Fast Rotation Search with Stereographic Projections for 3D Registration

Álvaro Parra Bustos, Tat-Jun Chin, *Member, IEEE,* Anders Eriksson, Hongdong Li, *Member, IEEE,* and David Suter, *Senior Member, IEEE*

Abstract—Registering two 3D point clouds involves estimating the rigid transform that brings the two point clouds into alignment. Recently there has been a surge of interest in using branch-and-bound (BnB) optimisation for point cloud registration. While BnB guarantees globally optimal solutions, it is usually too slow to be practical. A fundamental source of difficulty lies in the search for the rotational parameters. In this work, first by assuming that the translation is known, we focus on constructing a fast rotation search algorithm. With respect to an inherently robust geometric matching criterion, we propose a novel bounding function for BnB that is provably tighter than previously proposed bounds. Further, we also propose a fast algorithm to evaluate our bounding function. Our idea is based on using stereographic projections to precompute and index all possible point matches in spatial R-trees for rapid evaluations. The result is a fast and globally optimal rotation search algorithm. To conduct full 3D registration, we co-optimise the translation by embedding our rotation search kernel in a nested BnB algorithm. Since the inner rotation search is very efficient, the overall 6DOF optimisation is speeded up significantly without losing global optimality. On various challenging point clouds, including those taken out of lab settings, our approach demonstrates superior efficiency.

Index Terms—Point cloud registration, rotation search, branch-and-bound, stereographic projections, R-trees.

1 Introduction

Despite significant research efforts, point cloud registration remains a challenging problem. In general, given two point clouds, the goal of registration is to find a transform that maps points from one set to the other such that the two sets are aligned as well as possible. Many different formulations exist; see [1] for a survey. In this paper, we focus on registering two 3D point clouds that differ by a rigid transform. Such a problem occurs often in applications involving laser scanners or depth cameras, e.g., pose estimation and object detection.

For 3D rigid registration, many practical systems still rely on the classical ICP method [2], 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 [3], which also performs alternating optimisation. In many applications, the required initialisation is not available or is too laborious to be acquired. There is thus the need to consider algorithms that are globally convergent.

One of the earliest globally optimal registration methods was proposed by Breuel [4] for *geometric matching*, e.g., finding a previously seen configuration of 2D points in an input image. The method is based on branch-and-bound (BnB) optimisation [5], which guarantees global optimality. While Breuel's original formulation is fast enough for optimising 2D rigid transforms with 3DOF, a naive extension to estimate 3D rigid transforms with 6DOF is unwieldy, as the volume of the search space is significantly increased by going from 2D to 3D.

In practice the point clouds may only partially overlap, and points in the non-overlapping regions represent outliers. The geometric matching criterion is inherently robust to outliers since it only matches points within a distance threshold. In contrast, the original ICP [2] which subscribes to the maximum likelihood principle will attempt to match all the points, including outliers. This led to extensions such as ICP with M-estimators [6], [7] and trimmed ICP [8], which do not try to match all points. In general, the need to handle outliers further complicates the optimisation. Most robust ICP variants simply use alternating or iterative optimisation, which do not give globally optimal solutions. Further, trimmed ICP requires knowing the number of genuinely matching points beforehand, which is difficult to ascertain.

Typically, optimising the rotational parameters is less efficient than the translational parameters, due to the special structure of $SO(3)$. In this paper we focus on the rotation search subproblem: given two point clouds, calculate the rotation that best registers the points. By no means is this a trivial problem, and significant efforts have been devoted purely to rotation search [9], [10], [11], [12]. Our contribution is a fast BnB rotation search algorithm that optimises the geometric matching criterion. We exploit the geometry of rotational transforms

- *Parra Bustos, Chin and Suter are with the School of Computer Science, The University of Adelaide.*
  E-mail: {aparra, tjchin, dsuter}@cs.adelaide.edu.au
- *Eriksson is with the School of Electrical Engineering and Computer Science, Queensland University of Technology.*
  E-mail: anders.eriksson@qut.edu.au
- *Li is with the Research School of Engineering, College of Engineering and Computer Science, The Australian National University.*
  E-mail: hongdong.li@anu.edu.au

to derive a bounding function that is provably tighter than those proposed previously. This allows pruning of unpromising rotations to be conducted more aggresively in BnB, thus speeding up convergence.

Our bounding function is also amenable to very efficient evaluation. Specifically, we can precompute all possible point matches using *stereographic projections* [13] and index them in *circular R-trees* [14]. 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; our method can register up to 1000 points in 2 seconds.

To accomplish full 3D registration, we embed our rotation search "kernel" in a broader optimisation framework. Specifically, we optimise the 6DOF rigid transform with a nested BnB algorithm [15], whereby an outer BnB loop optimises the translation and the inner BnB loop conducts rotation search. The overall result is guaranteed to be 6DOF globally optimal. We show that our novel rotation search algorithm speeds up nested BnB tremendously without losing global optimality.

This paper is an extension of our previous work on pure rotation search [16]. The significant additions we have made include the usage of *matchlists* [17] to further speed up bound evaluations (Sec. 2.4), a plane sweep-inspired algorithm for pre-computing and indexing point matches (Sec. 3.3), the extension from pure rotation optimisation in [16] to 6DOF point cloud registration (Sec. 4), and a comprehensive benchmarking with state-of-the-art point cloud registration methods (Sec. 5).

1.1 Related work

1.1.1 Point cloud registration

Significant effort has been devoted to the rigid (Euclidean) registration of point clouds. Available techniques include randomised heuristics [18], [19], feature detection and matching [20], and methods that construct alternative representations for the point sets such as mixture models [21], [22] or Fourier transforms [23].

Of closer relevance to our work is the class of methods that employ mathematical optimisation to deterministically find the best solution. Within this class, two groups can be discerned. The first group [24], [25], [26] takes as input *a priori* established point matches between the two point clouds. Therefore, the quality of the registration is dependent on the veracity of the given point matches.

The second group of methods [4], [15], [27], [28] use the original point clouds without prior matching; our work belongs to this group. Though 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. In this paper, we focus on comparing against methods in the second group.

Breuel [4] was one of the earliest to apply BnB for point set registration. However, only 2D points were considered and a naive extension to 3D is unwieldy.

Li and Hartley [27] 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. [15] 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. Our 6DOF globally optimal algorithm (Sec. 4) is inspired by Yang et al.'s nested BnB approach. However, w.r.t. rotation BnB, stemming from the fact that we conduct geometric matching [4] instead of maximum likelihood [2], we can construct a much tighter bounding function. Further, our bounding function can be rapidly evaluated with innovations such as stereographic projections, R-trees and matchlists, all of which cannot be easily exploited under maximum likelihood; see Sec. 3 for details. By virtue of a much faster rotation search kernel, our 6DOF 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 [29], but only very small point sets can be handled. Enqvist et al. [28] 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.

1.1.2 Rotation search

Many recent works on rotation search were geared towards multi-view geometry problems [9], [10], [12], [30], [31], e.g., calculating relative camera poses. These methods can take advantage of image-based feature matches that relate data from one point set to the other. Although the matches can be erroneous (which necessitates the usage of robust costs to preclude wrong matches), the need for optimising point matching is obviated and rotation search is simplified tremendously. However, these methods are only optimal up to the set of "fixed" feature matches obtained by some other scheme. In contrast, since our technique operates directly on raw 3D points, it is not limited by the inaccuracy of feature matching. However, the associated optimisation job is harder. Indeed, Hartley and Kahl [9] remarked that *"the problems considered in (Breuel 2003) are harder in the sense that feature correspondences are not known a priori"*. We show how our novel bounding function significantly simplifies and speeds up rotation search.

Bazin et al. [12] also applied their method to rotational alignment of point clouds (3D keypoint matches must first be *a priori* established across the point clouds). We will compare our algorithm with [12] in the experiments.

2 Fast BnB rotation search

Here, we describe our BnB rotation search approach with a novel bounding function. In Sec. 3, we propose novel techniques to rapidly evaluate our bounding function, while in Sec. 4 we show how 6DOF registration can be achieved based on our rotation search kernel.

2.1 Objective function and BnB algorithm

Let $\mathcal{M} = \{\mathbf{m}_i\}_{i=1}^{M}$ and $\mathcal{B} = \{\mathbf{b}_j\}_{j=1}^{B}$ be two 3D point clouds which we assume are potentially related by a 3D rotation. Based on the geometric matching criterion, we seek the rotation $\mathbf{R} \in SO(3)$ that maximises

$$Q(\mathbf{R}) = \sum_i \max_j \lfloor \|\mathbf{R}\mathbf{m}_i - \mathbf{b}_j\| \leq \epsilon \rfloor, \tag{1}$$

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 (1) is robust since two points are matched only if their distance is less than the inlier threshold ϵ.

It is instructive to compare (1) against the maximum likelihood criterion in ICP, where we seek to minimise

$$E(\mathbf{R}) = \sum_i \min_j \|\mathbf{R}\mathbf{m}_i - \mathbf{b}_j\|. \tag{2}$$

For each point $\mathbf{R}\mathbf{m}_i$, its nearest neighbour in $\mathcal{B}$ must be found. Contrast this to (1) where as long as a sufficiently close point in $\mathcal{B}$ exists, we are not concerned with the distance of $\mathbf{R}\mathbf{m}_i$ to its nearest neighbour.

Algorithm 1 summarises our BnB algorithm for finding the globally optimal 3D rotation w.r.t. maximising (1). The basic idea is to recursively subdivide and prune the rotation space, until the global optimum is found. We employ the axis-angle parametrisation for rotations. A 3D rotation is represented as a 3-vector whose direction and norm specify the axis and angle of rotation. All rotations are thus contained in a π-ball. Initially we enclose the π-ball with a cube $\mathbb{B}$ of side 2π then successively subdivide it into eight smaller cubes. See [9] for details on the axis-angle parametrisation.

Crucially influencing the run time of Algorithm 1 is the tightness of the bounding function $\hat{Q}$, which satisfies

$$\hat{Q}(\mathbb{B}) \geq \max_{\mathbf{r} \in \mathbb{B}} Q(\mathbf{R}_\mathbf{r}), \tag{3}$$

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. We make contributions in both aspects, as described in Secs. 2.3 and 3.

2.2 Previous results

Given two rotation vectors $\mathbf{u}$ and $\mathbf{v}$ in the π-ball, it has been established [9] that

$$\angle(\mathbf{R}_\mathbf{u}\mathbf{m}, \mathbf{R}_\mathbf{v}\mathbf{m}) \leq \|\mathbf{u} - \mathbf{v}\|, \tag{4}$$

Algorithm 1 BnB algorithm to maximise (1).

Require: Point sets $\mathcal{M}$ and $\mathcal{B}$, threshold ϵ.
1: Initialise priority queue q, $\mathbb{B} \leftarrow$ cube of side 2π, $Q^* \leftarrow 0$, $\mathbf{R}^* \leftarrow \emptyset$.
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 $\mathbf{m}$ 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,

$$\mathbf{c} := (\mathbf{p} + \mathbf{q})/2 \tag{5}$$

is the centre of $\mathbb{B}$ with rotation matrix $\mathbf{R}_\mathbf{c}$. For any rotation $\mathbf{u}$ situated in the cube $\mathbb{B}$, we can see that

$$\begin{aligned} \angle(\mathbf{R}_\mathbf{c}\mathbf{m}, \mathbf{R}_\mathbf{u}\mathbf{m}) &\leq \max_{\mathbf{u} \in \mathbb{B}} \|\mathbf{c} - \mathbf{u}\| \\ &= \|\mathbf{p} - \mathbf{q}\|/2 := \alpha_\mathbb{B} \end{aligned} \tag{6}$$

as a direct consequence of (4). It thus follows that

$$\|\mathbf{R}_\mathbf{c}\mathbf{m} - \mathbf{R}_\mathbf{u}\mathbf{m}\| \leq \delta, \tag{7}$$

where the bound δ is based on the cosine rule

$$\delta = \sqrt{2\|\mathbf{m}\|^2(1 - \cos\alpha_\mathbb{B})}. \tag{8}$$

The result (7) immediately suggests the following bounding function for the objective function (1):

$$\hat{Q}_{br}(\mathbb{B}) = \sum_i \max_j \lfloor \|\mathbf{R}_\mathbf{c}\mathbf{m}_i - \mathbf{b}_j\| \leq \epsilon + \delta_i \rfloor, \tag{9}$$

where we define δ_i as (8) evaluated with $\mathbf{m}_i$. This bounding function was also originally proposed by Breuel; see [4] for proof that (9) is a valid bound for (1).

The bounding function (9) is unnecessarily conservative. Geometrically, the result (7) says that $\mathbf{R}_\mathbf{u}\mathbf{m}_i$ may lie anywhere within a ball of radius δ_i centred at $\mathbf{R}_\mathbf{c}\mathbf{m}_i$. Intuitively, we know that this is inaccurate, since the actions of all possible rotations in $\mathbb{B}$ may only allow $\mathbf{m}_i$ to lie on a patch on the surface of the sphere with radius $\|\mathbf{m}_i\|$; see Fig. 1. Our method exploits this key insight.

2.3 Improving the tightness of the bound

Let $S_\theta(\mathbf{m})$ represent the *spherical patch* (see Fig. 1) centred at $\mathbf{m}$ with angular radius θ, i.e.,

$$S_\theta(\mathbf{m}) = \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}\| = \|\mathbf{m}\|,\ \angle(\mathbf{m}, \mathbf{x}) \leq \theta\}. \tag{10}$$

$S_{2\pi}(\mathbf{m})$ is thus the sphere of radius $\|\mathbf{m}\|$ centred at the origin, and $S_\theta(\mathbf{m}) \subseteq S_{2\pi}(\mathbf{m})$. Further, the outline of

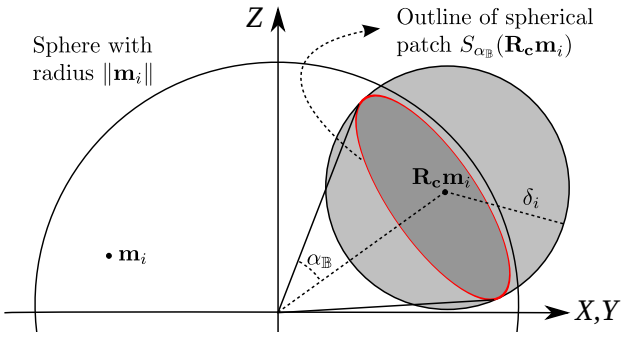


Fig. 1. Under the action of all possible rotations in $\mathbb{B}$, $\mathbf{m}_i$ may lie only on a spherical patch centered at $\mathbf{R_c m}_i$. However the bounding function (9) assumes that $\mathbf{m}_i$ may lie in the δ_i-ball centred at $\mathbf{R_c m}_i$.

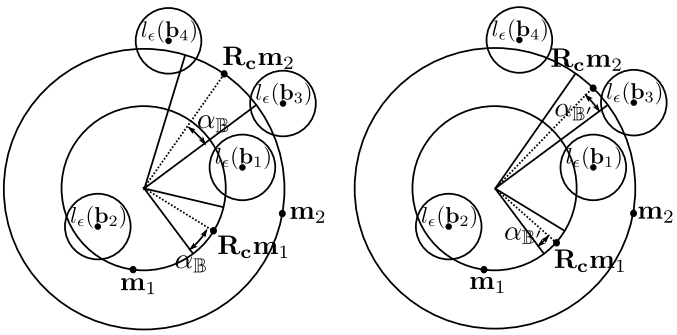


Fig. 2. Illustrating the idea of matchlists on 1D rotation search - an analogous idea exists for 3D rotation search. Here, the origin is at the centre of the largest circle. (Left) Under the actions of all possible rotations in an interval $\mathbb{B} \subseteq [0, 2\pi]$, $\mathbf{m}_1$ cannot match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

$S_\theta(\mathbf{m})$ is a circle on the surface of $S_{2\pi}(\mathbf{m})$. Using the above notation, we can reexpress the result (6) as

$$\mathbf{R_u m} \in S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}) \tag{11}$$

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

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

$$l_\epsilon(\mathbf{b}) = \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{b}\| \leq \epsilon\}. \tag{12}$$

The objective function (1) can be rewritten as

$$Q(\mathbf{R}) = \sum_i \max_j \lfloor \mathbf{R m}_i \in l_\epsilon(\mathbf{b}_j) \rfloor . \tag{13}$$

From (11), since $\mathbf{m}_i$ can only lie in $S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}_i)$ under all possible rotations in $\mathbb{B}$, determining if $\mathbf{m}_i$ can match with $\mathbf{b}_j$ under $\mathbb{B}$ amounts to checking if $S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}_i)$ intersects with $l_\epsilon(\mathbf{b}_j)$. This leads to the upper bound

$$\hat{Q}_{sp}(\mathbb{B}) = \sum_i \max_j \lfloor S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}_i) \cap l_\epsilon(\mathbf{b}_j) \neq \emptyset \rfloor . \tag{14}$$

To qualify as a valid bounding function for BnB, $\hat{Q}_{sp}$ has to meet several conditions, which we prove below.

Lemma 1. *For any cube* $\mathbb{B}$

$$\hat{Q}_{sp}(\mathbb{B}) \geq \max_{\mathbf{r} \in \mathbb{B}} Q(\mathbf{R_r}). \tag{15}$$

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

$$\hat{Q}_{sp}(\mathbb{B}) = Q(\mathbf{R_r}). \tag{16}$$

Proof. To prove (15), it is sufficient to show that if the pair $\mathbf{m}_i$ and $\mathbf{b}_j$ 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 $\mathbf{m}_i$ and $\mathbf{b}_j$ contribute 1 to $Q(\mathbf{R_r})$, then $\|\mathbf{R_r m}_i - \mathbf{b}_j\| \leq \epsilon$ and $\mathbf{R_r m}_i \in l_\epsilon(\mathbf{b}_j)$. Since $\mathbf{r}$ is in $\mathbb{B}$ then $\mathbf{R_r m}_i$ must lie in $S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}_i)$; see (11). This proves that the intersection $S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}_i) \cap l_\epsilon(\mathbf{b}_j)$ contains at least the item $\mathbf{R_r m}_i$ and is thus nonempty.

To prove (16), based on (6) as $\mathbb{B}$ collapses to a single point, $\mathbf{p} = \mathbf{q} = \mathbf{c}$ and $\alpha_{\mathbb{B}} = 0$. Thus $S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}_i)$ collapses to a single point $\mathbf{R_c m}_i$, rendering (14) to equal (13). □

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

Lemma 2. *For any cube* $\mathbb{B}$

$$\hat{Q}_{br}(\mathbb{B}) \geq \hat{Q}_{sp}(\mathbb{B}). \tag{17}$$

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

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.

2.4 Matchlists

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

$$\mathcal{N} = \{\mathbf{m} \in \mathcal{M} \mid \exists \mathbf{b} \in \mathcal{B},\ \ S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}) \cap l_\epsilon(\mathbf{b}) \neq \emptyset\}, \tag{18}$$

where $\mathbf{R_c}$ and $\alpha_{\mathbb{B}}$ are as defined in (6) for $\mathbb{B}$. For a subcube $\mathbb{B}' \subseteq \mathbb{B}$, it can be established that

$$\hat{Q}_{sp}(\mathbb{B}') \leq \hat{Q}_{sp}(\mathbb{B}) = |\mathcal{N}|. \tag{19}$$

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 $\hat{Q}_{sp}(\mathbb{B}')$. Fig. 2 illustrates the idea.

Breuel called $\mathcal{N}$ the *matchlist* of $\mathbb{B}'$ [17]. Using matchlists avoids redundant intersection queries in (14), especially in the later stages of BnB. To apply the idea in Algorithm 1, when inserting a cube $\mathbb{B}$ into the queue, we also record 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 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 maximum likelihood criterion (2), since each $\mathbf{m}_i$ must always be matched to its nearest point in $\mathcal{B}$ regardless of the distance.

3 Efficient bound evaluations

Using a tighter bound in BnB can be counterproductive if evaluating the bound itself takes significant time. Kd-tree is the main workhorse in [4] for evaluating Q and $\hat{Q}_{br}$. Points in $\mathcal{B}$ are indexed in a single kd-tree which is queried during BnB with rotated points from $\mathcal{M}$. This takes $\mathcal{O}(M \log B)$ effort per function evaluation.

To evaluate the proposed bound (14), we need to solve multiple queries of the following kind:

$$\max_j \lfloor S_{\alpha_{\mathbb{B}}}(\mathbf{R_c}\mathbf{m}_i) \cap l_\epsilon(\mathbf{b}_j) \neq \emptyset \rfloor . \tag{20}$$

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

$$\mathcal{B}_{\mathbf{m}_i} = \{\mathbf{b}_j \mid \mathbf{b}_j \in \mathcal{B}, |\|\mathbf{m}_i\| - \|\mathbf{b}_j\|| \leq \epsilon\}, \tag{21}$$

and the maximisation in (20) can be taken over just $\mathcal{B}_{\mathbf{m}_i}$. Sec. 3.3 will provide an efficient algorithm for finding $\mathcal{B}_{\mathbf{m}_i}$ for all i, given two point clouds $\mathcal{M}$ and $\mathcal{B}$.

3.1 Kd-tree approach

To evaluate (20) quickly, we can index each $\mathcal{B}_{\mathbf{m}_i}$ with a kd-tree. A total of M kd-trees are thus constructed. Given $\mathbb{B}$, we can perform a *range query* on the i-th kd-tree with point $\mathbf{R_c}\mathbf{m}_i$ and range $(\delta_i + \epsilon)$ (recall that $S_{\alpha_{\mathbb{B}}}(\mathbf{R_c}\mathbf{m}_i)$ is enclosed by the $(\delta_i + \epsilon)$-ball centred at $\mathbf{R_c}\mathbf{m}_i$). This disregards points in $\mathcal{B}$ that will never match with $\mathbf{m}_i$. Evaluating (14) thus takes $\mathcal{O}(M \log B_{av})$ effort, where $B_{av} \leq B$ is the average size of $\{\mathcal{B}_{\mathbf{m}_i}\}_{i=1}^M$.

3.2 Circular R-tree approach

While the M kd-tree approach permits faster bound evaluation than naive search, we propose another technique that gives bigger computational gains. Continuing the above observations, each $l_\epsilon(\mathbf{b}_j)$ for $\mathbf{b}_j \in \mathcal{B}_{\mathbf{m}_i}$ intersects $S_{2\pi}(\mathbf{m}_i)$ at a spherical patch — recall that a sphere-to-sphere intersection yields a circle, i.e., the outline of the spherical patch, see Fig. 3. The size of the patch depends on the distance of $\mathbf{b}_j$ to $S_{2\pi}(\mathbf{m}_i)$.

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

$$\mathbf{x} = \left[\frac{2p_1}{1+\mathbf{p}^T\mathbf{p}},\ \frac{2p_2}{1+\mathbf{p}^T\mathbf{p}},\ \frac{\mathbf{p}^T\mathbf{p}-1}{1+\mathbf{p}^T\mathbf{p}}\right]^T . \tag{22}$$

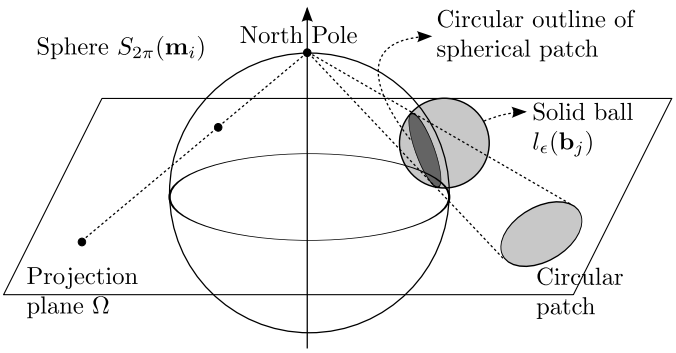


Fig. 3. A solid ball $l_\epsilon(\mathbf{b}_j)$ intersects the surface of the sphere $S_{2\pi}(\mathbf{m}_i)$ at a spherical patch, which has a circular outline on the sphere. Under stereographic projection, the spherical patch is projected to become a circular patch.

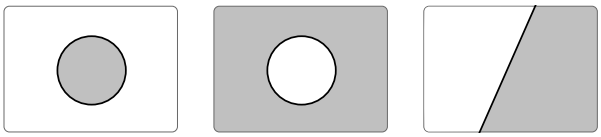


Fig. 4. The three types of patches arising from projecting spherical patches. (a) Interior patch. This is the case shown in Fig. 3. (b) Exterior patch. The spherical patch contains the pole, thus the "contents" of the spherical patch are projected outside the circle. (c) Half-plane. The pole lies exactly on the outline of the spherical patch.

The crucial property is that circles are projected as circles; see Fig. 3. 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 (22) into the plane equation yields

$$(c-d)(p_1^2 + p_2^2) + 2ap_1 + 2bp_2 - (c+d) = 0. \tag{23}$$

If $c \neq d$, (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 [13] for details.

A spherical patch is thus projected to become a *circular patch* in Ω; see Fig. 3. Given the circular patches from $\mathcal{B}_{\mathbf{m}_i}$, to solve (20) we first stereographically project $S_{\alpha_{\mathbb{B}}}(\mathbf{R_c}\mathbf{m}_i)$ to obtain the query patch $\mathcal{L}_q$, then check if $\mathcal{L}_q$ intersects any of the spherical patches from $\mathcal{B}_{\mathbf{m}_i}$. What makes this technique more efficient than the M kd-tree approach is the usage of *spatial access* data structures [14] to query for intersections. Note that the stereographic projection of a circle can be computed in closed form (constant time), thus it presents little overheads.

In the following, we 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 $\mathbf{R}\mathbf{m}_i$, the methods below can also be used to evaluate the quality (13).

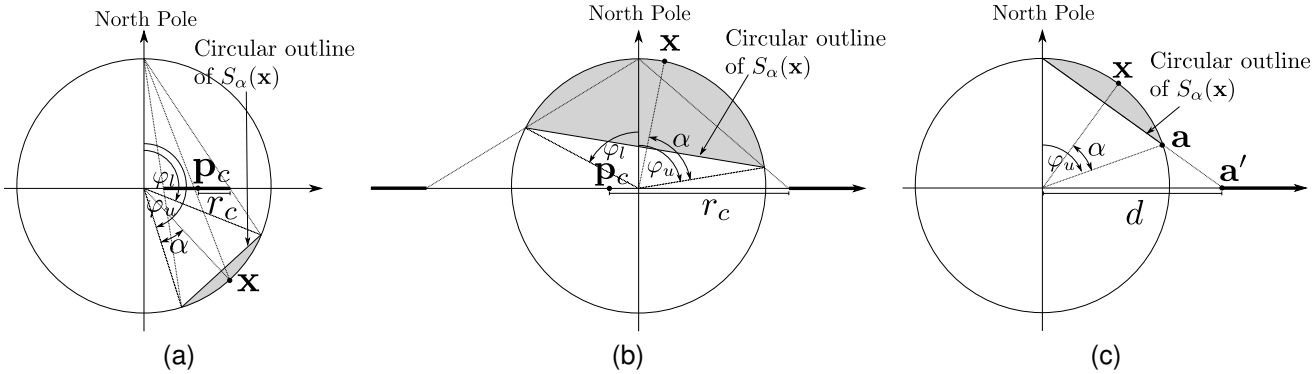


Fig. 5. Projection of a spherical patch $S_\alpha(\mathbf{x})$. The diagrams show the side view of Fig. 3, where the horizontal axis represents the xy-plane Ω. As explained in Fig. 4, three cases can arise: (a) an interior patch, (b) an exterior patch, or (c) a half-plane. In the above diagrams, the bolded segments on the horizontal axes indicate the resulting patches.

3.2.1 Projection of spherical patches

Discussing the full details of stereographic projection of circles is beyond the scope of this paper. We provide only the essential details in this subsection and refer the reader to the supplementary material or the text [13].

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 Fig. 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-planes*. Fig. 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 and azimuth. If $\varphi_\mathbf{x} - \alpha > 0$, $S_\alpha(\mathbf{x})$ does not contain the North Pole; see Fig. 5a. If $\varphi_\mathbf{x} - \alpha < 0$, the North Pole lies in the interior of $S_\alpha(\mathbf{x})$; see Fig. 5b. Finally, if $\varphi_\mathbf{x} - \alpha = 0$, the North Pole lies exactly on the circular outline of $S_\alpha(\mathbf{x})$; see Fig. 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. Define the radial distance of an inclination as

$$r(\varphi) = \frac{\sin(\varphi)}{1 - \cos(\varphi)}. \tag{24}$$

Let $\hat{\mathbf{x}}' \in \mathbb{R}^2$ be the unit vector of the *orthogonal* projection of $\mathbf{x}$ onto Ω. The maximum deviations in inclination of $\mathbf{x}$ in $S_\alpha(\mathbf{x})$ are given by $\varphi_l = \varphi_\mathbf{x} - \alpha$ and $\varphi_u = \varphi_\mathbf{x} + \alpha$. Then, the centre and radius of the circle in Ω are

$$\mathbf{p}_c = \frac{r(\varphi_l) + r(\varphi_u)}{2}\hat{\mathbf{x}}', \quad r_c = \frac{|r(\varphi_l) - r(\varphi_u)|}{2}. \tag{25}$$

See Figs. 5a and 5b. Now consider the $S_\alpha(\mathbf{x})$ that project to a half-plane. The half-plane is defined by

$$\hat{\mathbf{a}}'\mathbf{p} - d \geq 0 \quad \text{if } \varphi_u < \pi \tag{26}$$

$$\hat{\mathbf{a}}'\mathbf{p} - d < 0 \quad \text{if } \varphi_u \geq \pi \tag{27}$$

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), $\mathbf{a}$ is the point with inclination $\varphi_u = 2\alpha$ and $\hat{\mathbf{a}}'$ the unit vector in $\mathbb{R}^2$ corresponding to the *orthogonal* projection of $\mathbf{a}$ onto Ω. See Fig. 5c.

3.2.2 Indexation for fast intersection queries

Once the spherical patches from $\mathcal{B}_{\mathbf{m}_i}$ are projected onto the xy-plane Ω, we index them to facilitate efficient intersection queries. Here, we describe our indexing schemes for the three possible types of circular patches.

We first examine the indexation of exterior patches and half-planes which both arise from spherical patches containing or intersecting the North Pole. As these patches are infrequent in practice (in fact, no half-planes existed in our experiments), we simply index them in a list. Given a query patch $\mathcal{L}_q$, we scan the list to see if $\mathcal{L}_q$ intersects with any of the entries; as soon as a hit is encountered, we stop and return 1 to (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 (20) is dominated by testing the interior patches for overlaps with $\mathcal{L}_q$. To facilitate efficient querying, we index the interior patches (specifically, their circular outlines) in a *circular R-tree*. R-trees are indexing structures designed for *spatial access* queries; see [14] for a general exposition. In our 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, Fig. 6 shows a circular R-tree of depth three. The main parameter for tree building is the branching factor and maximum depth.

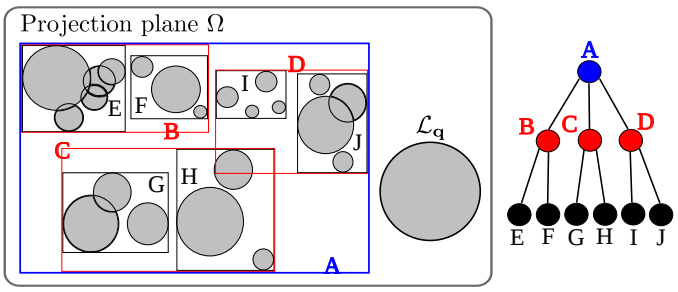


Fig. 6. A set of interior patches in the projection plane is indexed in a circular R-tree. The MBR at each node is also drawn. The tree structure is shown on the right. A query patch $\mathcal{L}_q$ is also shown; in this example, $\mathcal{L}_q$ does not intersect with the largest MBR at the root node, hence the search need *not* proceed beyond the root.

Regardless of the type of circular patch $\mathcal{L}_q$, querying the circular R-tree is conducted similarly; the distinction is just how 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 our circular R-tree, should (20) evaluate to 0, it is usually unnecessary to explore all tree levels. In many actual cases, only the first-few levels are descended; Fig. 6 shows an example. This difference in behaviour is the source of massive improvements in run time, as we will show in Sec. 5.1.

3.3 Modified plane sweep algorithm

As defined earlier, $\mathcal{B}_{\mathbf{m}_i} \subseteq \mathcal{B}$ is the set of points for which the ϵ-ball $l_\epsilon(\mathbf{b}_j)$ intersects with the sphere $S_{2\pi}(\mathbf{m}_i)$. A naive method to compute all $\{\mathcal{B}_{\mathbf{m}_i}\}_{i=1}^M$ is thus to test for each pair (i,j) whether $S_{2\pi}(\mathbf{m}_i)$ intersects with $l_\epsilon(\mathbf{b}_j)$. Testing all $M \times B$ pairs is wasteful, since not all the pairs intersect. We propose a more efficient method inspired by the plane sweep algorithm [32] used for calculating line segment intersections. See Algorithm 2.

We use concepts such as *status* and *events* from plane sweep. Specifically, we maintain a status containing the sorted values of the norms $\{\|\mathbf{m}_i\|\}_{i=1}^M$. We then iterate over events $\{\mathbf{b}_j\}_{j=1}^B$. For each event $\mathbf{b}_j$, we insert the values $\|\mathbf{b}_j\| - \epsilon$ and $\|\mathbf{b}_j\| + \epsilon$ into the status; let i_l and i_u respectively be the position of $\|\mathbf{b}_j\| - \epsilon$ and $\|\mathbf{b}_j\| + \epsilon$ in the status. Due to the presorting of the norms $\{\|\mathbf{m}_i\|\}_{i=1}^M$, any $\mathbf{m}_i$ whose index in status is below i_l or above i_u cannot intersect with $l_\epsilon(\mathbf{b}_j)$. Thus, the algorithm does not exhaustively test all $M \times B$ pairs of data for intersections. More specifically, the algorithm takes $\mathcal{O}(B \log M)$ effort, since we insert into a sorted array (the status) B times.

Algorithm 2 Modified plane sweep algorithm for finding the intersecting ϵ-ball set $\{\mathcal{B}_{\mathbf{m}_i}\}_{i=1}^M$.

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

3.4 Computational analysis

To evaluate the proposed bounding function (14), we will need 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 we will need to traverse the full depth of the tree and visit other branches (in both cases, we can bail out early since any overlapping interior patch or sufficiently close neighbour will do). In practice, however, we see significant speedups using circular R-trees. A reason behind this was given in Sec. 3.2.2.

Of secondary interest is the tree-building time, which occurs only once before the main loop of Algorithm 1. Given $\mathcal{B}_{\mathbf{m}_i}$, 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 [14] for details. Both types of trees will require finding $\{\mathcal{B}_{\mathbf{m}_i}\}_{i=1}^M$ using Algorithm 2.

In the next section, we will embark on utilising our novel rotation search algorithm for 6DOF registration. Readers who wish to first appraise the performance of our rotation search algorithm should skip to Sec. 5.1.

4 6 DOF REGISTRATION

Here, we show how our fast rotation search method can be used for full 3D (6DOF) point cloud registration.

4.1 Locally optimal method (Loc-GM)

We first present a locally optimal method. Whilst locally optimal methods for the maximum likelihood criterion are abundant [2], [7], [33], there appear to be no such algorithms for the geometric matching criterion.

Formally, we wish to maximise

$$Q(\mathbf{R}, \mathbf{t}) = \sum_i \max_j \lfloor \|\mathbf{R}\mathbf{m}_i + \mathbf{t} - \mathbf{b}_j\| \leq \epsilon \rfloor , \qquad (28)$$

where $\mathbf{t} \in \mathbb{R}^3$ is a translation vector. Our proposed algorithm is simple: given the current parameters $(\mathbf{R}^{(t)}, \mathbf{t}^{(t)})$,

we repeatedly update $\mathbf{R}$ and $\mathbf{t}$ by holding one of the components constant at each iteration. However, both subproblems are nonconvex. We conduct BnB to solve each subproblem *globally*. 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 [4]. Given $\mathbf{t}^{(t+1)}$, we obtain $\mathbf{R}^{(t+1)}$ via Algorithm 1. We solve the subproblems repeatedly until $Q(\mathbf{R}, \mathbf{t})$ does not change with the updates.

4.2 Globally optimal method (Glob-GM)

To conduct globally optimal 6DOF registration, we leverage on the *nested BnB* idea used in Go-ICP [15], 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, our inner BnB optimises rotation and the outer one optimises translation. See Algorithm 3.

Our goal is to maximise the objective function

$$Q(\mathbf{R}, \mathbf{t}) = \sum_i \max_j \lfloor \|\mathbf{R}(\mathbf{m}_i + \mathbf{t}) - \mathbf{b}_j\| \leq \epsilon \rfloor. \quad (29)$$

Here, to simplify exposition, we slightly abuse convention to define the rigid transform as $f(\mathbf{m}) = \mathbf{R}(\mathbf{m} + \mathbf{t})$; cf. (29) and (28). From the perspective of the outer BnB loop, the goal is to find the translation that maximises

$$V(\mathbf{t}) = \max_{\mathbf{R}} \sum_i \max_j \lfloor \|\mathbf{R}(\mathbf{x}_i + \mathbf{t}) - \mathbf{b}_j\| \leq \epsilon \rfloor, \quad (30)$$

where $V(\mathbf{t})$ is "evaluated" given $\mathbf{t}$ by invoking Algorithm 1 to rotationally align points $\mathcal{M} + \mathbf{t}$ and $\mathcal{B}$. Given a box of translations $\mathbb{T} \subset \mathbb{R}^3$ the upper bound

$$\hat{V}(\mathbb{T}) = \max_{\mathbf{R}} \sum_i \max_j \lfloor \|\mathbf{R}(\mathbf{m}_i + \mathbf{t}_c) - \mathbf{b}_j\| \leq \epsilon + \delta \rfloor, \quad (31)$$

can again be evaluated by using Algorithm 1 to rotationally align points $\mathcal{M} + \mathbf{t}_c$ and $\mathcal{B}$, 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 6DOF globally optimal [15].

Another crucial feature of Go-ICP that we adopt is an auxiliary local method to improve the quality of the current best solution - see Line 10 in Algorithm 3. It has been shown in [15] that such an auxiliary routine (ICP was used in their case) helps to significantly speed up the overall 6DOF algorithm. To this end, we use our locally convergent method Loc-GM described in Sec. 4.1.

Algorithm 3 Nested BnB algorithm to maximise (30).

Require: Point sets $\mathcal{M}$ and $\mathcal{B}$, threshold ϵ.
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 1 to align $\mathcal{M} + \mathbf{t_c}$ with $\mathcal{B}$ based on threshold ϵ.
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 4.1) 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 1 to align $\mathcal{M} + \mathbf{t_c}$ with $\mathcal{B}$ 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 $\mathbf{t}^*$.

5 Results

5.1 Rotation search

We first examine the efficiency of our 3D rotation search method (Sec. 2). The efficiency of our 6DOF point cloud registration algorithm (Sec. 4) will be analysed in Sec. 5.2.

Our experiment was designed as follows. We used scans of objects from three different sources, namely, the Stanford 3D Scanning Repository [34] (specifically *bunny*, *armadillo*, *dragon*, and *buddha*), Mian's dataset[1] (specifically *parasaur*, *t-rex* and *chicken*) and our own dataset of laser scans of underground mines (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. Fig. 7 shows the point clouds used in this experiment, whilst Columns 2–3 in Table 1 list the sizes of $\mathcal{V}_1$ and $\mathcal{V}_2$.

For each object, we generated realistic point clouds $\mathcal{M}$ and $\mathcal{B}$ as input for rotation search. This was accomplished by detecting 3D keypoints on the pair of input point clouds, extracting points in the neighbourhood of the keypoints; refer to supplementary material for the detailed procedure. To "normalise" the sizes we ensured $M \leq B$; since $\mathcal{B}$ is indexed in efficient data structures, its actual size is of less concern. See Column 4 in Table 1 for the average size of $\mathcal{M}$ for each object.

Note that our focus in this experiment is rotation search performance, thus we are not concerned with actually registering the point clouds (again, this will be examined in Sec. 5.2). We benchmark the following rotation search methods. All methods were implemented in C++ and executed on an Intel Core i7 3.40 GHz CPU.

- 1KDT: Breuel's original method [4], i.e., BnB with objective (1) and bound (9). The bound is evaluated using 1 kd-tree.
- MKDT: BnB with objective (13) and bound (14). The bound is evaluated using M kd-trees (see Sec. 3.1).
- MCIRC: BnB with objective (13) and bound (14). The bound is evaluated using steoreographic projection

1. http://staffhome.ecm.uwa.edu.au/~00053650/3Dmodeling.html

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 1
Comparing the performance of BnB rotation search methods using different bounds and bound evaluation methods. 1KDT: bound (9) using 1 kd-tree, MKDT: bound (14) using M kd-trees, MCIRC: bound (14) using M circular R-trees. 1KDT-ML, MKDT-ML and MCIRC-ML are variants of the above using matchlists (Sec. 2.4).

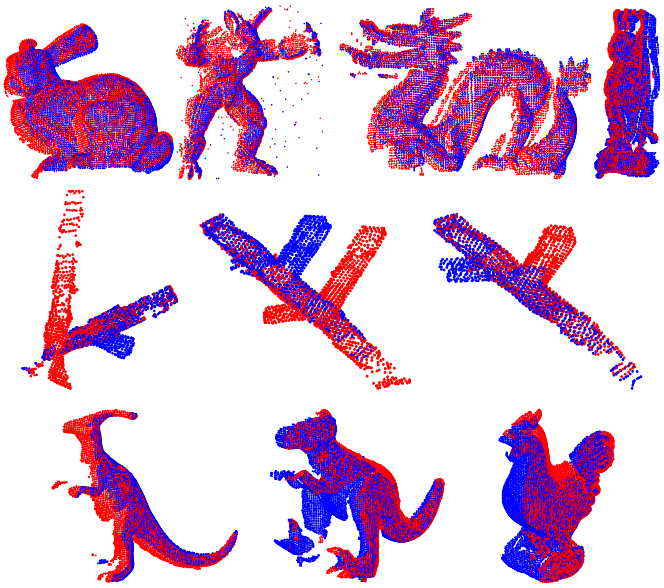


Fig. 7. Point clouds used in the evaluation of rotation search: *bunny*, *armadillo*, *dragon*, *buddha*, *mine-a*, *mine-b*, *mine-c*, *parasaur*, *t-rex* and *chicken*.

and M circular R-trees (see Sec. 3.2).
- 1KDT-ML, MKDT-ML and MCIRC-ML: Variants of the above with matchlists (see Sec. 2.4).

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

The average run time of all methods are listed in Columns 6–11 in Table 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 run time is an order of magnitude larger on the undergound mine dataset, possibly due to the more "organic" looking 3D structures. Also, using matchlists generally help 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, our 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 our previous method MCIRC [16].

5.1.1 Scalability of rotation search algorithm

To investigate the scalability of the rotation search algorithms, we repeated the above experiment with only the *armadillo* scans to avoid excessive run times. We also varied the neighbourhood size δ_{loc} to test a wider range of sizes of $\mathcal{M}$ and $\mathcal{B}$. In Fig. 8a, we plot the average run time as a function of the size of $\mathcal{M}$. Note that for the BnB rotation search algorithms, the size of $\mathcal{M}$ is a good representation of the problem size, since $\mathcal{B}$ is indexed in data structures and its size is not influential to run time. These results verify the superior performance of our algorithm on a large range of problem sizes.

5.1.2 Comparison with other BnB rotation search

Comparing with other formulations and techniques for rotation search is nontrivial, but we shall strive to quantitatively benchmark against [12]. While they also use BnB, there are crucial differences. First, their algorithm takes a set of point matches as input. In our case, Algorithm 1 does not require any *a priori* determined point matches between $\mathcal{M}$ and $\mathcal{B}$. Using point matches obviates the need to search for matches during BnB optimisation. Second, the error used in [12] is the *angular error* between matching points, while we use the l_2 distance. In the experiment of [12], 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 [12, Fig. 3] there seems to be approximately 100 keypoint matches. It was further reported that their algorithm "converges in a couple of seconds".

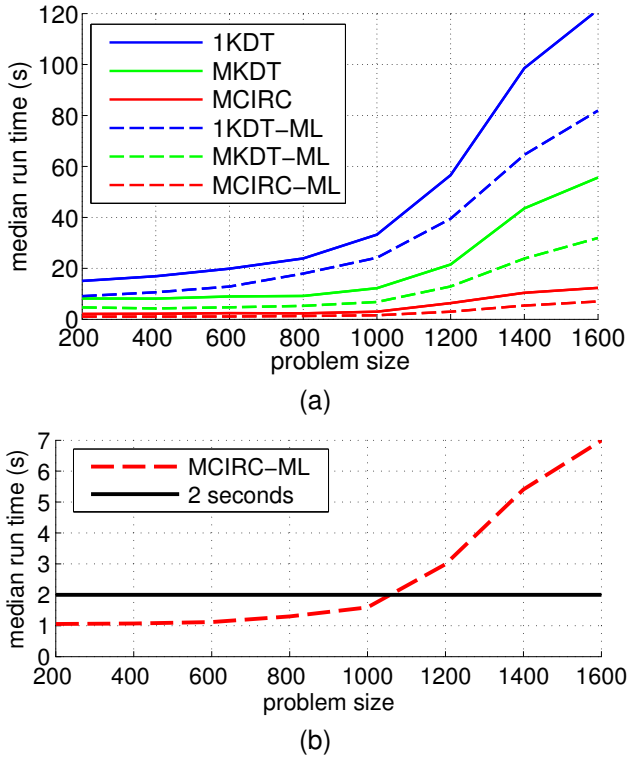


Fig. 8. (a) Median run time versus problem size M. (b) Median run time versus problem size M for MCIRC-ML. In this experiment, we ensured that the input point clouds $\mathcal{M}$ and $\mathcal{B}$ are of equal size so as compare against [12].

We reran the scalability experiment in Sec. 5.1.1. To obtain results that are comparable, here we ensured that the size of $\mathcal{B}$ is similar to the size of $\mathcal{M}$, 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). Fig. 8b shows the median run time of MCIRC-ML. Our algorithm is evidently superior, since it can align up to 1000 points within 2 seconds despite the need to conduct matching between $\mathcal{M}$ and $\mathcal{B}$ during the optimisation.

5.2 Globally optimal 6DOF registration

We now examine the performance of our globally optimal 3D registration method (Glob-GM, Algorithm 3) which encapsulates our fast rotation search algorithm. As described earlier in Sec. 1.1, the closest work to ours is Go-ICP [15], which globally minimises the maximum likelihood criterion (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 [2] was also incorporated to locally refine intermediate solutions. An equivalent step also exists in Glob-GM, where our novel local method Loc-GM (Sec. 4) is used to speed up convergence.

In the experiments in [15], nearest neighbours (NN) distance calculations were speeded up using distance transforms (DT) [7]. A DT is basically a discrete lookup table for NN distance values. If the data does not lie on an uniform grid, DT can only approximate the true NN distances. Thus, the global optimality guarantee of Go-ICP can be compromised by the approximations. We stress, however, that the original ideas of nested BnB and local refinement for speedup remain valid. In our experiment, we replace the DT in Go-ICP with a kd-tree, which gives exact NN distances (since the point sets lie in 3D, using kd-tree is ideal). Thus our implementation of Go-ICP can achieve the true global minimum.

We also compare Glob-GM against K-4PCS [35], 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 $\mathcal{B}$ are sampled and tested/matched against points from $\mathcal{M}$. 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. Since in our experiments, $\mathcal{M}$ and $\mathcal{B}$ were already subsampled (see below for details of our subsampling), we did not further conduct 3D keypoint detection for data reduction.

Two different settings were tested in this experiment:

- Full overlap: $\mathcal{M}$ is a subsample of $\mathcal{B}$, i.e., $\mathcal{M} \subset \mathcal{B}$. This is the same setting as that used in [15].
- Partial overlap: $\mathcal{M}$ and $\mathcal{B}$ are two different scans, thus not all the points in $\mathcal{M}$ have a match in $\mathcal{B}$.

Based on the objects used in Sec. 5.1, we created data for the above two settings as follows.

For the full overlap scenario, for each object, we selected and downsampled one of the scans to obtain $\mathcal{B}$. $\mathcal{M}$ is created as a sampled region of 100 points from $\mathcal{B}$. For the underground mine dataset, we performed the above for each individual scan, thus yielding six pairs of $\mathcal{M}$ and $\mathcal{B}$ which we named *mine-1* to *mine-6*. Fig. 10 shows the data in their initial (unregistered) poses.

For the partial overlap scenario, the point cloud pairs $\mathcal{M}$ and $\mathcal{B}$ were simply down-sampled versions of the original $\mathcal{V}_1$ and $\mathcal{V}_2$ used in Sec. 5.1. See Fig. 10 for the resulting data and their initial (unregistered) poses.

To avoid excessive run times, we fixed $\mathcal{M}$ between 400 and 1200 points for the full overlap case and between 180 and 410 points for the partial overlap scenario. The sizes of $\mathcal{M}$ and $\mathcal{B}$ are listed in Columns 2 and 3 of Tables 2 and 3 for both settings. Each point cloud set was uniformly scaled to fit the cube $[-50,\ 50]^3$. For further preprocessing details, see the supplementary material.

The following variants of our method were compared against Go-ICP and K-4PCS:

- Glob-GM-N: Algorithm 3 with local refinment (Step 10) disabled.
- Glob-GM: Algorithm 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 $\mathbf{m}_i$ in $\mathcal{M}$ must always be matched to a nearest point in $\mathcal{B}$ (see Sec. 2.4).

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 2 and 3 for the actual values. For Go-ICP, following the experiment in [15], the algorithm was terminated when the difference between the upper and lower bounds is $\leq \sqrt{0.05}$.

Two variants of K-4CPS were designed by changing the method settings (see supplementary material for details). The first variant K-4PCS-Quick was tuned to return fast approximate results, while the second variant K-4PCS-Quality was tuned to obtained high quality results. Since K-4PCS is randomised in nature, we report the median of 10 runs of each variant.

Tables 2 and 3 report the run times 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 (28).
- 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 $|\mathcal{M}| = 100$ 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 run times. In fact, in the case of full overlap, K-4PCS-Quality is as accurate as the globally optimal methods - we stress, however, that unlike the globally optimal algorithms, K-4PCS-Quality cannot guarantee 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 Fig. 10 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 (plots of results are presented in the supplementary material).

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 our 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.

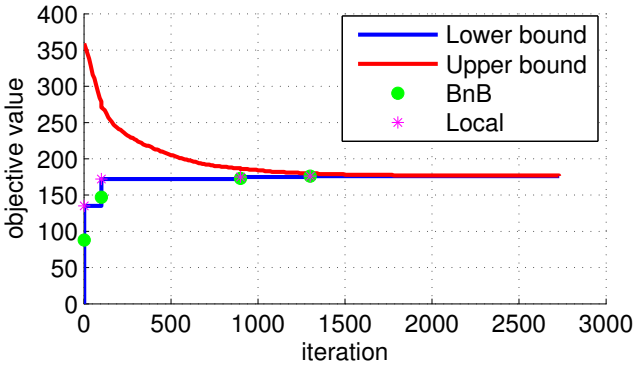


Fig. 9. Evolution of upper and lower bounds as a function of iteration count in our Glob-GM method (Algorithm 3). The time instances where a result is updated and then refined by the local method (Loc-GM) are also plotted. The algorithm is terminated only when the upper bound equals the lower bound (at $\approx$ the 2750-th iteration).

5.2.1 *Convergence of BnB algorithm*

To illustrate the convergence of Algorithm 3, Fig. 9 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. [15] to speed up BnB convergence.

Fig. 9 also shows that, similar to all BnB methods, the bulk of the time in Algorithm 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 we can terminate Algorithm 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 $\leq \sqrt{0.05}$).

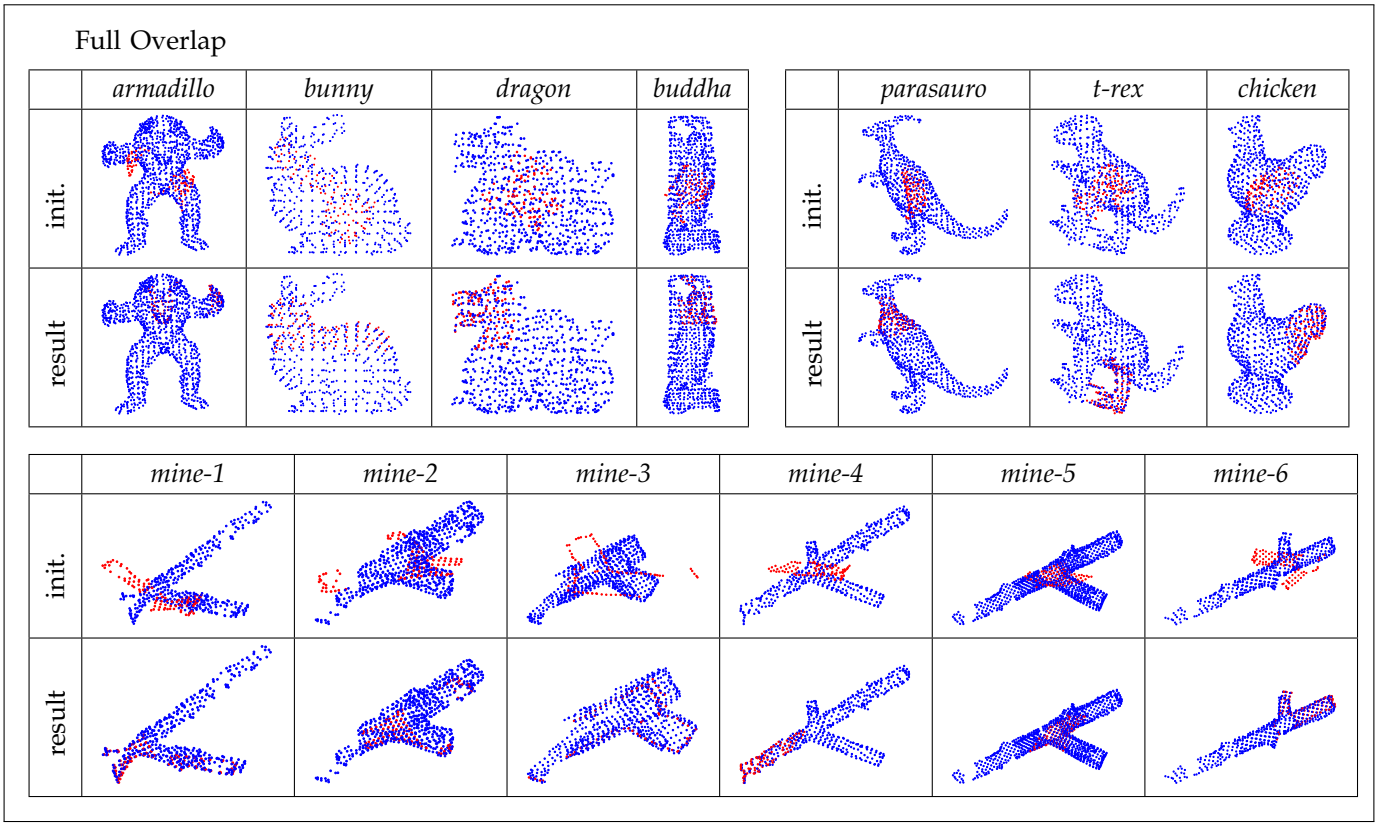
6 Conclusions

We have presented a novel BnB algorithm for 3D rotation search. Our method is based on the geometric matching criterion, and we proposed a novel bounding function that is provably tighter than previously available bounds. A very efficient algorithm to evaluate the bound based on stereographic projections and circular R-trees are also presented. Our globally optimal rotation search algorithm was shown to be an order of magnitude faster than the original BnB algorithm of Breuel [4].

We also presented a globally optimal 6DOF point cloud registration algorithm that encapsulates our rotation search method. Experimental results demonstrate the superior efficiency of our algorithm for point cloud registration, owing to a faster rotation search kernel.

Acknowledgments

This work was partly supported by the Australian Research Council grants DP130102524, DE130101775, DP120103896, DP130104567 and CE14-ACRV.



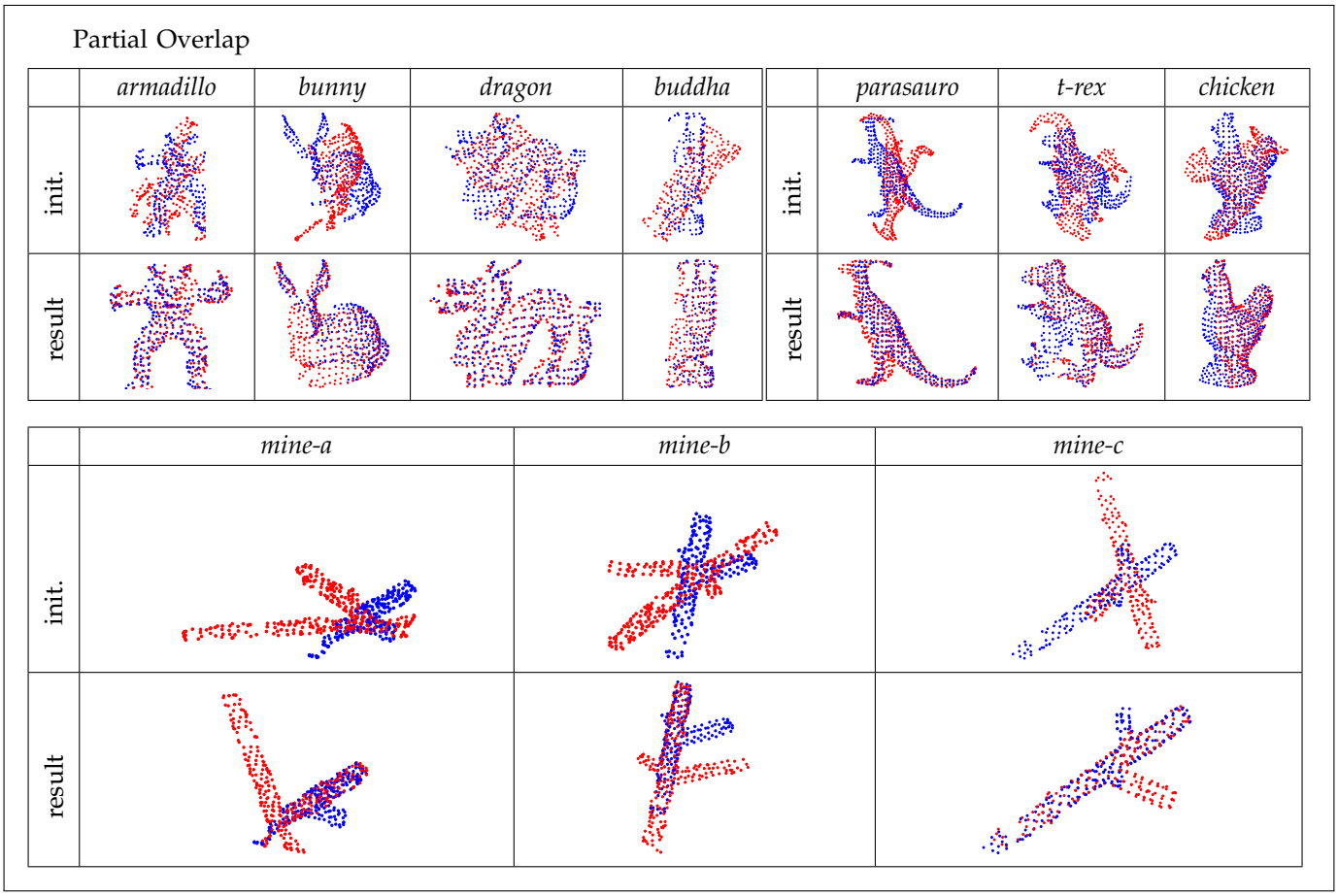


Fig. 10. Initial poses of point clouds and globally optimal results by Glob-GM under full and partial overlap scenarios.

	Full overlap (Glob-GM and variants)										
Dataset	$\|\mathcal{M}\|$	$\|\mathcal{B}\|$	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 and K-4PCS variants)															
Dataset	$\|\mathcal{M}\|$	$\|\mathcal{B}\|$	ϵ	Go-ICP					K-4PCS-Quick					K-4PCS-Quality				
				time (s)	Q	ang. err.	tr. err.	RMS	time (s)	Q	ang. err.	tr. err.	RMS	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	32.15	16	142.38	23.58	48.08	143.35	38	145.75	29.69	46.74
armadillo	100	1162	2.00	6321.26	100	4.5E-4	2.4E-4	3.0E-5	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	4314.57	100	3.5E-4	1.8E-4	2.3E-5	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	2027.46	100	1.4E-4	8.9E-5	1.4E-5	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	5171.75	100	1.8E-4	1.1E-4	1.2E-5	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	14137.80	100	4.4E-4	8.9E-5	1.8E-5	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	9936.01	100	2.2E-4	1.1E-4	1.7E-5	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	1270.40	100	1.3E-4	2.0E-5	1.0E-5	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	12496.60	100	1.1E-4	5.4E-5	1.4E-5	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	14723.70	100	5.7E-5	7.8E-5	1.9E-5	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	12189.10	100	7.7E-5	6.2E-6	6.0E-6	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 2

Comparing performance of 3D registration methods on point clouds with full overlap. Glob-GM: Alg. 3, Glob-GM-N: Alg. 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	$\|\mathcal{M}\|$	$\|\mathcal{B}\|$	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 and K-4PCS variants)															
Dataset	$\|\mathcal{M}\|$	$\|\mathcal{B}\|$	ϵ	Go-ICP					K-4PCS-Quick					K-4PCS-Quality				
				time (s)	Q	ang. err.	tr. err.	RMS	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 3

Comparing performance of 3D registration methods on point clouds with partial overlap. Glob-GM: Alg. 3, Glob-GM-N: Alg. 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.

REFERENCES

[1] G. K. L. Tam, Z.-Q. Cheng, Y.-K. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X.-F. Sun, and P. L. Rosin, "Registration of 3D point clouds and meshes: a survey from rigid to non-rigid," *IEEE TVCG*, vol. 19, no. 7, pp. 1199–1217, 2013.

[2] P. Besl and N. McKay, "A method for registration of 3D shapes," *IEEE TPAMI*, vol. 14, no. 2, pp. 239–256, 1992.

[3] S. Gold, A. Rangarajan, C. Lu, and E. Mjolsness, "New algorithms for 2D and 3D point matching: pose estimation and correspondence," *Pattern Recognition*, vol. 31, pp. 957–964, 1998.

[4] T. Breuel, "Implementation techniques for geometric branch-and-bound matching methods," *CVIU*, vol. 90, no. 3, pp. 258–294, 2003.

[5] R. Horst and H. Tuy, *Global optimization: deterministic approaches*. Springer, 2003.

[6] Z. Zhang, "Iterative point matching for registration of free-form curves," INRIA, Tech. Rep., 1992.

[7] A. Fitzgibbon, "Robust registration of 2D and 3D point sets," in *BMVC*, 2001.

[8] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *ICPR*, 2002.

[9] R. I. Hartley and F. Kahl, "Global optimization through rotation space search," *IJCV*, vol. 82, pp. 64–79, 2009.

[10] 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 *ICCV*, 2009.

[11] T. Ruland, T. Pajdla, and L. Kruger, "Globally optimal hand-eye calibration," in *CVPR*, 2012.

[12] J.-C. Bazin, Y. Seo, and M. Pollefeys, "Globally optimal consensus set maximization through rotation search," in *ACCV*, 2012.

[13] T. Needham, *Visual complex analysis*. Clarendon Press, 1997.

[14] Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos, and Y. Theodoridis, *R-trees: theory and applications*. Springer, 2006.

[15] J. Yang, H. Li, and Y. Jia, "Go-ICP: solving 3D registration efficiently and globally optimally," in *ICCV*, 2013.

[16] A. Parra Bustos, T.-J. Chin, and D. Suter, "Fast rotation search with stereographic projections for 3D registration," in *CVPR*, 2014.

[17] T. Breuel, "A comparison of search strategies for geometric branch and bound algorithms," in *ECCV*, 2002.

[18] 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," *IEEE TPAMI*, vol. 21, no. 11, pp. 1229–1234, 1999.

[19] D. Aiger, N. Mitra, and D. Cohen-Or, "4-points congruent sets for robust pairwise surface regisration," in *SIGGRAPH*, 2008.

[20] R. Gal and D. Cohen-Or, "Salient geometric features for partial shape matching and similarity," *ACM TOG*, vol. 25, pp. 130–150, 2006.

[21] B. Jian and B. Vemuri, "A robust algorithm for point set registration using mixture of Gaussians," in *ICCV*, 2005.

[22] A. Myronenko and X. Song, "Point set registration: coherent point drift," *IEEE TPAMI*, vol. 32, no. 1, pp. 2262–2275, 2010.

[23] A. Makadia, A. Patterson, and K. Daniilidis, "Fully Automatic Registration of 3D Point Clouds," in *CVPR*, 2006.

[24] N. Gelfand, N. Mitra, L. Guibas, and H. Pottmann, "Robust global registration," in *Eurographics*, 2005.

[25] C. Olsson, O. Enqvist, and F. Kahl, "A polynomial-time bound for matching and registration with outliers," in *CVPR*, 2008.

[26] E. Ask, O. Enqvist, and F. Kahl, "Optimal geometric fitting under the truncated L2-norm," in *CVPR*, 2013.

[27] H. Li and R. Hartley, "The 3D–3D registration problem revisited," in *ICCV*, 2007.

[28] O. Enqvist, K. Josephson, and F. Kahl, "Optimal correspondences from pairwise constraints," in *ICCV*, 2009.

[29] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *ICCV*, 2005.

[30] C. Olsson, F. Kahl, and M. Oskarsson, "Branch-and-bound methods for Euclidean registration problems," *IEEE TPAMI*, vol. 31, no. 5, pp. 783–794, 2009.

[31] J.-C. Bazin, H. Li, I. S. Kweon, C. Demonceaux, P. Vasseur, and K. Ikeuchi, "A branch and bound approach to correspondence and grouping problem," *IEEE TPAMI*, 2013.

[32] J. O'Rourke, *Computational geometry in C*. Cambridge University Press, 1998.

[33] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Int'l Conf. on 3-D Digital Imaging and Modeling (3DIM)*, 2001.

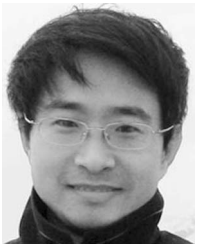
[34] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," *SIGGRAPH*, 1996.

[35] P. Theiler, J. Wegner, and K. Schindler, "Keypoint-based 4-points congruent sets - automated marker-less registration of laser scans," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 96, pp. 149 – 163, 2014.

Álvaro Parra received a BSc Eng. (2006), a Computer Science Engineer degree (2008) and a M.Sc. degree in Computer Science (2011) from *Universidad de Chile* (Santiago, Chile). He is currently a PhD student within the Australian Centre for Visual Technologies (ACVT) in the University of Adelaide, Australia. His main research interests include point cloud registration, 3D computer vision and optimisation methods in computer vision.

Tat-Jun Chin received a B.Eng. in Mechatronics Engineering from *Universiti Teknologi Malaysia* in 2003 and subsequently in 2007 a PhD in Computer Systems Engineering from Monash University, Victoria, Australia. He was a Research Fellow at the Institute for Infocomm Research in Singapore 2007- 2008. Since 2008 he is a Lecturer at The University of Adelaide, Australia. His research interests include robust estimation, computational geometry, and statistical learning methods in Computer Vision.

Anders Eriksson received the M.Sc. degree in electrical engineering and the PhD degree in mathematics from Lund University, Sweden, in 2000 and 2008, respectively. Currently, he is a senior research associate at the Queensland University of Technology, Australia. His research interests include optimization theory and numerical methods applied to the fields of computer vision and machine learning.



Hongdong Li received the M.Sc. and PhD degrees in information and electronics engineering in 1996 and 2000, respectively, both from Zhejiang University, China. Since 2004 he has joined the Australian National University (ANU) as a research fellow and is also seconded to National ICT Australia (NICTA) Canberra Labs. He is currently a faculty member with the Research School of Engineering, ANU. He is the recipient of the CVPR'12 Best Paper Award. His current research interests include geometric computer vision, pattern recognition, computer graphics, and combinatorial optimization.

David Suter received a B.Sc. degree in applied mathematics and physics (The Flinders University of South Australia 1977), a Grad. Dip. Comp. (Royal Melbourne Institute of Technology 1984), and a Ph.D. in computer science (La Trobe University, 1991). He was a Lecturer at La Trobe from 1988 to 1991; and a Senior Lecturer (1992), Associate Professor (2001), and Professor (2006-2008) at Monash University, Melbourne, Australia. Since 2008 he has been a professor in the school of Computer Science, The University of Adelaide. He served on the Australian Research Council (ARC) College of Experts from 2008-2010. He is on the editorial board of International Journal of Computer Vision and is co-Chair of the IEEE International Conference on Image Processing (ICIP2013).