The 3D-3D Registration Problem Revisited

Hongdong Li , Richard Hartley

RSISE, Australian National University
VISTA, National ICT Australia
Hongdong.Li@anu.edu.au

Abstract

We describe a new framework for globally *solving the 3D-3D registration problem with unknown point correspondences. This problem is significant as it is frequently encountered in many applications. Existing methods are not fully satisfactory, mainly due to the risk of local minima. Our framework is grounded on the Lipschitz global optimization theory. It achieves a guaranteed* global optimality *without any initialization. By exploiting the special structure of the problem itself and of the 3D rotation space SO(3), we propose a Box-and-Ball algorithm, which solves the problem efficiently. The main idea of the work can be applied to many other problems as well.*

1. Introduction

The *3D-3D registration* problem (also known as *3D-3D alignment*, *3D absolute orientation*, *3D pose*) is one of the outstanding and very basic problems in computer vision. In this problem, two sets of 3D points are given and the task is to optimally align these two sets of points by estimating a best transformation between them. Due to its fundamental importance, it arises as a subtask in many different applications (e.g., object recognition, tracking, range data fusion, graphics, medical image alignment, robotics and structural bioinformatics etc. [31][18][20][32][14][26]).

What makes the problem challenging (and interesting) is that point-wise correspondences between the two point sets are often unknown *a-priori*. Under this situation, the registration problem is also known as the *simultaneous pose and correspondence* problem (or SPC problem in short).

The goal of the paper is to *globally* solve the SPC problem. We emphasize that we want to find a *global* solution, meaning that the estimated transformation and correspondences are desirable to be globally optimal, i.e., giving rise to the globally minimal L_2 error. To our knowledge, no efficient global method is available yet.

The difficulty of the SPC problem comes from the problem's *chicken-and-egg* character: it consists of two mutually interlocked sub-problems, *pose estimation* and *point correspondence*—solving one is often the pre-condition for solving the other. Moreover, the problem itself is heavily non-convex.

1.1. Existing solutions

The most popular class of methods for solving the SPC problem is probably the EM-type algorithms. The key idea is to use an *alternative* minimization procedure. Specifically, it starts from an initial estimation and solves the overall problem by *alternatively* solving the two sub-problems.

For example, the famous ICP (Iterative Closest Point) algorithm [6][12][29], and SoftAssign algorithm [13], and their variants [15] [10] [33], all work in an alternative fashion. Previously, the authors have also proposed a simple implementation based on the nearest matrix and Newton's root finding [25].

While these EM-type algorithms have been popular, they suffer from common and serious drawbacks. Because they are all *local* methods, no global optimality is guaranteed and there is an evident risk of local minima; moreover, their performances all rely on a good initialization and heavily depend on the spatial configuration (distribution) of 3D points; additionally, to use them many parameters need to be tuned. Ideally, we would like an algorithm whose *optimality* is guaranteed and who does not depend on point configuration.

There is a second class of methods that attempts to *decouple* those two subproblems, in the sense that it intends to solve one subproblem without solving the other. The basic idea is to exploit certain geometric properties that are *invariant* to either the transformation or the point correspondences. For example, the *PCA alignment* [21] and modal and spectral matching [30] [22] are methods of this kind. However, these methods are very sensitive to point set configuration, and often fail badly when the 3D points are *degenerately* configured (e.g.,isotropically).

978-1-4244-1631-8/07/$25.00 ©2007 IEEE

Yet another class of methods adopts the hypothesis-and-test idea. Examples include Generalized Hough Transform, Geometric Hashing and RANSAC etc. These methods are fast and accurate in many cases, but, the optimality is not guaranteed due to their obvious heuristic nature.

1.2. Contribution of the paper

This paper presents a novel framework for globally solving the 3D-3D SPC problem. Our framework is grounded on a rigorous mathematical theory—*Lipschitz* optimization. Based on the theory, we propose a global search algorithm that simultaneously determines the transformation and point correspondences. The algorithm achieves arbitrarily high accuracy (up to machine precision) and is independent of point configuration. Moreover, it does not need any initialization, and has few parameters to be tuned.

Our algorithm makes use of the *branch-and-bound* search idea. We design an ***octree*** data structure to actually implement the search. Each of the tree nodes is a 3D box (voxel). Each box has a data member of a ball with radius ϵ. As such, we call our new algorithm the BnB (*box-and-ball*) algorithm.

It is *not* our intention to replace the already successful ICP or SoftAssign algorithm. However, our algorithm can be of help in places where these classic algorithms have difficulties (e.g.,when a global solution is desirable, or a good initialization is hard to obtain, or point distribution can be arbitrary, etc.).

2. Problem statement

Denote $\mathbf{x}_i^{\mathrm{T}}$ and $\mathbf{y}_i^{\mathrm{T}} (i \in [1, n])$ as row vectors from matrices $X \in \mathbb{R}^{n\times 3}$ and $Y \in \mathbb{R}^{n\times 3}$, respectively. X and Y represent two 3D point sets, and $\mathbf{x}_i$ and $\mathbf{y}_i$ are coordinates of the i-th points in the two sets, respectively.

Formally, the SPC problem aims at finding an optimal rigid motion and point correspondences that minimize the (squared) L_2 error, where the motion is represented by a translation $\mathbf{t}$ and a rotation $\mathtt{R}$, and the correspondences are represented by a *permutation matrix* $\mathtt{P} \in \mathbb{P}^n$ whose entries are $\mathtt{p}_{ij} = 1$ if $\mathbf{x}_i$ matches $\mathbf{y}_j$, and 0 otherwise.

Suppose $\mathbf{x}_i$ and $\mathbf{y}_{i(\mathtt{P})}$ are matched (under $\mathtt{P}$), then the L_2 error is given by:

$$d(X, Y|\mathtt{P}, \mathtt{R}) = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i - \mathtt{R}\mathbf{y}_{i(\mathtt{P})} - \mathbf{t}\|^2 \qquad (1)$$

In order to highlight our main contribution, we assume that both X and Y are of equal size ($|X| = |Y| = n$), and there is no missing-data (or outliers). Dealing with such an ideal and the simplest case does not compromise our method. This is because: (1) up to date, no global optimal method ever exists even for the simplest case; (2) in practice, the user can often apply some robust techniques beforehand to remove outliers; (3) our method, by some modifications, could possibly handle the missing-data situation.

Since the translation can be relatively easily found before solving other parameters, we assume that the centers of the two point sets are pre-aligned at the origin. Now the problem is neatly formulated as minimizing the mean L_2 error metric over $\mathtt{P} \in \mathbb{P}^n$ and rotation $\mathtt{R} \in SO(3)$:

$$d(X, Y|\mathtt{P}, \mathtt{R}) = \frac{1}{n}\|X - \mathtt{P}Y\mathtt{R}^{\mathrm{T}}\|^2. \qquad (2)$$

Note that in the problem setting we do not make any assumption about the points' configuration (e.g., they do not necessarily reside on a smooth surface, or have certain local connectivity). In addition, we deliberately avoid the use of any domain-dependent information (e.g. image patch correlation); Instead, we regard the point coordinates as the only available information. All these restrictions will render our method more generic, hence having wider applicabilities.

2.1. Two subproblems

It turns out that: even with the simplest SPC model the task of simultaneously finding the *optimal* $\mathtt{R}$ and $\mathtt{P}$ is extremely hard. The problem itself is non-convex, with binary entries in $\mathtt{P}$. Conventional gradient-descent methods can hardly reach a global optimum. The alternative EM-type algorithms do not work either.

It is noticed however: if either $\mathtt{R}$ or $\mathtt{P}$ is known or fixed in advance, then finding the other is relatively easy. In fact, solving one of the subproblems (conditioning on the other) proves to be a *convex* (hence *global*) problem.

If $\mathtt{P}$ is fixed, then the optimal $\mathtt{R}$ (for a given $\mathtt{P}$) can be found by e.g. SVD [2], quaternion method [18], or [27][11] optimally and in closed-form.
If $\mathtt{R}$ is fixed, then finding the optimal $\mathtt{P}$ (for a given $\mathtt{R}$) is a typical *linear assignment problem* [9], which can be solved globally by Linear Programming (LP), or the *Hungarian algorithm* more efficiently.

3. The main idea

The paper aims at finding a truly globally-optimal $\mathtt{R}$ and $\mathtt{P}$. This goal is non-trivial and little successful attempt was made before. To achieve the goal, we propose a novel framework, which basically conducts an *exhaustive search* over all possible combinations of $\mathtt{R}$ and $\mathtt{P}$.

This seemed to be impossible, because the entire combinatorial space to be searched is the *product space* of $\mathbb{P}^n$ and $SO(3)$), which is extremely huge. A naive *brute force* search is doomed to fail. Note that there are *infinitely many* rotations and *exponentially many* permutations.

To overcome this, we resort to the separable property of the problem mentioned above. Specifically, we observe that the problem can be separated into two sub-problems. For a given $\mathtt{R}$ one can easily find an optimal $\mathtt{P}$, and vice versa. This actually suggests that the search needs only be carried out over *half* of the parameter space.

At first thought, to perform such a half-space search it appears natural (and tempting) to start from searching over all *permutations* (as opposed to searching over all *rotations*). The reasons are obvious: (1) there is only a finite numbers of possible permutations, but there are infinitely many rotations; (2) the complexity of computing a rotation for a given permutation is no more than a 3×3 SVD which is highly efficient. If one did follow this direction, then a possible procedure would be as follows:

- *For every possible $\mathtt{P} \in \mathbb{P}^n$, compute the best $\mathtt{R}$ that gives the minimal residual error $d(\mathtt{P}, \mathtt{R})$. If the least residual error obtained so far is less than a prescribed accuracy, or the entire $\mathbb{P}^n$ space has been exhausted, then output the least error and its corresponding optimal parameter pair $(\mathtt{P}, \mathtt{R})$.*

While the above procedure is workable, it is however, very *inefficient*—when n grows big it becomes intractable, because the number of permutations increases exponentially. Check that $20! \approx 2.4 \times 10^{18}, 50! \approx 3.1 \times 10^{64}$ and $100! \approx 9.3 \times 10^{157}$. Whereas, in real applications, to match hundreds of feature points is *not uncommon*.

In our implementation we choose to use the *opposite* direction, namely, to search over all 3D rotations. But, the problem is: how can one exhaustively search a continuous space $SO(3)$? The next section will show that, by exploiting the special structure of the space one *can* indeed do such a search quite efficiently. This is perhaps surprising and counter-intuitive. The outline of our procedure is as follows:

- *For every $\mathtt{R} \in SO(3)$, compute the best $\mathtt{P}$ that gives the minimal residual error $d(\mathtt{R}, \mathtt{P})$. If the least residual error obtained so far is less than a prescribed accuracy, or the entire $SO(3)$ space has been exhausted, then output the least error and the corresponding optimal parameter pair $(\mathtt{R}, \mathtt{P})$.*

4. Theory: Lipschitz optimization

This section explains how a global search over the rotation space $SO(3)$ can be carried out effectively.

A key observation to the success of our new method is that, certain error functions do *not* change too abruptly. In other words, *upper bounds* exist for the speed of changing of such functions. This property can be used to derive global optimal optimization algorithm. A rigorous mathematical theory, the Lipschitz optimization [16], precisely addresses this issue. Our new method is grounded on the theory of Lipschitz optimization. To begin with, let us consider a simple case: 1-dimensional global optimization. Here are some useful results.

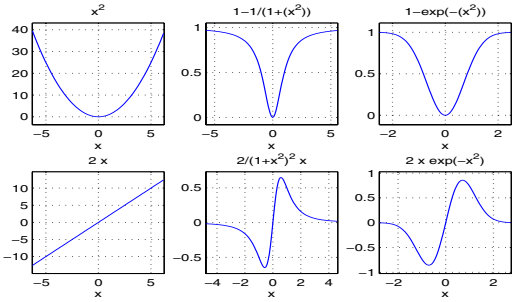

Figure 1. Top row: left: the original L_2 metric $f(x) = x^2$ (left); middle and right: two of its *Lipschitzized* metric functions, $f_1(x) = 1 - \frac{1}{(1+x^2)}$ and $f_2(x) = 1 - \exp(-x^2)$, respectively; Bottom row: their derivative functions. This figure shows that: while the function $f(x) = x^2$ has un-bounded derivatives in the real axis, the derivatives of its two Lipschitzized functions are bounded globally. For example, read form the figure we have $|f_1'(x)| < 0.65, \forall x \in \mathbb{R}$.

Definition 4.1. *A real-valued function f defined on the real domain $\mathbb{R}$ is a (global)* Lipschitz function *(or said satisfy a* Lipschitz condition*) if there exists a constant $L \geq 0$ such that for $\forall x, y \in \mathbb{R}$:*

$$|f(x) - f(y)| \leq L|x - y|. \tag{3}$$

The smallest such L is called the Lipschitz constant.

Result 4.2. *If f is a differentiable function with bounded derivatives $|f'(x)| \leq L$, then f is a global Lipschitz function with a Lipschitz constant estimate L.*

The Lipschitz condition says that a Lipschitz function cannot change too fast. Indeed, all its slopes are bounded by the constant L. This is the key insight that is used by the Lipschitz optimization to achieve global optimality.

4.1. Lipschitzize the L_2 metric

Note however that, the commonly adopted L_2 error function is *not globally Lipschitzian* in the entire domain of $\mathbb{R}$. This is easy to check: the slope of function $f(x) = x^2$ becomes arbitrary large as $x \to \infty$.

To salvage this, we introduce a "Lipschitzization" process to the L_2 metric. The idea is: by applying a proper algebraic transform to the traditional L_2 metric, we hope to reduce its speed-of-change while without altering the positions of its minima. This idea is explained in fig-1. There are many algebraic transforms that can be used for the purpose. Here we choose $f_1(x) = 1 - \frac{1}{(1+x^2)}$ (of fig-1).

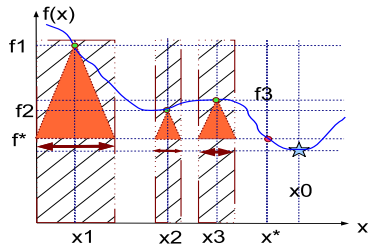

Figure 2. This figure shows a Lipschitz function $f(x)$ with Lipschitz constant L. The slope of the red regions equals L. f^* is the minimal value achieved so far. By evaluating the function at a single point x, one can safely eliminate a neighborhood region with radius $(|f(x) - f^*|/L)$ (shown in hatched lines) without losing the true global minimizer $x0$.

By plugging the transform $f_1(\cdot)$ into Eq.(2), we have

$$d_L = \frac{1}{n} \sum_{i=1..n} f_1(\|\mathbf{x}_i^{\mathbf{T}} - \mathtt{R}\mathbf{y}_i^{\mathbf{T}}\|^2). \tag{4}$$

This formula serves as our lipschitzized L_2 error function.

Besides making the traditional L_2 metric globally Lipschitzian, the Lipschitzization serves another purpose too. That is, it simplifies the estimation of the Lipschitz constant. This is seen from the following result.

Result 4.3. *The Lipschitz constant of eq.(4) is less than* $0.65\overline{\|\mathbf{y}_i\|}$*,where* $\overline{\|\mathbf{y}_i\|}$ *is the average length of all vectors* $\mathbf{y}_i, i = 1 \cdots n$.

Proof. Consider a single pair of matched points $(\mathbf{x}_i, \mathbf{y}_i)$ and its contribution to the error function. If we rotate $\mathbf{y}_i$ by an angle θ (in radian) about an arbitrary axis, then according to the *chain rule* the induced change to the error function is *at most* $(\theta \cdot |f_1'(\|\mathbf{y}_i\|)| \cdot \|\mathbf{y}_i\|)/n \leq (\theta \cdot 0.65 \cdot \|\mathbf{y}_i\|)/n$ (where 0.65 is read from fig-1 due to the Lipschitzization). Now consider all n matches. Using the fact that the Lipschitz constant of the sum of a set of Lipschitz functions is the sum of their constants, we reach: $L < 0.65\overline{\|\mathbf{y}_i\|}$. The inequality holds even when the correct point-correspondences are unknown. Such completes the proof. □

4.2. Search for global optimum

In this subsection we show that: by using the Lipschitz condition, one can quickly eliminate *un-promising* regions (in the *domain* space) which cannot possibly contain the global optimum.

Consider a one-dimensional Lipschitz function (shown in fig-2) with a Lipschitz constant L. We wish to find the global minimum within the given region D.

Let $f^* = f(x^*)$ be the minimal function value obtained so far. Suppose we are now evaluating the function at x. If $f(x) \leq f^*$, then we replace f^* with $f(x)$; Otherwise, we conclude that:

Within a neighborhood region of x with radius $\epsilon = \frac{|f(x)-f(x^*)|}{L}$, there cannot *be any domain variable y such that $f(y) > f(x^*)$.*

This result follows directly from $|f(x)-f(x^*)| \leq L|x-x^*|$ (i.e., the Lipschitz condition). Consequently, as shown in the figure, by evaluating the function at a single point, we can safely remove an ϵ-neighborhood region without losing the global minimum.

Repeating this process systematically within D, we will eventually find the global minimum in D. This is the basic idea of *univariate* Lipschitz global optimization algorithms (e.g., the Piyavskii-Shubert algorithm [16]).

4.3. Finding neighbors in SO(3)

Although the above idea of Lipschitz optimization can be extended to multi-dimensional *vector space*, where the ϵ-neighborhood becomes a hyper-cube inside the domain, it is however, *non-trivial* to extend it to the rotation space $SO(3)$. The major obstacle is that: it is not apparent at all how to define a proper neighborhood in $SO(3)$. Because all 3D rotation matrices do not form a vector space, the neighboring region to a given rotation is not a hyper-cube.

To overcome this: next we will first work out the shape of the neighborhood region in $SO(3)$, then we will approximate it by a simpler cubic region.

Parametrization of rotations A rotation matrix $\mathtt{R}$ with rotating angle at most π can be represented by the *exponential map* of a real 3-by-3 skew-symmetric matrix $[\mathbf{v}]_\times$ induced by a 3-vector $\mathbf{v}$, namely, $\mathtt{R} = \exp([\mathbf{v}]_\times)$, where $\exp(\cdot)$ is the *matrix exponential* defined by $\exp(\mathtt{M}) = \sum_{k=0}^{\infty} \frac{\mathtt{M}^k}{k!}$. The angle of the rotation equals to the norm $\|\mathbf{v}\|$, and its axis is given by the unit vector $\bar{\mathbf{v}}$. Conversely, $[\mathbf{v}]_\times = \log(\mathtt{R})$.

Geometrically, this parametrization is equivalent to using a 3D solid ball with radius $r \leq \pi$ (which we call the π-ball) to represent all 3D rotations (see fig-3) up to angle Π. The parametrization is one-to-one on the interior of the π-ball. Any voxel (cell) inside the π-ball represents a cluster of similar rotations. Because 3D rotations do not commute in general, namely, $\mathtt{R}_1\mathtt{R}_2 = \exp([\mathbf{v}_1]_\times)\exp([\mathbf{v}_2]_\times) \neq \exp([\mathbf{v}_1]_\times + [\mathbf{v}_2]_\times)$. In fact, the correct relationship is given by the celebrated Baker-Campbell-Hausdorff (**BCH**) formula:

$$\exp(\mathtt{A})\exp(\mathtt{B}) = \exp(\mathbf{BCH}(\mathtt{A},\mathtt{B})) =$$
$$\exp(\mathtt{A}+\mathtt{B} + \frac{1}{2}[\mathtt{A},\mathtt{B}] + \frac{1}{12}([\mathtt{A},[\mathtt{A},\mathtt{B}]] - [\mathtt{B},[\mathtt{A},\mathtt{B}]]) + \cdots) \tag{5}$$

where $[\cdot]$ denotes the Lie bracket.

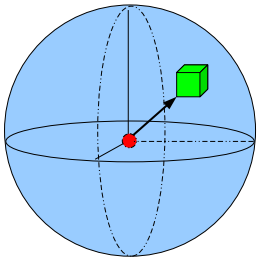

Figure 3. Parametrization of $SO(3)$ using a radius π ball in $\mathbb{R}^3$. A small cell inside the π-ball represents a cluster of similar rotations.

Angular distance between two rotations. The group of rotations has a natural metric structure, defined by the angular distance as follows. Let $\mathtt{R}_1$ and $\mathtt{R}_2$ be two rotations. Then, the distance between these two rotations is measured by the angle θ ($0 \leq \theta \leq \pi$) of the composite rotation of $\mathtt{R}_2^{-1}\mathtt{R}_1$, or $\mathtt{R}_1^{-1}\mathtt{R}_2$. By using the exponential-map we have

$$\theta = \|\log(\mathtt{R}_1^{-1}\mathtt{R}_2)\|/2 = \|\log(\exp(-[\mathbf{v}_1]_\times)\exp([\mathbf{v}_2]_\times))\|/2 = \|\mathbf{BCH}([\mathbf{v}_2]_\times, -[\mathbf{v}_1]_\times)\|/2. \quad (6)$$

Now, for any two given rotations we can compute their angle distance using the $\mathbf{BCH}$-formula. As a result, the (exact) neighborhood structure in SO(3) is properly defined.

Vector distance between two rotations. Directly computing the *exact* neighborhood in rotation space (using the **BCH** formula) can be very involving. We hence propose a simpler *approximation* which is based on the following result ([3]).

By exp-map any vector $\mathbf{v}$ inside the π-ball is mapped to a valid rotation $\mathtt{R} = \exp([\mathbf{v}]_\times)$. For two vectors inside the π-ball,$\mathbf{v}_1, \mathbf{v}_2$, we can naturally compute their *vector distance* $\|\mathbf{v}_1 - \mathbf{v}_2\|$. Then we have,

Theorem 4.4. *For any two 3D rotations, $\mathtt{R}_1$ and $\mathtt{R}_2$, the following inequality holds:*

$$\theta(\mathtt{R}_1, \mathtt{R}_2) = \|\mathbf{BCH}([\mathbf{v}_2]_\times, -[\mathbf{v}_1]_\times)\|/2 \leq \|\mathbf{v}_1 - \mathbf{v}_2\|. \quad (7)$$

Proof. It is only necessary to prove that the tangent-map T of the exp-map is of norm not exceed 1. But $T_{\mathbf{v}} \exp = R^e_{\exp(\mathbf{v})} \circ \Phi(\mathrm{ad}(\mathbf{v}))$, where $R^e_{\exp(\mathbf{v})}$ is right-translation from T_e to $T_{\exp(\mathbf{v})}$, ad is the adjoint operator and Φ the holomorphic function $\Phi(z) := \frac{e^z - 1}{z}$ (,z being complex, c.f. [3]). As $R^e_{\exp(\mathbf{v})}$ is an isometry and $\|\Phi(z)\| \leq 1$, the result follows. □

The inequality suggests a much simpler way to compute neighborhood. Recall that in Lipschitz optimization each time we need to remove a neighborhood region if it is not promising. Instead of computing an *exact* neighborhood using the the **BCH**-formula, we can compute the vector distance to obtain an approximate neighborhood. This approximation, remarkably, incurs no loss of optimality, because the approximate neighborhood-region to be removed is actually smaller than what is predicted by the **BCH** formula. In other words, the search is more conservative and impossible to miss global optimum.

5. Implementation: the BnB algorithm

So far we have done necessary preparations: now we have a globally Lipschitz error function; we have a concrete π-ball representing the space to be searched; we have also a well-defined ϵ-neighborhood and its vector-distance approximation. The only thing remaining is to design an algorithm to search the the π-ball systematically, eliminating those unpromising regions and keep the promising regions until the global optimum is found.

Our algorithm is based on *branch-and-bound*, a well-known technique in global optimization. It was adopted by vision researchers for object recognition and geometric matching a decade ago [7] [19], and recently is recurrent in new contexts with new bounding techniques [1] [28].

Branch-and-bound (minimization) works by recursively subdividing feasible transformation region into a set of subregions, and eliminating subregion which cannot contain global minima by evaluating lower *bounds* over the considered subregion. The process stops when the global minimum is bracketed into a small enough region that guarantees the desired accuracy.

For our SPC problem at hand, a Lipschitz constant ($L < 0.65\overline{\|\mathbf{y}_i\|}$) is used to predict the bounds.

5.1. Octree data structure

Manipulating (e.g., cut/combine/intersect) a set of ϵ-balls is not an easy task. Alternatively, we propose the use of regular cubic boxes. This results in a simple **octree** data structure—which is subsequently used to implement the branch and bound search.

We call our method the **box-and-ball** (BnB) algorithm. It starts from a 3D box circumscribing the π-ball. Suppose f^* is the best function value obtained so far. Evaluate the function at the center point (denoted by x) of the box currently under examined. Now compute the radius ϵ of a neighboring ball, $\epsilon = (f(x) - f(x^*))/L$. If the ϵ-ball encloses the entire volume of the current box, then this box can be eliminated safely; otherwise subdivide the box into eight sub-boxes. Repeat this process until the desired accuracy is reached. For visualization purpose we show a 2D slice of the octree in fig-4, i.e., a quadtree.

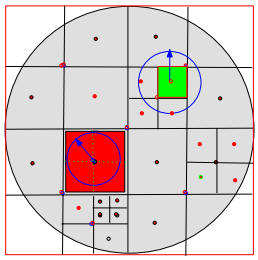

Figure 4. A 2D slice (viz. a quadtree) of our octree data structure. The blue circles indicate the ϵ-balls. During a branch-and-bound process, the green box can be safely removed as its volume is contained entirely inside its ϵ-ball, but the red box deserves further subdivision (the figure is better viewed in color).

5.2. The box-and-ball (BnB) algorithm

In detail, the BnB algorithm proceeds as follows:

1. (Preparation) Build an octree data structure, each node of which is a box. At the very beginning there is only one box which circumscribes the π-ball. Each box has an associated ϵ-ball. Radii of these balls are set to 0 initially, $\epsilon = 0$. Make a rough estimate of the rotation estimation, by any method at all (e.g., a random guess). Denote the best function value obtained so far as f^*. Make a proper estimate to the Lipschitz constant L. The desired accuracy is set to γ.
2. (Branch and bound search) Do repeated depth-first-search over the octree. For each tree node:
   (a) Compute a rotation matrix $\mathtt{R}$ using the center point x of the box.
   (b) Compute the best permutation matrix $\mathtt{P}$ for the given $\mathtt{R}$ by calling the Hungarian algorithm. Output the current function value $f(x)$. If $f(x) \leq f^*$, then update f^*; otherwise compute the radius of its ϵ-ball by $\epsilon = |f(x) - f^*|/L$.
   (c) If the box is entirely contained in its own ϵ-ball, then remove this box from the octree; otherwise, subdivide it into eight sub-boxes and insert these sub-boxes at the current position in the octree.
   (d) Stop the search when $f^* \leq \gamma$.
3. Report the center x_{opt} of one of the remaining boxes that produces the least function value as the optimal rotation $\mathtt{R}_{opt}$, and the corresponding permutation $\mathtt{P}_{opt}$ as the optimal correspondences. End.

Remark-1. Different tree-search strategies (such as breadth-first-search or best-first-search [4]) can be used as well[8]; **Remark-2.**The tighter the estimated Lipschitz constant is, the quicker the algorithm terminates; **Remark-3.** The Hungarian algorithm used in the inner loop is a guaranteed global solver. It always returns the best permutation for a hypothesized rotation, even when the rotation is not the correct one.

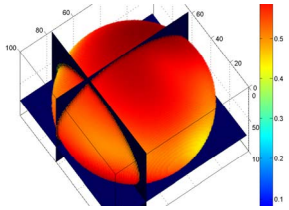

Figure 5. The π-ball with function evaluated at every grid point inside the ball. We also show the positions of three slices.

6. Experiment results

To validate our new framework, we have conducted experiments with both synthetic and real-world data. This section briefly presents some results, mainly demonstrating the effectiveness of our approach.

Synthetic data. We generate a set of Gaussian random vectors with zeros mean and identity covariance matrix, and use it as the point set X. Such an isotropic symmetric point distribution would disable the conventional PCA alignment algorithm. We rotate X by a random angle to get the point set Y. We then perturb both sets by adding Gaussian (or uniform) noise with different levels of std. (or limits).

The existence of local minima. First of all, for illustration purpose we would like to examine the existence of local minima. Recall that when the point correspondences $\mathtt{P}$ is known, then the subproblem of finding rotation is convex, possessing only a single minimum. Since our error function is defined within the π-ball, we thus can compute such a ball with function evaluated at every discretized grid points. We depict such a function-value ball in fig-5 based on an experiment with 50 synthetic points. For visualization purpose we also give three of its 2D slices (fig-6, top row). They clearly exhibit a single local minimum.

Similarly, we compute the function-value ball for the case when the point correspondences $\mathtt{P}$ are not known, see the bottom row of fig-6. It convincingly reveals the existence of multiple local minima. This explains why, in the journey of seeking of global optimum, most of the conventional methods, be it a gradient-descent method or an EM-type algorithm, unavoidably fail.

Algorithm performance. We have implemented the proposed BnB algorithm in C++, and tested it on a modest PC (Intel P4, 3Ghz, 1GB) running Linux. Initialization of the rotation matrix is chosen at the center of the π-ball, i.e., the

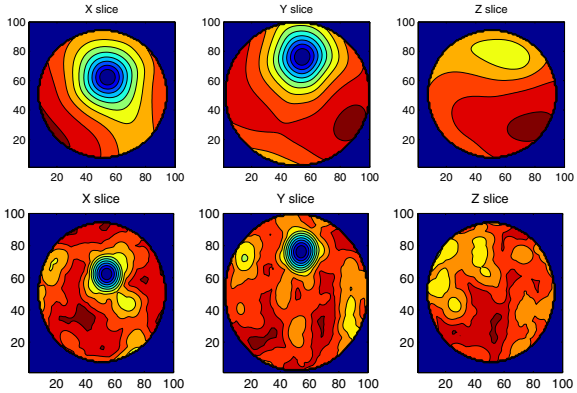

Figure 6. Three slices taken from the π-ball in fig-5 with error function contours. Top row: when correct correspondences are given, it exhibits only a single minimum; Bottom row: when correspondences are not known, there are multiple local minima. In the experiment the ground-truth rotation angle is 100 degrees.

identity matrix. The ground-truth rotation angles are randomly chosen among $[0, \pi)$.

Table-1 (in the next page) summarizes major aspects of the performance of our algorithm. The data are based on the average of 5 independent runs. Gaussian coordinates noise of level of 0.05 (in std.) is directly added to X. From the above table, we see that the algorithm achieves high accuracy,e.g. around 0.5 degrees in both rotating angle and rotation axis orientation. In fact, our algorithm is guaranteed to reach any prescribed accuracy (though the computation time may be long). In simulations,however, we simply stop the algorithm when the total volumes of unexplored boxes is below a threshold (e.g. 7% in our experiments).

Also, the estimated point correspondences are *correct* (, otherwise the overall L_2 error and the rotation error would be much bigger).

Our algorithm is much more efficient than brute-force search. At convergency, the numbers of-function-evaluations lie within the range of 10k to 300k. By contrast, this range is much less than the number of discrete cells inside a π-ball if the same accuracy is required. For example, if the brute-force search is used to achieve a 1-degree accuracy, then the space needs to be discretized into $\frac{4}{3}\frac{\pi^4}{(2\pi)^3} \times 360^3 \approx \mathbf{25}$ millions of cells.

The running time of our algorithm is acceptable to many applications (such as medical image landmark alignment). In fact, we consider our algorithm works quite fast, because otherwise in worst-case a general branch-and-bound search could have exponential complexity. The timing results in table-1 do not suggest a combinatorial explosion. The overall theoretic computational-complexity of our algorithm is not known yet. But its inner loop has only cubic complexity, because of the Hungarian algorithm [9].

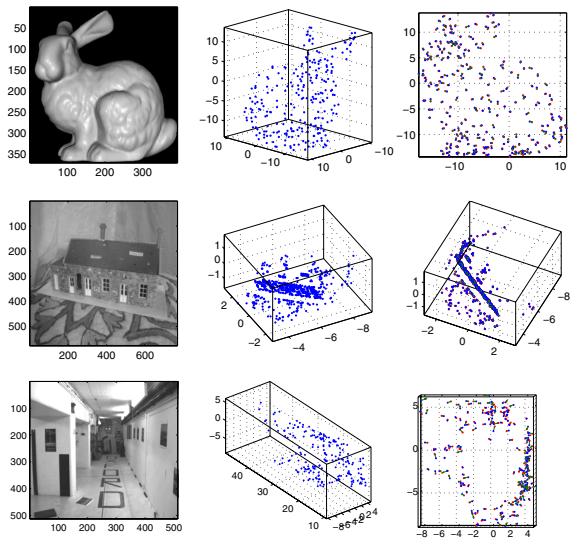

Figure 7. Results on real point clouds. Left: image; Middle: one of the 3D point sets; Right: registration result. The two point clouds are shown in red dots, and in blue dots, resp.

Real data. We test three sets of real 3D data. The first one is a down-sampled Stanford Bunny meshes with 500 facets. We artificially rotate the point cloud to generate a second instance and add extra noise. Our algorithm has successfully aligned the two point clouds as shown in fig-7 row-1.

We try our algorithm on point clouds which were reconstructed from images by triangulation algorithm [17][24]. The possible application context is to align two partially overlapping surfaces with shared landmarks. We manually identify and remove the outliers. The results are given in fig-7 row-2 and -3. We implement and test two existing algorithms, the ICP and SoftAssign. We find that on the same simulated data sets both of them work *only* when the initial rotation is within an angular distance of less than 50 degrees, i.e. a reasonably good initialization, while our algorithm always produces the right answer without any initialization. It is worth mentioning that, if a good initialization is available then our algorithm works much faster, as the good initial rotation already provides a near-optimal bound.

7. Discussion and conclusion

The novelty of the work lies in the Lipschitz framework and in the efficient implementation of the octree box-and-ball algorithm. The advantage of our method is that it reliably produces estimation with a certificated global optimality within reasonable time.

The effectiveness of searching over all possible rotations makes our method appealing to many other applications too. Particularly, the method can be extended to solve the motion estimation problem—to infer motion information from

num points(n)	10	20	40	60	80	100	150	200
angle err (deg)	0.21	0.15	0.16	0.32	0.51	0.83	0.72	0.3
axis err (deg)	0.47	0.89	0.91	0.62	1.20	0.46	0.26	0.73
num func evals	19509	23007	28328	28328	70312	118656	256320	255767
time spent (sec)	0.2	1.9	10.2	48.6	107.5	156.2	722.1	1103.2

Table 1. Algorithm performance

2D images—one of the central topics in multiview geometry [17][23]. The potential of applying our method there is clear.

There are areas for further improvements:(1) whilst the running time is acceptable, it is still very slow for large point sets. The ICP is much superior in this aspect. In future work we plan to explore alternative tree-searching strategies [8] [4]; (2) the inner loop of the algorithm can be easily augmented by more powerful matching algorithms using richer domain-dependent features such as mutual information [31], shape context [5], SIFT, EGI etc.; (3) although we argued that our framework could be adapted for dealing with outliers, at the moment how to achieve that goal is not all that apparent. The problem of outlier is however, in its own right, an interesting topic deserving specific research effort [24].

The main idea and the general framework of the paper will illuminate many other different problems in vision and beyond vision.

Acknowledgement. We wish to thank the anonymous reviewers for invaluable suggestions, and thank Dr F.Kahl for many inspiring discussions. NICTA is funded by Australian Government.

References

[1] S. Agarwal, M. Chandraker, F. Kahl, D. J. Kriegman, and S. Belongie. Practical global optimization for multiview geometry. In *ECCV*, pages 592–605, May 2006.

[2] K. Arun, T. Huang, and S. Blostein. Least-squares fitting of two 3-d point sets. In *PAMI*, volume 9, pages 698–701, 1987.

[3] C. J. Atkin. Weakly bounded banach lie groups. *Victoria International Conference*, pages 9–13, 2004.

[4] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. *CVPR*, 1997.

[5] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *Proc. of CVPR*, pages 26–33, 2005.

[6] P. Besl and N. McKay. A method for registration of 3-d shapes. In *PAMI*, volume 14, pages 239–256, 1992.

[7] T. Breuel. Fast recognition using adaptive subdivisions of transformation space. In *CVPR*, pages 445–451, 1992.

[8] T. Breuel. A comparison of search strategies for ggeometric branch and bound algorithms. In *ECCV*, pages 28–31, May 2002.

[9] C.Papadimitriou and K.Steiglitz. *Combinatorial optimization: algorithms and compelxity*, volume 5(7). Apr 1990.

[10] P. David, D. DeMenthon, R. Duraiswami, and H. Samet. Simultaneous pose and correspondence determination using line features. In *CVPR*, pages 424–431, June 2003.

[11] D. W. Eggert, A. Lorusso, and R. B. Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications*, 9:272–290, 1997.

[12] A. Fitzgibbon. Robust registration of 2d and 3d point sets. In *Image and Vision Computing*, pages 1145–1153, December 2003.

[13] S. Gold, A. Rangarajan, C. Lu, S. Pappu, and E. Mjolsness. New algorithms for 2d and 3d point matching: Pose estimation and correspondence. In *Pattern Recognition*, volume 31, pages 1019–1031, August 1998.

[14] J. Goldberger. Registration of multiple point sets using the *em* algorithm. In *ICCV*, volume 2, pages 730–736, Sep. 1999.

[15] S. Granger and X. Pennec. Multi-scale em-icp: A fast and robust approach for surface registration. In *ECCV*, pages 418–432, June 2002.

[16] P. Hansen and B. Jaumard. *Lipshitz Optimization. In: R. Horst and P. Pardalos. Handbook of Global Optimization.* Kluwer Academic Publishers, 1995.

[17] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[18] B. Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA-A*, 4:629–642, April 1987.

[19] D. Huttenlocher and S.Ullman. Recognizing solid object by alignment with an image. *IJCV*, 5(7):195–212, Apr 1990.

[20] F. Jurie. Solution of the simultaneous pose and correspondence problem using gaussian effor model. *CVIU*, 73:367–373, March 1999.

[21] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Shape matching and anisotropy. 2004.

[22] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, volume 2, pages 1482– 1489, 2005.

[23] H. Li. A simple solution to the six-point two-view focal-length problem. In *Proc. of ECCV*, pages 200–213, 2006.

[24] H. Li. A practical algorithm for l-infinity triangulation with outliers. In *CVPR*, 2007.

[25] H. Li and R. Hartley. A new and compact algorithm for simultaneously matching and estimation. *ICASSP*, 3:5–8, 2004.

[26] A. Makadia, A. P. IV, and K. Daniilidis. Fully automatic registration of 3d point clouds. In *CVPR*, pages 1297– 1304, June 2006.

[27] N. Ohta and K. Kanatani. Optimal estimation of three-dimensional rotation and reliability evaluation. In *ECCV*, pages 175–187, June 1998.

[28] C. Olsson, F. Kahl, and M. Oskarsson. The registration problem revisited: Optimal solutions from points, lines and planes. In *CVPR*, pages 1206–1213, June 2006.

[29] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3DIM*, May 2001.

[30] S. Sclaroff and A. Pentland. Model matching for correspondence and recognition. In *PAMI*, June 1995.

[31] P. Viola and I. W. M. Wells. Alignment by maximization of mutual information. In *ICCV*, pages 16–23, June 1995.

[32] L. Wolf, A. Shashua, and Y. Wexler. Join tensors: on 3d-to-3d alignment of dynamic sets. In *ICPR*, Sep 2000.

[33] W. Zhao, D. Nister, and S. Hus. Alignment of continuous video onto 3d point clouds. *PAMI*, 27:1305–1318, August 2005.