16-811 Course Project

Ruoyang Xu[*], Ruohai Ge[†], Zhihao Zhang[‡]

December 2020

1 Setup

Suppose we have two point cloud data sample contains N point before and after a rigid body transformation for k different object $O_1, O_2, \cdots, O_k$. The rigid body transformations for rotation and translation are $R_1, R_2, \cdots, R_k$ and $t_1, t_2, \cdots, t_k$ respectively. The data points before transformation could be represented as $x_1, x_2, \cdots, x_N$ and $x'_1, x'_2, \cdots, x'_N$ for after transformation respectively. The key assumption here is we have no point to point correspondence for the two set of data samples. Given only the random permuted data points, the problem is to figure out how to recover the point to point correspondence and R_i, t_i for each group of points. Suppose each object O_i contains N_i points, we have $\sum_{i=1}^{k} N_i = N$.

2 Methodology

The basic intuition for our algorithm is the fact that if two points belong to the same object, their relative euclidean distance will not change after the rigid body transformation. For the first two algorithm Direct Feature Matching (DFM) and Reference point Feature Matching (RFM), we keep this distance invariant property. However, when we move on to add scale transformation, this property will not hold anymore. On the other hand we will have the angle invariant property, which says that if three points belong to the same object, then after the transformation their relative angle will not change. Based on different assumption we need to derive different algorithms.

2.1 Direct Feature Matching (DFM)

For this algorithm, we first need to construct a feature descriptor for each data point. The feature descriptor is represented as a vector consisted of the euclidean distance with respect to all other point $f_i \in \mathbb{R}^N, i = 1, 2, \cdots, N$,

[*]ruoyangx
[†]ruohaig
[‡]zhihaoz3

$f_i = (d_{i1}, d_{i2}, \cdots, d_{iN})$ where $d_{ij} = \|x_i - x_j\|_2$. Then we need to define the metric to figure out the best point to point correspondence based on our feature descriptor. Since the elements in the distance vector is permuted after the transformation, we could only regard the vector as a set and define the metric for two points as the negative cardinality of the intersection of two points' feature vector set, $I(x_i, x'_j) = -|\text{set}(f_i) \cap \text{set}(f'_j)|$. Based on this metric, we could have the correspondence for any point x_i as $x' = \arg\min_{x'_j} I(x_i, x'_j)$. We construct the descriptor for all N points before and after the transformation, then match each pair of point based on our metric to get the correspondence followed by point registration to figure out the transformation matrix.

Algorithm 1 DFM

procedure EUCLID(X, X') ▷ construct feature descriptors for point clouds X, X'
  for $x_i \in X, x'_i \in X'$ **do**
    $f_i = (d_{i1}, d_{i2}, \cdots, d_{iN})$
    $f'_i = (d'_{i1}, d'_{i2}, \cdots, d'_{iN})$
procedure CORRES(X, X') ▷ get point correspondence based on feature descriptor
  for $x_i \in X$ **do**
    correspondence$(x_i) = \arg\min_{x'_j} I(x_i, x'_j)$
procedure REGISTRATION▷ get the transformation matrix based on point correspondence

2.2 Reference point Feature Matching (RFM)

We derive a updated version algorithm based on DFM, as we find out as long as we have 4 points' correspondence belong to one object, we could find all other points belong to the same object as well. This is due to the point belong to the same object if and only if their relative distance with respect to these 4 reference points haven't changed. It is necessary to choose 4 points here, as for 3 points there will exist a symmetrical ambiguity, however by using 4 points we could break this symmetry.

The feature descriptor is the same as before. But instead of matching for all N data point, we now randomly choose 4 points and perform a test to verify whether this 4 points belong to the same object.

We test this by first finding the corresponding points for the randomly sampled 4 points then verify if the distance between this 4 points remain the same before and after the transformation.

We then use this 4 points as reference points to find all other points belong

to the same object. We should notice that RFM algorithm could jointly solve the correspondence problem as well as the point classification problem.

Algorithm 2 RFM

procedure EUCLID(X, X') ▷ construct feature descriptors for point clouds X, X'
  for $x_i \in X, x'_i \in X'$ **do**
    $f_i = (d_{i1}, d_{i2}, \cdots, d_{iN})$
    $f'_i = (d'_{i1}, d'_{i2}, \cdots, d'_{iN})$
procedure RANSAC_CORRES(X, X') ▷ get point correspondence based on feature descriptor
  while exist points left unclassified **do**
    repeat
      $x_i, x_j, x_k, x_p \leftarrow$ Sample(X)
      find the correspondence of x_i, x_j, x_k, x_p
    until x_i, x_j, x_k, x_p belong to the same object
    for $x \in X, x' \in X'$ **do**
      if $d(x, x_i) = d(x', \text{Corres}(x_i))$ and $d(x, x_j) = d(x', \text{Corres}(x_j))$ and $d(x, x_k) = d(x', \text{Corres}(x_k))$ and $d(x, x_p) = d(x', \text{Corres}(x_p))$ **then**
        Corres(x) $= x'$ and they belong to the same object of the reference points.
procedure REGISTRATION▷ get the transformation matrix based on point correspondence

2.3 Scale Invariant

For scale invariance, the added assumption is that besides for rigid body transformation, we allow for scale transformation as well.

Similar to DFM but with a subtle difference, the property of distance invariance will not be true any more, instead of which we could use the property that any three points belong to one object will have the same relative angle before and after the transformation.

Therefore we could modify the feature descriptor for a point from the distance vector to an angle matrix whose element is the angle of every two points whose vertex is the selected point. Thus the feature size is N^2 for one point. So in order to satisfy scale invariance, we need to sacrifice algorithm running time from $O(N^3)$ to $O(N^4)$ for DFM. The outline of the Scale-Invariant DFM (SI-DFM) algorithm is listed below, angle($x_p, x_q | x_i$) represent the angle formed by line $\overrightarrow{x_p} - \overrightarrow{x_i}$ and $\overrightarrow{x_q} - \overrightarrow{x_i}$

Algorithm 3 SI-DFM

procedure EUCLID(X, X') ▷ construct feature descriptors for point clouds X, X'
  for $x_i \in X, x'_i \in X'$ **do**
    $f_i = \{\text{angle}(x_p, x_q | x_i) | x_p, x_q \in X\}$
    $f'_i = \{\text{angle}(x'_p, x'_q | x'_i) | x'_p, x'_q \in X'\}$
procedure CORRES(X, X') ▷ get point correspondence based on feature descriptor
  for $x_i \in X$ **do**
    correspondence$(x_i) = \arg\min_{x'_j} I(x_i, x'_j)$
procedure REGISTRATION▷ get the transformation matrix based on point correspondence

2.4 Registration

The registration is solved through reformulating the problem as a directly linear transform and then singular value decomposition. To robustify the algorithm against potential mis-matches in point correspondence, we optimize the solution through truncated least square method with a RANSAC solution as an initial guess.

2.4.1 Direct Linear Transform

Direct linear transform is an algorithm that solves variables from a set of similar relations, most commonly in the following form [3]:

$$\mathbf{x}_k \propto \mathbf{A}\mathbf{y}_k, k \in [1, N]$$

Define points in homogeneous representation $\mathbf{x}$ and correspondence $\hat{\mathbf{x}}$, and the transformation $\mathbf{A}$ a matrix representation of affine transformation, would be represented as

$$\mathbf{x} = \mathbf{A}\hat{\mathbf{x}} \tag{1}$$

To rewrite matrix A as

$$A = \begin{bmatrix} \mathbf{p_1} & \mathbf{p_2} & \mathbf{p_3} & \mathbf{p_4} \end{bmatrix}^T$$

with $\mathbf{x} = [x, y, z, 1]^T$ to account for scale factor:

$$x\mathbf{p_4}^T\hat{x} = \mathbf{p_1}^T\hat{x}$$
$$y\mathbf{p_4}^T\hat{x} = \mathbf{p_2}^T\hat{x}$$
$$z\mathbf{p_4}^T\hat{x} = \mathbf{p_3}^T\hat{x}$$

To solve $\mathbf{A}$ in the form of $h = [\mathbf{p_1}; \mathbf{p_2}; \mathbf{p_3}; \mathbf{p_4}]$ we transform equation 1, into the following

$$\mathbf{M} = \begin{bmatrix} -\hat{\mathbf{x}}^T & \mathbf{0} & \mathbf{0} & x\hat{\mathbf{x}}^T \\ \mathbf{0} & -\hat{\mathbf{x}}^T & \mathbf{0} & y\hat{\mathbf{x}}^T \\ \mathbf{0} & \mathbf{0} & -\hat{\mathbf{x}}^T & z\hat{\mathbf{x}}^T \end{bmatrix}, \mathbf{M}h = 0$$

We then perform singular value decomposition on **M** and the eigenvector with the smallest eigenvalue would be the solution to the least sqaure problem.

2.4.2 Optimization

The above solution would have two problems:

1. sensitive to outliers and noises
2. maynot exactly adhere to $SE(3)$ constraints

To address the issue we perform truncated least square on the solution. We perform RANSAC to obtain a good initial guess for the optimization process. The truncated least square aims for the following goal[2].

$$_{A \in SE(3)} \left(\Sigma_{x,\hat{x}} \min ||Ax - \hat{x}, c||\right)$$

In here we change the formulation of A to be a vector $u \in \mathbb{R}^6$ consisting of translation $t \in \mathbb{R}^3$ and rotation axis vector representation $v \in \mathbb{R}^3$. v can be transformed to and back from rotation matrix via Rodrigues' formula [1] and it is necessary to do so to ensure $SO(3)$ constraints.

3 Running time analysis

For running time analysis, we calculate the asymptotic estimation of the computational complexity for both DFM and RFM algorithm. For fair comparison, we will add RANSAC for DFM to figure out the point registration as well.

RFM algorithm mainly consists of three part: 1.reference points chosen 2. reference points matching 3. point classification. For clean data, we could calculate the expected iteration bound for step 1: choosing 4 consecutive points in one group as[1]:

$$O(\frac{1}{\ln(1 + \frac{1}{k^3 - 1})}) \tag{2}$$

For reference point matching the running time is $O(N^2)$ and for reference point verification the running time is $O(1)$. For point classification the running time is $O(N^2)$. Suppose we have k object, then we will need approximately k iterations. Therefore the worst case algorithm running time is

$$O(k)(O(\frac{1}{\ln(1 + \frac{1}{k^3 - 1})}) \cdot (O(N^2) + O(1)) + O(N^2)) \tag{3}$$

which is approximately $O(k^4 \cdot N^2)$ as $\ln(1 + a) \approx a, a \to 0$.

[1] derivation is included in the appendix section

For DFM algorithm, feature construction takes $O(N^2)$, feature matching takes $O(N^3)$ and RANSAC takes $O(k \cdot \frac{1}{\ln(1+\frac{1}{k^3-1})}) \approx O(k^4)$. From the running time analysis we could notice that if the object number is a constant which is usually true, then the running time of RFM is $O(N^2)$ which is more efficient than DFM who needs $O(N^3)$. On the other hand, if the object number is proportional to data points number $k = O(N)$, then DFM will take $O(N^4)$ and RFM will take $O(N^6)$, thus DFM will be more efficient if N goes to infinity. From this analysis we could easily see the pros and cons for both DFM and RFM algorithm.

4 Experiment

4.1 Experiment Setup

Our dataset comes from ModelNet 40 class subset, which is created by Princeton and a dataset used for 3D objects deep learning. The dataset contains 40 different categories including airplane, cup, sofa and ec. Each category contains different number of point clouds. Each point cloud (no matter under what category) has 2048 point.
We choose a random point clouds combination to form each test case. We then randomly choose two translation and rotation matrices, which are then used to create the original point cloud and the transformed point cloud. The point's x,y,z coordinates are saved in the .npz file for later analysis. The following pictures shows some example of the test case.

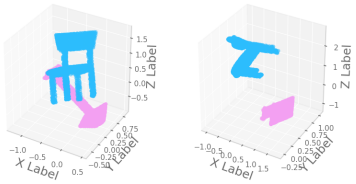

Figure 1: **2 objects:** example: On the left side is the original point cloud and on the right side is the transformed point cloud

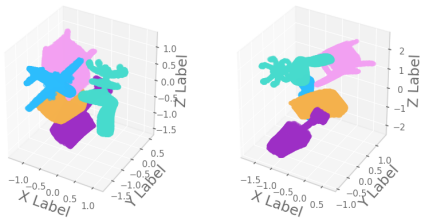

Figure 2: **5 objects:** example: On the left side is the original point cloud and on the right side is the transformed point cloud

4.2 Experiment with different number of objects

The original point cloud has 2048 points, which takes too long to run a comprehensive comparison using a regular laptop. We then enforce the total point number to be 400 by down sampling the original PCD. (for example, for 5 objects, each object will have around 80 points). **The experiment used RFM for clustering**.

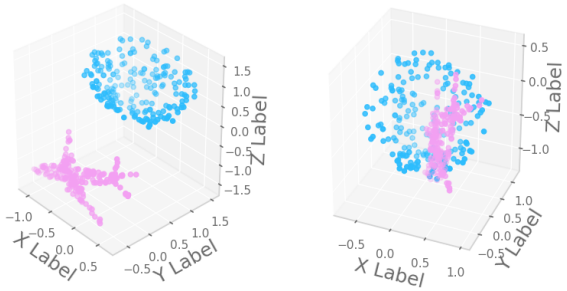

Figure 3: The effect after downsampling the original point cloud

We did experiments with 2 objects, 3 objects, 4 objects and 5 objects. Each experiment has been tested with 100 times. The following pictures will show their separate results and average results in comparison. The results contain runtime analysis and accuracy analysis (under L1 norm standard)

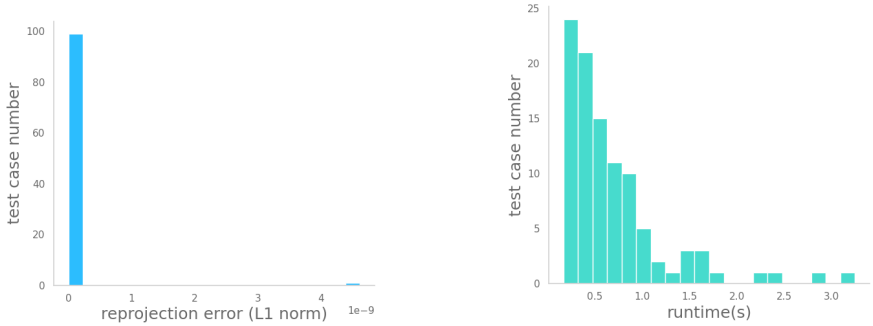

Figure 4: **2 objects:** Left side is accuracy analysis and right side is runtime analysis

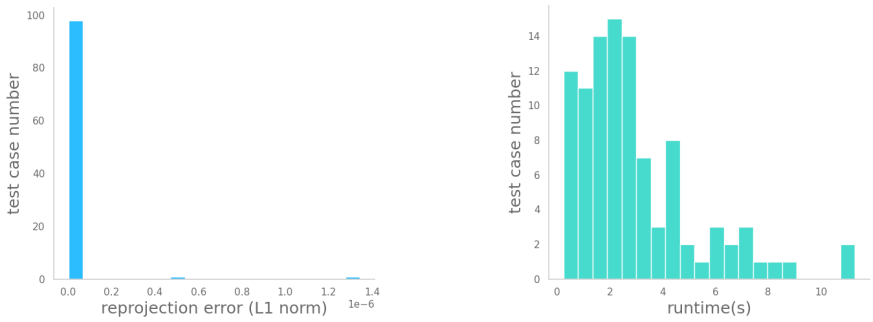

Figure 5: **3 objects:** Left side is accuracy analysis and right side is runtime analysis

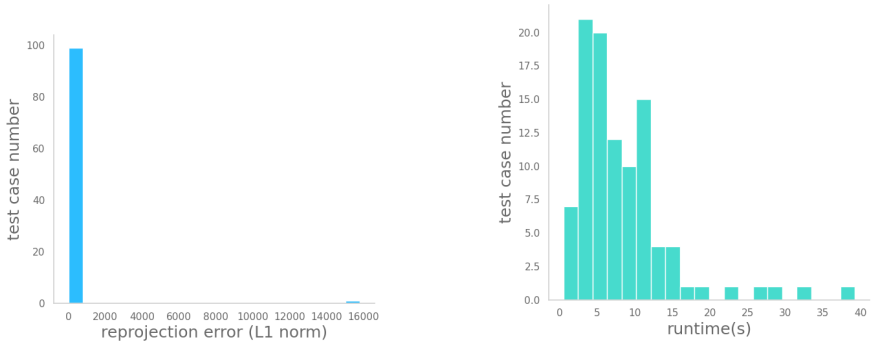

Figure 6: **4 objects:** Left side is accuracy analysis and right side is runtime analysis

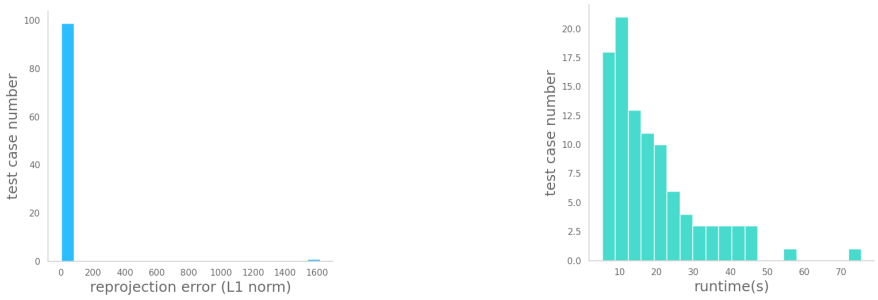

Figure 7: **5 objects:** Left side is accuracy analysis and right side is runtime analysis

We also did a test with original point cloud data (which is 2048) without downsampling. Due to the time it takes to run and our computer's capacity, we only did a two objects experiment.

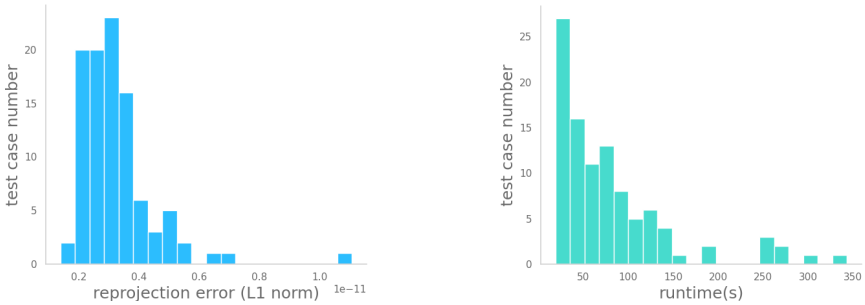

Figure 8: **2 objects:** Left side is accuracy analysis and right side is runtime analysis

The average result excludes those outliers to make the comparison more meaningful. One can view the number of outliers in the separate graph above.

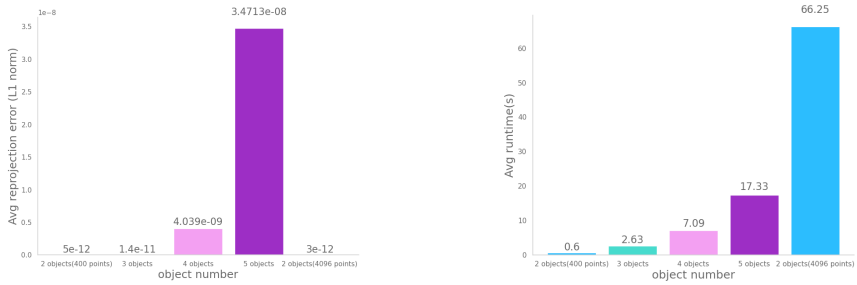

Figure 9: Left side is average accuracy analysis and right side is average runtime analysis

The result below looks pretty solid. Our error is nearly 0 under L1 norm standard and the runtime performs similar to our mathematical runtime analysis above. The runtime is related to object number K and object's point number N.

4.3 Experiment with different levels of noise

Now, Let's take a look at the robustness of our algorithm. The setup is similar to the clean dataset.We only did it for 2 objects for a preliminary result. $\frac{1}{8}$ of the points in both original PCD and transformed PCD has been added with different levels of Gaussian noise. The total point is still restricted to 400 points. **The experiment used RFM for clustering**.

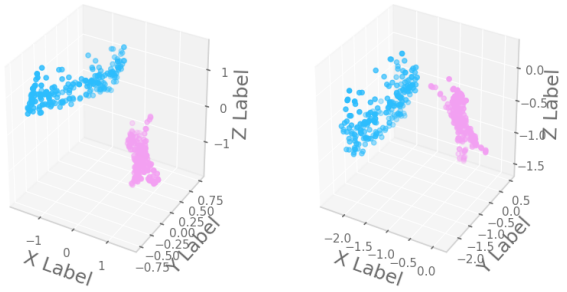

Figure 10: **noise level 0.01:** On the left side is the original point cloud and on the right side is the transformed point cloud

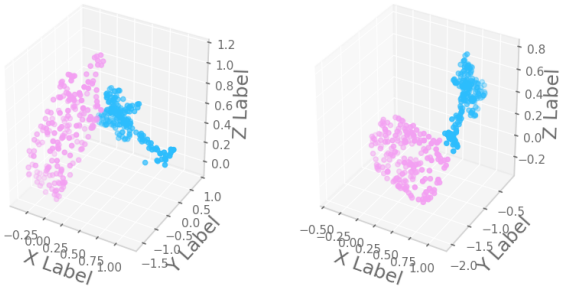

Figure 11: **noise level 0.05:** On the left side is the original point cloud and on the right side is the transformed point cloud

The noise level is from 0.01 to 0.05. Each noise level has been tested with 100 test groups. The RFM algorithm has been modified a little bit. The outer loop for finding cluster and the inner loop to find four suitable points have both been added with another stop condition when the iteration reaches the pre-set max_iteration limit. The outer loop's max_iteration has been set to 3 and the inner loop's max_iteration has been set to 100. This has been tested to reach a balance between accuracy and runtime.
The following pictures will show their separate results and average result in comparison. The result contains runtime analysis and accuracy analysis (under L1 norm standard)

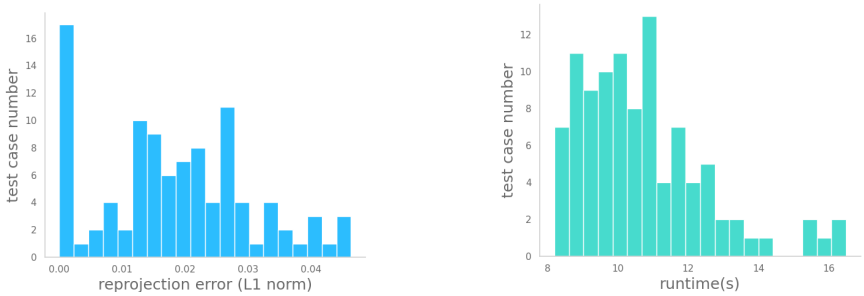

Figure 12: **noise level 0.01:** Left side is accuracy analysis and right side is runtime analysis

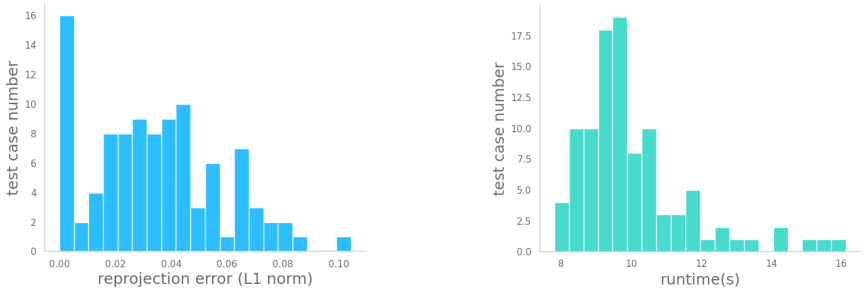

Figure 13: **noise level 0.02:** Left side is accuracy analysis and right side is runtime analysis

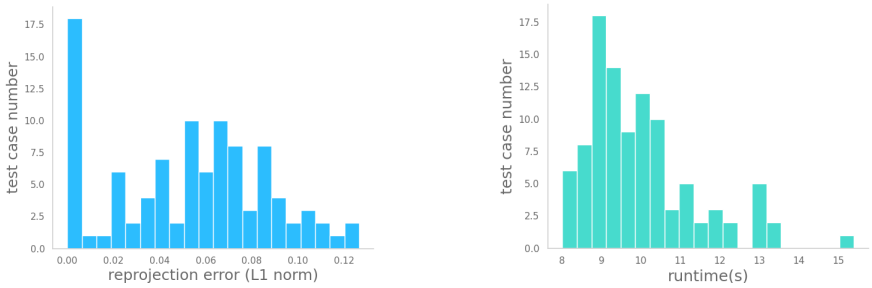

Figure 14: **noise level 0.03:** Left side is accuracy analysis and right side is runtime analysis

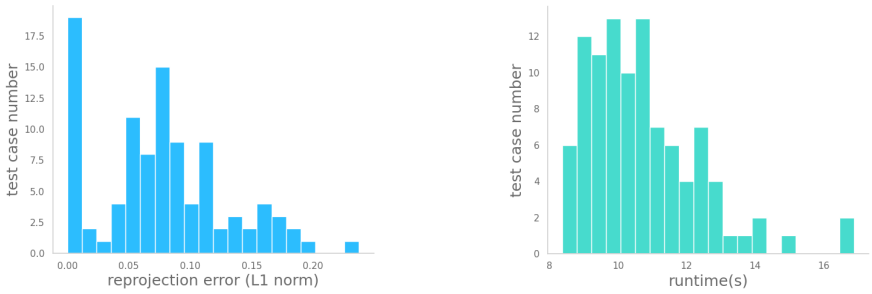

Figure 15: **noise level 0.04:** Left side is accuracy analysis and right side is runtime analysis

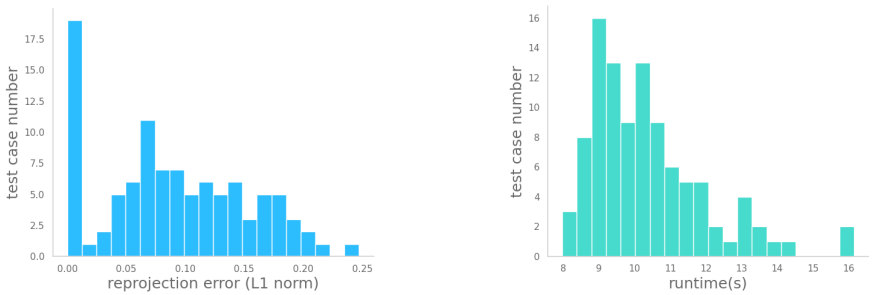

Figure 16: **noise level 0.05:** Left side is accuracy analysis and right side is runtime analysis

The average result excludes those outliers to make the comparison more meaningful. One can view the number of outliers in the separate graph above.

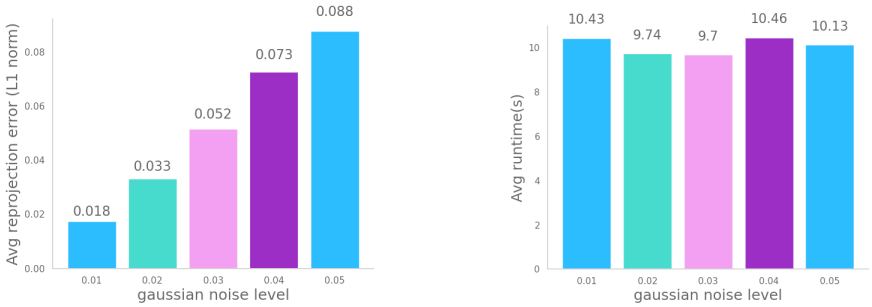

Figure 17: Left side is average accuracy analysis and right side is average runtime analysis

The current result shows a relatively good error, which is below 0.1 even with 0.05 noise level. The runtime is also stable with different noise levels, which proves our runtime analysis to be correct. Since it is only related to object's number and object's point number.

5 Appendix

5.1 Equation (1)

The probability for choosing 4 consecutive points belong to 1 group in M iteration is:

$$1-(1-\sum_{i=1}^{k}(\frac{N_i}{N})^4)^M \quad \geq \quad 1-(1-\frac{1}{k^3})^M \tag{4}$$

This inequality is derived by using Hölder's inequality on $\sum_{i=1}^{k}(\frac{N_i}{N})^4$, then we have $\sum_{i=1}^{k}(\frac{N_i}{N})^4 \geq \frac{1}{k^3}$. After acquiring the lower bound, we want to calculate how large M needed to be to get a $\frac{1}{2}$ lower bound. By setting $M \geq \frac{\ln 2}{\ln(1+\frac{1}{k^3-1})}$, we have $1-(1-\frac{1}{k^3})^M \geq \frac{1}{2}$, therefore the expected iteration to get 4 consecutive points belong to one object is $O(\frac{1}{\ln(1+\frac{1}{k^3-1})})$

5.2 Github Repo

https://github.com/RuohaiGe/3D-Registration-of-Mixed-2-objects

References

[1] Leonhard Euler. *Problema algebraicum ob affectiones prorsus singulares memorabile*. Berlin: Novi Comm. Acad. Sci. Petropolitanae, 1770, pp. 75–106.

[2] R. D. Fierro et al. "Regularization by Truncated Total Least Squares". In: *SIAM Journal on Scientific Computing* 18.4 (1997), pp. 1223–1241. DOI: 10.1137/S1064827594263837. eprint: https://doi.org/10.1137/S1064827594263837. URL: https://doi.org/10.1137/S1064827594263837.

[3] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, 2003.