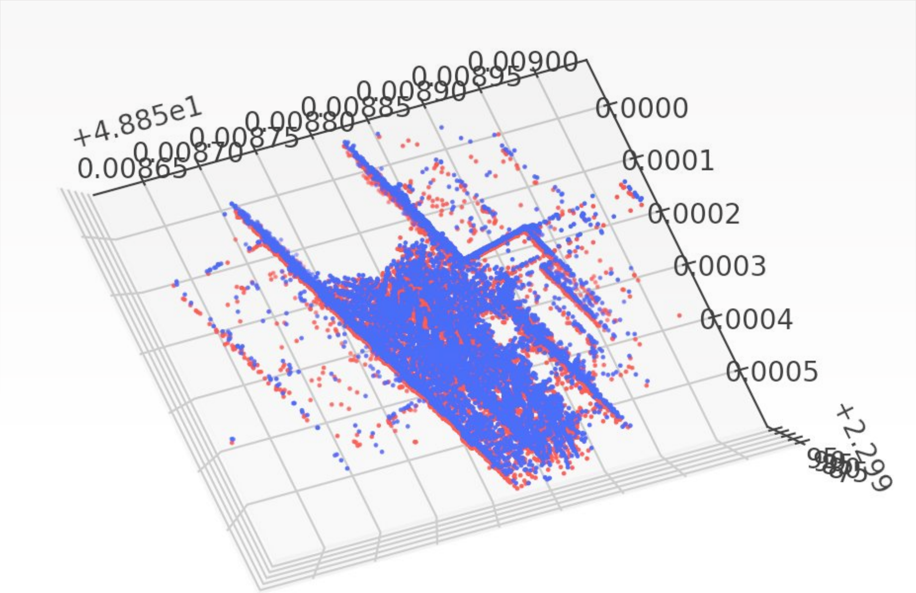
Point Clouds Registration

Solve the point-set matching problem using ICP algorithm.

Northwestern | McCORMICK SCHOOL OF ENGINEERING

The Problem Declaration

- Input: two point clouds X, Y
- Target: find a map function
- Challenge: no extra info.



Blue:Point Cloud X
Red:Point Cloud Y

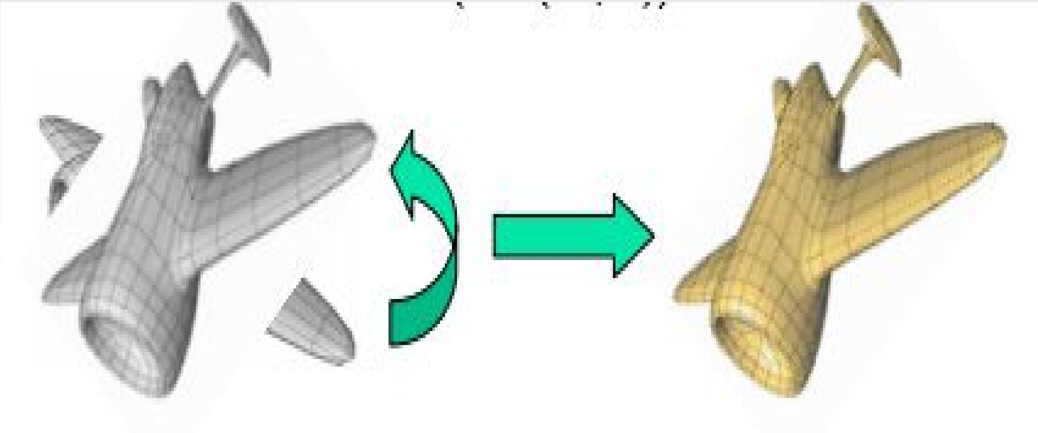
Iterative Closest Point (ICP) Algorithm

Solution: ICP

Result: find map Y = map(X)

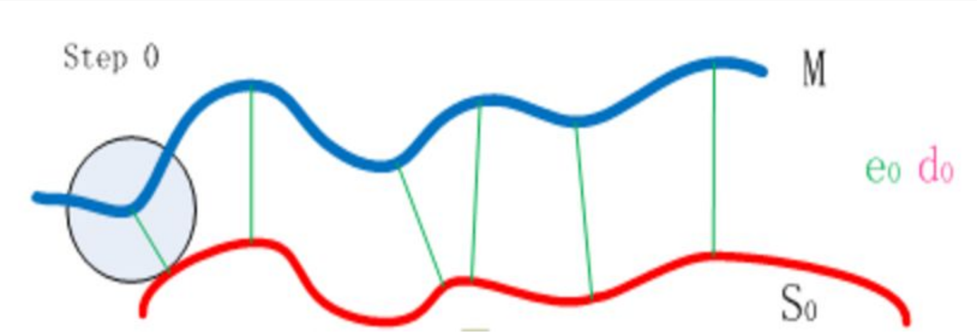
Output: a space transformation map matrix

Feature: two major action



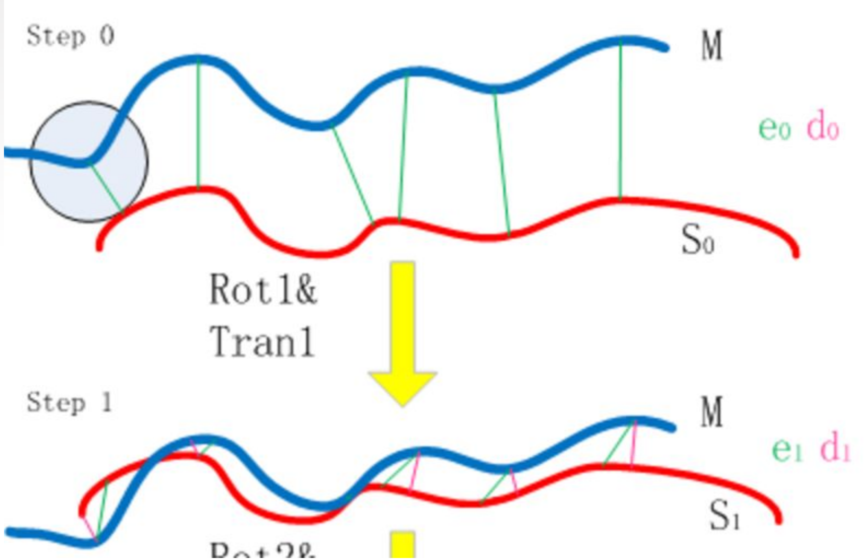
Iterative Closest Point (ICP) Algorithm

1. Find the closest match points in two point sets using inital R and T.



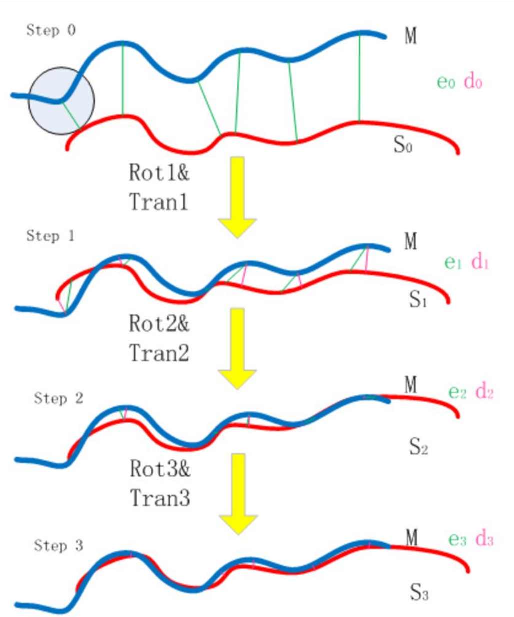
Iterative Closest Point (ICP) Algorithm

2.Update R and T to minimize error



Iterative Closest Point (ICP) Algorithm

3. Repeat previous steps untill R and T are converged.



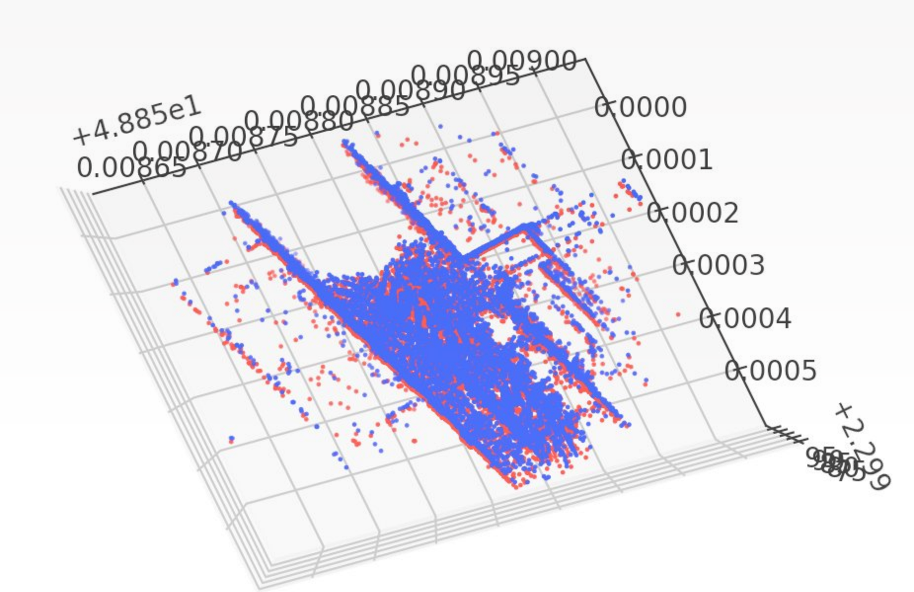
Implementation Details

Our method contains following steps:

1. Find the closest point
2. Calculate the alignment
3. Apply the alignment
4. Iterate to reduce the error
5. Improvement
6. Result and analysis

1. Find the closest points

Using KD-Tree

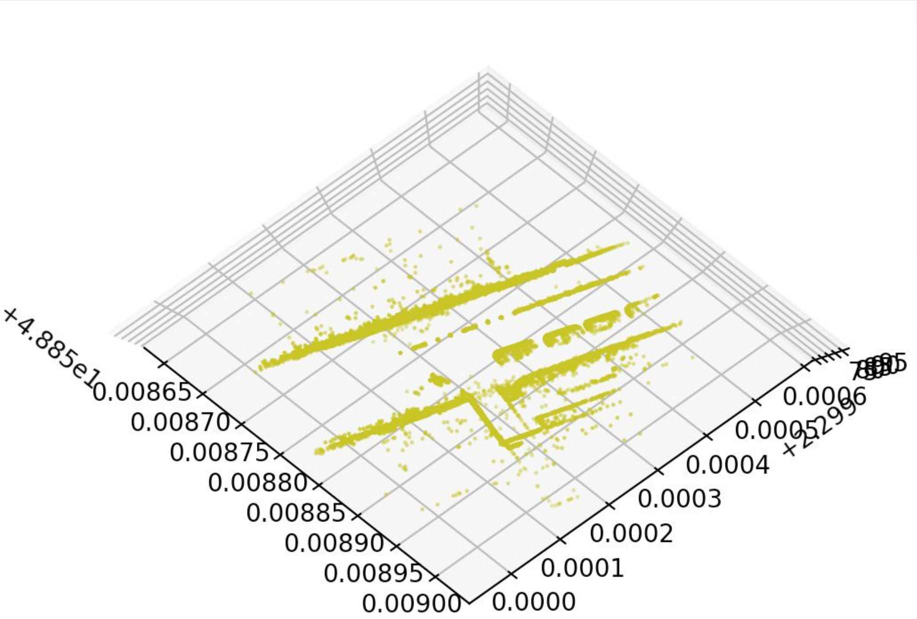


Blue:point Cloud A
Red:Point Cloud B

1. Find the closest points

4. Then we get a new point set Y. ($Y = \{y_i\}$ for i=1...N_a)

And Y is an injection from set A to set B.



Point Cloud Y

2. Calculate the alignment

From point set A to point set Y, we need to compute the registration.

1. Compute the “center of point” μ(Y) of the measured point set Y

$$\vec{u}_Y = \frac{1}{N_Y}\sum_{i=1}^{N_Y}\vec{y}_i$$

2. Compute the “center of mass” μ(A) for the point set A

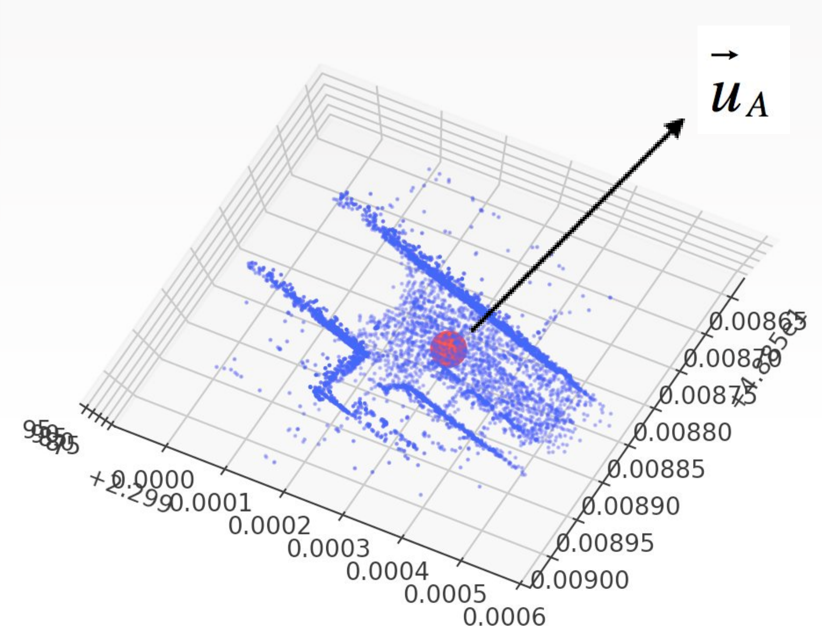
$$\vec{u}_A = \frac{1}{N_A}\sum_{i=1}^{N_A}\vec{a}_i$$

3. Compute cross-corvariance matrix of set Y and A is given by :

$$\Sigma_{AY} = \frac{1}{N_Y}\sum_{i=1}^{N_Y}\vec{y}_i[(\vec{y}_i - \vec{u}_Y)(\vec{a}_i - \vec{u}_A)^t]$$

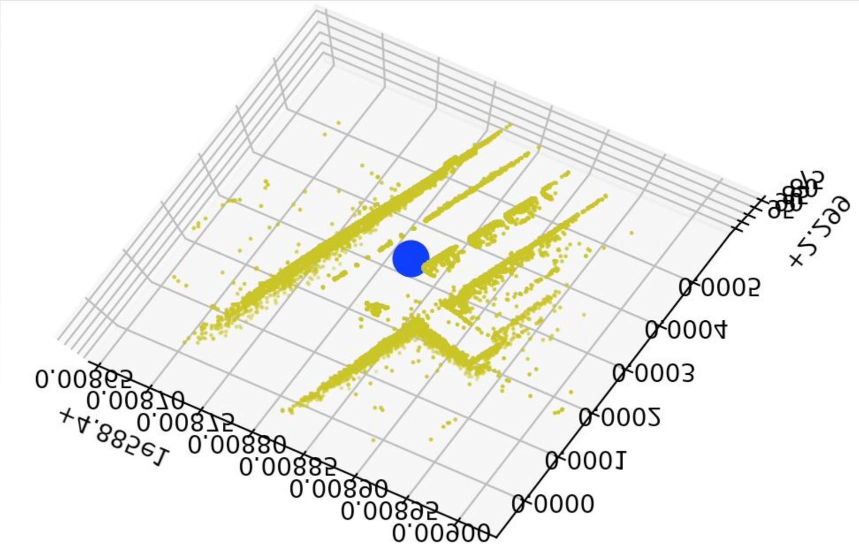
2. Calculate the alignment

The red dot is the center of mess for point set A



2. Calculate the alignment

The blue dot is the center of mess for point set Y



2. Calculate the alignment

4. The cyclic components of the anti-symmetric matrix :

$$A_{ij} = (\Sigma_{AY} - \Sigma_{AY}^{T})_{ij}$$

5. We use A_{ij} to form the column vector :

$$\Delta = \begin{bmatrix} A_{23} \\ A_{31} \\ A_{12} \end{bmatrix}$$

2. Calculate the alignment

6. Then we get a matrix : $Q(\Sigma_{AY})$

$$Q(\Sigma_{AY}) = \begin{bmatrix} tr(\Sigma_{AY}) & \triangle^T \\ \triangle & \Sigma_{AY} + \Sigma_{AY}^T - tr(\Sigma_{AY})I_3 \end{bmatrix}$$

7. The unit eigenvector of $Q(\Sigma_{AY})$ is selectd as optimal rotation vector :

$$\vec{q}_{Rotate} = \begin{bmatrix} q_0 & q_0 & q_2 & q_3 \end{bmatrix}^T$$

8.The optimal translation vector :

$$\vec{q}_{Translate} = \vec{u}_A - M(\vec{q}_{Rotate})\vec{u}_Y$$

2. Calculate the alignment

The transformation matrix we get :

```
Rotation vector:
[[ -1.00000000e+00   6.65738695e-16  -2.14054359e-08]
 [  6.65738695e-16   1.00000000e+00  -4.51438034e-18]
 [  2.14054359e-08  -4.51439459e-18  -1.00000000e+00]]
Translation vector
[[  9.77176626e+01]
 [ -1.49207093e-06]
 [  1.58641718e+02]]
```

3. Apply the alignment

1. Before applying the alignment, the mean square error(MSE) between point set A and B is :

$$e_0 = \frac{1}{N_A} \sum_{i=1}^{N_A} \left\| a_{i0} - b_{i0} \right\|$$

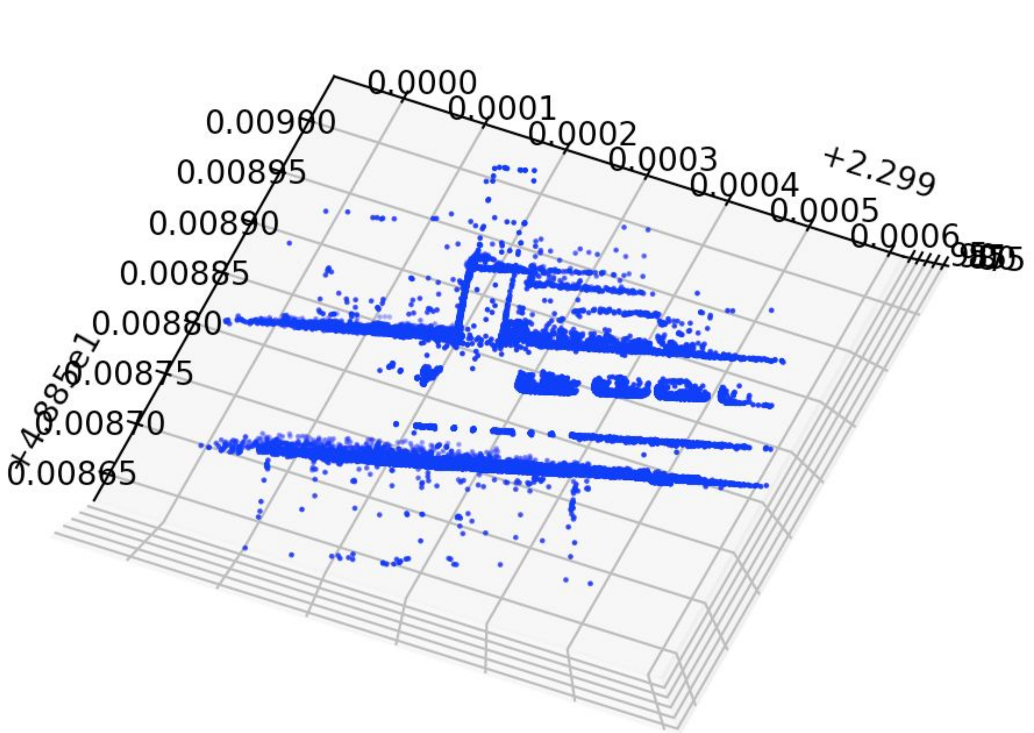
2. We transform the point set A using the obtained rotation and translation vectors into a new point set A_1.

3. After transformation, the MSE between point set A_1 and B :

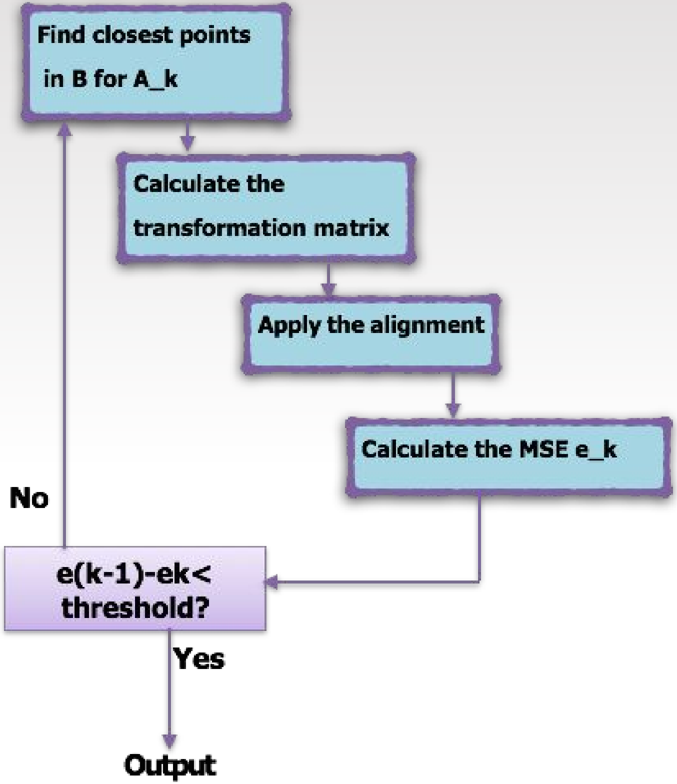
$$e_1 = \frac{1}{N_A} \sum_{i=1}^{N_A} \left\| a_{i1} - b_{i0} \right\| < e_0$$

3. Apply the alignment

Point set A_1 after alignment.



4. Iterate to reduce the error



4. Iterate to reduce the error

```
Iterate2:
  Rotation vector:
   [[ -9.99999439e-01  -9.65303545e-08   1.05884293e-03]
    [ -9.65302910e-08   1.00000000e+00   1.11098526e-10]
    [ -1.05884293e-03   8.88804732e-12  -9.99999439e-01]]
  Translation vector
   [[  9.76336609e+01]
    [ -9.10841343e-06]
    [  1.58881693e+02]]
```

....

After 6 iterations, we get a more accurate result:

```
Rotation Matrix `R` =
[[ -9.99989094e-01  -1.89318937e-06   4.67034984e-03]
 [ -1.89317817e-06   1.00000000e+00   6.81923714e-09]
 [ -4.67034984e-03  -2.02264160e-09  -9.99989094e-01]]
Translation Vector `T` =
[[  9.73466480e+01]
 [ -4.26934882e-05]
 [  1.59704227e+02]]
```

Thanks for your attention!