Fast Rotation Search for Real-Time Interactive Point Cloud Registration

Tat-Jun Chin* Álvaro Parra Bustos* Michael S. Brown† David Suter*
School of Computer Science, The University of Adelaide, Australia*
School of Computing, National University of Singapore†

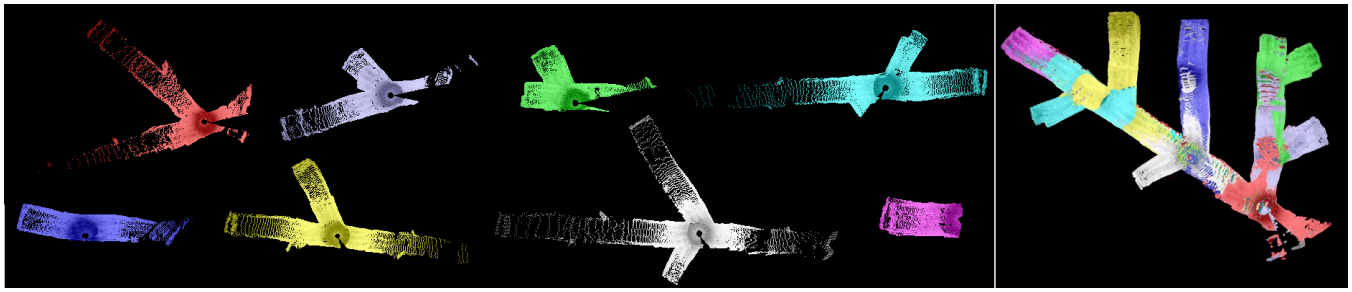


Figure 1: *A set of partially overlapping 3D laser scans (viewed from the top) from an underground mine and their correct registration.*

Abstract

Our goal is the registration of multiple 3D point clouds obtained from LIDAR scans of underground mines. Such a capability is crucial to the surveying and planning operations in mining. Often, the point clouds only partially overlap and initial alignment is unavailable. Here, we propose an *interactive user-assisted* point cloud registration system. Guided by the system, the user's role is simply to identify and search for overlapping regions across the point clouds. Specifically, given two point sets, the user clicks on a point in one set, then simply hovers the mouse on the other set to find a matching point. Each mouse position gives rise to a translation, and our system instantly optimises the rotation that aligns the point clouds.

Assuming that each individual point set is horizontally levelled with the ground by the level compensator on the LIDAR device, given a candidate 3D translation only one angular parameter needs to be estimated to rotationally align two 3D point sets. We propose a fast rotation search algorithm that delivers *globally optimal* results in *real time*. Our method conducts branch-and-bound optimisation with a novel bounding function whose evaluation amounts to simple sorted array operations. This provides smooth and accurate feedback to the user's search for overlapping regions. Our system is intuitive and helps to accelerate the registration of multiple scans.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Display Algorithms

Keywords: Point cloud registration, geometric matching, rotation optimisation, real-time interaction.

1 Introduction

Mining is the key economic driver in many countries, such as Australia where mining contributes almost 20% of the GDP or 300 billion dollars. As the industry expands there is continual pressure to maximise yield, minimise cost and maintain safety standards. To achieve these, accurate surveying of mines is crucial, and the usage of LIDAR scanning is prevalent. Due to the large expanse of the mines, there is the need to take multiple scans and then register the point clouds later. In this paper, our target application is aligning multiple 3D point clouds from LIDAR scans of underground mining sites. Fig. 1 shows a set of point clouds from an actual underground mine and their correct registration. Observe the small partial overlaps between the point clouds. This occurs frequently in practice, since a surveyor will economise the work by spreading the scans over the scene uniformly, thus reducing scan overlaps.

Despite significant efforts, fully automatic registration of multiple point clouds remains challenging [Tam et al. 2013]. Globally optimal methods do not require human input [Breuel 2003; Li and Hartley 2007], but they can be very slow. Approximate methods are fast, such as those based on randomised heuristics [Chen et al. 1999; Aiger et al. 2008; Papazov and Burschka 2011] or feature matching [Gelfand et al. 2005; Li and Guskov 2005; Gal and Cohen-Or 2006; Pottmann et al. 2009]. However, approximate methods do not guarantee good results. Moreover, the small partial overlaps and "self-similar" nature of the kind of data in Fig. 1 greatly raise the difficulty, e.g., feature matching may not be reliable.

Many practical systems still depend on ICP [Besl and McKay 1992], which is fast but requires careful initialisation. In practice, this means arranging the point clouds such that they are close to alignment already, and ICP simply refines the transformation parameters to "lock in" the fit. For underground scans, GPS localisation of the LIDAR devices cannot be used to intialise the registration. This leaves manual initialisation, which can be tricky and laborious when there are multiple point sets to align. We seek an approach that is less cumbersome and time-consuming for the user.

To this end, we propose a novel *user-assisted* point cloud registration system with *real-time interaction*. Guided by the system, the user's role is to identify overlapping regions across the point clouds. More specifically, given two input point clouds, the user first selects a point $\mathbf{p}$ in one point cloud, then hovers the mouse over the second point cloud to look for a matching point. Each mouse position gives rise to a potential corresponding point $\mathbf{q}$, and the first point cloud is translated by vector $\mathbf{q} - \mathbf{p}$ such that $\mathbf{p}$ and $\mathbf{q}$ coincide. In real time, the system calculates the rotation centred on $\mathbf{q}$ that best aligns the point clouds, providing instant verification to the match $\mathbf{p} \leftrightarrow \mathbf{q}$. If the result is not satisfactory, the user can simply move the mouse to

a different location, or restart by reselecting point $\mathbf{p}$. *Please watch the supplementary video or Figs. 7, 8 and 9 for sample results.*

Our system takes advantage of the level compensator on board most LIDAR devices to simplify rotation search. Level compensators horizontally aligns each individual scan with respect to the ground plane, and they are commonly used in surveying. Thus, given the translation component of a 3D rigid transform, only one angular parameter (i.e., the azimuth) needs to be obtained to rotationally align the point sets. The crux of our system is a rotation search algorithm that delivers *globally optimal* results in *real time*. We propose a method based on branch-and-bound (bnb) optimisation [Horst and Tuy 2003] with a novel bounding function, whose evaluation amounts to very simple sorted-array insertions. This yields an interactive system that gives smooth and accurate feedback.

From a user's standpoint, identifying corresponding regions across point clouds is much easier than initialising their alignment as required in ICP. Arguably, manually initialising alignment involves first finding where the point clouds overlap. Our system thus requires much less effort on the part of the user; for example, the point sets in Fig. 1 can be aligned in mere minutes. Note that $\mathbf{p}, \mathbf{q}$ need not be salient points or points with high curvature [Gal and Cohen-Or 2006; Pottmann et al. 2009]. A human observer can draw upon structural characteristics of the scene (e.g., end points or intersections of tunnels; see Figs. 7 and 8) to quickly find overlaps.

Finally, although we concentrate on LIDAR scans of underground mines, our system is certainly applicable to other kinds of objects or structures, as long as each point cloud is level with respect to the ground plane. Examples include the Stanford range data [Curless and Levoy 1996] where the objects seem to have been rotated on a turntable; see the supplementary video or Fig. 9.

The rest of the paper is as follows: in Sec. 2, we formulate the rotation search problem, and describe our algorithm in detail in Sec. 3. Sec. 4 experimentally compares our rotation search algorithm against other techniques, and show sample results of our interactive point set registration system. We then conclude in Sec. 5.

2 Rotation search for point set registration

Let $\mathcal{M} = \{\mathbf{m}_i\}_{i=1}^{M}$ and $\mathcal{B} = \{\mathbf{b}_j\}_{j=1}^{B}$ be two sets of 3D points. We assume that $\mathcal{M}$ and $\mathcal{B}$ have been translated such that the corresponding points $\mathbf{p} \in \mathcal{M}$ and $\mathbf{q} \in \mathcal{B}$ are at the origin in $\mathbb{R}^3$. We seek the rotation matrix $\mathbf{R} \in SO(3)$ that best aligns $\mathcal{M}$ and $\mathcal{B}$. Following Breuel's method [Breuel 2003], we maximise the *geometric matching* criterion

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

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

Optimising over the space of 3D rotations $\mathbf{R}$ still presents a major problem, as observed in other works. For example, the box-and-ball algorithm presented by Li and Hartley requires more than $150s$ to rotationally align two point clouds of size of just 100 points each [Li and Hartley 2007]. Moreover, they considered an easier problem with no outliers, i.e., each point in $\mathcal{M}$ must have a matching point in $\mathcal{B}$. Rotation search has also been investigated for other applications, e.g., for estimating camera poses [Hartley and Kahl 2009; Seo et al. 2009; Bazin et al. 2012; Bazin et al. 2013]. Most of these methods are not amenable to real time performance.

It is thus necessary to simplify (1). As mentioned earlier, we rely on the level compensator existing on many LIDAR devices to remove two angular degrees of freedom from $\mathbf{R}$, leaving only the azimuth θ unknown. The criterion (1) now takes the form

$$Q(\theta) = \sum_i \max_j \lfloor \|\mathbf{R}_z(\theta)\mathbf{m}_i - \mathbf{b}_j\| \leq \epsilon \rfloor, \tag{2}$$

where $\mathbf{R}_z(\theta)$ is a rotation matrix of θ radians about the z-axis. We assume in (2) that each $\mathcal{M}$ and $\mathcal{B}$ is level with respect to the ground plane. Secondly, given a correct correspondence $\mathbf{p} \leftrightarrow \mathbf{q}$, it is unnecessary to try to rotationally align all the points, since the point clouds only partially overlap. Thus, we exclude from (2) points in $\mathcal{M}$ and $\mathcal{B}$ whose norm is greater than a threshold ϵ_n. While this reduces the size of the problem, on dense point clouds such as those in Fig. 1, the size of the remaining point sets can still be large.

To account for potential inaccuracies in level compensation, for each $\mathbf{m}_i$ we optionally add to ϵ an extra tolerance ϵ_i, defined as

$$\epsilon_i = \sqrt{2\|\mathbf{m}_i\|^2 - 2\|\mathbf{m}_i\|^2 \cos\phi} \tag{3}$$

where ϕ is the angular uncertainty of the plumb-line (the z-axis); see Fig. 2. To illustrate the resulting $Q(\theta)$, Fig. 3 plots the function for two of the point clouds from Fig. 1.

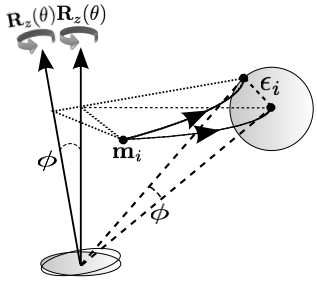


Figure 2: *If the angular deviation of the z-axis is within ϕ, then all possible variations of $\mathbf{R}_z(\theta)\mathbf{m}_i$ due to the uncertainty lie within a ball of radius ϵ_i, where ϵ_i can be obtained based on the cosine rule.*

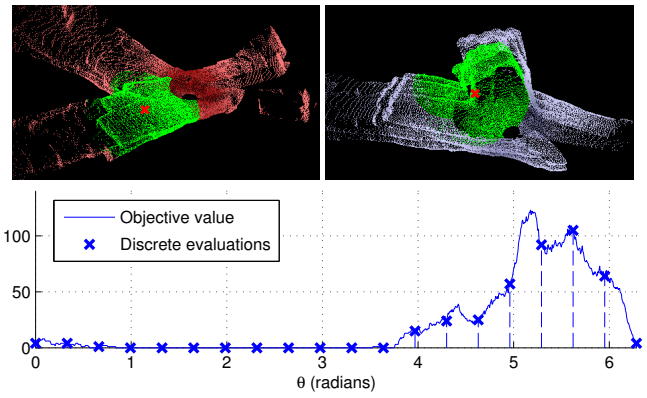


Figure 3: *The top figures show $\mathcal{M}, \mathcal{B}$ with $\mathbf{p}, \mathbf{q}$ indicated by red crosses. Only the points (coloured green) with distance $\leq \epsilon_n$ to $\mathbf{p}, \mathbf{q}$ are used. The objective function* (2) *is plotted at the bottom, where the sampling of* (2) *at 20 discrete positions is also shown.*

It is clear that $Q(\theta)$ has multiple local maxima, and a simple strategy to maximise the function by sampling at discrete intervals is unlikely to be successful and efficient. Recall that each evaluation

Algorithm 1 Bnb rotation search to solve (2).

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

Set $Q_{\max} = 0$, $\mathbb{T}_0 = [0, 2\pi]$, and $list = (\mathbb{T}_0, \hat{Q}(\mathbb{T}_0))$.
while $list$ is not empty **do**
  Remove from $list$ the interval $\mathbb{T}$ with the highest $\hat{Q}(\mathbb{T})$.
  if $\hat{Q}(\mathbb{T}) > Q_{\max}$ **then**
    $\theta_c \leftarrow$ centre of $\mathbb{T}$.
    if $Q(\theta_c) = \hat{Q}(\mathbb{T})$ **then**
      $\theta^* \leftarrow \theta_c$ and terminate algorithm.
    else if $Q(\theta_c) > Q_{\max}$ **then**
      $Q_{\max} \leftarrow Q(\theta_c)$ and $\theta^* \leftarrow \theta_c$.
    end if
    $\mathbb{T}_l \leftarrow [\min(\mathbb{T}), \theta_c]$ and $\mathbb{T}_r \leftarrow [\theta_c, \max(\mathbb{T})]$.
    $list \leftarrow list \cup (\mathbb{T}_l, \hat{Q}(\mathbb{T}_l)) \cup (\mathbb{T}_r, \hat{Q}(\mathbb{T}_r))$.
  end if
end while

of (2) requires checking whether each $\mathbf{R}_z(\theta)\mathbf{m}_i$ matches a point in $\mathcal{B}$, which can be an expensive nearest neighbour problem if $\mathcal{M}$ and $\mathcal{B}$ are large. Moreover, as exemplified in Fig. 3 many of the evaluations will be wasted on totally unpromising values of θ. The existence of multiple local maxima in (2) also rules out the usage of line search or iterative optimisation methods (e.g., gradient descent) to find the best θ, since these can only converge to a local maximum.

3 Bnb rotation search

A more promising approach is bnb optimisation [Horst and Tuy 2003], which was previously applied by Breuel [Breuel 2003]. The method recursively explores and prunes the search space $[0, 2\pi]$ by maintaining a list of intervals. Given an interval, we test whether a better rotation than the best found so far exists in the interval. If the test fails then the whole interval is discarded; else the interval is divided into smaller subintervals. The process is repeated until the global maximum is found. Algorithm 1 summarises the procedure.

A key component of Algorithm 1 is the bounding function $\hat{Q}(\mathbb{T})$, which gives the upper bound of $Q(\theta)$ in the interval $\mathbb{T} \subseteq [0, 2\pi]$:

$$\hat{Q}(\mathbb{T}) \geq \max_{\theta \in \mathbb{T}} Q(\theta). \tag{4}$$

As shown in Algorithm 1, $\hat{Q}(\mathbb{T})$ plays the crucial of pruning unpromising search intervals. Applying the Breuel's geometric matching bounding function [Breuel 2003] for rotational transform yields

$$\hat{Q}_{\text{gm}}(\mathbb{T}) = \sum_i \max_j \lfloor \|\mathbf{R}_z(\theta_c)\mathbf{m}_i - \mathbf{b}_j\| \leq \epsilon + \delta_i \rfloor, \tag{5}$$

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

$$\delta_i = \sqrt{2\|\mathbf{m}_i\|^2 - 2\|\mathbf{m}_i\|^2 \cos(\max(\mathbb{T}) - \theta_c)}, \tag{6}$$

where the additional tolerance δ_i arose due to the actions of all possible rotations in $\mathbb{T}$. It has been proven that (5) satisfies (4) [Breuel 2003]. As we will show in the experiments in Sec. 4, applying (2) and (5) in Algorithm 1 does not produce real-time performance.

3.1 A novel rotational bounding function

Clearly, the run time of Algorithm 1 depends on the total number of angular intervals that are tested before the optimal θ is found. A tighter bounding function will reduce the number of intervals that needs to be tested, since it can conduct a more aggressive pruning. Our core insight is that a direct application in (5) of Breuel's bounding function ignores the *limited range* of rotational transforms, i.e., a point can only lie a circular arc under all possible $\mathbf{R}_z(\theta)$ for $\theta \in \mathbb{T}$. Thus, it is unnecessary to expect that $\mathbf{m}_i$ may lie anywhere within the δ_i-ball centred at $\mathbf{R}_z(\theta_c)\mathbf{m}_i$, as assumed in (5).

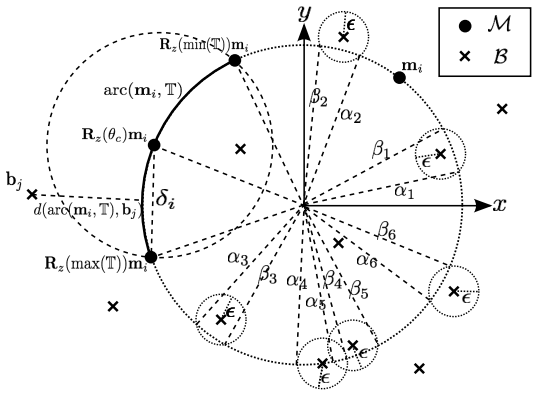


Figure 4: *Top view illustration of our rotation search problem. For simplicity, only a point $\mathbf{m}_i$ from $\mathcal{M}$ is shown. Under all possible rotation angles in a range $\mathbb{T}$, a direct application* (5) *of Breuel's bounding function conservatively assumes $\mathbf{m}_i$ may lie anywhere within a δ_i-sphere centred at $\mathbf{R}_z(\theta_c)\mathbf{m}_i$. In reality, $\mathbf{m}_i$ can only lie on an arc. Our bounding function* (7) *exploits this knowledge. Further, if $\mathbf{b} \in \mathcal{B}$ can possibly match with $\mathbf{m}_i$, then an ϵ-ball centred at $\mathbf{b}$ intersects with circular trajectory of $\mathbf{m}_i$ at an arc. In Sec. 3.2, we describe a method that indexes the set of possible matches with $\mathbf{m}_i$ in a simple array for rapid evaluations of* (2) *and* (7)*.*

To crystalise our idea, we propose the new bounding function

$$\hat{Q}_{\text{arc}}(\mathbb{T}) = \sum_i \max_j \lfloor d(\text{arc}(\mathbf{m}_i, \mathbb{T}), \mathbf{b}_j) \leq \epsilon \rfloor, \tag{7}$$

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

$$\text{arc}(\mathbf{m}_i, \mathbb{T}) = \{\, \mathbf{R}_z(\theta)\mathbf{m}_i \mid \theta \in \mathbb{T} \,\}, \tag{8}$$

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

$$d(\text{arc}(\mathbf{m}_i, \mathbb{T}), \mathbf{b}_j) = \min_{\mathbf{x} \in \text{arc}(\mathbf{m}_i, \mathbb{T})} \|\mathbf{x} - \mathbf{b}_j\|. \tag{9}$$

Fig. 4 illustrates this idea. Next, we prove that $\hat{Q}_{\text{arc}}(\mathbb{T})$ is a valid bounding function that is also tighter than $\hat{Q}_{gm}(\mathbb{T})$.

Lemma 1. *The condition*

$$\hat{Q}_{arc}(\mathbb{T}) \geq \max_{\theta \in \mathbb{T}} Q(\theta) \tag{10}$$

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

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

$$\|\mathbf{R}_z(\theta)\mathbf{m}_i - \mathbf{b}_j\| \leq \epsilon \tag{11}$$

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

$$d(\text{arc}(\mathbf{m}_i, \mathbb{T}), \mathbf{b}_j) \leq \epsilon \tag{12}$$

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

Another important condition for a valid bounding function is that $\hat{Q}(\mathbb{T})$ approaches $Q(\theta)$ as the volume of $\mathbb{T}$ approaches zero [Horst and Tuy 2003]. This can be easily seen in $\hat{Q}_{\text{arc}}(\mathbb{T})$, since as $\mathbb{T}$ collapses to a single point θ_c (volume of $\mathbb{T}$ becomes 0), then $d(\text{arc}(\mathbf{m}_i, \theta_c), \mathbf{b}_j) = \|\mathbf{R}_z(\theta_c)\mathbf{m}_i - \mathbf{b}_j\| \implies \hat{Q}_{\text{arc}}(\theta_c) = Q(\theta_c)$.

Lemma 2. *The condition*

$$\hat{Q}_{gm}(\mathbb{T}) \geq \hat{Q}_{arc}(\mathbb{T}) \tag{13}$$

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

Proof. Showing that $\hat{Q}_{\text{gm}}(\mathbb{T})$ at least equals $\hat{Q}_{\text{arc}}(\mathbb{T})$ can be done by simply appealing to be fact that both functions satisfy (4).

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

$$\mathbf{m}_i \text{ and } \mathbf{b}_j = \mathbf{R}_z(\theta_c)\mathbf{m}_i \left(1 + \frac{\epsilon + \delta_i}{\|\mathbf{m}_i\|}\right) \tag{14}$$

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

$$d(\text{arc}(\mathbf{m}_i, \mathbb{T}), \mathbf{b}_j) = \epsilon + \delta_i \tag{15}$$

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

3.2 Efficient evaluations for real-time search

The efficiency of evaluating the objective and bounding functions is also crucial to the speed of Algorithm 1, since these functions are called repeatedly. Kd-trees are the workhorse in Breuel's method [Breuel 2003] for speeding up the function evaluations. As Sec. 4 will show, this is still insufficient for real-time performance.

We propose a much faster method for evaluating (2) and (7). Based on the same insight in Sec. 3.1, as $\mathbf{m}_i$ is subjected to all allowable rotations $\mathbf{R}_z(\theta)$, only a subset $\mathcal{B}_{\mathbf{m}_i}$ of $\mathcal{B}$ can possibly match with $\mathbf{m}_i$, since $\mathbf{R}_z(\theta)\mathbf{m}_i$ is limited to lie on a circular arc. Define $circ_{\mathbf{m}_i}$ as the trajectory resulting from rotating $\mathbf{m}_i$ about the z-axis by 2π radians. An ϵ-ball centred at a point $\mathbf{b} \in \mathcal{B}_{\mathbf{m}_i}$ intersects with $circ_{\mathbf{m}_i}$ at a circular arc; see Fig. 4. Intuitively, solving the query

$$\max_j \lfloor \|\mathbf{R}_z(\theta)\mathbf{m}_i - \mathbf{b}_j\| \leq \epsilon \rfloor \tag{16}$$

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

To implement the above idea, let $[\alpha, \beta]$ be the angular interval subtended by the intersection between an ϵ-ball and $circ_{\mathbf{m}_i}$; see Fig. 4. We store the intervals relating to all $\mathbf{b} \in \mathcal{B}_{\mathbf{m}_i}$ in the array

$$int_{\mathbf{m}_i} = [\alpha_1, \beta_1, \alpha_2, \beta_2, \dots] \tag{17}$$

which is maintained to be *sorted*. If an ϵ-ball intersects $circ_{\mathbf{m}_i}$ at one point, then α equals β (there may exist repeated values in $int_{\mathbf{m}_i}$). Also, overlapping intervals in $int_{\mathbf{m}_i}$ are merged into a single interval; for example, in Fig. 4 $[\alpha_4, \beta_4]$ and $[\alpha_5, \beta_5]$ will be merged. An efficient algorithm to compute $int_{\mathbf{m}_i}$ will be given later. To compute (16) we simply find the position of the angle $\theta + \theta_{\mathbf{m}_i}$ in $int_{\mathbf{m}_i}$, where the offset $\theta_{\mathbf{m}_i}$ is the azimuth of $\mathbf{m}_i$

$$\theta_{\mathbf{m}_i} = \arcsin\left(\frac{\mathbf{m}_i(2)}{\sqrt{\mathbf{m}_i(1)^2 + \mathbf{m}_i(2)^2}}\right) \in [0, 2\pi]. \tag{18}$$

If $\theta + \theta_{\mathbf{m}_i}$ lies within any neighbouring $[\alpha, \beta]$, then the solution to (16) is 1; else, it is 0. A similar idea can be used for the range query

$$\max_j \lfloor d(\text{arc}(\mathbf{m}_i, \mathbb{T}), \mathbf{b}_j) \leq \epsilon \rfloor \tag{19}$$

contained in (7), where $\mathbb{T} = [\theta_1, \theta_2]$. If $\theta_1 + \theta_{\mathbf{m}_i}$ and $\theta_2 + \theta_{\mathbf{m}_i}$ both lie in the *same* position in $int_{\mathbf{m}_i}$, and that position is *not* within any neighbouring $[\alpha, \beta]$, then the solution to (19) is 0; else it is 1. We have thus reduced (2) and (7) to a series of sorted array insertion operations, which are extremely simple and efficient.

We now discuss how to build the arrays $\{int_{\mathbf{m}_i}\}_{i=1}^M$ given point sets $\mathcal{M}$ and $\mathcal{B}$. Since $\mathcal{M}$ and $\mathcal{B}$ change rapidly in our real-time interactive system, it is vital to compute $\{int_{\mathbf{m}_i}\}_{i=1}^M$ very quickly. Our proposed solution is inspired by the classical sweep plane algorithm [O'Rourke 1998, Sec. 7.7]. We rotate a sweep plane about the z-axis, and each time the sweep plane "hits" a point $\mathbf{b}_j \in \mathcal{B}$, we insert the angular intervals corresponding to $\mathbf{b}_j$ into the relevant arrays; Fig. 5 illustrates the idea, and Algorithm 2 summarises the method. Note that the subsets $\{\mathcal{B}_{\mathbf{m}_i}\}_{i=1}^M$ are implicitly contained in $\{int_{\mathbf{m}_i}\}_{i=1}^M$, thus they are not output by Algorithm 2.

Similar to the original sweep plane algorithm, the key to Algorithm 2's efficiency is the maintenance of a *status* array, which allows to avoid exhaustively checking all $M \times B$ potential matches

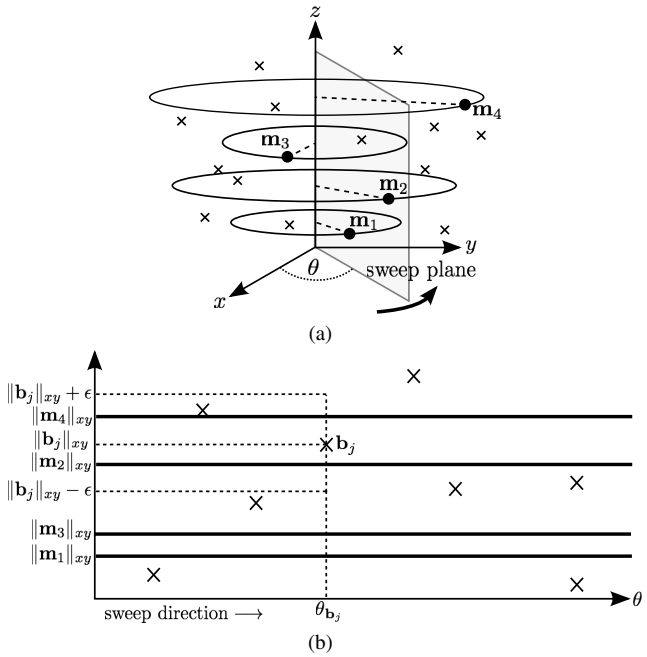


Figure 5: *(a) An illustration of our modified sweep plane algorithm. The sweep plane is rotated about the z-axis for* 2π *radians. (b) This is the top view of (a) which is unrolled to lie flat on the plane. In this example, "status" in Algorithm 2 has the content* $[\|\mathbf{m}_1\|_{xy}, \|\mathbf{m}_3\|_{xy}, \|\mathbf{m}_2\|_{xy}, \|\mathbf{m}_4\|_{xy}]$*. Like in the classical sweep plane algorithm, our algorithm visits a finite set of "events", corresponding to the points* $\mathcal{B}$*, in the order of their azimuth. In this example,* $\mathbf{b}_j$ *has the potential to match* $\mathbf{m}_2$ *and* $\mathbf{m}_4$.

Algorithm 2 Modified sweep plane algorithm.

Require: Point sets $\mathcal{M}$ and $\mathcal{B}$, threshold ϵ.
1: Set $int_{\mathbf{m}_i} = \emptyset$ for all $i = 1, \ldots, M$.
2: Reorder the points in $\mathcal{B}$ based on their azimuth (18).
3: $status \leftarrow$ sort $\{\|\mathbf{m}_i\|_{xy}\}_{i=1}^{M}$ for all $\mathbf{m}_i \in \mathcal{M}$; see (20).
4: Reorder the points in $\mathcal{M}$ based on their position in $status$.
5: **for** $j = 1, \ldots, B$ **do**
6: $\quad i_l \leftarrow$ smallest i such that $status(i) \geq \|\mathbf{b}_j\|_{xy} - \epsilon$.
7: $\quad i_u \leftarrow$ largest i such that $status(i) \leq \|\mathbf{b}_j\|_{xy} + \epsilon$.
8: $\quad$ **if** both i_l and i_u are not null **then**
9: $\quad\quad$ **for** $i = i_l, \ldots, i_u$ **do**
10: $\quad\quad\quad$ **if** $(\|\mathbf{m}_i\|_{xy} - \|\mathbf{b}_j\|_{xy})^2 \leq \epsilon^2 - (\mathbf{m}_i(3) - \mathbf{b}_j(3))^2$ **then**
11: $\quad\quad\quad\quad arc \leftarrow$ intersect $circ_{\mathbf{m}_i}$ and ϵ-ball centred at $\mathbf{b}_j$.
12: $\quad\quad\quad\quad [\alpha, \beta] \leftarrow$ angular interval subtended by arc.
13: $\quad\quad\quad\quad$ Insert (or, if required, merge) $[\alpha, \beta]$ into $int_{\mathbf{m}_i}$.
14: $\quad\quad\quad$ **end if**
15: $\quad\quad$ **end for**
16: $\quad$ **end if**
17: **end for**
18: **return** Set of arrays $\{int_{\mathbf{m}_i}\}_{i=1}^{M}$.

between $\mathcal{M}$ and $\mathcal{B}$. Here, *status* contains the sorted norms $\{\|\mathbf{m}_i\|_{xy}\}_{i=1}^{M}$, where $\|\mathbf{m}_i\|_{xy}$ is the norm of $\mathbf{m}_i$ on the xy-plane

$$\|\mathbf{m}_i\|_{xy} = \sqrt{\mathbf{m}_i(1)^2 + \mathbf{m}_i(2)^2}. \tag{20}$$

Essentially $\|\mathbf{m}_i\|_{xy}$ is the radius of $circ_{\mathbf{m}_i}$, and this quantity is "perpendicular" to the sweep direction; see Fig. 5(b). In addition, analogous to the sorting of *intersection events* in the original algorithm, Algorithm 2 visits the points in $\mathcal{B}$ in the order of their azimuth angle (18). This ensures that in Step 13 the intervals are inserted in sorted order into $int_{\mathbf{m}_i}$. Merging potentially overlapping intervals on-the-fly can be done efficiently by maintaining a stack along with each sorted interval array $int_{\mathbf{m}_i}$; since this is a very common problem we will not discuss the details here.

Using similar analysis from computational geometry [O'Rourke 1998, Sec. 7.7], for each point in $\mathcal{B}$ we need to query the sorted array *status* of length M twice. The complexity of Algorithm 2 is $\mathcal{O}(B \log M + k)$, where k is the total number of possible matches between $\mathcal{M}$ and $\mathcal{B}$. We will demonstrate in Sec. 4 that Algorithm 2 consumes very little overhead. In turn this enables very efficient evaluations of functions (2) and (7), leading to real-time optimal rotation search.

4 Results

We first test and benchmark the performance of our rotation search algorithm by evaluating the following methods:

- **bnb-M-circ**: the proposed algorithm.
- **bnb-1-tree**: Algorithm 1 using Breuel's original bounding function $\hat{Q}_{\text{gm}}$ (5). Given $\mathcal{M}$ and $\mathcal{B}$, a kd-tree is used to index $\mathcal{B}$ to speed-up the evaluation of (2) and (5).
- **bnb-M-tree**: same as **bnb-1-tree** but to further speed up (2) and (5), for each $\mathbf{m}_i \in \mathcal{M}$, we presearch the set of possible matches $\mathcal{B}_{\mathbf{m}_i}$ and index $\mathcal{B}_{\mathbf{m}_i}$ in a kd-tree. This yields M kd-trees in total. To check if $\mathbf{R}_z(\theta)\mathbf{m}_i$ matches a point in $\mathcal{B}$, only the kd-tree associated with $\mathbf{m}_i$ needs to be queried.
- **icp-rotate**: a special case of ICP whereby given $\mathcal{M}$ and $\mathcal{B}$, the optimised transformation is restricted to $\mathbf{R}_z(\theta)$.

Theoretically, since in **bnb-M-tree** only the possible matches $\mathcal{B}_{\mathbf{m}_i}$ of $\mathbf{m}_i$ are searched for a match, the bound $\hat{Q}_{\text{gm}}$ is equal to $\hat{Q}_{\text{arc}}$, i.e., **bnb-M-circ** and **bnb-M-tree** will take the same number of iterations in Algorithm 1. The fundamental difference is thus the efficiency required to evaluate the objective and bounding functions. The performance metrics used to compare the above methods are:

- Total run time, which includes all data structure preparation time (kd-trees, sorted arrays, etc.) and searching for θ until convergence (all methods compared can provably converge). All the methods were implemented in C and run on a machine with an Intel Core i7 3.40GHz CPU.
- Number of matched points (2) at convergence. All the bnb methods will yield the same globally maximal value. For **icp-rotate**, we simply use the converged θ value to evaluate (2).

Further, to evaluate the metrics for **icp-rotate** we run the method with 10 different random initialisations and take the average result.

To simulate the user-assisted search for matching regions, given two input point clouds we sample local point sets $\mathcal{M}$ and $\mathcal{B}$ as follows:

1. On each point cloud, 3D keypoints are detected. While many methods are applicable, for convenience we use *ISS* [Zhong 2009] as implemented on Point Cloud Library (PCL)[1].
2. We randomly sample 100 keypoint matches across the two point clouds. For each match, the local point cloud (within radius ϵ_n) around each keypoint are taken as $\mathcal{M}$ and $\mathcal{B}$.
3. ISS also gives the surface normal at the keypoints. We orient each $\mathcal{M}$ and $\mathcal{B}$ pair such that the normals are pointing up. We then optimise the rotation $\mathbf{R}_z(\theta)$ that best aligns $\mathcal{M}$ and $\mathcal{B}$.

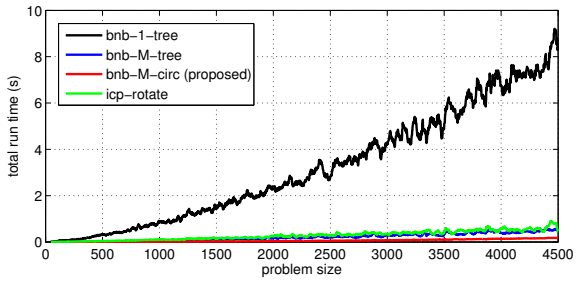
Note that the sampled keypoint matches here need not be genuine matches; our purpose is to examine the speed of various methods in rotation search, not their accuracy in matching keypoints. Also, we vary ϵ_n across ten different values for each keypoint match, thus yielding $\mathcal{M}$ and $\mathcal{B}$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

To conduct our experiment, we choose the following pairs of point clouds from the underground mine data in Fig. 1, namely Set 1 vs Set 2, Set 2 vs Set 3, and Set 1 vs Set 4 (the numbering is taken in the left-right order in Fig. 1). As output by the LIDAR device, these point clouds are individually level with respect to the ground plane. For completeness, we also use the range data from the Stanford 3D Scanning Repository [Curless and Levoy 1996], namely, Bunny, Dragon and Armadillo (for each object, we choose point clouds from two different views). The objects seem to have been rotated on a turntable so the data satisfies our assumption; see Fig. 9.

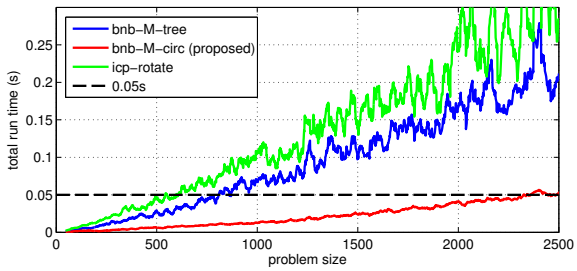
Fig. 6 shows the overall result on all datasets. We plot both run time and number of matches against problem size. To enable a consistent way of measuring problem size, *before* performing rotation search, for each $\mathcal{M}$ and $\mathcal{B}$ we ensure that $\mathcal{M}$ is smaller than $\mathcal{B}$ by swapping the point sets if required. We then take M as the size of each rotation search problem. Doing this also ensures that the quality value is bounded by M, as defined in the objective function (2).

The run time results in Fig. 6(a) clearly show that **bnb-M-circ** has a much better computational complexity than **bnb-1-tree**. It is also evident that **bnb-1-tree** cannot provide real-time performance. Fig. 6(b) gives a closer comparison between **bnb-M-circ**, **bnb-M-tree** and **icp-rotate**. The two methods **bnb-M-circ** and **bnb-M-tree** show a clear divergence starting from small problem sizes. Taking 20 frames-per-second ($0.05s$ per rotation search) as real time, **bnb-M-tree** can only handle up to 800 points, whereas

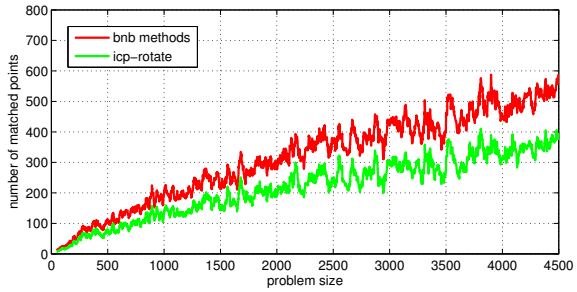
[1] `http://pointclouds.org/`



(a) Total run time versus problem size.



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



(c) Number of matched points versus problem size.

Figure 6: *Comparing run time and quality (number of matched points) of various rotation search methods.*

bnb-M-circ is good up to 2500 points. This indicates the superior efficiency of our novel bound evaluation techniques in Sec. 3.2 (recall that since both methods theoretically evaluate the same bounds, their core difference is in the bound evaluation time). In problems involving high resolution LIDAR scans such as surveying the underground mines in Fig. 1 (e.g., an individual scan can contain up to $500,000$ points), extremely fast rotation search is crucial for aligning the local point clouds $\mathcal{M}$ and $\mathcal{B}$ with sizes up to the range of thousands. Somewhat surprisingly, the results show that **icp-rotate** is slower than **bnb-M-circ** (note that the plot in Fig. 6 show the *average* time and quality for **icp-rotate** over 10 repetitions). This implies that the number of iterations required in **icp-rotate** is typically higher than our bnb method. In any case, as Fig. 6(c) shows, the quality of **icp-rotate** is lower than bnb (note that all the bnb methods return the same globally optimal result), and its quality will vary more erratically without the averaging conducted here.

optimal with respect to the set of discrete correspondences between $\mathcal{M}$ and $\mathcal{B}$, and not to all the available points.

Other globally optimal methods. As surveyed in Sec. 1 there exist other methods that promise global optimality in point cloud registration. Most of them tackle full 3D rigid transforms, thus a direct comparison is only possible if significant changes are made to those methods. For an indication of how the other methods compare, Li and Hartley [Li and Hartley 2007] used their method to optimise pure rotation in an experiment, where one of the Bunny point clouds was downsampled and then rotated to yield two point sets $\mathcal{M}$ and $\mathcal{B}$ (their method requires $\mathcal{M}$ and $\mathcal{B}$ to have the same size and each point in $\mathcal{M}$ must have a matching point in $\mathcal{B}$). For point clouds of size 200 their method requires more than $1000s$. A globally optimal rotation search algorithm was proposed by Bazin et al. [Bazin et al. 2012], however, they require 3D point correspondences to be estalished between the point clouds $\mathcal{M}$ and $\mathcal{B}$ to guide the search. Thus, they are tackling a different problem to ours. Moreover, their method is only optimal with respect to the set of discrete correspondences between $\mathcal{M}$ and $\mathcal{B}$, and not to all the available points.

4.1 Overall system

Based on our rotation search method, we construct a user-assisted 3D registration system that provides real-time feedback. Figs. 7, 8 and 9 show several screenshots of our system in action. As described in the introduction, the user's role is to identify potential overlapping areas. First, a point $\mathbf{p}$ is selected in the first point cloud, then the user hovers the mouse pointer over the second point cloud to seek a matching point $\mathbf{q}$ of $\mathbf{p}$. Each mouse pointer location gives rise to a candidate $\mathbf{q}$ and triggers the rotation search kernel to validate the correspondence by aligning the local point clouds around $\mathbf{p}$ and $\mathbf{q}$. This provides instant guidance to the user to either continue searching for $\mathbf{q}$, or restart by reselecting $\mathbf{p}$. As described in Sec. 2, the rotation optimisation step only considers points within a neighbourhood (of radius ϵ_n) of the tentative point match $\mathbf{p} \leftrightarrow \mathbf{q}$. Once a satisfactory rotational alignment based on $\mathbf{p} \leftrightarrow \mathbf{q}$ is obtained, the user can refine the overall alignment by using ICP to estimate a full 3D rigid transform using all the points. Our full system (shown in the accompanying video) allows to register multiple overlapping point clouds by simply repeating the steps above.

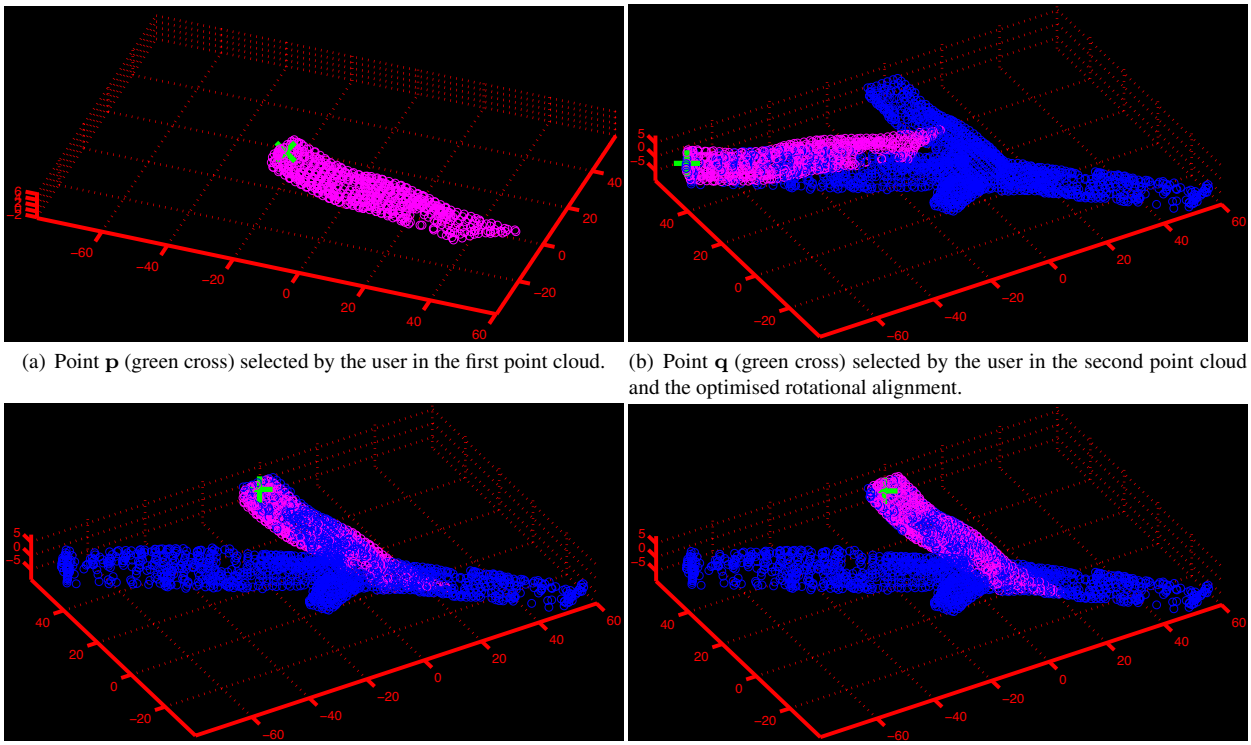
5 Conclusions

We have proposed a novel user-assisted point cloud registration system that provides real-time interaction. Our system is useful for aligning multiple point clouds for large-scale 3D modelling and surveying. In particular, in settings where fully automatic registration may not be feasible due to large problem sizes or lack of information for initialisation, our system provides an efficient means for registering the point clouds with minimal user effort. At the core of our system is a novel rotation search algorithm that is able to globally optimise the rotation in real time. To construct this algorithm, we devised a highly efficient bounding function for use in a bnb algorithm. We have also experimentally showed that our system is orders of magnitude faster than previous approaches.

Possible extensions. Although here we focus on LIDAR scans of underground sites, our system is applicable to other settings where each point set is individually level with respect to a reference plane. Nonetheless, it would be useful to extend our rotation search algorithm to optimise full 3D rotations, such that assumptions on level compensation on LIDAR devices may be removed. Globally optimising full 3D rotations in real-time is challenging. Another possible extension is to leverage on 3D keypoint detectors to provide suggestions to the user on how to align the point clouds.

Acknowledgements

We thank our industry partner Maptek Pty Ltd for generously providing the LIDAR scan data used in this paper.



(a) Point $\mathbf{p}$ (green cross) selected by the user in the first point cloud.

(b) Point $\mathbf{q}$ (green cross) selected by the user in the second point cloud and the optimised rotational alignment.

(c) A better point $\mathbf{q}$ and the rotational alignment.

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

Figure 7: *Sample result of our interactive 3D registration system on underground mine scans.*

References

AIGER, D., MITRA, N., AND COHEN-OR, D. 2008. 4-points congruent sets for robust pairwise surface regisration. In *Proceedings ACM SIGGRAPH 2008*.

BAZIN, J.-C., SEO, Y., AND POLLEFEYS, M. 2012. Globally optimal consensus set maximization through rotation search. In *Proceedings ACCV 2012*.

BAZIN, J.-C., LI, H., KWEON, I. S., DEMONCEAUX, C., VASSEUR, P., AND IKEUCHI, K. 2013. A branch and bound approach to correspondence and grouping problem. *IEEE Trans. PAMI 35*, 7, 1565–1576.

BESL, P., AND MCKAY, N. 1992. A method for registration of 3d shapes. *IEEE Trans. PAMI 14*, 2, 239–256.

BREUEL, T. 2003. Implementation techniques for geometric branch-and-bound matching methods. *CVIU 90*, 3, 258–294.

CHEN, C.-S., HUNG, Y.-P., AND CHENG, J.-B. 1999. RANSAC-based DARCES: A new approach to fast automatic registration of partially overlapping range images. *IEEE Trans. PAMI 21*, 11, 1229–1234.

CURLESS, B., AND LEVOY, M. 1996. A volumetric method for building complex models from range images. *Computer Graphics (SIGGRAPH 1996 Proceedings)*.

GAL, R., AND COHEN-OR, D. 2006. Salient geometric features for partial shape matching and similarity. *ACM Trans. Graph. 25*, 130–150.

GELFAND, N., MITRA, N., GUIBAS, L., AND POTTMANN, H. 2005. Robust global registration. In *Proc. Eurographics 2005*.

HARTLEY, R., AND KAHL, F. 2009. Global optimization through rotation space search. *IJCV 82*, 64–79.

HORST, R., AND TUY, H. 2003. *Global optimization: deterministic approaches*. Springer.

LI, X., AND GUSKOV, I. 2005. Multi-scale features for approximate alignment of point-based surfaces. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Geometry Processing 2005*.

LI, H., AND HARTLEY, R. 2007. The 3d–3d registration problem revisited. In *Proceedings ICCV 2007*.

O'ROURKE, J. 1998. *Computational geometry in C*. Cambridge University Press.

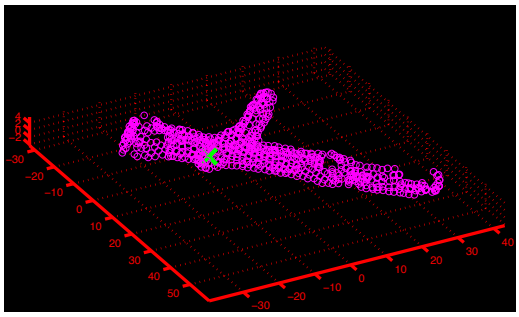
PAPAZOV, C., AND BURSCHKA, D. 2011. Stochastic global optimization for robust point set registration. *CVIU 115*, 1598–1609.

POTTMANN, H., WALLNER, J., HUANG, Q.-X., AND YANG, Y.-L. 2009. Integral invariants for robust geometry processing. *Computer Aided Geometric Design 26*, 1, 37–60.

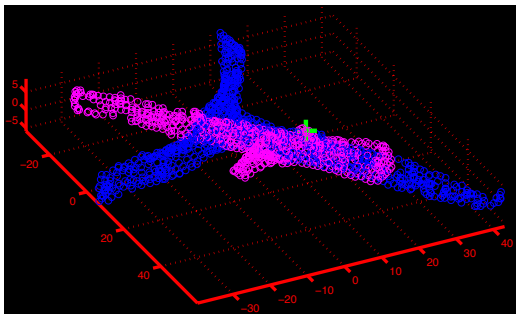
SEO, Y., CHOI, Y.-J., AND LEE, S. W. 2009. A branch-and-bound algorithm for globally optimal calibration of a camera-and-rotation-sensor system. In *Proceedings ICCV 2009*.

TAM, G. K. L., CHENG, Z.-Q., LAI, Y.-K., LANGBEIN, F. C., LIU, Y., MARSHALL, D., MARTIN, R. R., SUN, X.-F., AND ROSIN, P. L. 2013. Registration of 3d point clouds and meshes: a survey from rigid to non-rigid. *IEEE Trans. Visual. Comp. Graph. 19*, 7, 1199–1217.

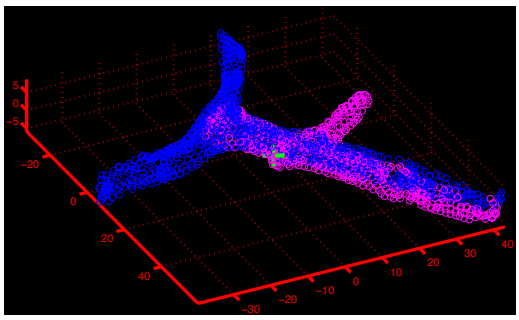
ZHONG, Y. 2009. A shape descriptor for 3d object recognition. In *Proceedings ICCV 2009 Workshop 3DRR*.



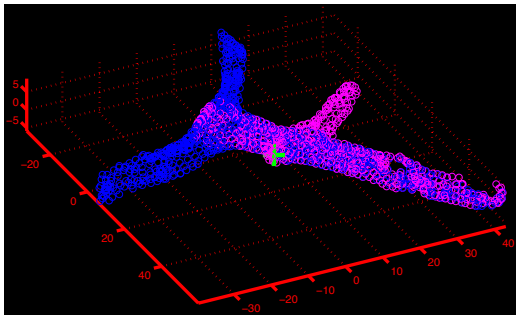
(a) Point $\mathbf{p}$ selected by the user in the first point cloud.



(b) Point $\mathbf{q}$ (green cross) selected by the user in the second point cloud and the optimised rotational alignment.

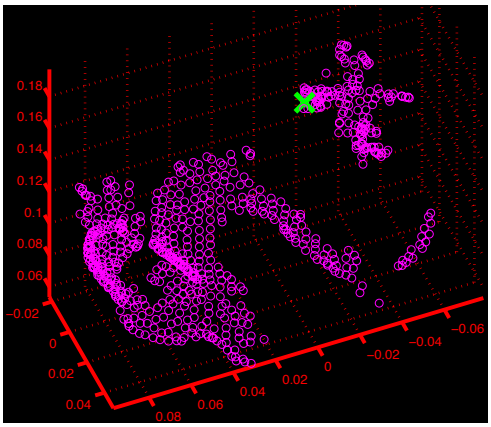


(c) A better point $\mathbf{q}$ and the rotational alignment.

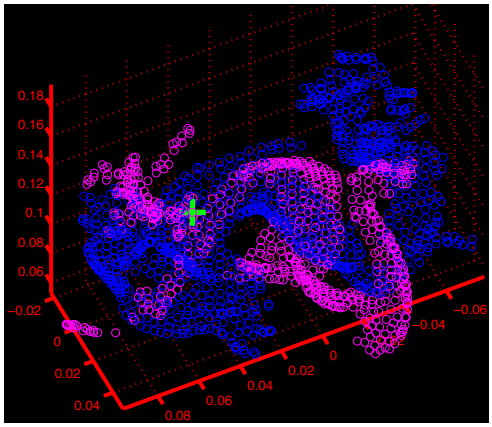


(d) ICP refinement of the result in (c).

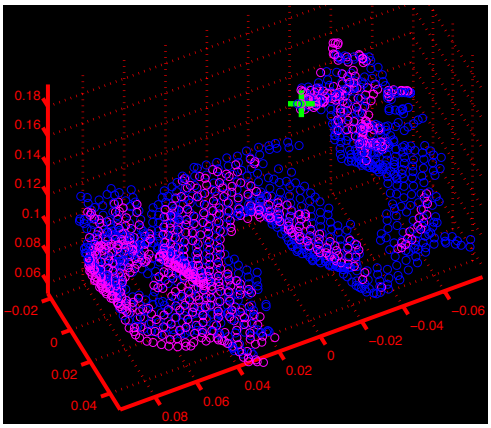
Figure 8: *Sample result of our interactive 3D registration system on underground mine scans.*



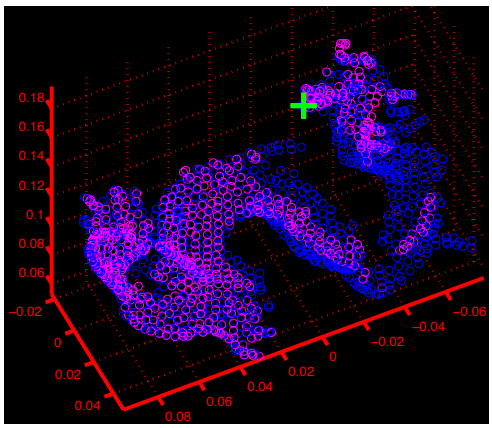
(a) Point $\mathbf{p}$ selected by the user in the first point cloud.



(b) Point $\mathbf{q}$ (green cross) selected by the user in the second point cloud and the optimised rotational alignment.



(c) A better point $\mathbf{q}$ and the rotational alignment.



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

Figure 9: *Sample result of our interactive 3D registration system on the Stanford Dragon (views* 96° *and* 120°*).*