Fast Rotation Search with Stereographic Projections for 3D Registration

Álvaro Parra Bustos, Tat-Jun Chin and David Suter
School of Computer Science, The University of Adelaide
{aparra, tjchin, dsuter}@cs.adelaide.edu.au

Abstract

Recently there has been a surge of interest to use branch-and-bound (bnb) optimisation for 3D point cloud registration. While bnb guarantees globally optimal solutions, it is usually too slow to be practical. A fundamental source of difficulty is the search for the rotation parameters in the 3D rigid transform. In this work, assuming that the translation parameters are 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 allows rapid evaluation. Underpinning our bounding function is the usage of stereographic projections to precompute and spatially index all possible point matches. This yields a robust and global algorithm that is significantly faster than previous methods.

To conduct full 3D registration, the translation can be supplied by 3D feature matching, or by another optimisation framework that provides the translation. On various challenging point clouds, including those taken out of lab settings, our approach demonstrates superior efficiency.

1. Introduction

Despite significant research efforts, 3D point cloud registration remains a formidable challenge; see [15] for a recent survey. The goal is to find the rigid transforms that bring two or more 3D point clouds into alignment. Many practical systems still rely on the classical ICP method [3], 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 can only converge to local optima. A similar weakness afflicts the well-known SoftAssign method [7], which also performs alternating optimisation. In many applications, the required initialisations may not be dependable or are too laborious to be acquired. There is thus the need to consider algorithms that are globally optimal.

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 edge map. The method is based on branch-and-bound (bnb) optimisation, 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 increased exponentially.

In practice the point clouds may only partially overlap, and points in the non-overlapping regions represent outliers. The geometric matching criterion [4] is inherently robust to outliers since it only matches points within a distance threshold. In contrast, the original ICP [3] 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 [17, 6] and trimmed ICP [5]. In general, however, the need to handle outliers further complicates the optimisation. Most robust ICP variants simply use alternating [17, 5] or iterative optimisation [6], which do not give globally optimal solutions.

In this paper, we focus on the rotation search subproblem: given the translation component of the 3D rigid transform, calculate the rotation to register the points. By no means is this a trivial problem, as we will show in the experiments. Significant efforts have also been devoted purely to rotation search [8, 14, 12, 2]. Our contribution is a fast bnb rotation search algorithm that optimises the geometric matching criterion [4]. We exploit the geometry of rotational transforms to derive a tight bounding function that is also amenable to very efficient evaluation. Specifically, we precompute all possible point matches using *stereographic projections* [11] and index them in *circular R-trees* [10]. This facilitates fast bound computations and speeds up the overall bnb algorithm. The result is a rotation search method that is robust, globally optimal *and* fast; our method can register up to 1000 points in 2 seconds.

To accomplish full 3D registration, our rotation search "kernel" can be embedded in a broader optimisation framework. One possibility is to use it in the nested bnb algorithm of [16] that contains two nested bnb loops for translation and rotation; the overall result is guaranteed to be globally optimal. Here, we suggest to generate candidate translations using 3D keypoint detection and matching tech-

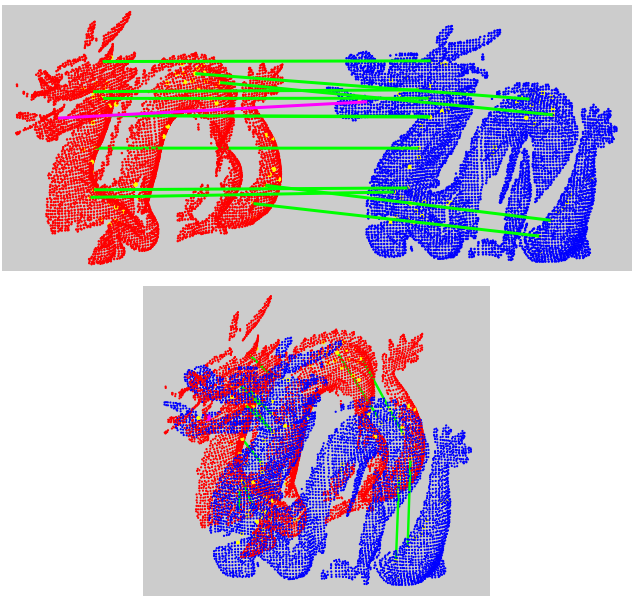


Figure 1. (Top) Our framework first obtains 3D keypoint matches across the input point clouds. Here, only a small subset of the matches are plotted to minimise clutter. (Bottom) Each keypoint match is used to translate the point clouds such that the matched points coincide. If the keypoint match is a true positive, the point clouds can be aligned by just a 3D rotation. Our main contribution is a fast and globally optimal algorithm for 3D rotation search. Note that our rotation search method uses the original (unmatched) points as input, not the keypoint matches.

niques [15]; note that a point match across two point clouds defines a 3D translation. Given a candidate translation we conduct rotation search to find the optimal rotational alignment. The best rotation and translation pair naturally gives rise to an aligning 3D rigid transform. Fig. 1 illustrates.

The rest of the paper is organised as follows: Sec. 2 defines our problem and surveys related work to put our paper in the right context. Sec. 3 describes the proposed rotation search method, while Sec. 4 provides experimental results and comparisons. Finally, we draw conclusions in Sec. 5.

2. Problem definition

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, e.g., $\mathcal{M}$ and $\mathcal{B}$ have been previously translated according to a keypoint match; see Fig. 1 (bottom). Following [4], we find the rotation $\mathbf{R} \in SO(3)$ that maximises

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

where $\lfloor \cdot \rfloor$ is the indicator function. The geometric matching criterion (1) is robust since two points are matched only if their distance is less than the inlier threshold ϵ. Our main contribution is a fast algorithm to globally maximise (1).

2.1. Related work

Many of the recent works on rotation search concentrated on multi-view geometry problems [8, 14, 12, 2]. Perhaps the most relevant to our work is [2], who also considered rotational alignment of point clouds. Given two input point clouds which are assumed to differ only by rotation, keypoint matches are first established. Their goal is to find the maximum consensus rotation (the rotation that agrees with the most number of keypoint matches), and a bnb algorithm was proposed. Note that in [2], only the keypoint matches are considered in the optimisation; contrast this to (1) where the original (unmatched) input points are used. This implies that at least three true-positive keypoint matches must exist for the result to be meaningful.

It is worthwhile to further compare our work with [2]. In our proposed 3D registration framework, each keypoint match gives rise to a pair $\mathcal{M}$ and $\mathcal{B}$, whose aligning rotation is then determined by maximising (1) — there are thus as many rotation search instances as the number of keypoint matches. For our approach to be successful, at least one true-positive keypoint match must be available. In [2], the authors only considered point clouds that differ purely by rotation. To allow for general 3D rigid transforms, we can apply their algorithm in our framework, whereby given a particular keypoint match, instead of translating the point clouds, we translate the set of keypoint matches to form the input to their algorithm [2]; see also Fig. 1 (bottom). On this basis, we will benchmark our result against [2].

Decoupling translation and rotation and estimating them separately is not a new idea — note that our main novelty is a highly efficient 3D rotation search method, not the framework in Fig. 1. In [1], towards the goal of estimating the motion of catadioptric cameras, the authors proposed to calculate the rotation via detecting parallel catadioptric lines, and the translation by a robust 2-point algorithm.

More recently, a nested bnb algorithm was proposed in [16] for 3D point cloud registration, where an "outer" bnb optimises the rotation, while an "inner" bnb searches for the translation *given* the rotation. Another interesting contribution of [16] is the usage of ICP to tighten the bound for the overall algorithm, thus speeding up convergence. We see our work as complementary to [16] by envisioning a nested bnb algorithm where the inner bnb conducts rotation search (using our novel method) *given* the translation. Moreover, the idea of using ICP to hasten convergence also applies.

An earlier work that adopted bnb for rotation search is [9]. A "Lipschitzised" objective function is formulated and globally optimised over the space of rotations. The method assumes one-to-one matching of the point clouds (i.e., no partial overlaps) which can be a severe limitation. Further, the performance is too slow to be practical.

Algorithm 1 Bnb rotation search to maximise (1).

Require: Point sets $\mathcal{M}$ and $\mathcal{B}$, threshold ϵ.

Initialise priority queue q, $\mathbb{B} \leftarrow$ cube of side 2π, $Q^* \leftarrow 0$, $\mathbf{R}^* \leftarrow null$.
Insert $\mathbb{B}$ into q.
while q is not empty **do**
  Obtain a box $\mathbb{B}$ from q.
  $\mathbf{R_c} \leftarrow$ centre rotation of $\mathbb{B}$.
  If $Q(\mathbf{R_c}) = Q^*$ then terminate.
  If $Q(\mathbf{R_c}) > Q^*$ then $Q^* \leftarrow Q(\mathbf{R_c})$, $\mathbf{R}^* \leftarrow \mathbf{R_c}$.
  Subdivide $\mathbb{B}$ into $\mathbb{B}_l$ and $\mathbb{B}_r$.
  If $\hat{Q}(\mathbb{B}_l) > Q^*$, insert $\mathbb{B}_l$ with priority $\hat{Q}(\mathbb{B}_l)$ into q.
  If $\hat{Q}(\mathbb{B}_r) > Q^*$, insert $\mathbb{B}_r$ with priority $\hat{Q}(\mathbb{B}_r)$ into q.
end while
return Optimal rotation $\mathbf{R}^*$ with quality Q^*.

3. Fast globally optimal rotation search

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 optima 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 [8]; all rotations are thus contained in a π-ball. Initially we enclose the π-ball with a cube of side 2π then successively subdivide it into smaller boxes. We employ binary subdivision: a box $\mathbb{B}$ is split equally along its longest side into two smaller boxes $\mathbb{B}_l$ and $\mathbb{B}_r$, with arbitrary tie-breaking if $\mathbb{B}$ is a cube. Octal subdivision [8] is also possible; if memory is not a limitation, the branching factor is not critical to the run time.

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

$$\hat{Q}(\mathbb{B}) \geq \max_{\mathbf{r} \in \mathbb{B}} Q(\mathbf{R_r}), \tag{2}$$

where $\mathbf{R_r}$ is the matrix form of rotation $\mathbf{r}$. A tighter $\hat{Q}$ will prune more agressively and result in fewer iterations. Of equal importance is the efficiency of evaluating Q and $\hat{Q}$, since they are called repeatedly. We make novel contributions in both aspects, as we describe in Secs. 3.2 and 3.3.

3.1. Previous results

Given two rotation vectors $\mathbf{u}$ and $\mathbf{v}$ in the π-ball, it has been established [8] that

$$\angle(\mathbf{R_u m}, \mathbf{R_v m}) \leq \|\mathbf{u} - \mathbf{v}\|, \tag{3}$$

where $\mathbf{m}$ is a 3D point, and $\angle(\cdot, \cdot)$ gives the angular distance. Further, given a box $\mathbb{B}$, let $\mathbf{p}$ and $\mathbf{q}$ be the lower-left-front and upper-right-back corner of $\mathbb{B}$ (see [8]). Then,

$$\mathbf{c} := 0.5(\mathbf{p} + \mathbf{q}) \tag{4}$$

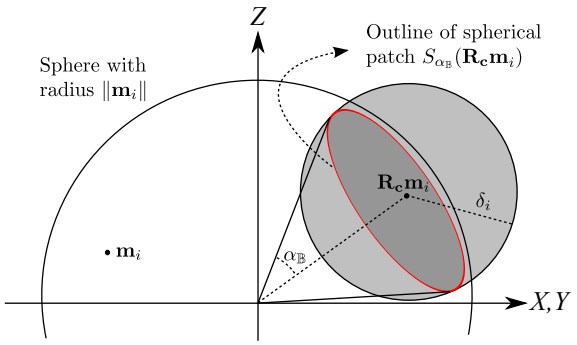


Figure 2. Under the action of all rotations in $\mathbb{B}$, $\mathbf{m}_i$ may lie only on a spherical patch centered at $\mathbf{R_c m}_i$. However the bounding function (8) assumes $\mathbf{m}_i$ may lie in the δ_i-ball centred at $\mathbf{R_c m}_i$.

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

$$\begin{aligned} \angle(\mathbf{R_c m}, \mathbf{R_u m}) &\leq \max_{\mathbf{u} \in \mathbb{B}} \|\mathbf{c} - \mathbf{u}\| \\ &= 0.5\|\mathbf{p} - \mathbf{q}\| := \alpha_\mathbb{B} \end{aligned} \tag{5}$$

as a direct consequence of (3). It thus follows that

$$\|\mathbf{R_c m} - \mathbf{R_u m}\| \leq \delta, \tag{6}$$

where the bound δ is based on the cosine rule

$$\delta = \sqrt{2\|\mathbf{m}\|^2(1 - \cos\alpha_\mathbb{B})}. \tag{7}$$

The result (6) immediately suggests the following bounding function for the geometric matching criterion (1):

$$\hat{Q}_{br}(\mathbb{B}) = \sum_i \max_j \lfloor \|\mathbf{R_c m}_i - \mathbf{b}_j\| \leq \epsilon + \delta_i \rfloor, \tag{8}$$

where we define δ_i as (7) evaluated with $\mathbf{m}_i$. This bounding function was also originally proposed by Breuel; see [4] for proof that (8) is a valid bounding function for (1).

The bounding function (8) is unnecessarily conservative. Geometrically, the result (6) says that $\mathbf{R_u m}_i$ may lie anywhere within a ball of radius δ_i centred at $\mathbf{R_c m}_i$ — 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. 2. Our method exploits this key insight.

3.2. Improving the tightness of the bound

Let $S_\theta(\mathbf{m})$ represent the *spherical patch* (see Fig. 2) centred at $\mathbf{m}$ with angular radius θ, i.e.,

$$S_\theta(\mathbf{m}) = \{\mathbf{x} \mid \|\mathbf{x}\| = \|\mathbf{m}\|,\ \angle(\mathbf{m}, \mathbf{x}) \leq \theta\}. \tag{9}$$

$S_{2\pi}(\mathbf{m})$ 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 $S_\theta(\mathbf{m})$

is a circle on the surface of $S_{2\pi}(\mathbf{m})$. Using the above notation, we can reexpress the result (5) as

$$\mathbf{R_u m} \in S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}) \tag{10}$$

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

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

$$l_\epsilon(\mathbf{b}) = \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{b}\| \leq \epsilon\}. \tag{11}$$

The objective function (1) can be rewritten as

$$Q(\mathbf{R}) = \sum_i \max_j \lfloor \mathbf{R m}_i \in l_\epsilon(\mathbf{b}_j) \rfloor . \tag{12}$$

From (10), 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 possibly 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

$$\hat{Q}_{sp}(\mathbb{B}) = \sum_i \max_j \lfloor S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}_i) \cap l_\epsilon(\mathbf{b}_j) \neq \emptyset \rfloor . \tag{13}$$

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 box* $\mathbb{B}$

$$\hat{Q}_{sp}(\mathbb{B}) \geq \max_{\mathbf{r} \in \mathbb{B}} Q(\mathbf{R_r}). \tag{14}$$

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

$$\hat{Q}_{sp}(\mathbb{B}) = Q(\mathbf{R_r}). \tag{15}$$

Proof. To prove (14), it is sufficient to show that if the pair $\mathbf{m}_i$ and $\mathbf{b}_j$ contribute 1 to $Q(\mathbf{R_r})$, they must also contribute 1 to $\hat{Q}_{sp}(\mathbb{B})$. If $\|\mathbf{R_r m}_i - \mathbf{b}_j\| \leq \epsilon$ then $\mathbf{R_r m}_i \in l_\epsilon(\mathbf{b}_j)$. 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 (10). 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.

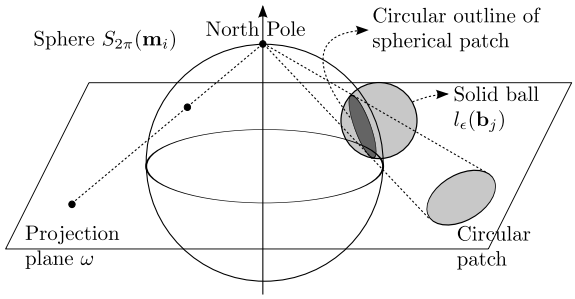
To prove (15), based on (5) as $\mathbb{B}$ collapses to a single point, $\mathbf{p} = \mathbf{q} = \mathbf{c}$ and $\alpha_{\mathbb{B}} = 0$. Thus $S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}_i)$ collapses to a single point $\mathbf{R_c m}_i$, rendering (13) to equal (12). □

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 the spherical patch while $\hat{Q}_{br}$ allows $\mathbf{m}_i$ to vary within a ball of radius that encloses the spherical patch. A formal proof is as follows.

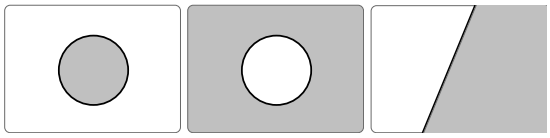
Lemma 2. *For any box* $\mathbb{B}$

$$\hat{Q}_{br}(\mathbb{B}) \geq \hat{Q}_{sp}(\mathbb{B}). \tag{16}$$

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 contributing 0 to $\hat{Q}_{sp}$. □



(a)



(b) (c) (d)

Figure 3. (a) A solid ball intersects the surface of a sphere at a spherical patch, which has a circular outline on the sphere. Under stereographic projection, a spherical patch is projected to become a circular patch. (b–d) The three types of circular patches: interior patch (this is the case in panel (a)), exterior patch (the spherical patch contains the pole; the "contents" of the spherical patch is projected outside of the circle), and half-plane (the pole lies on the circular outline of the spherical patch). See [11] for diagrams of projections that yield the two latter types.

3.3. Speeding up function evaluations

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 points from $\mathcal{M}$. Evaluating (1) or (8) thus takes $\mathcal{O}(M \log B)$ computational effort.

To evaluate the proposed bound (13) efficiently, we need to answer multiple queries like the following quickly:

$$\max_j \lfloor S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}_i) \cap l_\epsilon(\mathbf{b}_j) \neq \emptyset \rfloor \tag{17}$$

From Fig. 2, since $S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}_i)$ 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 $S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}_i)$. Further, this subset (which we call $\mathcal{B}_{\mathbf{m}_i}$) contains the $\mathbf{b}_j$'s with distance $\leq \epsilon$ to $S_{2\pi}(\mathbf{m}_i)$.

To evaluate (17) quickly, we can presearch $\mathcal{B}_{\mathbf{m}_i}$ and index it in a kd-tree. Given $\mathbb{B}$, we can perform a *range query* with point $\mathbf{R_c m}_i$ and range δ_i (recall that $S_{\alpha_{\mathbb{B}}}(\mathbf{R_c m}_i)$ is enclosed by the δ_i-ball centred at $\mathbf{R_c m}_i$). This disregards points in $\mathcal{B}$ that will never match with $\mathbf{m}_i$. By querying M separate kd-trees, evaluating (13) takes $\mathcal{O}(M \log B_{av})$ effort, where $B_{av} \leq B$ is the average size of $\{\mathcal{B}_{\mathbf{m}_i}\}_{i=1}^M$.

While the M kd-tree approach permits faster bound evaluation, 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(a). The size of the patch depends on the distance of $\mathbf{b}_j$ to $S_{2\pi}(\mathbf{m}_i)$.

Our idea is to use *stereographic projection* to project the spherical patches onto the xy-plane ω. Assuming a unit-sphere and a projection pole at $[0,0,1]^T$, a point $\mathbf{x}$ on the sphere and its stereographic projection $\mathbf{p}$ are related by [11]

$$\mathbf{x} = \left[\frac{2\mathbf{p}_1}{1+\mathbf{p}^T\mathbf{p}}, \frac{2\mathbf{p}_2}{1+\mathbf{p}^T\mathbf{p}}, \frac{\mathbf{p}^T\mathbf{p}-1}{1+\mathbf{p}^T\mathbf{p}}\right]^T. \quad (18)$$

The crucial property is that circles are projected as circles; see Fig. 3(a). To see this, recall that a circle results from the intersection between a plane τ and the sphere. Let τ be $[a\ b\ c]^T\mathbf{x} = d$. Putting (18) into the plane equation yields

$$(c-d)(\mathbf{p}_1^2+\mathbf{p}_2^2) + 2a\mathbf{p}_1 + 2b\mathbf{p}_2 - (c+d) = 0. \quad (19)$$

If $c \neq d$, (19) 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; see [11].

A spherical patch is projected to become a *circular patch* in ω. Recall that the outline of a spherical patch is a circle. Typically the vast majority of spherical patches do not intersect or contain the 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. A small minority of spherical patches that contain the pole will be projected to yield *exterior patches* or *half-planes* (both are special cases of circular patches). Figs. 3(b)–(d) show the three types.

Given the circular patches from $\mathcal{B}_{\mathbf{m}_i}$, to solve (17) we first stereographically project $S_{\alpha_{\mathbb{B}}}(\mathbf{R}_\mathbf{c}\mathbf{m}_i)$ to obtain the query patch $\mathcal{L}_q$, then check if $\mathcal{L}_q$ intersects any of the circular patches from $\mathcal{B}_{\mathbf{m}_i}$. In the following, we discuss efficient indexing and search schemes, based on the different types of circular patches from $\mathcal{B}_{\mathbf{m}_i}$. 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 objective function (12).

Exterior patches and half-planes. Due to the small number of occurrences in practice (in fact, no half-planes existed in our experiments), we simply store the exterior patches and half-planes in a list. Given $\mathcal{L}_q$, we scan the list to see if the non-empty region of $\mathcal{L}_q$ overlaps with the non-empty region of any of the entries; as soon as a hit is encountered, we stop and return 1 to (17). In fact, if $\mathcal{L}_q$ is itself an exterior patch or a half-plane, it will always overlap with an entry (since all the originating spherical patches contain and intersect at the North Pole) and the scan can be avoided.

Interior patches. Solving (17) is dominated by testing the interior patches from $\mathcal{B}_{\mathbf{m}_i}$ for overlaps. 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, e.g., is a polygon contained in another polygon; see [10] 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). Fig. 4 shows a tree of depth three. The main parameter for building the tree is the branching factor.

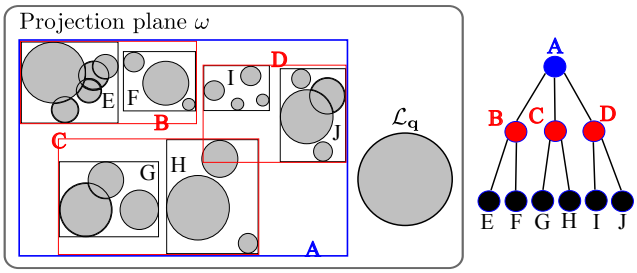


Figure 4. A set of interior patches in the projection plane is indexed in a circular R-tree. The MBR at each node is also drawn. The tree structure is shown on the right. A query patch $\mathcal{L}_q$ is also shown; in this example, $\mathcal{L}_q$ does not intersect with the largest MBR at the root 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 (17) evaluate to 0, it is usually unnecessary to explore all tree levels. In many actual cases, only the first-few levels are descended; Fig. 4 shows an example. This difference in behaviour is the source of massive improvements in run time, as we will show in Sec. 4.

Computational analysis. To evaluate the proposed bounding function (13), we will need to build and query M circular R-trees (the same number of trees required in the naive M-kd-tree approach). Search efficiency is of greater interest since querying occurs multiple times during bnb. Theoretically, R-trees and kd-trees have similar search complexities; 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.

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. Finding $\mathcal{B}_{\mathbf{m}_i}$ can be efficiently done using a kd-tree that indexes the *norms* of $\mathcal{M}$; any $\mathbf{b}_j$ with $|\|\mathbf{b}_j\| - \|\mathbf{m}_i\|| \leq \epsilon$ is placed in $\mathcal{B}_{\mathbf{m}_i}$.

4. Results

We first test our 3D rotation search method in the context of the 3D registration framework depicted in Fig. 1. We used 3D scan data from the Stanford repository, namely, *bunny*, *dragon*, *armadillo*, *asian dragon*, *buddha* and *statuette*. For each object, two partially overlapping scans $\mathcal{V}_1$ and $\mathcal{V}_2$ were chosen and downsampled; columns 2 and 3 in Table 1 list the size of each point cloud. Since only one scan is available for *statuette* and *asian dragon*, we generated a second scan by adding Gaussian noise to the points.

Apart from the Stanford data, we also tested our method on laser scans of underground mines. Laser scanning is a frequently used technique in mine surveying. Scan registration is needed since a single scan cannot image the whole mine. Two pairs of scans were used in our experiment: *mineA* and *mineB*; see Fig. 5. Such data is much more challenging due to the small partial overlap between scans.

To find the rigid transform $(\mathbf{R}, \mathbf{t})$ that registers $\mathcal{V}_1$ and $\mathcal{V}_2$, in our framework we first detect and match 3D keypoints across $\mathcal{V}_1$ and $\mathcal{V}_2$; we used ISS3D [18] and PFH [13] as implemented in Point Cloud Library[1]. The PFH descriptors were compared using the l_2 norm, and we chose a threshold such that exactly 100 keypoint matches were produced per $\mathcal{V}_1$ and $\mathcal{V}_2$ pair. The matching precision varied across the type of structure (column 5 in Table 1), but we verified that at least three true positive matches existed per $\mathcal{V}_1$ and $\mathcal{V}_2$ pair.

For each match $\mathbf{p} \leftrightarrow \mathbf{q}$, we translated $\mathcal{V}_1$ by $-\mathbf{p}$ and $\mathcal{V}_2$ by $-\mathbf{q}$ such that they potentially differ by a rotation about the origin. Further, instead of using all $\mathcal{V}_1$ and $\mathcal{V}_2$, we took only points within a radius δ_{loc} from $\mathbf{p}$ and $\mathbf{q}$; this produced the pair $\mathcal{M}$ and $\mathcal{B}$ as input for rotation search. The value δ_{loc} was chosen as a ratio of the point cloud extent and fixed for each $\mathcal{V}_1$ and $\mathcal{V}_2$ pair. Since our contribution is globally optimal rotation search, taking local point clouds does not detract from our contribution, as only the size of $\mathcal{M}$ and $\mathcal{B}$ matter for benchmarking. To "normalise" the sizes we ensured $|\mathcal{M}| \leq |\mathcal{B}|$ by swapping $\mathcal{M}$ and $\mathcal{B}$ if needed; note that in all the objective and bounding functions the main loop has exactly $|\mathcal{M}|$ iterations, thus it would be more strategic make $\mathcal{M}$ the smaller set. Column 4 in Table 1 lists the average $|\mathcal{M}|$ across all matches in the datasets.

We benchmark the following rotation search methods. All methods were implemented in C and executed on a machine with an Intel Core i7 3.40 GHz CPU.

- **bnb-M-circ**: the proposed method, i.e., bnb with objective (12) and bound (13) using M circular R-trees.
- **bnb-1-tree**: Breuel's original method [4], i.e., bnb with objective (1) and bound (8) using 1 kd-tree.
- **bnb-M-tree**: the same as **bnb-M-circ**, but using M kd-trees to index $\{\mathcal{B}_{\mathbf{m}_i}\}_{i=1}^{M}$; see Sec. 3.3.

[1] http://pointclouds.org/

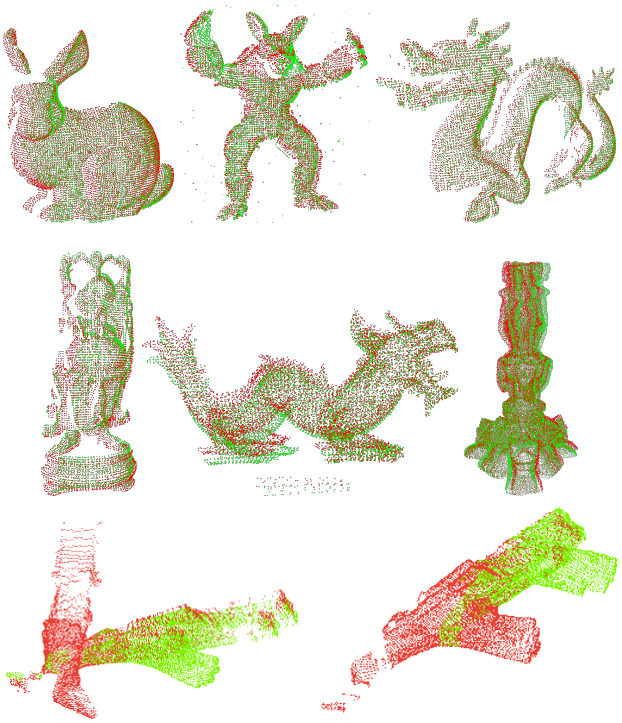


Figure 5. Point clouds successfully registered by our method: from the top-left, *bunny*, *armadillo*, *dragon*, *buddha*, *asian dragon*, *statuette*, *mineA* and *mineB*.

On each $\mathcal{V}_1$ and $\mathcal{V}_2$ pair, we performed rotation search on the local point clouds originating from the 100 keypoint matches. The durations required to perform 100 rotation searches are listed in columns 7–9 in Table 1; note that the recorded durations include time for all data structure preparations (e.g., building M kd-trees or circular R-trees).

Observe that **bnb-M-circ** can attain more than an order of magnitude speedup over **bnb-1-tree**. Compared to **bnb-M-tree**, our method gave an average of 5 times speedup; note that these two methods use exactly the same bounding function, hence, both will require the same number of iterations in Algorithm 1; their gap in actual run time is thus due primarily to the different bound evaluation techniques.

To obtain the desired registration between $\mathcal{V}_1$ and $\mathcal{V}_2$, across all 100 *optimised* local rotations we selected the one with the highest normalised match score, i.e., $Q(\mathbf{R}^*)/M$. Using that rotation and its originating keypoint match, we rotationally aligned $\mathcal{V}_1$ and $\mathcal{V}_2$ and evaluated the global match score Q_{glob}; this is simply (1) calculated using all available points. The result is in column 6 in Table 1. Visually, as shown in Fig. 5, all objects were satisfactorily registered. Note that all bnb methods give the same quality.

Given that our 3D registration framework employs keypoint matches, one may wonder how our rotation search approach compares with RANSAC. A minimal subset of three keypoint matches is sufficient to estimate a 3D rigid

	Data characteristics				3D rotation search				RANSAC		
Dataset	$\lvert\mathcal{V}_1\rvert$	$\lvert\mathcal{V}_2\rvert$	avg $\lvert\mathcal{M}\rvert$	% inliers	Q_{glob}	**bnb-1-tree** time (s)	**bnb-M-tree** time (s)	**bnb-M-circ** time (s)	med Q_{glob}	med time (s)	med iter
bunny	7055	6742	378.47	43	5377	376.47	128.59	22.11	3292	456.93	49978
armadillo	5619	5483	353.85	15	3243	406.50	152.66	33.09	2795	311.13	42210
dragon	6991	6200	351.45	29	5873	280.34	81.87	13.63	4433	405.88	48316
buddha	5312	5109	372.89	20	4512	331.18	102.45	19.61	3283	348.18	50000
a. dragon	8413	8413	103.16	13	5823	183.89	127.64	20.06	6813	241.96	18755
statuette	44156	44156	205.36	22	18328	163.87	70.83	9.13	14855	2169.25	49448
mineA	7629	7727	187.61	10	3197	2816.35	900.53	218.56	1089	418.54	50000
mineB	7496	5487	330.02	3	4870	6025.66	1978.23	631.13	1265	374.60	50000

Table 1. Quantitative registration results. Time for bnb methods is the total duration for 100 rotation searches.

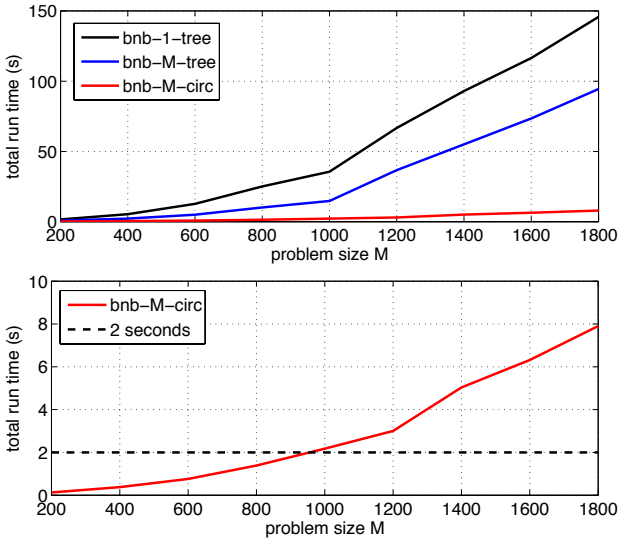


Figure 6. (top) Median run time versus problem size M. (bottom) Median run time for **bnb-M-circ** only.

transform. The quality was taken as (1) evaluated on all the points, i.e., Q_{glob}. We stopped the RANSAC sampling as soon as Q_{glob} equalled or surpassed the Q_{glob} of the bnb methods (note: this is possible since Q_{glob} is not directly optimised by the bnb methods), or if the number of iterations hit the limit of 50000. We then recorded the elapsed time (note: due to the 50000 iteration cutoff, RANSAC may have shorter run times). The median results from 20 instances of RANSAC are in columns 10-12 of Table 1.

On easier pairs like *asian dragon* where both scans fully overlap (hence, keypoint matching is more accurate), the quality of RANSAC can surpass bnb rotation search; however this was achieved using a much longer run time. On the other pairs RANSAC required significantly more time than **bnb-M-circ** without reaching close to the latter's quality. To achieve a quality close to the optimal, RANSAC will require iterations many times greater than the 50000 limit.

To investigate the scaling property of our method, we repeated the above experiment, but varied the neighbourhood size δ_{loc} such a larger range of sizes of $\mathcal{M}$ and $\mathcal{B}$ could be tested. In Fig. 6 (top), we plot the median run time of all bnb methods as a function of the size of $\mathcal{M}$. These results verify the superior performance of the proposed method.

Comparisons with [2]. Comparing with other formulations and techniques for rotation search is nontrivial, but we shall strive to quantitatively benchmark against [2]. 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 matches between $\mathcal{M}$ and $\mathcal{B}$ (in our registration framework, keypoint matches are only used for translating the points). Using point matches obviates the need to search for matches during function evaluations. Second, the error used in [2] is the *angular error* between matching points, while we use the l_2 distance.

In the real data experiments using Stanford *bunny* [2, Sec. 4.2], keypoint matches were first obtained on two partially overlapping scans that differed only by rotation. The sizes of point clouds, number of keypoint matches and outlier proportion were not explicitly reported. The authors did mention that, in the context of using RANSAC in their experiment, there were "8 inliers out of 1191 *points*" — we take this to mean that there were 1191 *keypoint matches* of which 8 were genuine[2]. Further, in the overview of their experiments [2, Sec. 4], it was reported that the algorithm "converges in a couple of seconds" — we shall take this as indicative of the run time in point cloud registration.

Fig. 6 (bottom) shows Fig. 6 (top) with only the run times from **bnb-M-circ**. At $\lvert\mathcal{M}\rvert = 1200$ (comparable to the 1191 data used in [2]), our method achieved a median run time of 3s, which is very close to the 2s in [2]. It should be noted that our excellent run time was achieved despite the need to search for matches online between $\mathcal{M}$ and $\mathcal{B}$.

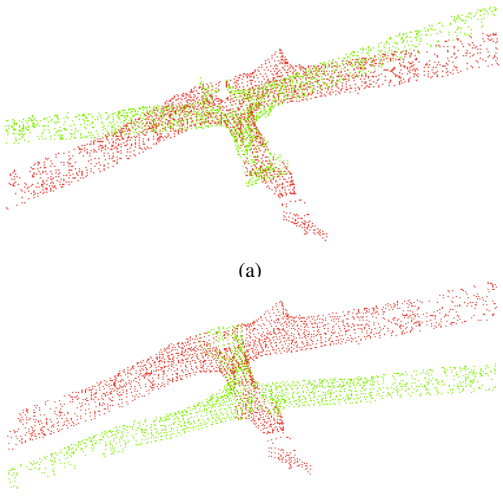
Comparisons with [16]. A globally optimal *nested* bnb algorithm was proposed to find the 3D rigid transform $(\mathbf{R}, \mathbf{t})$ that aligns two point clouds. The outer bnb loop optimises the rotation, while the inner bnb loop estimates the translation. Also, ICP was integrated into the algorithm to bootstrap the computation of the bound, such that convergence to the global optimum can be accelerated. Impressive run times were reported: on the Stanford *bunny* dataset, only 30 seconds were needed to register 1000 data points (equivalent to $\mathcal{V}_1$) to 30000–40000 model points (equivalent to $\mathcal{V}_2$). It should be noted, however, that nearest neigh-

[2]Observing Fig. 3 in [2], there seemed to be < 100 keypoint matches.

bours (NN) distance calculations were speeded up by using the distance transform (DT) [16, Sec. 7] which is basically a discrete lookup table. If the data does not lie on a uniform grid, DT can only approximate the true NN distances, and global optimality with respect to the original ICP criterion may be affected if that discrepancy is not taken into account.

Actually we see our rotation search algorithm as a possible component in [16]. In principle we can search for the translation in the outer bnb loop, and use our method in the inner loop for rotation optimisation. An auxiliary local method can also be used to hasten convergence. Note that all our function evaluations are exact, thus our method will speed up nested bnb without affecting global optimality.

A note on globally optimal registration. We are aware of practical cases where the globally optimal 3D rigid transform $(\mathbf{R}, \mathbf{t})$ does not provide the desired result. Fig. 7(a) shows the globally optimal *full 3D* registration (using the geometric matching criterion [4]) of two laser scans of an underground mine, which is *not* satisfactory. Despite using a robust criterion, since the structure is highly "organic" and self-similar (unlike the well defined Stanford objects) the result was heavily biased. It such cases, it will be useful to employ human assistance. Fig. 7(b) shows the correct outcome obtained by our rotation search method, after translating the data according to a point match identified by the user. Our fast rotation search algorithm can play a valuable role in such a user-assisted registration approach.



(a) (b)

Figure 7. (a) Globally optimal *full 3D* registration. (b) Globally optimal rotational alignment, based on a point match identified by the user. The robust matching score of (a) is greater than (b).

5. Conclusions

Our globally optimal rotation search algorithm was shown to be an order of magnitude faster than the original bnb algorithm of Breuel [4]. It also has good performance relative to other methods based on different formulations. Under our registration framework that uses keypoint matches, our algorithm was demonstrated to be accurate and efficient in registering partially overlapping point clouds.

Acknowledgements

This work is partly supported by the Australian Research Council grant DP130102524.

References

[1] J.-C. Bazin, C. Demonceaux, P. Vasseur, and I. S. Kweon. Motion estimation by decoupling rotation and translation in catadioptric vision. *CVIU*, 114:254–273, 2012. 2

[2] J.-C. Bazin, Y. Seo, and M. Pollefeys. Globally optimal consensus set maximization through rotation search. In *ACCV*, 2012. 1, 2, 7

[3] P. Besl and N. McKay. A method for registration of 3d shapes. *IEEE TPAMI*, 14(2):239–256, 1992. 1

[4] T. Breuel. Implementation techniques for geometric branch-and-bound matching methods. *CVIU*, 90(3):258–294, 2003. 1, 2, 3, 4, 6, 8

[5] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek. The trimmed icp algorithm. In *ICPR*, 2002. 1

[6] A. Fitzgibbon. Robust registration of 2D and 3D point sets. In *BMVC*, 2001. 1

[7] S. Gold, A. Rangarajan, C. Lu, and E. Mjolsness. New algorithms for 2d and 3d point matching: pose estimation and correspondence. *Pattern Recognition*, 31:957–964, 1998. 1

[8] R. I. Hartley and F. Kahl. Global optimization through rotation space search. *IJCV*, 82:64–79, 2009. 1, 2, 3

[9] H. Li and R. Hartley. The 3d–3d registration problem revisited. In *ICCV*, 2007. 2

[10] Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos, and Y. Theodoridis. *R-trees: theory and applications*. Springer, 2006. 1, 5

[11] T. Needham. *Visual complex analysis*. Clarendon Press, 1997. 1, 4, 5

[12] T. Ruland, T. Pajdla, and L. Kruger. Globally optimal hand-eye calibration. In *CVPR*, 2012. 1, 2

[13] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *IROS*, 2008. 6

[14] 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. 1, 2

[15] 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*, 19(7):1199–1217, 2013. 1, 2

[16] J. Yang, H. Li, and Y. Jia. Go-ICP: solving 3d registration efficiently and globally optimally. In *ICCV*, 2013. 1, 2, 7, 8

[17] Z. Zhang. Iterative point matching for registration of free-form curves. Technical report, INRIA, 1992. 1

[18] Y. Zhong. A shape descriptor for 3d object recognition. In *Proceedings ICCV 2009 Workshop 3DRR*, 2009. 6