LieTrICP: An improvement of trimmed iterative closest point algorithm

Jianmin Dong a, Yaxin Peng a,*, Shihui Ying a, Zhiyu Hu b

a Department of Mathematics, Shanghai University, Shanghai 200444, China
b Institute of NanoMicroEnergy, Shanghai University, Shanghai 200444, China

1. Introduction

Point set registration is a technique widely used in areas such as computer vision, medical image analysis and pattern recognition. The purpose of registration between two point sets is to find their correspondence and transformation, transforming one point set to its counterpart. A good algorithm for point set registration should (1) match the two point sets correctly, and (2) be robust to noisy points, outliers and missing points, etc.

In [1], a variety of algorithms for point set registration have been reviewed and compared, including the popular method of Singular Value Decomposition (SVD) [2]. The Iterative Closest Point (ICP) algorithm, proposed by Besl and McKay [3], and Chen and Medioni [4], is a well-known efficient method for point set registration. The ICP algorithm defines an energy function which indicates the summation of squared distances for point pairs in two sets, and finds the optimal transformation by minimizing this energy function. However, the ICP algorithm has some obvious disadvantages: it requires a rough initial registration, is applicable to only rigid registration, and is sensitive to noise and outliers.

Regarding these issues, many researchers have dedicated to improve the ICP algorithm. In order to provide initial transformations close to the optimal solution, rough estimation methods, such as the principal component analysis (PCA) and the independent component analysis, were proposed [5–7]. In order to extend the transformation type from rigid to non-rigid transformations, Zha et al. [8] extended the rigid registration to the case of isotropic scale deformation; Du et al. [9–11] proposed non-rigid ICP algorithms; Ying et al. [5,7] proposed the so-called Scale-ICP algorithm. Many efforts have also been made to improve the accuracy for the point pairs. Liu added geometric constraints and weights to the ICP algorithm [12,13]. However, all the algorithms mentioned above no longer work when the overlap rate between two point sets (i.e., the number of points in the first set that really have matched points in the second set divided by the number of all points in the first set) is low.

To overcome the low overlap rate problem, Chetverikov et al. [14] proposed the Trimmed ICP (TrICP) algorithm. The TrICP algorithm ticks out outliers and then conducts ICP by minimizing the Trimmed Squared Distance (TSD). On this basis, Phillips et al. [15] proposed a fast automatic overlap rate estimation method. Du et al. [16] proposed an iterative closest point algorithm by introducing a new objective function. Later, Du et al. introduced an isometric scale methods for partial registration [17]. However, these algorithms [14–16] consider only the rigid transformation, and they may fail when the anisotropic scale transformation exists.

In this paper we introduce Lie group parametrization [18,19] to optimize the 2D and 3D point set registrations, which can put the registration under a unified framework. We conduct the registration via iteration. In each iteration, we first compute the point-to-point correspondence for every point, then make use of the trimmed strategy to select overlapped point pairs in two sets, and optimize the transformation by minimizing the TSD of these
selected points via Lie group parametrization. This algorithm can deal with registration for the cases of anisotropic scale transformation and low overlap rate, and provides accurate and robust results. We term our algorithm as LieTrICP.

This paper is organized as follows. In Section 2 we introduce the classic ICP and the TrICP algorithms. Section 3 describes our LieTrICP algorithm. Experiments and analysis are shown in Section 4. Section 5 concludes the paper.

2. Related works

2.1. ICP algorithm

Given two point sets, i.e., the source set \( X = \{x_i|i = 1, \ldots, m\} \) and the target set \( Y = \{y_j|j = 1, \ldots, n\} \), the ICP algorithm aims to find a transformation \( T \), such that \( T\cdot X \) matches \( Y \). Given the \( k \)-th iteration, the \((k+1)\)-th iteration is as follows:

1. Having the transformation \( T^k \) fixed, for each \( x_i \in X \), select \( z_i^k = \arg \min_{y \in Y} \| T^k \cdot x_i - y \|^2 \) as the closest point for \( T^k \cdot x_i \). Then we obtain the correspondence \( \mu(x_i, T^k) \), which matches point \( x_i \) to \( z_i^k = \mu(x_i, T^k) \in Y \), for \( i = 1, \ldots, m \).

2. Having the correspondence \( \mu(x_i, T^k) \) fixed, update the transformation from \( T^k \) to \( T^{k+1} \) by

\[
T^{k+1} = \arg \min_{T} \sum_{i=1}^{m} \| T \cdot x_i - z_i^{k} \|^2.
\]

The optimal transformation can be obtained by iterating these two steps until convergence. However, ICP often fails since noise and outliers affect the match, as ICP takes all points into account.

2.2. TrICP algorithm

The TrICP algorithm [14], an adaptive matching algorithm using the overlap rate \( r \), improves ICP by trimming out unmatched points. Given a coarse transformation \( T^0 \) from \( X \) to \( Y \) and an initial overlap rate \( r \), the \((k+1)\)-th iteration of TrICP can be expressed as follows:

1. For each point \( x_i \in X \), find \( T^0 \cdot x_i \)'s closest point \( z_i^0 = \mu(x_i, T^0) \) in \( Y \). Compute the individual squared distance \( d_i^0 = \| T^0 \cdot x_i - z_i^0 \|^2 \).

2. Sort \( d_i^0 \) in an increasing order, and select the first \( m_r = \lfloor r \cdot m \rfloor \) point pairs, where \( \lfloor \cdot \rfloor \) is the round-off function. Calculate the TSD

\[
e^k = \sum_{i=1}^{m_r} d_i^k.
\]

3. Update the transformation from \( T^k \) to \( T^{k+1} \) by

\[
T^{k+1} = \arg \min_{T} \sum_{i=1}^{m_r} \| T \cdot x_i - z_i^{k} \|^2.
\]

4. If any of the given stopping conditions is satisfied, then exit; otherwise, continue the above three steps.

The overlap rate \( r \) is fixed in the above procedures. In what follows, we discuss how to choose a suitable \( r \) in the interval \([r_{\text{min}}, r_{\text{max}}]\). Under this \( r \), the registration error should be minimized. The error we adopt in the paper is defined as

\[
\psi(r) = \frac{e(r)}{r^{1+s}}
\]

where \( e(r) = (1/m_r) \sum_{i=1}^{m_r} d_i^r \) is the trimmed mean squared error.

The minimizer of (2) can be obtained by the Golden Section method. Typically we set \( r_{\text{min}} = 0.4 \) and \( r_{\text{max}} = 1 \). However, this method is time consuming. We will use a fast automatic overlap rate estimation technique [15,16] for anisotropic transformation in Section 3.2.1.

Comparing TrICP and ICP, the difference is that TrICP introduces the overlap rate and only uses the corresponding trimmed point pairs. In fact, ICP is a special case of TrICP for \( r = 1 \). TrICP approaches ICP, as the overlap rate \( r \) approaches one.

3. LieTrICP: TrICP under Lie group framework

The TrICP algorithm only considers the rigid transformation. As a result, it fails in the case of isotropic or anisotropic scale transformations. In this section, we describe the proposed LieTrICP which can solve this problem.

The solution for the isotropic scale transformation is relatively simple, as some implicit methods, e.g., the methods using SVD [1,2,5], already exist. When the anisotropic scale transformation is involved, the SVD methods fail due to the non-commutativity of the diagonal and the rotation matrices. We introduce the Lie group parametrization method to handle the anisotropic cases.

The details of LieTrICP for 2-D are given as follows. The case for 3-D is similar.

3.1. Initialization

We assume that the initial transformation \( T^1 \) is a combination of an anisotropic scale transformation \( S^1 \) followed by a rotation \( R^1 \) and a translation \( t^1 \), i.e., \( T^1(x_i) = R^1 \cdot S^1 \cdot x_i + t^1 \) for \( x_i \in X \). Following the idea of [7], we give the expressions of initializations \( S^1, R^1, \) and \( t^1 \) below:

1. Anisotropic scale transformation \( S^1 \): We set \( S^1 = \text{diag}(s_1, s_2) \), where \( s_1 \) is the scale coefficient along the \( x \)-direction, and \( s_2 \) is the coefficient along the \( y \)-direction. The expression \( s_1 \) is given by

\[
s_1 = \sqrt{\frac{\beta_1}{\lambda_1}}
\]

where \( \lambda_1 \) and \( \beta_1 \) are the eigenvalues of covariance matrices \( M_x \) for \( X \) and \( M_y \) for \( Y \) respectively, with the assumptions that \( \lambda_1 \leq \lambda_2 \) and \( \beta_1 \leq \beta_2 \).

2. Rotation \( R^1 \): We set \( R^1 \) as

\[
R^1 = [v_1, v_2][u_1, u_2]^{-1},
\]

where \( u_1 \) and \( v_1 \) are corresponding eigenvectors of \( \lambda_1 \) and \( \beta_1 \).

3. Translation \( t^1 \): We set \( t^1 \) as

\[
Y_c - R^1 X_c
\]

where \( X_c \) and \( Y_c \) are centroids for sets \( X \) and \( Y \) respectively.

3.2. Iteration

Let \( k \) be the current iteration, \( T^k = (S^k, R^k, t^k) \) the current transformation.

3.2.1. Estimating trimmed correspondence

We update the correspondence as ICP, and compute the individual squared distance \( d_i^k \). Then, the useful matched point pairs should be selected by the trimmed strategy. We sort the squared distance \( d_i^k \) in an increasing order to find \( m_r \), such that \( r = m_r/m \) reaches the minimum of (2), and then select the first \( m_r \) pairs of points corresponding to the current overlap rate \( r \).

The above step makes the computation time of one iteration in our method comparable with that of ICP. The reason is that both algorithms spend almost the same amounts of time on computing...
correspondence and transformation, which are the most time-consuming part. The computation time of other calculations is not significant.

The problem to find $T^{k+1}$ becomes
\[
(s^{k+1}, r^{k+1}, t^{k+1}) = \arg \min_{S \in D^+ \cdot R \cdot SO(n), t \in \mathbb{R}^3} \sum_{i=1}^m \| R \cdot S \cdot x_i + t - z_i^k \|^2,
\]
where $D^+$ is a group of diagonal matrices with non-negative diagonal elements, and $SO(n)$ is an $n$-dimensional Euclidean space rotation group.

In order to prevent possible degeneration which may occur if there are no constraints for the scale factor [5], we set a constraint for scale interval as $S \in D^+ \cdot [S^L \leq S \leq S^U]$. Here $S^L \in D^+$ and $S^U \in D^+$ are the lower and upper bounds of $S$ for the constraints respectively.

3.2.2. Solving transition $t$

The iteration formula for $t$ can be obtained by setting $\partial E(S, R, t)/\partial t = 0$ for the objective function Eq. (3), i.e.,
\[
t^{k+1} = \sum_{i=1}^m z_i^k/m_i - R \cdot S \cdot \left( \sum_{i=1}^m x_i/m_i \right)/m_i,
\]
which is equivalent to
\[
t^{k+1} = Z^k - R \cdot S \cdot X_t,
\]
where $Z^k = \sum_{i=1}^m z_i^k/m_i$ and $X_t = \sum_{i=1}^m x_i/m_i$ are centroids of the two point sets $Z^k = [z_i^k]_{m_i=1}$ and $[x_i]_{m_i=1}$, respectively. Let $\bar{x}_i = x_i - X_t$ and $\bar{z}_i = z_i^k - Z^k$. Eq. (3) can be re-arranged as
\[
(s^{k+1}, t^{k+1}) = \arg \min_{S \in D^+ \cdot R \cdot SO(n)} \sum_{i=1}^m \| Z^k - R \cdot S \cdot \bar{x}_i \|^2.
\]

Then solving Eq. (3) is reduced to the optimization problem (5) with respect to scale and rotation.

3.2.3. Lie group parameterizations

The arguments in Eq. (5) can be viewed as elements in Lie groups, $SO(n)$ and $D^+$. More knowledge on Lie group can be found in [20,21].

Thus, under the bases of Lie algebras, we introduce the local approximation of the transformation sequence in anisotropic scale transformations and rotations. Specifically, letting $so(2)$ and $b^+$ be Lie algebras of Lie groups $SO(2)$ and $D^+$ respectively, we have
\[
\begin{align*}
so(2) & = \text{span}[E_{11} - E_{12}], & b^+ & = \text{span}[E_{11}, E_{22}],
\end{align*}
\]
where $E_q$ is a $2 \times 2$ matrix with the element of $i$-th row and $j$-th column being 1 and else being 0. Then, the transformation iterative sequence can be expressed as
\[
\begin{align*}
S & = S^k \exp(a_1 E_1 + a_2 E_2) \quad & R & = R^k \exp(a_3 E_3),
\end{align*}
\]
where $a_i \in \mathbb{R}$ for $i = 1, 2; E_1 = E_{11}, E_2 = E_{22}$ and $E_3 = E_{21} - E_{12}$.

Solving Eq. (5) then can be decomposed into the following series of optimization problems:
\[
a^* = \arg \min_{a = [a_1, a_2, a_3] \in \mathbb{R}^3} \sum_{i=1}^m \| Z^k_i - R^k \exp(a_1 E_1 + a_2 E_2) X_t^i \|^2
\]
\[
\text{s.t. } S^k \leq S^k \exp(a_1 E_1 + a_2 E_2) \leq S^U.
\]
(6)

In fact, when $a = [a_1, a_2, a_3] \in \mathbb{R}^3$ is sufficiently small, we have
\[
\begin{align*}
\exp(a_1 E_1 + a_2 E_2) & \approx I + a_1 E_1 + a_2 E_2 + \frac{1}{2} (a_1^2 E_1 + a_2^2 E_2), \\
\exp(a_3 E_3) & \approx I + a_3 E_3 - \frac{1}{2} a_3^2 E_3.
\end{align*}
\]
The objective function in Eq. (6) can be approximated by the following quadratic form:
\[
a^T H a + 2L^T a + C_0.
\]
(8)
Here $C_0 \in \mathbb{R}$; $H = [h_{pq}]_{i,j}$ and $L = [l_{pq}]_{i,j}$ are two matrices defined by
\[
\begin{align*}
h_{pq} & = \frac{1}{m_i} \left( \sum (w_{pi} w_{qj}) + \sum (\mu_{pq} w_{0i} - z_i^k) \right), & p, q = 1, 2, 3,
\end{align*}
\]
\[
l_p = \frac{1}{m_i} \sum (w_{pi} w_{0j} - z_i^k), & p = 1, 2, 3, i = 1, \ldots, m_i,
\]
where
\[
\begin{align*}
S^k & = \text{diag}(s_{11}^k, s_{22}^k), & S^U & = \text{diag}(s_{11}^U, s_{22}^U) & S^L & = \text{diag}(s_{11}^L, s_{22}^L),
\end{align*}
\]
In the minimization of Eq. (8), $C_0$ is negligible since it is a constant. The constraints (7) to the anisotropic scale is equal to
\[
(S^k)^{-1} S^L \preceq \exp \left( \sum_{i=1}^2 a_i E_i \right) \preceq (S^k)^{-1} S^U.
\]
Specifically, if $S^k = \text{diag}(s_{11}^k, s_{22}^k)$, $S^L = \text{diag}(s_{11}^L, s_{22}^L)$ and $S^U = \text{diag}(s_{11}^U, s_{22}^U)$, the minimization problem becomes the following constraint programming:
\[
\begin{align*}
\min f(a) & = a^T Ha + 2L^T a \\
\text{s.t. } (s_{11}^L/s_{11}^k) & \leq a_{11} \leq \log(s_{11}^U/s_{11}^k), & i, j = 1, 2.
\end{align*}
\]
(9)

3.3. Stopping condition

Note that at $k$-th iteration, the current TSD is
\[
e^k = \left| \sum_{i=1}^m (T^k \cdot x_i - z_i^k) \right|^2.
\]
(10)

Then, after updating transformation, the new error is
\[
\begin{align*}
e^{k+1} & = \left| \sum_{i=1}^m (T^{k+1} \cdot x_i - z_i^k) \right|^2.
\end{align*}
\]
(11)

In the algorithm, the stopping condition is chosen as (1) the maximum iteration number $N_{iter}$ has reached; (2) the mean TSD $e^{k}/m$ is sufficiently small; or (3) $|e^k - E^k|$ is sufficiently small.

3.4. Summary of algorithm

To summarize, the LieTriCP algorithm is listed as Algorithm 1.

Algorithm 1. LieTriCP algorithm.

**Step 0** (pre-registration): Input the estimated initial transformation $T^0$ for iteration $k = 1 : N_{iter}$ do

**Step 1**: Fix the transformation $T^k$, update the correspondence: For each $x_i \in X$, find $T^k \cdot x_i$’s closest point $z_i^k = \mu(x_i, T^k)$ in $Y$, and compute the squared distance $d_i^2 = |T^k \cdot x_i - z_i^k|^2$.

**Step 2**: Sort $d_i^2$ increasingly, find $m_r$ s.t. $r = m_r/m$ reaches the minimum of (2), then select the first $m_r$ point pairs, and compute the TSD $e^k = \sum_{i=1}^{m_r} d_i^2$. 
Proof. From the formulas (10) and (11), we have
\[ E^k \geq E^{k+1} \] (12)
\[ E^k \geq E^{k+1} \]

In addition, we also have
\[ \| T^{k+1} \cdot x_i - z_i \|^2 \geq \| T^{k+1} \cdot x_i - z_i \|^2, \]
since the correspondence has been updated in Step 1 of Algorithm 1 for next iteration. Thus
\[ E^k \geq E^{k+1}. \] (13)

From Eqs. (12) and (13), we have
\[ E^1 \geq \ldots \geq E^k \geq E^{k+1} \geq E^{k+1} \geq \ldots \geq 0. \]

The sequence of TSD values is non-increasing and bounded by the lower bound zero. Then the sequence converges to the local minimum. □

Note that we can only prove the local minimum from the above theorem. Therefore, the initial estimation is crucial for obtaining a global minimum. In our algorithm we estimate \( T^1 \) by using the method introduced in the beginning of this section.

4. Experimental results and analysis

In order to show the effectiveness of the LieTrICP algorithm and compare it with other algorithms, four algorithms, i.e., ICP, TrICP, LieICP and LieTrICP, under different parameters are implemented in Matlab 2012a using Pentium Intel(R) CPU T2080 @2.1 GHz with 3 GB RAM.

4.1. Results for 2-D

We perform several experiments on point sets derived from the images in MPEG-7 shape B database, and compare the LieTrICP algorithm with the ICP, TrICP and LieICP [5] algorithms, showing that our algorithm outperforms the others.

Fig. 1 shows 10 pictures used in our experiments, i.e., turtle, bell, bat, camel, butterfly, frog, hammer, bird, beetle and tree pictures, from this database.

The two point sets are preprocessed by the following procedures. (1) Select an image from the database, and use the extracted edge points by the Sobel edge detector to form a point set, then make two identical copies of the point set. (2) Scale anisotropically, rotate, and translate one copy. Therefore now we have two different sets, denoted by the source data \( X \) and the target data \( Y \). (3) Conduct some operations on either or both point sets. Here the operations refer to adding, removing and perturbing some points, or the combinations of the above adding, removing and perturbation procedures. The new point sets are still denoted by \( X \) and \( Y \). Remember that our goal is to find an optimal transformation from \( X \) to \( Y \).

Fig. 2 illustrates some of the results for the 10 examples mentioned above. The data are produced by randomly perturbing 20% points and adding 10% outliers to \( X \) and removing 20% points from \( Y \). Without loss of generality, in Fig. 2 we only show the results for turtle, bell and butterfly pictures. Red points indicate the point set \( X \), and blue points indicate the point set \( Y \). The first column illustrates the situations before registration, while the other four columns show the results for transforming red points to blue points under ICP, TrICP, LieICP and our LieTrICP. Visual inspection shows that our algorithm performs best. In order to provide a clearer view, as an example, Fig. 3 shows the zoom-in of the fourth (LieICP) and fifth (LieTrICP) columns in the first row (turtle) of Fig. 2.

The convergence rates are also compared for the four algorithms. The convergence is measured by the Root-Mean-TSD (RMTSD) between \( Y \) and its approximation \( T \cdot X \), when \( T \) is determined. The RMTSD error is defined as
\[ \sqrt{\frac{1}{m} \sum_{i=1}^{m} \| T \cdot x_i - \mu(x_i, T) \|^2}. \]

Again, without loss of generality, in Fig. 4 we only show for the results of turtle (Fig. 4a) and butterfly (Fig. 4b). In these two examples, the LieTrICP has the fastest decay of RMTSD, as shown in red.

4.2. Results for 3-D

In this part, our method is tested to demonstrate its robustness and efficiency for 3D data. We use the data of Stanford dragon and
bunny from Stanford 3D Scanning Repository (http://graphics.stanford.edu/data/3Dscanrep) and the data of hippocampus in brains from the Open Access Series of Imaging Studies (OASIS) (http://www.oasis-brains.org).

First, we make two copies of the data set. Then, random noise or missing points are also generated for one or both copies, and geometric transformation is applied to one point set. Our aim is to find the optimal transformation from one point set to the other. The results for the Stanford dragon and bunny data are shown in Figs. 5–7; results for the hippocampus data are shown in Fig. 8. The corresponding data for quantitative analysis are provided in Table 1.

Observing from Figs. 5, 7, 8 and Table 1, our algorithm can achieve satisfactory results, no matter what random anisotropic transformation is. This demonstrates that our algorithm is robust under anisotropic transformation. SVD, ICP and TriCP, on the other hand, cannot complete the anisotropic registration. LieICP has a similar result compared with our algorithm, but a zoom-in check (Fig. 6) shows that it is not as accurate as ours.

Fig. 5 illustrates the results for the Stanford dragon. The data sets X and Y have only 50% overlap area. Red indicates the data set X, and blue indicates the data set Y. Fig. 5a illustrates the situations before registration, Fig. 5b shows the result of SVD, which is used as a preprocessing registration, while others (Fig. 5c, d, e and f) show the results under ICP, TriCP, LieICP and our LieTriCP respectively. Visual inspection shows that our algorithm performs best, and the result of LieICP is the most approximated one to ours. In order to provide a clearer view, Fig. 6 shows the zoom-in of the result for LieICP and LieTriCP.

In Fig. 7, the above-mentioned methods are conducted for the Stanford bunny under anisotropic scale [0.9, 0.88, 0.82] for data set Y. The data sets X and Y have about 50% overlap area. The results show that our algorithm performs best. For more results, refer to Table 1.

Fig. 8 shows the results for the hippocampus under the anisotropic scale [1.2, 1.2, 1.1] on data set Y. About 10% points are removed from data set Y and some outliers are added. We find that our method obtains reasonable result, but SVD and TriCP algorithms fail.

4.3. Computation time

We compare the computation times for SVD, ICP and our LieTriCP, and the results are shown in Table 1. TriCP is not put into comparison, since its time for computing the overlap rate is...
Fig. 4. RMTSD graphs for each iteration of four algorithms. (a) Turtle; (b) butterfly.

Fig. 5. Registration results for Stanford dragon. (a) Before registration, red indicates X and blue indicates Y; (b) SVD; (c) ICP; (d) TrICP; (e) LieICP; (f) LieTrICP. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)
LieTrICP also adopts the trimmed strategy, but it makes use of a fast automatic overlap rate estimation method for general transformation problem, inspired by the techniques in [15,16]. As a result, the computation time of our algorithm in every iteration is comparable to that of ICP.

As the computation time is concerned, the speed of SVD mainly depends on the data size. Note that ICP can use SVD to obtain the optimal geometric transformation. Thus, ICP can be viewed as the alternating iterated SVD. Therefore, the computational time of ICP is related to the number of iterations and the data size.

Fig. 6. Zoom-in of registration results for Stanford dragon in Fig. 5. (a) LieICP; (b) LieTrICP.

Fig. 7. Registration results for Stanford bunny. (a) Before registration, red indicates X and blue indicates Y; (b) SVD; (c) ICP; (d) TrICP; (e) LieICP; (f) LieTrICP. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)
Our method, on the other hand, may have more iterations with the increment of the noise and the transformation complexity. As a result, the computation time will also increase. However, our objective function decays fast by using the trimmed strategy and the optimization of parameters, and the iteration number is often less than that of ICP. Thus our method has good performances in terms of accuracy, stability and speed.

Table 1 lists the comparison results for Stanford bunny and hippocampus under different transformations and noises. Bunny I is a sub-sample from the Stanford Bunny (Bunny II). In order to cancel out the effect of possible bias caused by individual result, the results given here are the average of 10 experiments. Note that the computation times listed here are the times after the initial registration after SVD. In other words, the real times for the ICP, TrICP and LieTrICP should also plus the computation time of SVD.

From the table, the computation times of SVD, ICP and LieTrICP are comparable, and the results are all good, under the circumstances of rigid transformations and without the presence of noise. However, when non-rigid transformation and noise appear, even very small, SVD will have high errors, which is the worst among all these algorithms. ICP can slightly reduce the error via alternatively iterating geometric transformations and correspondence, but it still fails under noise and complicated geometric transformations. TrICP and our method can work under the noise, but TrICP fails under the non-rigid transformations and the error increases rapidly.

With the increment of point number, the computation times of SVD, ICP and our method increase similarly, the errors of SVD and ICP also increase, especially in the presence of noise, while the error of our method almost remains the same.

Please cite this article as: J. Dong, et al., LieTrICP: An improvement of trimmed iterative closest point algorithm, Neurocomputing (2014), http://dx.doi.org/10.1016/j.neucom.2014.03.035
4.3.1. Statistical results

In order to make the comparison more statistically meaningful, we analyze the results from 10 cases in Section 4.1 as a whole to avoid possible bias from some individual cases.

Figs. 9, 10, and 11 use boxplots to illustrate the RMTSD for all 10 pairs of point sets \((X, Y)\) derived from images in Fig. 1, representing the cases of missing points in \(Y\), perturbations in \(X\), and outliers in \(X\) respectively.

The boxplot graph can be interpreted as follows. The x-axis indicates four algorithms; the y-axis indicates their respective RMTSD. The central red mark in the box indicates the median of the 10 RMTSD errors; the two horizontal edges of the box indicate the 25th and 75th percentiles; the two short bars indicate the most extreme data points which are not considered as outliers, i.e., the values larger than the upper short bar or smaller than the lower bar will be considered as outliers. For more details about boxplot, refer to the help file of boxplot for Matlab. Boxplot shows not only the results but also the stability of the results. Obviously, the best algorithm in all experiments should have the lowest median and variance. Note that the height of the box is an indicator of the variance.

Fig. 9 shows three results in Fig. 9a–c, for the case of missing points in \(Y\) under different parameters. In what follows we will analyze, as shown in the figures, that the performance of our LieTrICP is the best in each case, especially when the number of outliers and the anisotropic extent increase.

Fig. 9a is the result for 80% overlap rate and 10° rotation. As expected, TrICP performs better than ICP and LieICP, since TrICP ticks out outliers. Our LieTrICP has a similar result compared with TrICP.

Fig. 9b is the result using the parameters of Fig. 9a and anisotropic scale transformation. The anisotropic scale transformation is \([0.9, 1]\), which means the point \(x = (x_1, x_2)^T\) is scaled into \(x = (0.9x_1, x_2)^T\). Due to this scale transformation, ICP and TrICP become less effective; LieICP performs better than ICP, as roughly speaking LieICP is the anisotropic scale version of ICP under Lie group framework. Our LieTrICP outperforms LieICP, as LieTrICP ticks out outliers while LieICP not.

Fig. 9c furthers the experiment by decreasing the overlap rate to 60% and increasing the anisotropic extent to \([0.8, 1]\). The behaviors of our counterparts all become worse; LieTrICP still

Fig. 9. Boxplot for the results of four algorithms, in the case of missing points in \(Y\). (a) Overlap 80%, rotation 10°; (b) overlap 80%, rotation 10°, scale \([0.9,1]\); (c) overlap 60%, rotation 10°, scale \([0.8,1]\).

Fig. 10. Boxplot for the results of four algorithms, in the case of perturbations in \(X\). (a) Overlap 80%, rotation 10°; (b) overlap 80%, rotation 10°, scale \([0.9,1]\); (c) overlap 80%, rotation 10°, scale \([0.8,1]\).

Fig. 11. Boxplot for the results of four algorithms, in the case of outliers in \(X\). (a) Overlap 80%, rotation 10°; (b) overlap 80%, rotation 10°, scale \([0.9,1]\); (c) overlap 60%, rotation 10°, scale \([0.9,1]\).
remains stable, because LieTrICP ticks out outliers and conducts anisotropic transformation while none of others performs both operations.

Figs. 10 and 11 show the results for the cases of perturbation in $X$ (Fig. 10) and outliers in $X$ (Fig. 11). All parameters are listed in the figures. Due to similar reasons analyzed in the case of Fig. 9, LieTrICP outperforms others.

5. Conclusions

We have proposed a robust algorithm called LieTrICP for registering two low overlapped point sets using Lie group parameterization. This algorithm improves the TrICP algorithm by representing the transformations in the registration problem as elements in a Lie group, therefore finding the optimal transformation is reduced to finding a suitable element in the group. Moreover, this algorithm generalizes the TrICP algorithm from the rigid transformations to transformations with anisotropic scaling.

Experimental results demonstrated the effectiveness and robustness of the LieTrICP algorithm, compared with the ICP, TrICP and LieICP algorithms.

This framework is not specific to TrICP. As far as certain transformations form a Lie group, this method is applicable. Thus, it can be easily extended to more complex geometric transformations and high dimension cases.

Acknowledgments

This work is supported by the 973 Programme (No. 2011CB707104), the National Science Foundation of China (Nos. 61273298, 11101260 and 61005002), the First-class Discipline of Universities in Shanghai, and the Discipline Project at the corresponding level of Shanghai (A13-0101-12-005).

References


Jianmin Dong is a master Candidate of Computational Mathematics, School of Science, Shanghai University. Her current research interest is image processing.

Yaxin Peng received her B.Eng. degree in mechanical engineering and Ph.D. degree in applied mathematics from Xijing University, China, in July 2001 and April 2008, respectively. He is currently with the Department of Mathematics, School of Science, Shanghai University, China. His research interests include differential geometry, remote sensing, and machine learning.

Shihui Ying received her B.Eng. degree in mechanical engineering and Ph.D. degree in applied mathematics from Jiaotong University, China, in 2007, and M.A. (1995) from Fisk University respectively. He, National Chair Professor of China, is the founding director of the Institute of NanoMicroEnergy of Shanghai University and is also holding a Zhi Yuan Chair professorship at Shanghai Jiao Tong University and serving as the director of National Key Laboratory of Science and Technology on Micro-/Nano-Fabrication. Dr. Hu is an adjunct professor at University of Tennessee, Knoxville, Tennessee, USA plus other honorary positions in several universities and institutes.