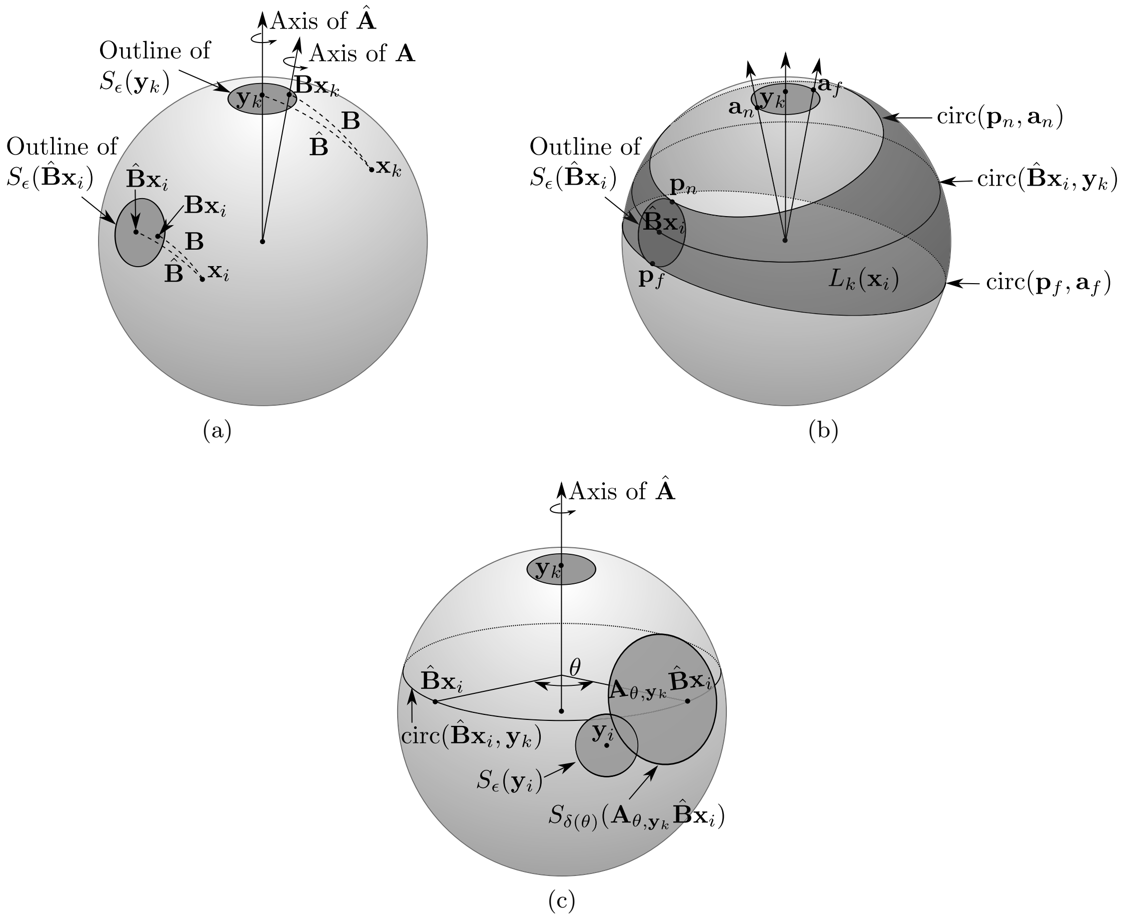

Axis of $\hat{\mathbf{A}}$
Axis of $\mathbf{A}$
Outline of $S_\epsilon(\mathbf{y}_k)$
$\mathbf{y}_k$
$\mathbf{B}\mathbf{x}_k$
$\mathbf{B}$
$\hat{\mathbf{B}}$
$\mathbf{x}_k$
Outline of $S_\epsilon(\hat{\mathbf{B}}\mathbf{x}_i)$
$\hat{\mathbf{B}}\mathbf{x}_i$
$\mathbf{B}\mathbf{x}_i$
$\mathbf{B}$
$\hat{\mathbf{B}}$
$\mathbf{x}_i$
(a)
$\mathbf{a}_n$
$\mathbf{y}_k$
$\mathbf{a}_f$
$\text{circ}(\mathbf{p}_n, \mathbf{a}_n)$
Outline of $S_\epsilon(\hat{\mathbf{B}}\mathbf{x}_i)$
$\mathbf{p}_n$
$\text{circ}(\hat{\mathbf{B}}\mathbf{x}_i, \mathbf{y}_k)$
$\hat{\mathbf{B}}\mathbf{x}_i$
$\mathbf{p}_f$
$L_k(\mathbf{x}_i)$
$\text{circ}(\mathbf{p}_f, \mathbf{a}_f)$
(b)
Axis of $\hat{\mathbf{A}}$
$\mathbf{y}_k$
θ
$\hat{\mathbf{B}}\mathbf{x}_i$
$\mathbf{A}_{\theta,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i$
$\text{circ}(\hat{\mathbf{B}}\mathbf{x}_i, \mathbf{y}_k)$
$\mathbf{y}_i$
$S_\epsilon(\mathbf{y}_i)$
$S_{\delta(\theta)}(\mathbf{A}_{\theta,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i)$
(c)

FIGURE 5.1: Geometry interpretations. (a) Interpreting rotation $\mathbf{R}_k$ according to (5.4). (b) The uncertainty region $L_k(\mathbf{x}_i)$ (5.15). (c) This figure shows $S_{\delta(\theta)}(\mathbf{A}_{\theta,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i)$ intersecting with $S_\epsilon(\mathbf{y}_i)$ for a particular θ. The aim is to find a bounding interval $\Theta_i \subset [-\pi, \pi]$ on θ for which the intersection is non-empty.

where $\mathbf{B}$ is defined as a rotation that honors the condition

$$\angle(\mathbf{B}\mathbf{x}_k, \mathbf{y}_k) \le \epsilon, \tag{5.5}$$

and $\mathbf{A}$ as a rotation about axis $\mathbf{B}\mathbf{x}_k$. Since $\mathbf{A}$ leaves $\mathbf{B}\mathbf{x}_k$ unchanged, the condition (5.5) and hence constraint (5.3) are always satisfied. Figure 5.1a illustrates this interpretation.

Solving P_k thus amounts to finding the combination of the rotation $\mathbf{B}$ (a 2 DoF problem, given (5.5)) and the rotation angle of $\mathbf{A}$ (a total of 3 DoF) that maximises the objective.

5.3.1 The ideal case

In the absence of noise and outliers, $\mathbf{x}_i$ can be aligned exactly with $\mathbf{y}_i$ for all i. Based on (5.4), the rotation that solves P_k under this ideal case is denoted as

$$\hat{\mathbf{R}}_k = \hat{\mathbf{A}}\hat{\mathbf{B}}, \tag{5.6}$$

which can be solved as follows (refer also to Figure 5.1a). First, find a rotation $\hat{\mathbf{B}}$ that aligns $\mathbf{x}_k$ exactly with $\mathbf{y}_k$, i.e.,

$$\hat{\mathbf{B}}\mathbf{x}_k = \mathbf{y}_k. \tag{5.7}$$

For example, take $\hat{\mathbf{B}}$ as the rotation that maps $\mathbf{x}_k$ to $\mathbf{y}_k$ with the minimum geodesic motion. To solve for $\hat{\mathbf{A}}$, take any $i \neq k$, then find the angle $\hat{\theta}$ of rotation about axis $\hat{\mathbf{B}}\mathbf{x}_k$ that maps $\hat{\mathbf{B}}\mathbf{x}_i$ to $\mathbf{y}_i$. Then $\hat{\mathbf{A}} = \exp\left([\hat{\theta}\hat{\mathbf{B}}\mathbf{x}_k]_\times\right)$, where $\exp(\cdot)$ is the exponential map; see Section 2.2.2 for details. The above steps affirm that rotation estimation requires a minimum of two point matches (see Section 2.4.1).

5.3.2 Uncertainty bound

In the usual case, data is contaminated with noise and outliers. The aim of this section is to establish a bound on the position of $\mathbf{x}_i$ when acted upon by the set of feasible rotations $\mathbf{R}_k$, i.e., those that satisfy (5.3) for P_k.

The set of $\mathbf{B}$ that maintain (5.5) cause $\mathbf{B}\mathbf{x}_k$ to lie within a spherical region of angular radius ϵ centred at $\mathbf{y}_k$, i.e.,

$$\mathbf{B}\mathbf{x}_k \in S_\epsilon(\mathbf{y}_k), \tag{5.8}$$

$$\text{where } \; S_\epsilon(\mathbf{y}_k) := \{\mathbf{x} \in \mathbb{R}^3 \,|\, \angle(\mathbf{x}, \mathbf{y}_k) \leq \epsilon \text{ and } \|\mathbf{x}\| = 1\}. \tag{5.9}$$

Since $\mathbf{B}\mathbf{x}_k$ is the rotation axis of $\mathbf{A}$, the interior of $S_\epsilon(\mathbf{y}_k)$ also represents the set of possible rotation axes for $\mathbf{A}$. Further, for any $i \neq k$,

$$\angle(\mathbf{B}\mathbf{x}_i, \hat{\mathbf{B}}\mathbf{x}_i) = \angle(\mathbf{B}\mathbf{x}_k, \hat{\mathbf{B}}\mathbf{x}_k) \tag{5.10}$$

$$= \angle(\mathbf{B}\mathbf{x}_k, \mathbf{y}_k) \leq \epsilon, \tag{5.11}$$

where (5.10) is based on the fact that applying the same pair of rotations on different points will transport the points across the same angular distance. Hence, (5.11) also shows that the set of feasible $\mathbf{B}$ cause $\mathbf{B}\mathbf{x}_i$ to lie in a spherical region, i.e.,

$$\mathbf{B}\mathbf{x}_i \in S_\epsilon(\hat{\mathbf{B}}\mathbf{x}_i). \tag{5.12}$$

Figure 5.1a also shows $S_\epsilon(\mathbf{y}_k)$ and $S_\epsilon(\hat{\mathbf{B}}\mathbf{x}_i)$. The bound on $\mathbf{R}_k\mathbf{x}_i$ can thus be analysed based on these two regions.

To make explicit the dependence of $\mathbf{A}$ on a rotation axis $\mathbf{a}$ and angle θ, the notation $\mathbf{A}_{\theta,\mathbf{a}}$ is adopted, where

$$\mathbf{A}_{\theta,\mathbf{a}} = \exp\left([\theta\mathbf{a}]_\times\right). \tag{5.13}$$

Let $\mathbf{p}$ be an arbitrary unit-norm point. Define

$$\operatorname{circ}(\mathbf{p},\mathbf{a}) := \{\mathbf{A}_{\theta,\mathbf{a}}\mathbf{p} \mid \theta \in [-\pi,\pi]\} \tag{5.14}$$

as the circle traced by $\mathbf{p}$ when acted upon by rotation $\mathbf{A}_{\theta,\mathbf{a}}$ for all θ at a particular axis $\mathbf{a}$.

The set of possible positions of $\mathbf{R}_k\mathbf{x}_i$ is then defined by

$$L_k(\mathbf{x}_i) := \{\operatorname{circ}(\mathbf{p},\mathbf{a}) \mid \mathbf{p} \in S_\epsilon(\hat{\mathbf{B}}\mathbf{x}_i), \mathbf{a} \in S_\epsilon(\mathbf{y}_k)\}. \tag{5.15}$$

Figure 5.1b illustrates this feasible region, which exists on the unit-sphere. The region is bounded within the two circles

$$\operatorname{circ}(\mathbf{p}_n,\mathbf{a}_n) \ \text{ and } \ \operatorname{circ}(\mathbf{p}_f,\mathbf{a}_f), \tag{5.16}$$

which are highlighted in Figure 5.1b. Intuitively, $\mathbf{p}_n$ and $\mathbf{a}_n$ (resp. $\mathbf{p}_f$ and $\mathbf{a}_f$) are the closest (resp. farthest) pair of points from $S_\epsilon(\hat{\mathbf{B}}\mathbf{x}_i)$ and $S_\epsilon(\mathbf{y}_k)$. Mathematically,

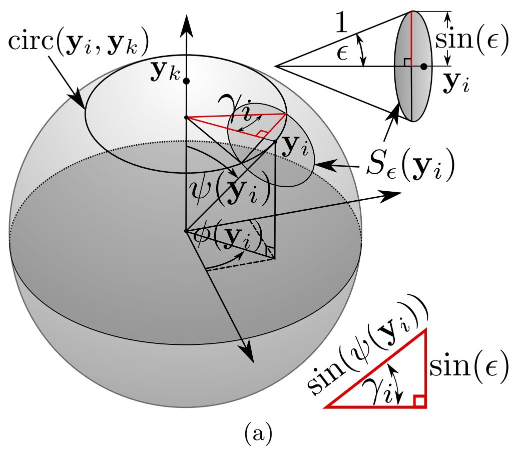
$$\mathbf{p}_n = \exp\left(\left[\epsilon\,\frac{\hat{\mathbf{B}}\mathbf{x}_i \times \mathbf{y}_k}{\|\hat{\mathbf{B}}\mathbf{x}_i \times \mathbf{y}_k\|}\right]_\times\right)\hat{\mathbf{B}}\mathbf{x}_i; \tag{5.17}$$

$$\mathbf{a}_n = \exp\left(\left[\epsilon\,\frac{\mathbf{y}_k \times \hat{\mathbf{B}}\mathbf{x}_i}{\|\mathbf{y}_k \times \hat{\mathbf{B}}\mathbf{x}_i\|}\right]_\times\right)\mathbf{y}_k; \tag{5.18}$$

$$\mathbf{p}_f = \exp\left(\left[\epsilon\,\frac{\mathbf{y}_k \times \hat{\mathbf{B}}\mathbf{x}_i}{\|\mathbf{y}_k \times \hat{\mathbf{B}}\mathbf{x}_i\|}\right]_\times\right)\hat{\mathbf{B}}\mathbf{x}_i; \text{ and} \tag{5.19}$$

$$\mathbf{a}_f = \exp\left(\left[\epsilon\,\frac{\hat{\mathbf{B}}\mathbf{x}_i \times \mathbf{y}_k}{\|\hat{\mathbf{B}}\mathbf{x}_i \times \mathbf{y}_k\|}\right]_\times\right)\mathbf{y}_k. \tag{5.20}$$

Note that if $\hat{\mathbf{B}}\mathbf{x}_i$ is antipodal to $\mathbf{y}_k$, the feasible region reduces to the spherical region $S_{3\epsilon}(\hat{\mathbf{B}}\mathbf{x}_i)$.



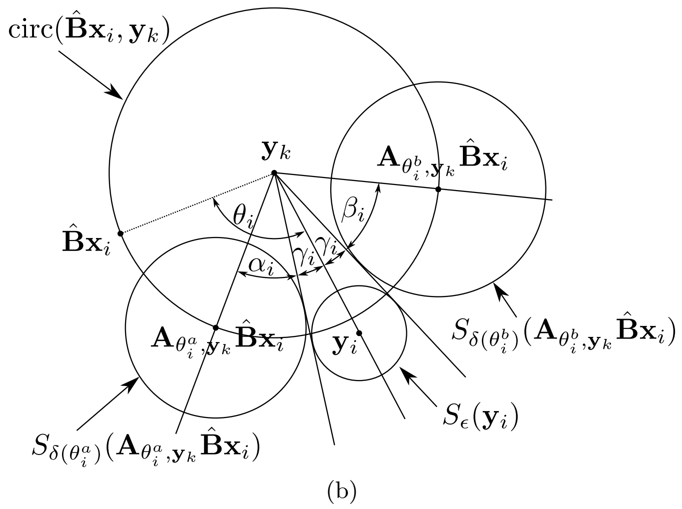

circ($\hat{\mathbf{B}}\mathbf{x}_i, \mathbf{y}_k$)
$\mathbf{y}_k$
$\mathbf{A}_{\theta_i^b,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i$
$\hat{\mathbf{B}}\mathbf{x}_i$
θ_i
β_i
γ_i
γ_i
α_i
$\mathbf{A}_{\theta_i^a,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i$
$\mathbf{y}_i$
$S_{\delta(\theta_i^b)}(\mathbf{A}_{\theta_i^b,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i)$
$S_\epsilon(\mathbf{y}_i)$
$S_{\delta(\theta_i^a)}(\mathbf{A}_{\theta_i^a,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i)$
(b)

FIGURE 5.2: Geometric considerations of the angular bounding interval. (a) Solving for γ_i in the red triangle. Its cathetus is half of the longest segment connecting points in $S_\epsilon(\mathbf{y}_i)$ and its hypotenuse is the radius of circ$(\mathbf{y}_i, \mathbf{y}_k)$. (b) To simplify the diagram and to aid intuition, the sphere in Figure 5.1 is stereographically projected to the 2D plane using the North Pole $(\mathbf{y}_k)$ as the projection pole. Recall that the stereographic projection preserves circles [49], thus the shapes of all the circles and spherical regions on the sphere are preserved. Note that stereographic projection is only for presentation and is not required in practice.

Result 1. For any $i \neq k$, if $S_\epsilon(\mathbf{y}_i)$ does not intersect with $L_k(\mathbf{x}_i)$, then $(\mathbf{x}_i, \mathbf{y}_i)$ cannot be aligned by any rotation $\mathbf{R}_k$ that satisfies (5.3). The correspondence $(\mathbf{x}_i, \mathbf{y}_i)$ can then be safely removed without affecting the result f_k of P_k.

5.3.3 Reducing the uncertainty

For each point match $(\mathbf{x}_i, \mathbf{y}_i)$ that survives the pruning by Result 1, its uncertainty bound (5.15) is reduced into an *angular interval.* This reduction is crucial for the efficient upper bound algorithm to be introduced in Section 5.3.4.

Consider rotating an arbitrary unit-norm point $\mathbf{p}$ with $\mathbf{A}_{\theta,\mathbf{u}}$ for a fixed angle θ and an axis $\mathbf{u} \in S_\epsilon(\mathbf{y}_k)$. The aim is to bound the possible locations of $\mathbf{A}_{\theta,\mathbf{u}}\mathbf{p}$ given the uncertainty in $\mathbf{u}$. To this end, it can be established that

$$\begin{aligned}\max_{\mathbf{u}\in S_\epsilon(\mathbf{y}_k)} \angle(\mathbf{A}_{\theta,\mathbf{u}}\mathbf{p}, \mathbf{A}_{\theta,\mathbf{y}_k}\mathbf{p}) &\leq \max_{\mathbf{u}\in S_\epsilon(\mathbf{y}_k)} \|\theta\mathbf{u} - \theta\mathbf{y}_k\|_2 \\ &= 2|\theta|\sin(\epsilon/2), \end{aligned} \tag{5.21}$$

where the first line is based on a well-known result of the axis-angle representation (see Lemma 2.32), and the second line occurs since $S_\epsilon(\mathbf{y}_k)$ has an angular radius of ϵ.

Now (5.21) is extended to accommodate the uncertainty of $\mathbf{p}$ itself as a point from $S_\epsilon(\hat{\mathbf{B}}\mathbf{x}_i)$:

$$\begin{aligned}&\max_{\substack{\mathbf{p}\in S_\epsilon(\hat{\mathbf{B}}\mathbf{x}_i)\\ \mathbf{u}\in S_\epsilon(\mathbf{y}_k)}} \angle(\mathbf{A}_{\theta,\mathbf{u}}\mathbf{p}, \mathbf{A}_{\theta,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i) \\ \leq &\max_{\substack{\mathbf{p}\in S_\epsilon(\hat{\mathbf{B}}\mathbf{x}_i)\\ \mathbf{u}\in S_\epsilon(\mathbf{y}_k)}} \angle(\mathbf{A}_{\theta,\mathbf{u}}\mathbf{p}, \mathbf{A}_{\theta,\mathbf{y}_k}\mathbf{p}) + \angle(\mathbf{A}_{\theta,\mathbf{y}_k}\mathbf{p}, \mathbf{A}_{\theta,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i) \\ \leq\, & 2|\theta|\sin(\epsilon/2) + \epsilon. \end{aligned} \tag{5.22}$$

The second line is due to the triangle inequality, while the third line applies (5.21) on the first term of the second line. Define

$$\delta(\theta) = 2|\theta|\sin(\epsilon/2) + \epsilon. \tag{5.23}$$

The inequality (5.22) states that for a fixed θ and for all $\mathbf{u} \in S_\epsilon(\mathbf{y}_k)$ and $\mathbf{B}\mathbf{x}_i \in S_\epsilon(\hat{\mathbf{B}}\mathbf{x}_i)$, the point $\mathbf{A}_{\theta,\mathbf{u}}\mathbf{B}\mathbf{x}_i$ lies in

$$S_{\delta(\theta)}(\mathbf{A}_{\theta,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i). \tag{5.24}$$

Figure 5.1c depicts this spherical region. Observe that for all $\theta \in [-\pi, \pi]$, the centre of the region lies in $\mathrm{circ}(\hat{\mathbf{B}}\mathbf{x}_i, \mathbf{y}_k)$. Intuitively, this is a circle of a fixed latitude on the globe when $\mathbf{y}_k$ is the North Pole. Further, the spherical region attains the largest angular radius at $\theta = \pm\pi$.

For a pair $(\mathbf{x}_i, \mathbf{y}_i)$, the idea is to obtain a bound Θ_i (an interval) on the range of θ that enable $\mathbf{A}_{\theta,\mathbf{u}}\mathbf{B}\mathbf{x}_i$ to align with $\mathbf{y}_i$, given the uncertainties $\mathbf{u} \in S_\epsilon(\mathbf{y}_k)$ and $\mathbf{B}\mathbf{x}_i \in S_\epsilon(\hat{\mathbf{B}}\mathbf{x}_i)$. This is analogous to seeking a bound on the θ that allows $S_{\delta(\theta)}(\mathbf{A}_{\theta,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i)$ to "touch" $S_\epsilon(\mathbf{y}_i)$; see Figure 5.1c.

Henceforth, concepts from the spherical coordinate system are used with reference to $\mathbf{y}_k$ as the North Pole.

5.3.3.1 Degenerate cases

If $\hat{\mathbf{B}}\mathbf{x}_i$ is close to $\mathbf{y}_k$, the North Pole may lie in $L_k(\mathbf{x}_i)$. If this occurs, Θ_i is taken as $[-\pi, \pi]$.

5.3.3.2 Non-degenerate cases

Define $\phi(\mathbf{y}_i)$ and $\psi(\mathbf{y}_i)$ respectively as the azimuth and inclination of $\mathbf{y}_i$. The spherical region $S_\epsilon(\mathbf{y}_i)$ is contained between the meridians $\phi(\mathbf{y}_i) - \gamma_i$ and $\phi(\mathbf{y}_i) + \gamma_i$, where

$$\gamma_i = \arcsin\left(\frac{\sin(\epsilon)}{\sin(\psi(\mathbf{y}_i))}\right) \tag{5.25}$$

following the geometric considerations in Figure 5.2a. Let $\theta_i \in [-\pi, \pi]$ be the rotation angle such that the point $\mathbf{A}_{\theta_i,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i$ is on the meridian $\phi(\mathbf{y}_i)$. Refer to Figure 5.2b.

Case 1: $\theta_i \in [0, \pi]$

This case is shown in Figure 5.2b. Define $\Theta_i = [\theta_i^a, \theta_i^b]$. The desired bounding interval Θ_i can be obtained by taking

$$\theta_i^a = \theta_i - \gamma_i - \alpha_i \quad \text{and} \quad \theta_i^b = \theta_i + \gamma_i + \beta_i, \tag{5.26}$$

where α_i is the largest value such that the spherical region

$$S_{\delta(\theta_i^a)}(\mathbf{A}_{\theta_i^a,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i) \tag{5.27}$$

still touches the meridian $\phi(\mathbf{y}_i) - \gamma_i$, and β_i is the largest value such that the spherical region

$$S_{\delta(\theta_i^b)}(\mathbf{A}_{\theta_i^b,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i) \tag{5.28}$$

still touches the meridian $\phi(\mathbf{y}_i) + \gamma_i$. Refer to Figure 5.2b. α_i and β_i must be found to determine Θ_i. From (5.23),

$$\delta(\theta_i^a) = 2|\theta_i - \gamma_i - \alpha_i| \sin(\epsilon/2) + \epsilon \quad \text{and} \tag{5.29}$$

$$\delta(\theta_i^b) = 2|\theta_i + \gamma_i + \beta_i| \sin(\epsilon/2) + \epsilon. \tag{5.30}$$

Applying the same geometric considerations in Figure 5.2a on the spherical regions (5.27) and (5.28),

$$\sin(\alpha_i) = \frac{\sin(\delta(\theta_i^a))}{\sin(\psi(\mathbf{x}_i))} \qquad \text{and} \qquad \sin(\beta_i) = \frac{\sin(\delta(\theta_i^b))}{\sin(\psi(\mathbf{x}_i))}. \tag{5.31}$$

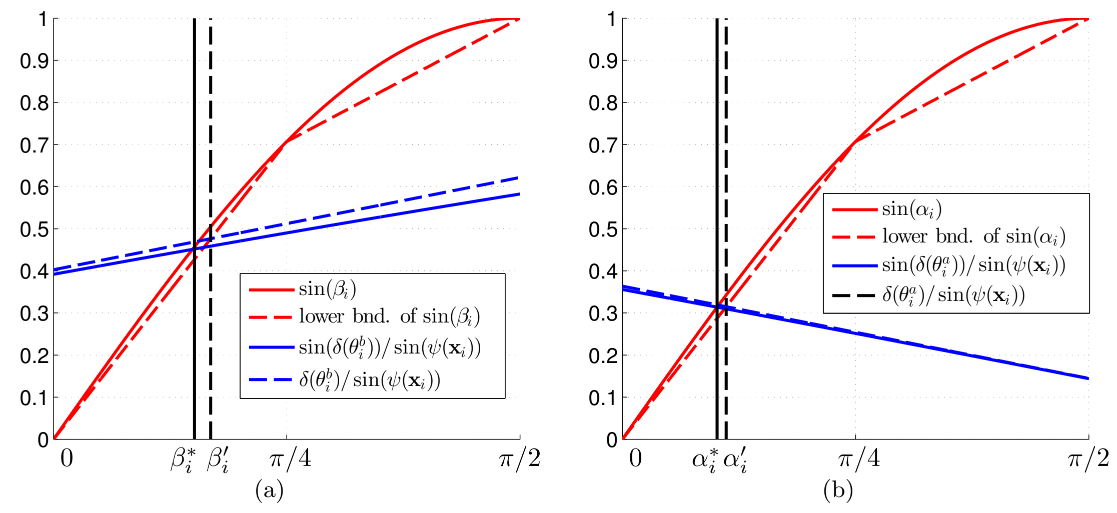


FIGURE 5.3: (a) Solving for β_i in (5.31) using the proposed linear approximations. (b) Solving for α_i in (5.31) using the proposed linear approximations. Note that the obtained solution α_i' and β_i' are always greater than the exact solution α_i^* and β_i^*, thus guaranteeing $[\theta_i^a, \theta_i^b]$ is a valid bounding interval.

Note that the functions on both sides of each equation have the unknowns α_i and β_i respectively.

Figure 5.3a plots the two sine functions $\sin(\beta_i)$ and $\sin(\delta(\theta_i^b))/\sin(\psi(\mathbf{x}_i))$. β_i is considered only to be in $[0, \pi/2]$, since the condition where the two functions do not intersect before $\beta_i \leq \pi/2$ corresponds to the degeneracies in Section 5.3.3.1; proof is given in Section 5.3.3.3. Further, since usually $\epsilon \ll \pi$, the period of the second sine function

$$\frac{2\pi}{2\sin(\epsilon/2)} \gg 2\pi \tag{5.32}$$

is much greater than 2π, thus explaining the almost linear trend of the second sine function for $\beta_i \in [0, \pi/2]$. A largely identical plot occurs for the functions involving α_i (see Figure 5.3b).

Analytically solving the equations in (5.31) is non-trivial. However, since all that is required is a bounding interval Θ_i, sine functions can be replaced with more amenable approximations that yield a valid bounding interval. An identical technique is used to solve for α_i and β_i respectively, thus here a solution is described only for β_i.

To solve for β_i, $\sin(\beta_i)$ is replaced with a lower-bounding two-piece linear function $g\colon [0, \pi/2] \to [0, 1]$ defined as

$$g(t) = \begin{cases} \dfrac{2\sqrt{2}}{\pi} t & \text{if } 0 \leq t < \pi/4 \\ \dfrac{2(2-\sqrt{2})}{\pi} t + \sqrt{2} - 1 & \text{if } \pi/4 \leq t < \pi/2. \end{cases} \tag{5.33}$$

See Figure 5.3a. Jordan's inequality

$$\frac{2}{\pi}t \leq \sin(t) \leq t \;\text{ for }\; t \leq \frac{\pi}{2} \tag{5.34}$$

is used to obtain an upper-bounding line to $\sin(\delta(\theta_i^b))/\sin(\psi(\mathbf{x}_i))$, which enables to replace the second sine function with

$$\frac{\delta(\theta_i^b)}{\sin(\psi(\mathbf{x}_i))} = \frac{2|\theta_i + \gamma_i + \beta_i|\sin(\epsilon/2) + \epsilon}{\sin(\psi(\mathbf{x}_i))}. \tag{5.35}$$

This upper-bounding line is legitimate for

$$2|\theta_i + \gamma_i + \beta_i|\sin(\epsilon/2) + \epsilon \leq \pi/2, \tag{5.36}$$

where in the worst case requires

$$2\pi\sin(\epsilon/2) + \epsilon \leq \pi/2 \tag{5.37}$$

or $\epsilon \leq \pi/(2\pi+2) \equiv 21.7°$, which is more than adequate for practical applications. Solving for β_i in the manner above allows to compute the upper limit θ_i^b in constant time.

Note that the resulting upper limit θ_i^b may extend beyond π; to "wrap around" the interval, $\Theta_i = [\theta_i^a, \theta_i^b]$ is broken into two connected intervals $[\theta_i^a, \pi]$ and $[\theta_i^b - 2\pi, -\pi]$.

Case 2: $\theta_i \in [-\pi, 0]$

Case 2 is simply a mirror of **Case 1** and the same steps apply with the "directions" reversed.

Result 2. For any $i \neq k$, if $S_\epsilon(\mathbf{y}_i)$ intersects with $L_k(\mathbf{x}_i)$, the range of angles θ such that $\angle(\mathbf{A}_{\theta,\mathbf{u}}\mathbf{B}\mathbf{x}_i, \mathbf{y}_i) \leq \epsilon$ for all $\mathbf{u} \in S_\epsilon(\mathbf{y}_k)$ and $\mathbf{B}\mathbf{x}_i \in S_\epsilon(\hat{\mathbf{B}}\mathbf{x}_i)$ is bounded by Θ_i computed according to Section 5.3.3.

5.3.3.3 Range of α_i and β_i

$\delta(\theta^a)$ and $\delta(\theta^b)$ are consequence of the bound result of Lemma 2.32. Thus, $\delta(\theta_i^a)$ (resp. $\delta(\theta_i^b)$) is maximised when $|\theta_i^a| = \pi$ (resp. $|\theta_i^b| = \pi$).

To solve (5.31) is equivalent to solving

$$\alpha_i = \arcsin\left(\frac{\sin(\delta(\theta_i^a))}{\sin(\psi(\mathbf{x}_i))}\right), \quad \beta_i = \arcsin\left(\frac{\sin(\delta(\theta_i^b))}{\sin(\psi(\mathbf{x}_i))}\right). \tag{5.38}$$

α_i and β_i have solution if $\psi(\mathbf{x}_i) > 0$ and

$$\sin(\delta(\theta_i^a)) \leq \sin(\psi(\mathbf{x}_i)) \qquad \text{and} \qquad \sin(\delta(\theta_i^b)) \leq \sin(\psi(\mathbf{x}_i)), \tag{5.39}$$

as arcsin yields non negative values when evaluated in $[0,1]$. Note that $\sin(\psi(\mathbf{x}_i)) \in [0,1]$ since $\psi(\mathbf{x}_i)$ takes values in $[0,\pi]$. Note also that $\delta(\theta_i^a)$, $\delta(\theta_i^b) \in [\epsilon, \pi/2]$ if $\epsilon < 21.7°$, then $\sin(\delta(\theta_i^a))$ and $\sin(\delta(\theta_i^b))$ take values in $(0,1]$. By using Jordan's inequality (5.34) in (5.39), the following bound over the position of $\mathbf{x}_i$ guarantees the existence of a solution to 5.38:

$$\lambda(\mathbf{x}_i) \geq \epsilon \frac{\pi(\pi+1)}{2} \equiv 6.506\epsilon, \tag{5.40}$$

where $\lambda(\mathbf{x}_i)$ is defined as the distance to the "closest pole" and taking values in $[0,\pi/2]$ as required by Jordan's inequality.

Result (5.40) shows that there exist a solution of the equations in (5.38) if $\mathbf{x}_i$ is not too close to the poles. Recall that rather than finding the exact solutions of (5.31), bounds of such exact solutions are obtained by intersecting linear approximations: (5.33) and (5.35). Thus, it is sufficient to check if the approximate method has solution to obtain a bounding interval Θ_i. If the approximate method does not have a solution then Θ_i is set to $[-\pi,\pi]$.

Lemma 5.2. *α and β are well defined in* $[0,\pi/2]$.

Proof. Let the outline of the spherical region $S_{\delta(\theta_i^b)}(\mathbf{A}_{\theta_i^b,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i)$ be infinitesimally close to either the North or the South Pole. In such a limiting case, the meridians containing $S_{\delta(\theta_i^b)}(\mathbf{A}_{\theta_i^b,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i)$ will be at azimuthal angle π apart, thus $\beta_i \leq \pi/2$. If either the North or the South Pole is contained in $S_{\delta(\theta_i^b)}(\mathbf{A}_{\theta_i^b,\mathbf{y}_k}\hat{\mathbf{B}}\mathbf{x}_i)$, then this spherical region cannot be bounded between two meridians since the spherical region can intersect points with azimuth in all the azimuthal domain $[0,2\pi]$. The same analysis can be done for α_i. □

5.3.3.4 Validity of the proposed method

The following lemma makes it possible to establish that the solution of intersecting $\sin(\delta(\theta_i^a))/\sin(\psi(\mathbf{x}_i))$ (resp. $\sin(\delta(\theta_i^b))/\sin(\psi(\mathbf{x}_i))$) with the sine function is upper bounded by the solution of intersecting $\delta(\theta_i^a)/\sin(\psi(\mathbf{x}_i))$ (resp. $\delta(\theta_i^b)/\sin(\psi(\mathbf{x}_i))$) with the sine function.

Lemma 5.3. *If $\hat{f}^a$ and $\hat{f}^b$ are upper bounding functions of $\sin(\delta(\theta_i^a))/\sin(\psi(\mathbf{x}_i))$ and $\sin(\delta(\theta_i^b))/\sin(\psi(\mathbf{x}_i))$, and $\hat{\alpha}_i$, $\hat{\beta}_i$ are such that $\hat{\alpha}_i = \arcsin(\hat{f}^a(\hat{\alpha}_i))$, $\hat{\beta}_i = \arcsin(\hat{f}^b(\hat{\beta}_i))$, then $\hat{\alpha}_i \geq \alpha_i^*$ and $\hat{\beta}_i \geq \beta_i^*$, where α_i^* and β_i^* are the solutions of* (5.31).

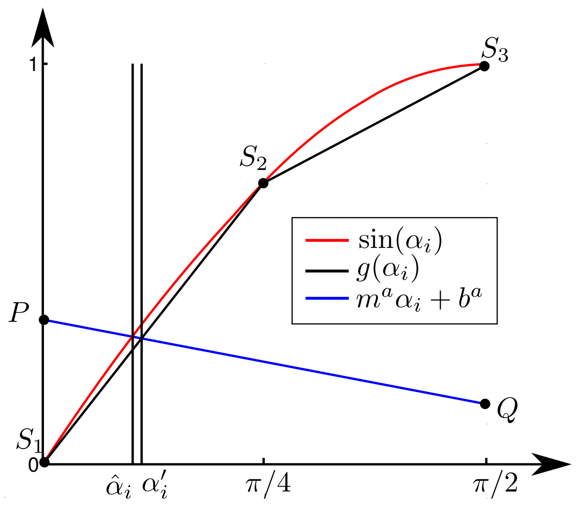


FIGURE 5.4: Geometry of solving α_i'.

Proof. Let f^a be the right hand function in the left equation in (5.38). Define $y_0^a := f^a(0)$

$$y_0^a = \arcsin\left(\frac{\sin\left(2(\gamma_i + \alpha_i)\sin(\epsilon/2) + \epsilon\right)}{\sin(\psi(\mathbf{x}_i))}\right) \tag{5.41}$$

$y_0^a > 0$ since, as will be shown, arcsin is evaluated in $(0, 1]$ raising values > 0. Since $\sin(\delta(\theta_i^a))/\sin(\psi(\mathbf{x}_i)) < 1$ if condition (5.40) is true, it is sufficient to show the argument in arcsin is > 0 to prove that $y_0^a > 0$. $\sin(2(\gamma_i + \alpha_i)\sin(\epsilon/2) + \epsilon) > 0$ if $2\pi\sin(\epsilon/2) < \pi$, which is a less restrictive condition over ϵ than in (5.37). $\sin(\psi(\mathbf{x}_i)) > 0$ since it is a necessary condition to (5.31) has a solution.

f^a intersects the identity at some α_i^* in $[0, \hat{\alpha}_i]$ since f^a is a continuous function lying at opposites sides of the identity at 0 and $\hat{\alpha}_i$:

$$f^a(0) = y_0^a > 0 \qquad \text{and} \qquad f^a(\hat{\alpha}_i) \leq \hat{f}^a(\hat{\alpha}_i) \leq \arcsin(\hat{f}^a(\hat{\alpha}_i)) = \hat{\alpha}_i, \tag{5.42}$$

since $\arcsin(t) \geq t$ in $[0, 1]$. The same analysis can be done to show that $\hat{\beta}_i \geq \beta_i^*$. □

The linear approximations for the right hand side of equations in (5.31) are

$$\hat{f}^a(\alpha_i) := \frac{\delta(\theta_i^a)}{\sin\psi(\mathbf{x}_i)} = m^a\alpha_i + b^a \quad \text{and} \quad \hat{f}^b(\beta_i) := \frac{\delta(\theta_i^b)}{\sin\psi(\mathbf{x}_i)} = m^b\beta_i + b^b, \tag{5.43}$$

where (m^a, b^a) and (m^b, b^b) are the parameters of the linear functions.

Define $\hat{\alpha}_i$ and $\hat{\beta}_i$ as the solutions of

$$\sin(\hat{\alpha}_i) = m^a\hat{\alpha}_i + b^a \qquad \text{and} \qquad \sin(\hat{\beta}_i) = m^b\hat{\beta}_i + b^b. \tag{5.44}$$

Define α_i' and β_i' as the solutions of

$$g(\alpha_i') = m^a\alpha_i' + b^a \qquad \text{and} \qquad (\beta_i') = m^b\beta_i' + b^b, \tag{5.45}$$

where g is the piece-wise linear function defined in (5.33).

From Lemma 5.3, $\hat{\alpha}_i \geq \alpha_i^*$ and $\hat{\beta}_i \geq \beta_i^*$. Here it is shown that $\alpha_i' \geq \hat{\alpha}_i$, $\beta_i' \geq \hat{\beta}_i$, and hence they are valid bounds. Geometrically, α_i' is at the intersection of a segment $\overline{PQ}$ with a segment $\overline{S_jS_{j+1}}$, where $P = (0, b^a)$ with $b^a \geq 0$, and segments $\{\overline{S_jS_{l+j}}\}_{l=1}^2$ defining g; see Figure 5.4. Recall g was defined to be a lower bounding function of sine. $\overline{PQ}$ intersects sine at the left than $\{\overline{S_jS_{j+1}}\}_{j=1}^2$.

Lemma 5.4. *α_i' and β_i' that solve (5.45) are such that $\alpha_i' \geq \alpha_i^*$ and $\beta_i' \geq \beta_i^*$.*

Proof. Proof $\alpha_i' \geq \alpha_i^*$ is given by showing that $\alpha_i' \geq \hat{\alpha}_i$. g is a lower bounding function of sin (see (5.33)). Then, for all line equations $\{p_j\alpha_i + q_j\}_{j=1}^2$ defining g

$$p_j\alpha_i + q_j \leq \sin(\alpha_i). \tag{5.46}$$

It follows from (5.44) and (5.46) that

$$p_j\hat{\alpha}_i + q_j \leq \sin(\hat{\alpha}_i) = m^a\hat{\alpha}_i + b^a, \ \forall j = 1, 2. \tag{5.47}$$

Finally, $\hat{\alpha}_i \leq \alpha_i'$ is obtained by replacing

$$b^a = (p_j - m^a)\alpha_i' + q_j \tag{5.48}$$

from (5.45) in (5.47):

$$p_j\hat{\alpha}_i + q_j \leq m^a\hat{\alpha}_i + (p_j - m^a)\alpha_i' + q_j \ \Leftrightarrow \ \hat{\alpha}_i \leq \alpha_i'. \tag{5.49}$$

Proving $\beta_i' \geq \beta_i^*$ can be done similarly. □

5.3.4 Interval stabbing

For problem P_k, on the input point matches that remain after pruning by the application of Result 1, Result 2 can be used to convert them into a set of angular intervals $\{\Theta_j\}$, where each $\Theta_j = [\theta_j^a, \theta_j^b]$. The aim is to find the largest number of point matches that can be aligned by the same rotation angle θ. More formally, this is the solution of

$$O_k = \underset{\theta \in [-\pi, \pi]}{\text{maximize}} \ \sum_j \left\lfloor \theta \in [\theta_j^a, \theta_j^b] \right\rfloor \tag{5.50}$$

where $\lfloor \cdot \rceil$ is an indicator function that returns 1 if the input predicate is true and 0 otherwise. This is the well-known *interval stabbing* problem, for which efficient deterministic algorithms exist [8, Chap. 10]. $\hat{f}_k := O_k + 1$ is taken as an upper bound to the solution f_k to P_k.

Theorem 5.5. $\hat{f}_k := O_k + 1 \geq f_k$.

Proof. By Result 2, each interval Θ_j is an over-estimation of the range of angles of rotation $\mathbf{A}_{\theta,\mathbf{u}}$ that permit the associated point match to be aligned. The number $O_k + 1$ must thus be greater than or equal to the maximum number of point matches that can be aligned under problem P_k. □

As a by-product of interval stabbing, it can be derived

$$\tilde{\mathbf{R}}_k = \mathbf{A}_{\tilde{\theta},\hat{\mathbf{B}}\mathbf{x}_k}\hat{\mathbf{B}}, \tag{5.51}$$

where $\tilde{\theta}$ is an angle that globally solves (5.50). Aligning the input data with $\tilde{\mathbf{R}}_k$ thus provides a lower bound to the original rotation search problem (5.1).

5.4 Main algorithm

This section presents GORE for the rotation search problem (5.1). Algorithm 5.1 summarizes the method. Given a set of input point matches $\mathcal{H}$, GORE iterates over each point match $(\mathbf{x}_k, \mathbf{y}_k)$ and performs two operations: seek an improved lower bound l to problem (5.1) and an upper bound $\hat{f}_k$ to subproblem P_k; both steps are conducted simultaneously using the given techniques in Section 5.3. Both values are then compared to attempt to reject the current point match as an outlier. The output is a reduced set of point matches $\mathcal{H}' \subseteq \mathcal{H}$ guaranteed to include the globally optimal solution $\mathcal{I}^*$ to (5.1).

GORE is a deterministic algorithm, unlike RANSAC. The worst case time complexity can be established as follows: for each k, the bounding interval Θ_i for each $i \neq k$ is obtained in constant time. Given N intervals, the stabbing problem (5.50) can be solved in $\mathcal{O}(N \log N)$ time [8, Chap. 10]. Thus, Line 5 in Algorithm 5.1 takes $\mathcal{O}(N \log N)$ time. In the worst case, Line 5 is performed N times, and GORE thus consumes $\mathcal{O}(N^2 \log N)$ time.

As a whole, GORE contains only very simple geometric operations. Section 5.5 demonstrates the extreme efficiency of GORE in processing large input data sizes.

Algorithm 5.1 Guaranteed outlier removal for rotation search (GORE).

```
Require: Point matches $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$, inlier threshold $\epsilon$.
 1: $\mathcal{H} \leftarrow \{1, 2, \ldots, N\}$.
 2: $\mathcal{H}' \leftarrow \mathcal{H}$, $\mathcal{O} \leftarrow \mathcal{H}$, $\mathcal{V} \leftarrow \emptyset$, and $l \leftarrow 0$.
 3: for all $k \in \mathcal{O}$ do
 4:     $\mathcal{V} \leftarrow \mathcal{V} \cup \{k\}$.
 5:     Compute upper bound $\hat{f}_k$ and suboptimal rotation $\tilde{\mathbf{R}}_k$ (Section 5.3) for problem
        $P_k$ on data indexed by $\mathcal{H}'$.
 6:     $\mathcal{C}_k \leftarrow \{i \mid i \in \mathcal{H}', \angle(\tilde{\mathbf{R}}_k \mathbf{x}_i, \mathbf{y}_i) \le \epsilon\}$.
 7:     $l_k \leftarrow |\mathcal{C}_k|$.
 8:     if $l_k > l$ then
 9:         $l \leftarrow l_k$.
10:         $\mathcal{O} \leftarrow \mathcal{H}' \setminus \mathcal{C}_k$.
11:     end if
12:     if $\hat{f}_k < l$ then
13:         $\mathcal{H}' \leftarrow \mathcal{H}' \setminus \{k\}$.
14:     end if
15:     $\mathcal{O} \leftarrow \mathcal{O} \setminus \mathcal{V}$.
16: end for
17: return $\{(\mathbf{x}_i, \mathbf{y}_i) \mid i \in \mathcal{H}'\}$.
```

5.5 Results

All algorithms were implemented in C++. GORE's implementation is public available [57]. Experiments were conducted on a standard PC with a 2.7 GHz CPU.

5.5.1 Synthetic data

A data instance was generated as follows: N points on the unit-sphere were randomly produced to obtain set $\mathcal{X}$. Set $\mathcal{X}$ was randomly rotated to produce set $\mathcal{Y}$, which was then added with Gaussian noise of $\sigma = 0.5°$ (recall that the angular distance was used here). For a given outlier rate ρ, ρN point matches $(\mathbf{x}_i, \mathbf{y}_i)$ were randomly chosen from $(\mathcal{X}, \mathcal{Y})$ and resampled uniformly on the sphere to create outliers. In the experiments, $N \in \{100, 250, 500\}$ and $\rho = \{0, 0.05, \ldots, 0.9\}$ were used. For each (N, ρ) combination, 1000 data instances were generated and $\epsilon = 0.5°$ was used in (5.1). The following approaches were tested:

- RANSAC: A confidence level of 0.99 was used for the stopping criterion [24]. For each data instance, median runtime over 100 runs were taken.
- GORE: Algorithm 5.1. No particular ordering for the data was conducted beyond the order of generation.
- BnB: Following the method of [7, 34].

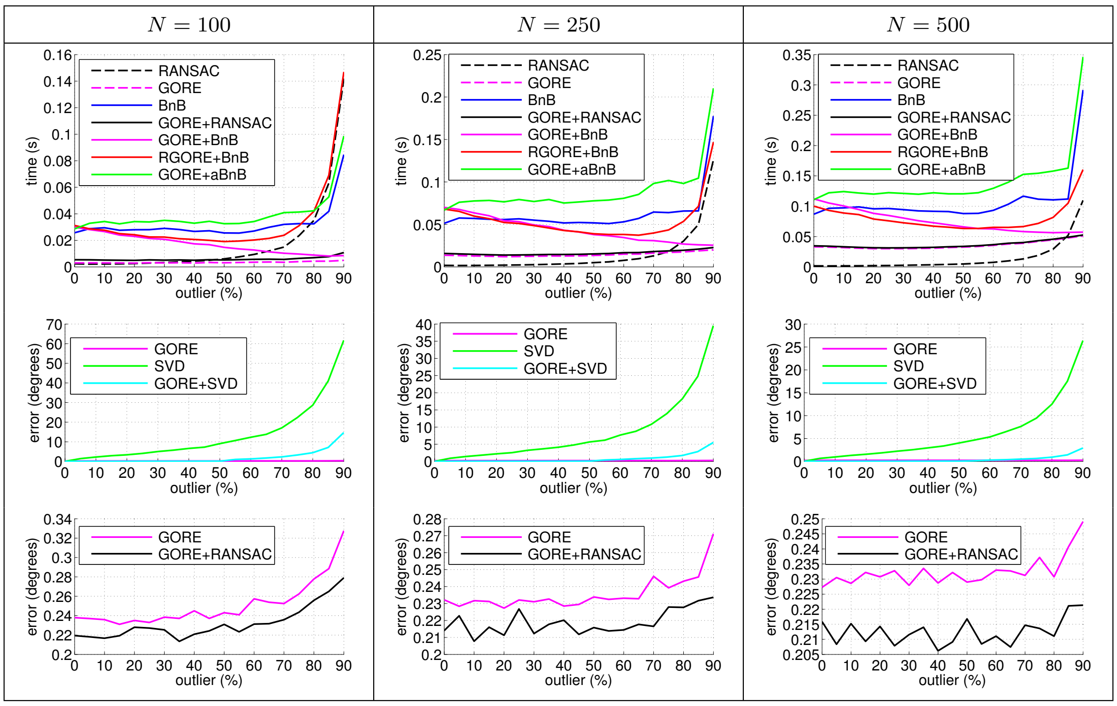


Figure 5.5: Results on synthetic data. Row 1: Runtimes for different outlier ratios. Rows 2 and 3: Angular error of estimated rotations. To avoid clutter, GORE+RANSAC's error is not plotted in Row 2.

- GORE+BnB: Data remaining after GORE was fed to BnB. The lower bound of BnB was also initialised as the value of l at the termination of Algorithm 5.1.
- GORE+RANSAC: Data remaining after GORE was fed to RANSAC. Global optimality is not guaranteed.
- RGORE+BnB: Same as GORE+BnB, but the initial value of l in Algorithm 5.1 for GORE was obtained by first running RANSAC to yield a suboptimal result $\tilde{\mathcal{I}}$.
- GORE+aBnB: Same as GORE+BnB, but all the original data was given to BnB (GORE was only used to initialize the lower bound of BnB).

Runtime comparisons The first row of Figure 5.5 shows the median *total* runtime over all data instances for the methods.

While RANSAC was faster than GORE at low outlier rates, as the outlier rate increased, the runtime of RANSAC increased exponentially. In contrast, the runtime of GORE grew at a much lower rate. As expected, the runtime of BnB also grew rapidly as the outlier rate increased. This contrasts with the trend exhibited by GORE+BnB—as the outlier rate increased, the total runtime decreased. This is because at higher outlier rates, GORE removed more outliers and reduced the overall data population more aggressively.

Hence, BnB was able to find the global solution using less time. Since the bulk of the runtime in GORE+BnB was due to BnB, the total runtime decreased with outlier rate.

At lower outlier rates, GORE+BnB took longer than "raw" BnB since there were fewer outliers to remove, but steadily GORE+BnB began to outperform BnB. The performance gain became significant at $\approx 70\%$ outliers. At 90% outliers, GORE+BnB was an order of magnitude faster than BnB. As it is shown in Section 5.5.2, outlier rates greater than 95% are actually very common in real data.

The results of RGORE+BnB at low outlier rates show that initializing GORE with RANSAC only marginally reduced the total runtime. However, at high outlier rates, the total runtime increased dramatically following the slowing down of RANSAC. Crucially, the trend of GORE+aBnB shows clearly that the dominating factor in speeding up BnB is in reducing the data amount and outlier rate, not in initializing BnB with a good lower bound. Hence, hot starting BnB with the suboptimal result $|\tilde{\mathcal{I}}|$ of RANSAC will not reduce runtime (not to mention that at high outlier rates, the computation of RANSAC itself is a major burden).

Evaluation of suboptimal rotation Here is provided empirical evidence that, although GORE cannot completely eliminate all outliers, the best suboptimal rotation $\tilde{\mathbf{R}}_k$ calculated by Algorithm 5.1 is actually a good approximate solution. On each data instance generated above, the error of the best $\tilde{\mathbf{R}}_k$ to the globally optimal solution $\mathbf{R}^*$ was calculated, where the error is measured by $d\angle(\tilde{\mathbf{R}}_k, \mathbf{R}^*) = \|\log(\tilde{\mathbf{R}}_k(\mathbf{R}^*)^\top)\|_2$ with $\log(\cdot)$ the inverse of the exponential map (see Section 2.3.1 for more details). The distance is interpreted as the minimum geodesic motion between $\tilde{\mathbf{R}}_k\mathbf{p}$ and $\mathbf{R}^*\mathbf{p}$, where $\mathbf{p}$ is an arbitrary point [33]. The rotations estimated using SVD [2] from the raw data, and from the data after outlier removal with GORE, were evaluated in the same way. The second row in Figure 5.5 shows error from all three rotations for increasing outlier rate, while the third row shows error from $\tilde{\mathbf{R}}_k$ and GORE+RANSAC's result.

As expected, SVD (LS) rotation estimation is easily biased by outliers. Also, the non-negligible error of GORE+SVD points to the presence of remaining outliers after GORE. The error of $\tilde{\mathbf{R}}_k$, however, remains low ($\leq 0.5°$) even for high outlier rates. This indicates the efficacy of GORE as a suboptimal rotation search method. Results also show that a further improvement to $\tilde{\mathbf{R}}_k$ can be achieved by GORE+RANSAC at a small additional runtime.

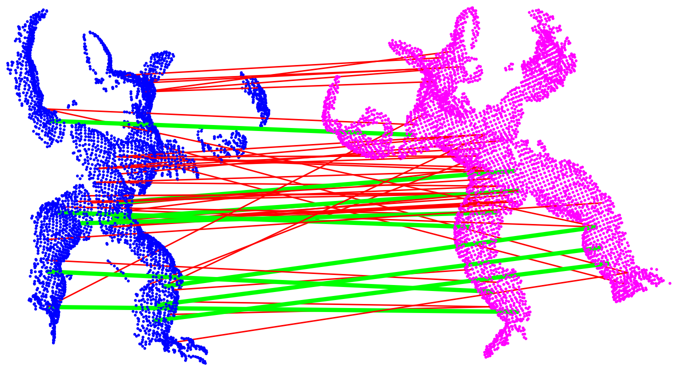


Figure 5.6: Data instance for *armadillo* for $N = 100$. Green lines represent the 10 inlier matches found by BnB. To avoid excessive clutter, only half of the outlier matches (red lines) are displayed.

Object	N	irat	GORE				RANSAC			BnB			RANSAC +BnB	GORE +BnB
			lwbnd	err (°)	out	time (s)	lwbnd	err (°)	time (s)	opt	err (°)	time (s)	time (s)	time (s)
buddha $\lvert S_1\rvert = 4151$ $\lvert S_2\rvert = 3901$	100	0.09	6	0.23	53	0.009	7	0.36	0.164	9	0.37	0.225	0.389	0.074
	250	0.05	9	0.31	178	0.040	10	0.24	0.583	12	0.22	0.980	1.561	0.116
	500	0.03	13	0.35	390	0.112	14	0.31	1.366	17	0.27	2.875	4.211	0.237
	750	0.02	13	0.34	590	0.304	14	0.32	4.127	17	0.27	7.565	11.827	0.630
	1000	0.01	13	0.32	807	0.447	14	0.30	6.494	17	0.27	12.610	19.470	1.018
bunny $\lvert S_1\rvert = 6533$ $\lvert S_2\rvert = 6226$	100	0.18	16	0.19	74	0.003	16	0.20	0.032	18	0.13	0.030	0.062	0.003
	250	0.10	20	0.27	209	0.015	21	0.24	0.133	24	0.13	0.145	0.278	0.024
	500	0.06	27	0.23	442	0.056	26	0.23	0.342	30	0.22	0.520	0.881	0.076
	750	0.04	31	0.18	684	0.127	29	0.25	0.659	32	0.23	1.245	1.946	0.147
	1000	0.04	32	0.19	924	0.219	30	0.24	1.220	35	0.14	2.445	3.764	0.269
armadillo $\lvert S_1\rvert = 4508$ $\lvert S_2\rvert = 4362$	100	0.10	7	0.17	80	0.003	8	0.30	0.125	10	0.21	0.095	0.215	0.013
	250	0.06	10	0.17	229	0.014	12	0.31	0.501	14	0.26	0.350	0.875	0.021
	500	0.03	10	0.69	469	0.055	12	0.31	1.783	15	0.24	1.430	3.198	0.066
	750	0.02	13	0.34	713	0.146	13	0.29	3.270	16	0.24	3.435	7.002	0.161
	1000	0.01	13	0.34	958	0.233	13	0.31	6.843	16	0.50	7.150	14.505	0.264
dragon $\lvert S_1\rvert = 5332$ $\lvert S_2\rvert = 4683$	100	0.20	19	0.22	71	0.004	18	0.20	0.024	20	0.24	0.060	0.079	0.014
	250	0.12	29	0.11	205	0.016	29	0.15	0.068	30	0.25	0.175	0.241	0.034
	500	0.07	30	0.18	446	0.055	31	0.17	0.257	33	0.22	0.565	0.827	0.065
	750	0.05	33	0.15	693	0.167	33	0.16	0.506	35	0.17	1.340	1.908	0.184
	1000	0.04	36	0.12	939	0.226	36	0.16	0.870	38	0.14	2.635	3.557	0.283

Table 5.1: Point cloud registration results.

5.5.2 Point cloud registration

This experiment tests the use of GORE for rotational registration. This experiment uses data from the Stanford repository, namely *buddha*, *bunny*, *armadillo* and *dragon*. Two partially overlapping scans S_1 and S_2 were selected for each object. Sizes of S_1 and S_2 are listed on the Column 1 in Table 5.1. S_1 and S_2 were centred and scaled such as their centroids coincided with the origin and both point sets were contained in the cube $[-50, 50]^3$. Point matches between S_1 and S_2 were obtained using ISS3D [74] keypoint detector and matching with the PFH [66] descriptor as available in Point Cloud Library [61].

A correspondence set was created by retaining N of the best point matches based on the L_2-norm of the PFH descriptors. For $N \in \{100, 250, 500\}$, the obtained inlier ratios based on the threshold $\epsilon = 0.5^\circ$ are listed in Column 3 in Table 5.1. Observe that the outlier rates in this problem are extremely high, even reaching 99% in some data instances. For each correspondence set, 10 different randomized rotations were applied on S_1 to produce 10 data instances for rotation search; Figure 5.6 depicts one such instance.

For GORE, a straightforward variant was used; the main loop in Algorithm 5.1 was iterated until no more outliers could be removed. Typically this required 3 to 10 passes through the data. While this increased the duration of the method, the total runtime was still relatively minuscule, as evidenced in Table 5.1. Also, the following methods were executed: RANSAC, BnB, RANSAC+BnB (the suboptimal RANSAC result $|\tilde{\mathcal{I}}|$ was used to initialize the lower bound of BnB) and GORE+BnB. The following measures were recorded:

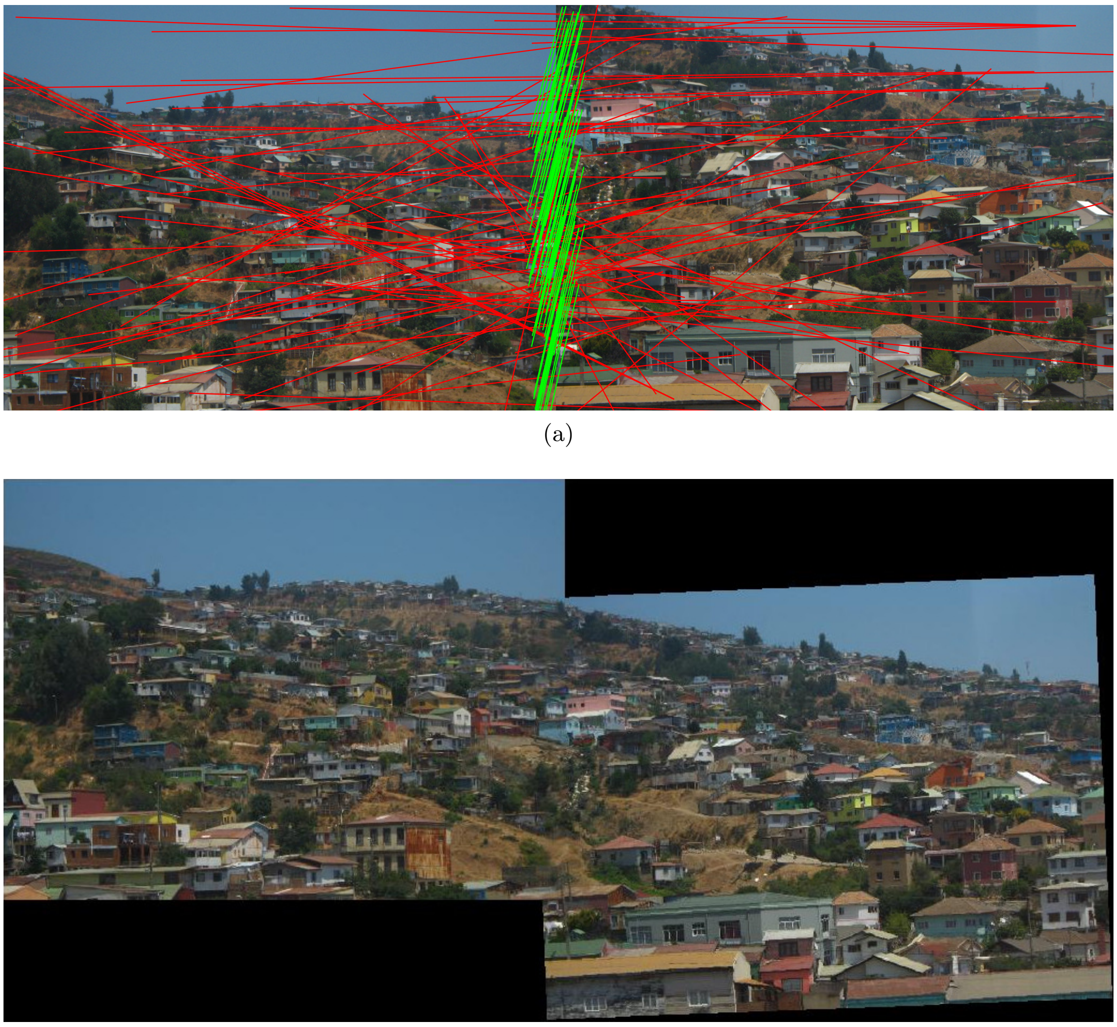
- lwbnd: objective value (5.1) of best suboptimal solution.
- err (°): angular error in degrees of best suboptimal rotation to true rotation.
- time (s): total runtime in seconds.
- opt: objective value (5.1) of global solution.

Table 5.1 lists the median values over all 10 data instances.

Due to the extremely high outlier rates, RANSAC was an order of magnitude slower than GORE. On all the data instances, GORE was able to terminate within 1 second, even with multiple passes over the data. The most crucial outcome is that the combination GORE+BnB was able to find the globally optimal result with an order of magnitude less time than raw BnB. This was due to the massive reduction of true outliers before BnB—in this experiment, after GORE the median problem size to BnB was just 50. Additionally, the fact that RANSAC+BnB was slower than raw BnB indicates the ineffectiveness of hot starting using RANSAC.

5.5.3 Image stitching

This experiment follows the image stitching experiment in [19]. SIFT correspondences are obtained across an image pair taken with known camera intrinsics $\mathbf{K}_1$ and $\mathbf{K}_2$. The scene is sufficiently far away to justify a homography $\mathbf{H} = \mathbf{K}_2\mathbf{R}\mathbf{K}_1^{-1}$ as an alignment function, where $\mathbf{R}$ is the rotation between the views. The rotation $\mathbf{R}$ can be estimated



(a)

(b)

Figure 5.7: SIFT correspondences and stitching result of GORE. Source: Flickr (left and right images).

by registering the matching vectors backprojected from the SIFT keypoint coordinates using the inverse calibration matrix.

Figure 5.7 presents a challenging image pair where there is a very small overlapping area. A total of 147 SIFT matches were detected and 79 of them correspond to inliers. The inlier threshold used was $\epsilon = 0.04°$. GORE with multiple repetitions eliminated all outliers in 10 ms; running BnB after GORE would thus terminate immediately, since the best solution found by GORE equals to the global solution.

The experiment was run over the four additional image pairs presented in Figures 5.8–5.11.

Image pair	N	irat	GORE				BnB		GORE +BnB
			lwbnd	err (°)	out	time (s)	opt	time (s)	time (s)
valparaiso	147	0.54	79	1.55	68	0.01	79	0.21	0.01
machu-picchu	194	0.33	51	1.10	60	0.02	64	1.94	1.27
paris1	718	0.07	49	0.01	584	0.33	51	16.68	2.03
paris2	921	0.22	181	0.16	311	1.07	200	16.24	11.33
rio	675	0.18	115	0.69	379	0.20	120	11.43	4.74

Table 5.2: Image stitching results.

5.5.3.1 Quantitative results

Table 5.2 presents quantitative results and experiment details. For brevity, RANSAC results are not included here, since it is clear from results in Section 5.5.1 that running RANSAC does not assist in cutting down the overall runtime of globally optimal rotation search. For all tested image pairs, removing true outliers found by GORE helped to decrease runtime.

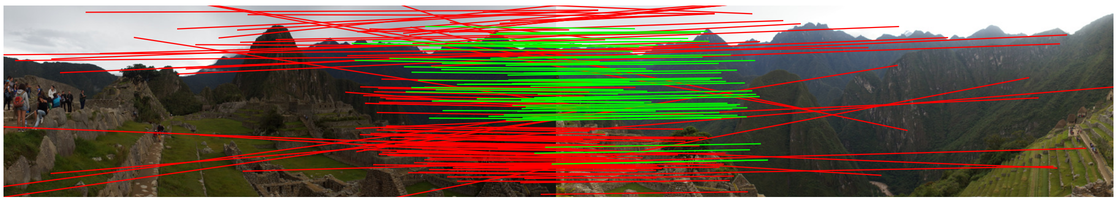
5.5.3.2 Qualitative results

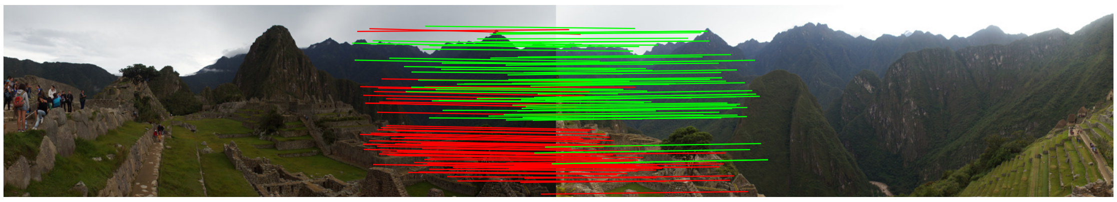
Panel (b) in Figures 5.8–5.11 show the SIFT keypoint matches. Green lines designate the inliers that agree with the globally optimal rotation, while red lines designate "true" outliers with respect to the globally optimal rotation.

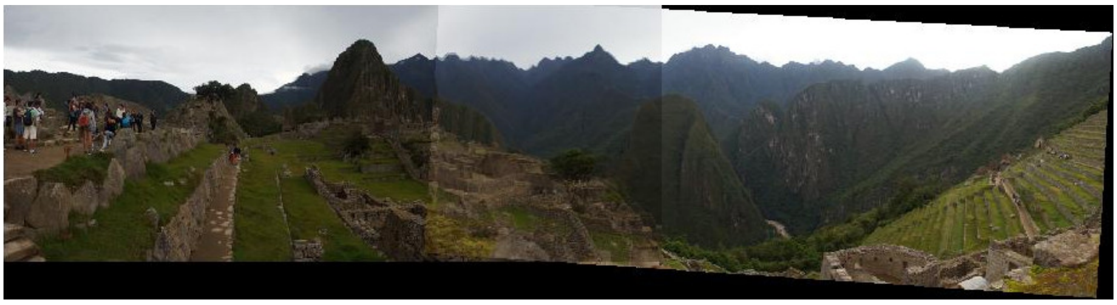
Panel (c) show the matches that remain after GORE; observe that the matches that have been removed do not exist in the true inlier set.

To demonstrate that the suboptimal rotation produced by GORE is very close to the optimal one, panel (d) show the stitched images using the homographies defined using the suboptimal rotations. Compare with the stitching results in panel (e) where the homographies were defined using the globally optimal rotation found by BnB.

(a) Input images.


(b) SIFT keypoint matches (green = true inliers, red = true outliers).


(c) Matches that remain after preprocessing with GORE.


(d) Stitching result using suboptimal rotation by GORE.

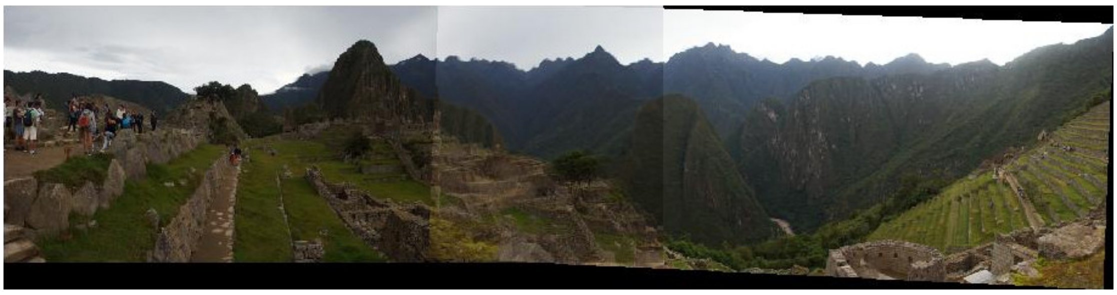
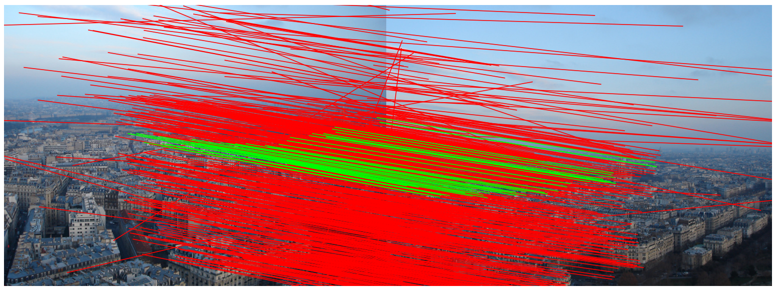
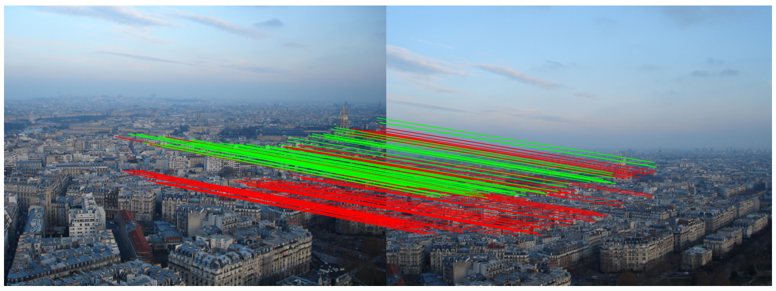
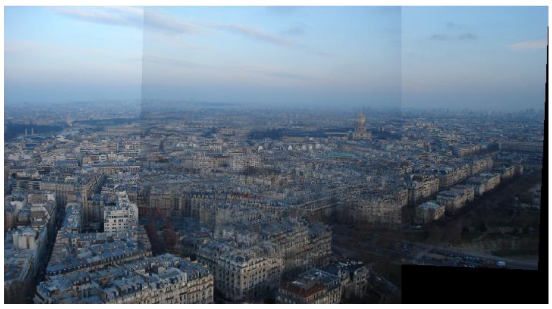

(e) Stitching result using globally optimal rotation by BnB.

FIGURE 5.8: Results for the *machu-picchu* image pair. Source: Flickr (left and right images).

(a) Input images.


(b) SIFT keypoint matches (green = true inliers, red = true outliers).


(c) Matches that remain after preprocessing with GORE.


(d) Stitching result using suboptimal rotation by GORE.

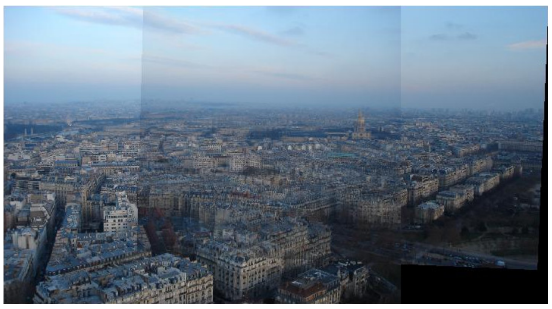
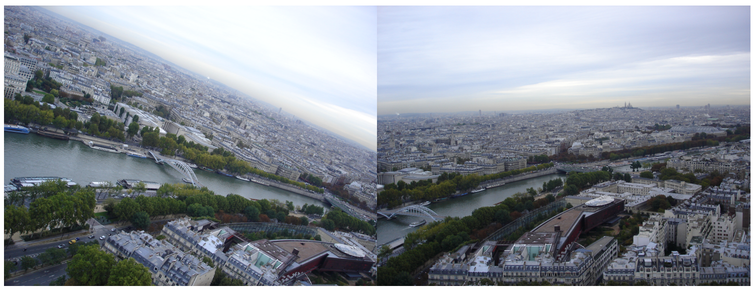
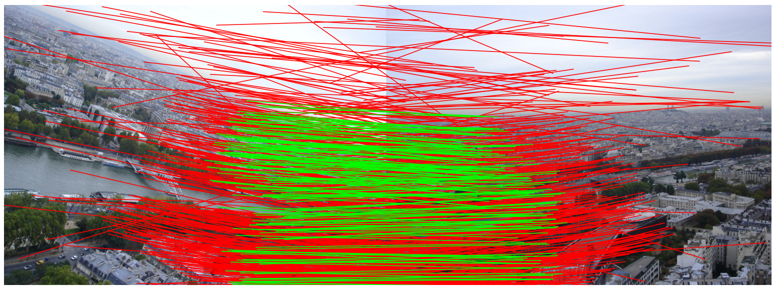
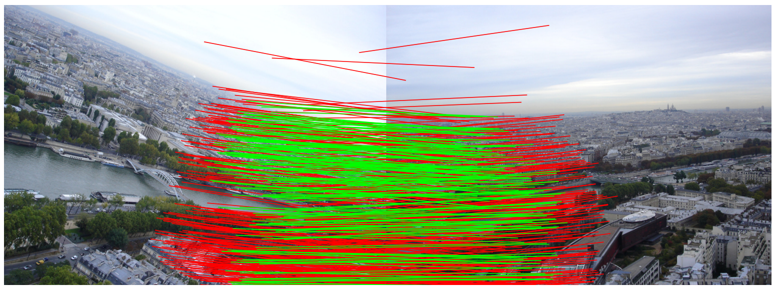
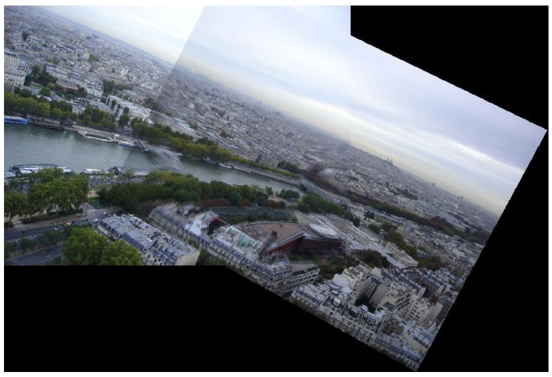

(e) Stitching result using globally optimal rotation by BnB.

FIGURE 5.9: Results for the *paris1* image pair. Source: Flickr (left and right images).


(a) Input images.


(b) SIFT keypoint matches (green = true inliers, red = true outliers).


(c) Matches that remain after preprocessing with GORE.


(d) Stitching result using suboptimal rotation by GORE.

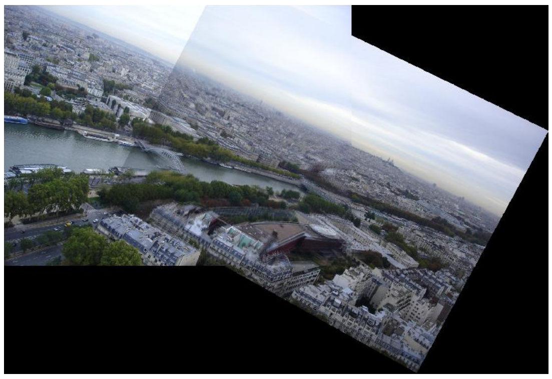
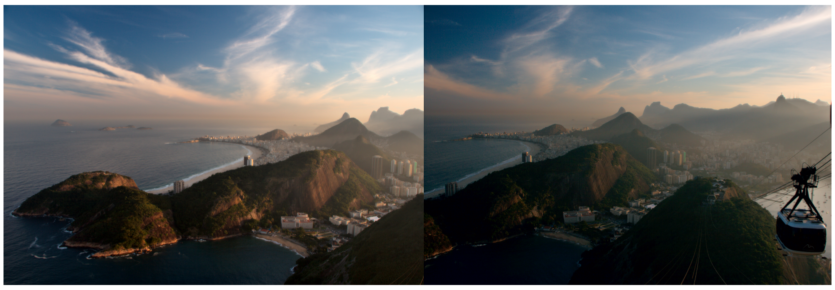
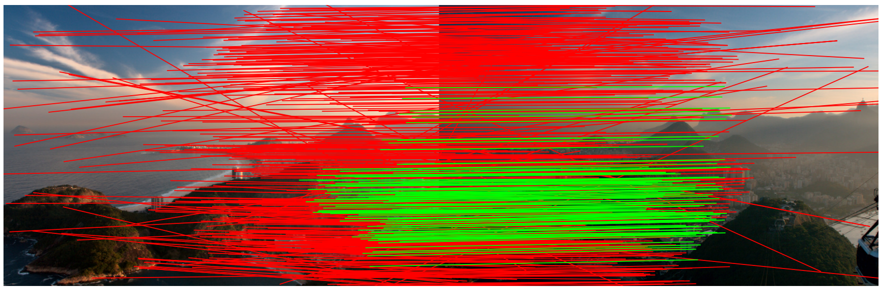
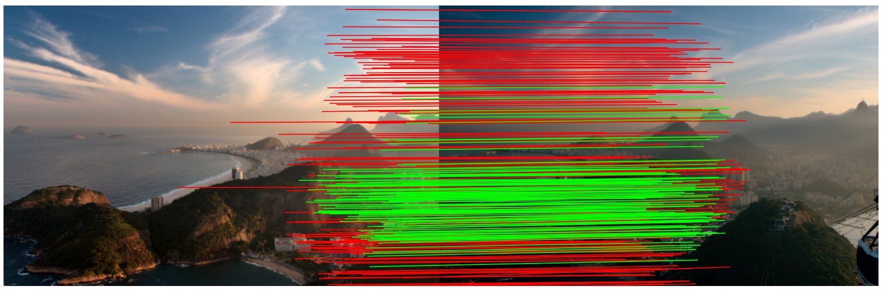
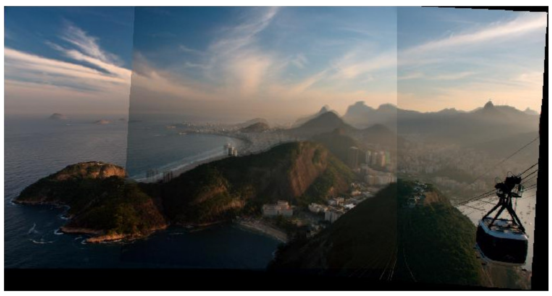

(e) Stitching result using globally optimal rotation by BnB.

FIGURE 5.10: Results for the *paris2* image pair. Source: Flickr (left and right images).


(a) Input images


(b) SIFT keypoint matches (green = true inliers, red = true outliers).


(c) Matches that remain after preprocessing with GORE.


(d) Stitching result using suboptimal rotation by GORE.

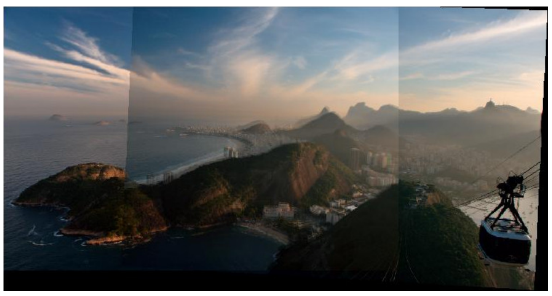

(e) Stitching result using globally optimal rotation by BnB.

FIGURE 5.11: Results for the *rio* image pair. Source: Flickr (left and right images).

5.6 Summary

This chapter presented a guaranteed outlier removal technique for problem Type 1, in the sense that any datum it removes cannot be in the globally optimal solution. Based on simple geometric operations, GORE (Algorithm 5.1) is deterministic and efficient. Experiments show that, by significantly reducing a significant amount of the outliers, GORE greatly speeds up globally optimal rotation search.

The suboptimal rotation of GORE could be a good alternative for applications that require a quick solution.

Chapter 6

Conclusions and Future Work

This thesis investigated finding globally optimal solutions for rotation search. Rotation search is an important sub-routine in many geometric computer vision problems such as point cloud registration, camera pose estimation, and image stitching, as surveyed in Chapter 1. Rotation search is a computationally challenging problem. Most of the useful objective functions such as GM and CSM are nonconvex/nonconcave, as surveyed in Chapter 2.

This thesis made several contributions towards improving the performance of solving robust rotation search. For 1D rotation search, Chapter 3 proposed a real-time algorithm. For 3D rotation search, Chapter 4 proposed an algorithm that is an order of magnitude faster than previous formulations. When rotation search is solved on point correspondences, Chapter 5 proposed an effective outlier removal method that only removes true outliers, thus reducing the time for obtaining the optimal rotation by a subsequent exact algorithm.

6.1 Future research directions

6.1.1 Extensions for registration of LIDAR scans

A possible extension for the user-assisted point cloud registration system of Chapter 3 is to leverage on 3D keypoint detectors to provide suggestions to the user on how to align the point clouds. For example, solutions could be proposed efficiently by solving rotation search for the best keypoint matches. Alternatively, if the number of keypoints is small (≈ 100 for each point cloud), rotation could be solved by Algorithm 3.1 for all possible keypoint matches in a few minutes.

6.1.2 Improvements for 3D rotation search

The BnB rotation search algorithm of Chapter 4 uses R-trees to index all possible matches (see Section 4.4.2). The proposed upper bounding function (4.14) is evaluated by querying multiple R-trees. There is a trade-off between insertion and search time in R-trees [45]. Thus, the runtime of Algorithm 4.1 could be improved by using a more sophisticated insertion strategy if the further reduction in querying time is more significant than the extra cost of the insertion strategy. Also, static versions for R-trees could be examined. Such variants of R-trees "bulk load" the data with a possible reduction on the building time. For a discussion on insertion strategies and static R-trees the reader can refer to [45].

6.1.3 Removal of true outliers on point cloud registration

A further research direction is to investigate the removal of true outliers on point cloud registration:

$$\begin{aligned} &\underset{\mathbf{R}\in SO(3),\, \mathbf{t}\in\mathbb{R}^3,\, \mathcal{I}\subseteq\mathcal{H}}{\text{maximize}} && |\mathcal{I}| \\ &\text{subject to} && \|\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\| \leq \epsilon,\ \forall i \in \mathcal{I}, \end{aligned} \tag{6.1}$$

where $\mathcal{H} = \{1, \ldots, N\}$ indexes the set of all point correspondences $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$.

For 4 DoF registration, true outliers might be removed using a similar method to as that used in Chapter 5: assuming a correspondence to be a true inlier, bound the constrains of (6.1) into angular intervals. As in Chapter 5, an upper bound of the solution of the problem can be obtained on such angular intervals. If that upper bound is lower than a known lower bound, then the assumed inlier can be identified as a true outlier.

For 6 DoF registration, if the translation is fixed such that $(\mathbf{x}_k, \mathbf{y}_k)$ remains a valid correspondence for any rotation, an upper bound of the solution of (6.1) can be obtained as $N - m$, where m is a lower bound on the the number of true outliers of the resulting rotation search problem resulting of fixing the translation. Thus, m can be obtained by using the Euclidean version of Algorithm 5.1. From Figure 6.1, the angular error ϵ can be expressed in terms of the Euclidean error ϵ':

$$\epsilon = \arccos\left(\frac{d}{\|\mathbf{x}_i\|}\right). \tag{6.2}$$

Thus, Algorithm 5.1 can be adapted to the Euclidean error by considering independent angular errors (6.2) associated with each point correspondence $(\mathbf{x}_i, \mathbf{y}_i)$.

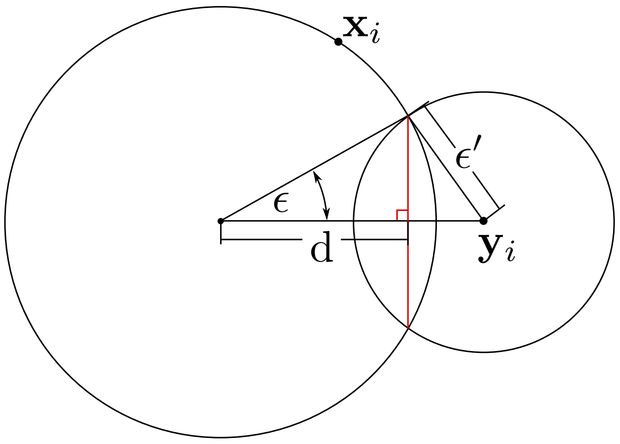


FIGURE 6.1: Relating the Euclidean error ϵ' and the angular error ϵ for a point correspondence $(\mathbf{x}_i, \mathbf{y}_i)$ related by a rotation $\mathbf{R}$ such that $\|\mathbf{R}\mathbf{x}_i - \mathbf{y}_i\| \leq \epsilon'$. d can be obtained by solving the circle–circle intersection problem.

Bibliography

[1] D. Aiger, N. J. Mitra, and D. Cohen-Or. “4-points congruent sets for robust pairwise surface registration”. In: *Proceedings ACM SIGGRAPH 2008*. SIGGRAPH. New York, NY, USA: ACM, Aug. 1, 2008, 85:1–85:10. ISBN: 978-1-4503-0112-1. DOI: 10.1145/1399504.1360684.

[2] K. Arun, T. Huang, and S. Blostein. “Least-Squares Fitting of Two 3-D Point Sets”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* PAMI-9.5 (Sept. 1987), pp. 698–700. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1987.4767965.

[3] E. Ask, O. Enqvist, and F. Kahl. “Optimal Geometric Fitting under the Truncated L2-Norm”. In: *Computer Vision and Pattern Recognition (CVPR)*. Computer Vision and Pattern Recognition. Portland, OR: IEEE, June 23–28, 2013, pp. 1722–1729. DOI: 10.1109/CVPR.2013.225.

[4] C. Atkin. “Weakly bounded banach lie groups”. In: *Victoria Interna- tional Conference*. 2004, pp. 9–13.

[5] H. Bay, T. Tuytelaars, and L. Van Gool. “SURF: Speeded Up Robust Features”. In: *European Conference on Computer Vision (ECCV)*. Vol. 3951. 3951. Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33832-1. DOI: 10.1007/11744023_32.

[6] J.-C. Bazin, H. Li, I. S. Kweon, C. Demonceaux, P. Vasseur, and K. Ikeuchi. “A Branch-and-Bound Approach to Correspondence and Grouping Problems”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.7 (July 2013), pp. 1565–1576. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.264.

[7] J.-C. Bazin, Y. Seo, and M. Pollefeys. “Globally Optimal Consensus Set Maximization through Rotation Search”. In: *Asian Conference on Computer Vision (ACCV)*. Vol. 7725. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 539–551. ISBN: 978-3-642-37443-2. DOI: 10.1007/978-3-642-37444-9_42.

[8] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational geometry*. 3rd. Springer, 2008.

[9] P. J. Besl and N. D. McKay. "A method for registration of {3-D} shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (Feb. 1992), pp. 239–256. ISSN: 0162-8828. DOI: 10.1109/34.121791.

[10] T. M. Breuel. "A comparison of search strategies for geometric branch and bound algorithms". In: *European Conference on Computer Vision (ECCV)*. Vol. 2352. Copenhagen, Denmark: Springer Berlin Heidelberg, Apr. 2002, pp. 837–850. ISBN: 978-3-540-43746-8. DOI: 10.1007/3-540-47977-5_55.

[11] T. M. Breuel. "Implementation techniques for geometric branch-and-bound matching methods". In: *Computer Vision and Image Understanding* 90.3 (2003), pp. 258 –294. ISSN: 1077-3142. DOI: 10.1016/S1077-3142(03)00026-2.

[12] M. Brown, D. Windridge, and J.-Y. Guillemaut. "Globally Optimal 2D-3D Registration from Points or Lines Without Correspondences". In: International Conference on Computer Vision (ICCV). 2015, pp. 2111–2119. URL: http://www.cv-foundation.org/openaccess/content_iccv_2015/html/Brown_Globally_Optimal_2D-3D_ICCV_2015_paper.html.

[13] C.-S. Chen, Y.-P. Hung, and J.-B. Cheng. "RANSAC-based DARCES: A new approach to fast automatic registration of partially overlapping range images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 21.11 (Nov. 1999), pp. 1229–1234. ISSN: 0162-8828. DOI: 10.1109/34.809117.

[14] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek. "The trimmed iterative closest point algorithm". In: *International Conference on Pattern Recognition (ICPR)*. Vol. 3. IEEE, 2002, pp. 545–548. ISBN: 0-7695-1695-X. DOI: 10.1109/ICPR.2002.1047997.

[15] T.-J. Chin, Á. Parra Bustos, and M. S. Brown. *Demostration video for: Fast Rotation Search for Real-Time Interactive Point Cloud Registration*. 2014. URL: https://youtu.be/tg8XfU13GGQ.

[16] T.-J. Chin, Á. Parra Bustos, M. S. Brown, and D. Suter. "Fast Rotation Search for Real-time Interactive Point Cloud Registration". In: *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '14. New York, NY, USA: ACM, 2014, pp. 55–62. ISBN: 978-1-4503-2717-6. DOI: 10.1145/2556700.2556712.

[17] B. Curless and M. Levoy. "A volumetric method for building complex models from range images". In: *Computer Graphics (SIGGRAPH 1996 Proceedings)*. ACM, 1996, pp. 303–312. DOI: 10.1145/237170.237269.

[18] D. W. Eggert, A. Lorusso, and R. B. Fisher. “Estimating 3-D rigid body transformations: a comparison of four major algorithms”. In: *Machine Vision and Applications* 9.5-6 (1997), pp. 272–290. ISSN: 0932-8092. DOI: 10.1007/s001380050048.

[19] O. Enqvist, E. Ask, F. Kahl, and K. Åström. “Robust Fitting for Multiple View Geometry”. In: *European Conference on Computer Vision (ECCV)*. ECCV. Vol. 7572. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 738–751. ISBN: 978-3-642-33717-8. DOI: 10.1007/978-3-642-33718-5_53.

[20] O. Enqvist, E. Ask, F. Kahl, and K. Åström. “Tractable Algorithms for Robust Model Estimation”. In: *International Journal of Computer Vision* 112.1 (Mar. 1, 2014), pp. 115–129. ISSN: 0920-5691. DOI: 10.1007/s11263-014-0760-2.

[21] O. Enqvist, K. Josephson, and F. Kahl. “Optimal Correspondences from Pairwise Constraints”. In: *International Conference on Computer Vision (ICCV)*. IEEE, Sept. 29–Oct. 2, 2009, pp. 1295–1302. ISBN: 978-1-4244-4420-5. DOI: 10.1109/ICCV.2009.5459319.

[22] O. Enqvist and F. Kahl. “Robust Optimal Pose Estimation”. In: *European Conference on Computer Vision (ECCV)*. Vol. 5302. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 141–153. ISBN: 978-3-540-88681-5. DOI: 10.1007/978-3-540-88682-2_12.

[23] O. D. Faugeras and M. Hebert. “The Representation, Recognition, and Locating of 3-D Objects”. In: *The International Journal of Robotics Research* 5.3 (1986), pp. 27–52. ISSN: 0278-3649. DOI: 10.1177/027836498600500302.

[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: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692.

[25] A. W. Fitzgibbon. “Robust Registration of 2D and 3D Point Sets”. In: *British Machine Vision Conference (BMVC)*. Manchester: BMVA Press, 2001, pp. 411–420. ISBN: 1-901725-16-2. DOI: 10.5244/C.15.43.

[26] A. W. Fitzgibbon. “Robust registration of 2D and 3D point sets”. In: *Image and Vision Computing* 21.13–14 (2003). British Machine Vision Computing 2001, pp. 1145 –1153. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2003.09.004.

[27] R. Gal and D. Cohen-Or. “Salient geometric features for partial shape matching and similarity”. In: *ACM Transactions on Graphics (TOG)* 25.1 (Jan. 2006), pp. 130–150. DOI: 10.1145/1122501.1122507.

[28] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. “Robust global registration”. In: *Eurographics*. Eurographics. Vol. 2. 2005, pp. 197–206. ISBN: 3-905673-24-X. URL: http://dl.acm.org/citation.cfm?id=1281920.1281953.

[29] S. Gold, A. Rangarajan, C.-P. Lu, S. Pappu, and E. Mjolsness. "New algorithms for 2D and 3D point matching: Pose estimation and correspondence". In: *Pattern recognition* 31.8 (1998), pp. 1019–1031. DOI: 10.1016/S0031-3203(98)80010-1.

[30] Google Street View. *Google Photo Sphere's website*. Google Photo Sphere. URL: https://www.google.com/maps/streetview/publish/.

[31] R. Hartley and F. Kahl. "Global optimization through searching rotation space and optimal estimation of the essential matrix". In: *International Conference on Computer Vision (ICCV)*. Rio de Janeiro: IEEE, 2007, pp. 1–8. ISBN: 978-1-4244-1630-1. DOI: 10.1109/ICCV.2007.4408896.

[32] R. Hartley and F. Kahl. "Optimal Algorithms in Multiview Geometry". In: *Asian Conference on Computer Vision (ACCV)*. Vol. 4843. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 13–34. ISBN: 978-3-540-76385-7. DOI: 10.1007/978-3-540-76386-4_2.

[33] R. Hartley, J. Trumpf, Y. Dai, and H. Li. "Rotation Averaging". In: *International Journal of Computer Vision* 103.3 (July 2013), pp. 267–305. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-012-0601-0.

[34] R. I. Hartley and F. Kahl. "Global Optimization through Rotation Space Search". In: *International Journal of Computer Vision* 82.1 (2009), pp. 64–79. ISSN: 0920-5691. DOI: 10.1007/s11263-008-0186-9.

[35] J. Heller, M. Havlena, and T. Pajdla. "A branch-and-bound algorithm for globally optimal hand-eye calibration". In: *Computer Vision and Pattern Recognition (CVPR)*. CVPR. Providence, RI: IEEE, June 2012, pp. 1608–1615. ISBN: 978-1-4673-1226-4. DOI: 10.1109/CVPR.2012.6247853.

[36] R. Horaud and F. Dornaika. "Hand-Eye Calibration". In: *The International Journal of Robotics Research* 14.3 (June 1, 1995), pp. 195–210. ISSN: 0278-3649. DOI: 10.1177/027836499501400301.

[37] B. K. P. Horn. "Closed-form solution of absolute orientation using unit quaternions". In: *J. Opt. Soc. Am. A* 4.4 (Apr. 1987), pp. 629–642. DOI: 10.1364/JOSAA.4.000629.

[38] R. Horst and H. Tuy. *Global optimization: Deterministic approaches*. Berlin, Heidelberg: Springer, 2003. 730 pp. ISBN: 978-3-540-61038-0.

[39] B. Jian and B. C. Vemuri. "A robust algorithm for point set registration using mixture of Gaussians". In: *International Conference on Computer Vision (ICCV)*. Vol. 2. IEEE, 2005, pp. 1246–1251. ISBN: 0-7695-2334-X. DOI: 10.1109/ICCV.2005.17.

[40] M. Leordeanu and M. Hebert. “A spectral technique for correspondence problems using pairwise constraints”. In: *International Conference on Computer Vision (ICCV)*. Vol. 2. Beijing: IEEE, 2005, pp. 1482–1489. ISBN: 0-7695-2334-X. DOI: 10.1109/ICCV.2005.20.

[41] H. Li and R. Hartley. “The 3D-3D Registration Problem Revisited”. In: *International Conference on Computer Vision (ICCV)*. Rio de Janeiro: IEEE, Oct. 2007, pp. 1–8. ISBN: 978-1-4244-1630-1. DOI: 10.1109/ICCV.2007.4409077.

[42] X. Li and I. Guskov. “Multiscale Features for Approximate Alignment of Point-based Surfaces.” In: *ACM SIGGRAPH/Eurographics Symposium on Geometry Processing*. Vol. 255. Vienna, Austria: Citeseer, 2005, pp. 217–226. URL: http://dl.acm.org/citation.cfm?id=1281955.

[43] D. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110. ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94.

[44] A. Makadia, A. Patterson, and K. Daniilidis. “Fully automatic registration of 3D point clouds”. In: *Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. New York, USA: IEEE, 2006, pp. 1297–1304. ISBN: 0-7695-2597-0. DOI: 10.1109/CVPR.2006.122.

[45] Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos, and Y. Theodoridis. *R-trees theory and applications*. London: Springer, 2006. ISBN: 978-1-84628-293-5. URL: http://public.eblib.com/choice/publicfullrecord.aspx?p=645397.

[46] MAPTEK. *Introducing I Site Studio 6*. URL: http://www.maptek.com/products/i-site/i-site_studio_6.html.

[47] A. Mian, M. Bennamoun, and R. Owens. “Three-Dimensional Model-Based Object Recognition and Segmentation in Cluttered Scenes”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.10 (Oct. 2006), pp. 1584–1601. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2006.213.

[48] A. Myronenko and Xubo Song. “Point Set Registration: Coherent Point Drift”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.12 (Dec. 2010), pp. 2262–2275. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2010.46.

[49] T. Needham. *Visual complex analysis*. Clarendon Press, 1997.

[50] N. Ohta and K. Kanatani. “Optimal estimation of three-dimensional rotation and reliability evaluation”. In: *European Conference on Computer Vision (ECCV)*. Vol. 1406. Lecture Notes in Computer Science. Springer Berlin Heidelberg, Jan. 1, 1998, pp. 175–187. ISBN: 978-3-540-64569-6. DOI: 10.1007/BFb0055666.

[51] C. Olsson, F. Kahl, and M. Oskarsson. "Optimal Estimation of Perspective Camera Pose". In: *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on.* Vol. 2. 2006, pp. 5–8. DOI: 10.1109/ICPR.2006.909.

[52] C. Olsson, O. Enqvist, and F. Kahl. "A polynomial-time bound for matching and registration with outliers". In: *Computer Vision and Pattern Recognition (CVPR).* Anchorage, AK: IEEE, June 2008, pp. 1–8. ISBN: 978-1-4244-2242-5. DOI: 10.1109/CVPR.2008.4587757.

[53] C. Olsson, F. Kahl, and M. Oskarsson. "Branch-and-Bound Methods for Euclidean Registration Problems". In: *Pattern Analysis and Machine Intelligence (TPAMI)* 31.5 (May 2009), pp. 783–794. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2008.131.

[54] C. Olsson, F. Kahl, and M. Oskarsson. "The registration problem revisited: Optimal solutions from points, lines and planes". In: *Computer Vision and Pattern Recognition (CVPR).* Vol. 1. IEEE, 2006, pp. 1206–1213. ISBN: 0-7695-2597-0. DOI: 10.1109/CVPR.2006.307.

[55] J. O'Rourke. *Computational geometry in C.* 2nd ed. Cambridge, UK: Cambridge University Press, 1998. 376 pp. ISBN: 978-0-521-64010-7 978-0-521-64976-6.

[56] C. Papazov and D. Burschka. "Stochastic global optimization for robust point set registration". In: *Computer Vision and Image Understanding* 115.12 (Dec. 2011), pp. 1598–1609. ISSN: 10773142. DOI: 10.1016/j.cviu.2011.05.008.

[57] Á. Parra Bustos. *Author's website.* URL: http://cs.adelaide.edu.au/~aparra.

[58] Á. Parra Bustos and T.-J. Chin. "Guaranteed Outlier Removal for Rotation Search". In: *International Conference on Computer Vision (ICCV).* 2015, pp. 2165–2173. URL: http://www.cv-foundation.org/openaccess/content_iccv_2015/html/Bustos_Guaranteed_Outlier_Removal_ICCV_2015_paper.html.

[59] Á. Parra Bustos, T.-J. Chin, A. Eriksson, H. Li, and D. Suter. "Fast Rotation Search with Stereographic Projections for 3D Registration". In: (2015). DOI: 10.1109/TPAMI.2016.2517636.

[60] Á. Parra Bustos, Chin, Tat-Jun, and D. Suter. "Fast Rotation Search with Stereographic Projections for 3D Registration". In: *Computer Vision and Pattern Recognition (CVPR).* June 23–28, 2014, pp. 3930–3937. ISBN: 978-1-4673-1226-4. DOI: 10.1109/CVPR.2014.502.

[61] PCL. *Point Cloud Library's website.* URL: http://pointclouds.org.

[62] H. Pottmann, J. Wallner, Q.-X. Huang, and Y.-L. Yang. "Integral invariants for robust geometry processing". In: *Computer Aided Geometric Design* 26.1 (Jan. 2009), pp. 37–60. ISSN: 01678396. DOI: 10.1016/j.cagd.2008.01.002.

[63] F. Remondino. “Heritage Recording and 3D Modeling with Photogrammetry and 3D Scanning”. In: *Remote Sensing* 3.12 (May 30, 2011), pp. 1104–1138. ISSN: 2072-4292. DOI: 10.3390/rs3061104.

[64] T. Ruland, T. Pajdla, and L. Kruger. “Globally optimal hand-eye calibration”. In: *Computer Vision and Pattern Recognition (CVPR)*. Providence, RI: IEEE, June 2012, pp. 1035–1042. ISBN: 978-1-4673-1226-4. DOI: 10.1109/CVPR.2012.6247781.

[65] S. Rusinkiewicz and M. Levoy. “Efficient variants of the ICP algorithm”. In: *International Conference on 3-D Digital Imaging and Modeling*. Quebec City, Que.: IEEE, May 2001, pp. 145–152. ISBN: 0-7695-0984-3. DOI: 10.1109/IM.2001.924423.

[66] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. “Aligning point cloud views using persistent feature histograms”. In: *International Conference on Intelligent Robots and Systems (IROS)*. Nice: IEEE, Sept. 2008, pp. 3384–3391. ISBN: 978-1-4244-2057-5. DOI: 10.1109/IROS.2008.4650967.

[67] J. Salvi, C. Matabosch, D. Fofi, and J. Forest. “A review of recent range image registration methods with accuracy evaluation”. In: *Image and Vision Computing* 25.5 (May 2007), pp. 578–596. ISSN: 02628856. DOI: 10.1016/j.imavis.2006.05.012.

[68] Y. Seo, Y.-J. Choi, and S. W. Lee. “A branch-and-bound algorithm for globally optimal calibration of a camera-and-rotation-sensor system”. In: *International Conference on Computer Vision (ICCV)*. ICCV. Kyoto: IEEE, Sept. 2009, pp. 1173–1178. ISBN: 978-1-4244-4420-5. DOI: 10.1109/ICCV.2009.5459343.

[69] L. Svarm, O. Enqvist, M. Oskarsson, and F. Kahl. “Accurate Localization and Pose Estimation for Large 3D Models”. In: *Computer Vision and Pattern Recognition (CVPR)*. Columbus, OH: IEEE, June 2014, pp. 532–539. DOI: 10.1109/CVPR.2014.75.

[70] G. K. L. Tam, Zhi-Quan Cheng, Yu-Kun Lai, F. C. Langbein, Yonghuai Liu, D. Marshall, R. R. Martin, Xian-Fang Sun, and P. L. Rosin. “Registration of 3D Point Clouds and Meshes: A Survey from Rigid to Nonrigid”. In: *IEEE Transactions on Visualization and Computer Graphics* 19.7 (July 2013), pp. 1199–1217. ISSN: 1077-2626. DOI: 10.1109/TVCG.2012.310.

[71] P. W. Theiler, J. D. Wegner, and K. Schindler. “Fast registration of laser scans with 4-point congruent sets - what works and what doesn’t”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* II-3 (Aug. 7, 2014), pp. 149–156. ISSN: 2194-9050. DOI: 10.5194/isprsannals-II-3-149-2014.

[72] F. Tombari, S. Salti, and L. Di Stefano. “Performance Evaluation of 3D Keypoint Detectors”. In: *International Journal of Computer Vision* 102.1-3 (Mar. 2013), pp. 198–220. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-012-0545-4.

[73] J. Yang, H. Li, and Y. Jia. “Go-ICP: Solving 3D Registration Efficiently and Globally Optimally”. In: *International Conference on Computer Vision (ICCV)*. Sydney, NSW: IEEE, Dec. 2013, pp. 1457–1464. DOI: 10.1109/ICCV.2013.184.

[74] Y. Zhong. “Intrinsic shape signatures: A shape descriptor for 3D object recognition”. In: *International Conference on Computer Vision Workshops (ICCV Workshops)*. Kyoto: IEEE, 2009, pp. 689–696. ISBN: 978-1-4244-4442-7. DOI: 10.1109/ICCVW.2009.5457637.