Programming Assignments 3 and 4 – 601.455/655 Fall 2021

Score Sheet (hand in with report) Also, PLEASE INDICATE WHETHER YOU ARE IN 601.455 or 601.655

(one in each section is OK)

Name 1	Shuran Zhang
Email	srzhang@jhu.edu
Other contact information (optional)	
Name 2	Qiyuan Wu
Email	qwu37@jhu.edu
Other contact information (optional)	
Signature (required)	I (we) have followed the rules in completing this assignment Shuran Zhang Qiyuan Wu

Grade Factor		
Program (40)		
Design and overall program structure	20	
Reusability and modularity	10	
Clarity of documentation and programming	10	
Results (20)		
Correctness and completeness	20	
Report (40)		
Description of formulation and algorithmic approach	15	
Overview of program	10	
Discussion of validation approach	5	
Discussion of results	10	
TOTAL	100	

PA4

2021年11月30日 星期二 下午9:11

ICP brief work flow :

0. Given initial frame transform T_0, point set d_k, surface mesh M

1. Find closest point $c_k \in M$ for each $T \cdot d_k$

$c_k = \underset{c}{\operatorname{argmin}} \| T \cdot d_k - c \|$. This step can use simple search or octree

2. Throw out outlier points by choosing the effective ones :

c_k', $(T \cdot d_k)'$, s.t.

$\| (T \cdot d_k)' - c_k' \| \leq \eta$

η is related to square distance σ

or average distance $\bar{\varepsilon}$

here we use $\eta = 3\bar{\varepsilon}$

3. Use c_k' and d_k' to do point cloud registration

$T \cdot d_k' = c_k'$

Update $T(n)$.

4. Calculate mean square distance (error)

$$\sigma = \frac{\sum_k \| (T \cdot d_k)' - c_k' \|^2}{\mathrm{Num}(k)}.$$

and mean error

$$\bar{\varepsilon} = \frac{\sum_k \| (T \cdot d_k)' - c_k' \|}{\mathrm{Num}(k)}$$

3. Judge if the iterations converged.

Condition: $\sigma < 0.005$ or $\sigma_n / \sigma_{n-1} = 1$.

If not converge, go back to step 2.

Speeding up methods:

① Use bounding boxes for each triangular mesh unit.

We have lower bound and upper bound $\vec{L}_i$ and $\vec{U}_i$

current minimum distance is denoted as dis.

for a point $T \cdot \vec{d}_k$

only when
$$\begin{cases} \vec{L}_{ix} - dis \le (T \cdot \vec{d}_k)_x \le \vec{U}_{ix} + dis \\ \vec{L}_{iy} - dis \le (T \cdot \vec{d}_k)_y \le \vec{U}_{iy} + dis \\ \vec{L}_{iz} - dis \le (T \cdot \vec{d}_k)_z \le \vec{U}_{iz} + dis \end{cases}$$
satisfied,

projection and find closest point would be executed.

② Use OCtree.

Only when a treenode satisfies:

$$\begin{cases} \vec{L}_{tx} - dis \le (T \cdot \vec{d}_k)_x \le \vec{U}_{tx} + dis \\ \vec{L}_{ty} - dis \le (T \cdot \vec{d}_k)_y \le \vec{U}_{ty} + dis \\ \vec{L}_{tz} - dis \le (T \cdot \vec{d}_k)_z \le \vec{U}_{tz} + dis \end{cases}$$

do we still need to execute search and splittree.

- An overview of the structure of the computer program, sufficient to enable someone with reasonable skill (the grader) to understand your approach and follow your code.

Header: XX.h

--CatesianMathBasic.h

-- `Vector3d crossp(Vector3da, Vector3db)`

-- `double dotp(Vector3da, Vector3db)`

-- `Matrix3d invR(Matrix3dR)`

-- `Matrix3d Rot3(Vector3d w, double theta)`

--F.h

-- **Class F:**

--Class members: Rotation matrix R and translation vector p.

--Class functions: F operates F; F operates point; Inverse of F.

--3Dpoint_sets_registration.h

--F Point3D_Sets_Registration(vector<Vector3d>&a, vector<Vector3d>&b)

--ProjectOnSegment.h

--Function: `Vector3d ProjectOnSegment(const Vector3d& c, const Vector3d& v1, const Vector3d& v2)`

--BoundingBoxTreeNode.h

--Class BoundingBoxTreeNode:

--Class members:

Vector3d Center; //Splitting point

Vector3d UB; //Corners of box

Vector3d LB;

int HaveSubtrees;

int nBoxes;

BoundingBoxTreeNode* SubTrees[2][2][2];

BoundingBox** Boxes;

--Class functions:

BoundingBoxTreeNode(BoundingBox** BB, int nB);

void ConstructSubtrees();

void SplitSort(Vector3d SplittingPoint, BoundingBox** Boxes, int& nnn, int& npn, int& npp, int& nnp, int& pnn, int& ppn, int& ppp, int& pnp);

void FindClosestPoint(Vector3d v, double& bound, Vector3d& closest);

~BoundingBoxTreeNode();

Class BoundingBox:

--Class members:

Vector3d UB, LB, V1, V2, V3;

Vector3d Center;

--Class functions:

BoundingBox(Vector3d v1, Vector3d v2, Vector3d v3);

--FindClosestPoint.h/ ClosestPoint.h(They only change the function name.)

--Function: `Vector3d FindClosestPoint(Vector3d d, Vector3d v1, Vector3d v2, Vector3d v3)`

--matching_boundingbox.h

--Function: `vector<Vector3d> matching_boundingbox(int N_tri, F F_guess, const vector<Vector3d>& d, const vector<Vector3d>& Ver, const vector<VectorXi>& Tri)`

Library:

--<pch.h>

--< vector >

--<stdio.h>

--<stdlib.h>

--<iostream>

--<fstream>

--< iomanip >

--<Eigen\Dense>

Program: XX.cpp

-- F.cpp

-- F operates F: Multiplicate two transformations.

Input: Two transformations.

Output: $F_1F_2 = [R_1R_2,\ R_1p_2 + p_1]$.

-- F operates point: Compute a transformation of a point.

Input: Transformation F and a point.

Output: $p_{new} = R_Fp + p_F$.

-- F inverse: Compute the inverse of transformation F.

Input: Transformation F.

Output: $F.inv() = [R^T, -R^T * p]$.

-- CatesianMathBasic.cpp

-- crossp: Compute cross product of two vevtors. Use the function in Eigen.

Input: 2 vectors.

Output: cross product of these two vectors.

--dotp: Compute dot product of two vevtors. Use the function in Eigen.

Input: 2 vectors.

Output: inner product of these two vectors.

--invR: Compute inverse matrix of the rotation matrix R, which is equal to the transpose of matrix R. Use the function in Eigen.

Input: 1 rotation matrix.

Output: inverse of the input.

--Rot3: Build the rotation matrix R using Rodrigues's formula given rotation axis and angle.

Input: rotation axis, angle of rotating.

Output: corresponding rotation matrix.

--3Dpoint_sets_registration.cpp

--Input: 3D point set a & 3D point set b

--Output: Frame from coordinate system a to b.

--Process:

- Compute the average value of 3D point set a and b.
- Compute new value of centralized points in a and b.
- Quaternion method for R.
  1. Compute H.
  2. Compute G.

3. Compute eigen value decomposition of G.
4. The eigenvector corresponding to the largest eigenvalue is a unit quaternion corresponding to the rotation.
5. Compute the Rotation matrix using the unit quaternion.

- Output desired transformation frame F.

-- ProjectOnSegment.cpp

--Input: Point needs to be projected and two triangle vertexes.

--Output: The projected point.

--Process:

- Compute the norm of each point in the dataset q with all the points in the dataset c, using the following equation:$\lambda = \frac{(c-p)\cdot(q-p)}{(q-p)\cdot(q-p)}$.
- Compute the projected point with the following equation: $c^* = p + \lambda \times (q-p)$.

-- FindClosestPoint.cpp/ ClosestPoint.cpp

--Input: point *d*, and vertices of the triangle mesh unit *p, q, r*.

--Output: The closest projection point on the triangle *prj*

--Process:

- For each point d, use SVD decomposition approach to solve the least square problem to get coefficient matrix of the interpolation.
- Judge which region the projection lies in.
- Find the closest point *prj,* using different methods for different regions.

-- matching_boundingbox.cpp

--Input: Number of triangle mesh units, transformation T, point dataset q, surface vertices dataset Ver, and triangle dataset Tri.

--Output: The coordinates of the point set c* which are the closest point on the surface to each T*q.

--Process:

- Judge if this box need to calculate projection.
- If the box is within search range, then use **FindClosestPoint** to find the closest point *prj*.
- Compute the norm of each point in the dataset q with all the closest points for every searchable box, keep updating the smallest distance for each searched box.
- Output the set of the projected(closest) points corresponding to each point in dataset q.

--Main program: PA4-BoundingBox.cpp

--Input: Data files(a, b, A, B and several numbers).

--Outputs: Dataset s, c, and their norm distances.

--Process: Fig1 shows the flowchart of the main program.

--BoundingBoxTreeNode.cpp

--Class BoundingBox:

--Inputs: The coordinates of the three vertices of a triangle.

--Outputs: The BoundingBox (including LB, UB, and Center) of the trangle.

--Class BoundingBoxTreeNode:

--Inputs: The pointer points to the BoudingBox array: BoundingBox** BB and the number of BoundingBoxes int nB;

--Outputs: The LB, UB, and Center of the BoundingBox tree. When you call the function of FindClosestPoint, you will need to input an initial bound value and closest point, and then it will build subtrees based on the criteria and search each subtree to update the bound and the closest point.

--Main program: PA4-Octree.cpp

--Input: Data files(a, b, A, B and several numbers).

--Outputs: Dataset s, c, and their norm distances.

--Process: Fig2 shows the flowchart of the main program.

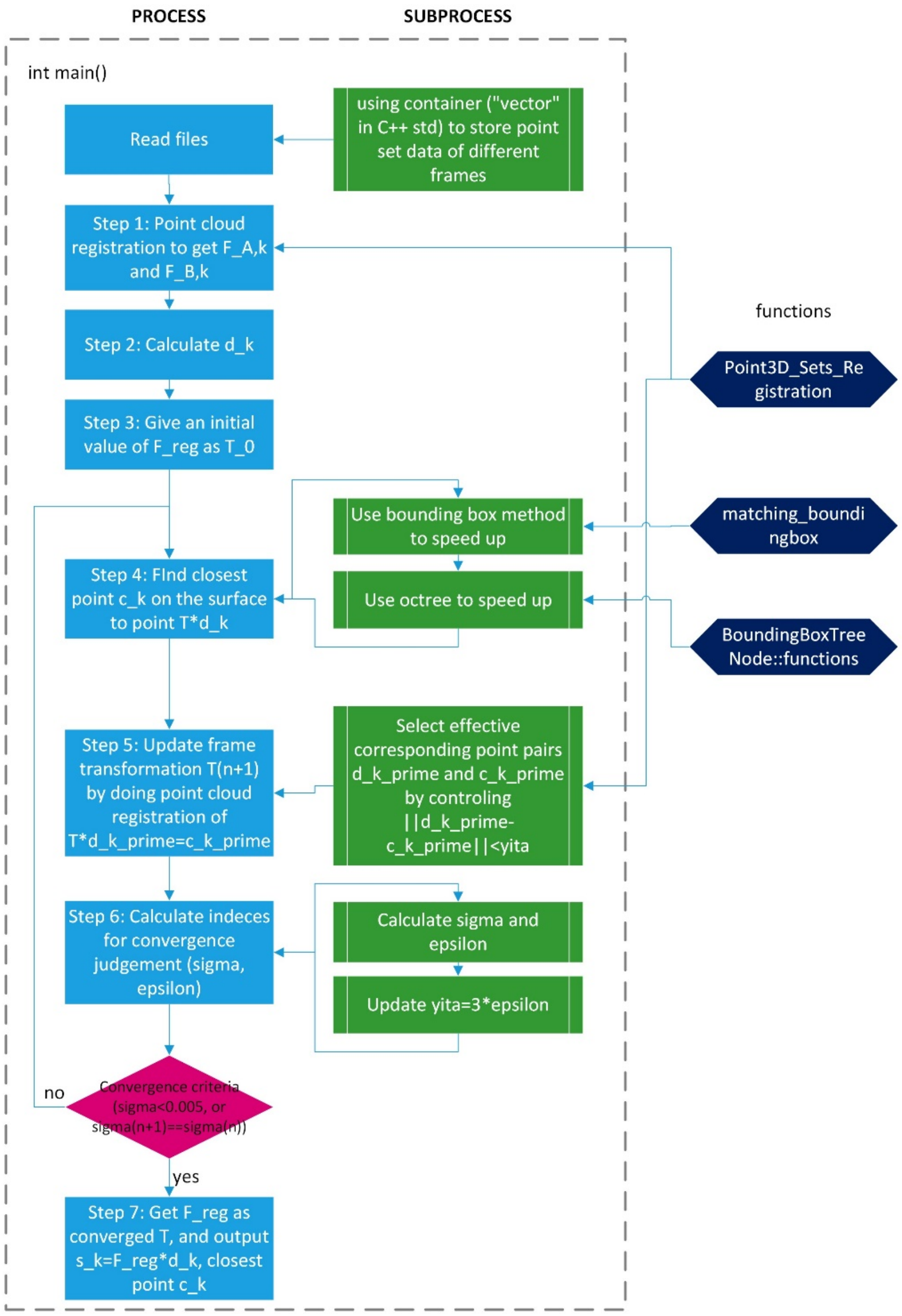
PROCESS
SUBPROCESS
int main()
Read files
using container ("vector" in C++ std) to store point set data of different frames
Step 1: Point cloud registration to get F_A,k and F_B,k
Step 2: Calculate d_k
Step 3: Give an initial value of F_reg as T_0
functions
Point3D_Sets_Registration
Use bounding box method to speed up
Use octree to speed up
matching_boundingbox
Step 4: FInd closest point c_k on the surface to point T*d_k
BoundingBoxTreeNode::functions
Step 5: Update frame transformation T(n+1) by doing point cloud registration of T*d_k_prime=c_k_prime
Select effective corresponding point pairs d_k_prime and c_k_prime by controling ||d_k_prime-c_k_prime||<yita
Step 6: Calculate indeces for convergence judgement (sigma, epsilon)
Calculate sigma and epsilon
Update yita=3*epsilon
no
Convergence criteria (sigma<0.005, or sigma(n+1)==sigma(n))
yes
Step 7: Get F_reg as converged T, and output s_k=F_reg*d_k, closest point c_k

Figure 1 Flowchart of the main program for PA4-BoundingBox

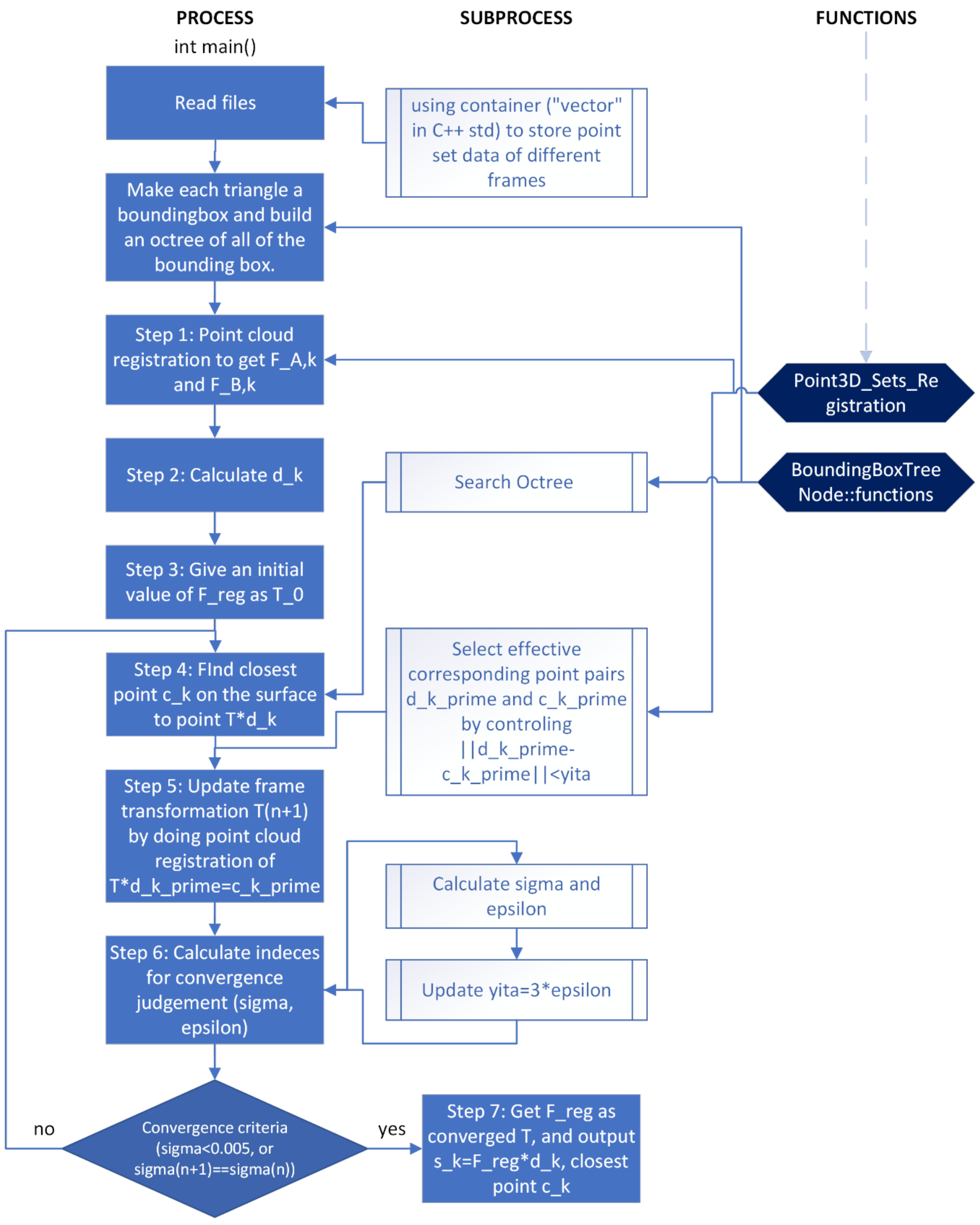
PROCESS
int main()
SUBPROCESS
FUNCTIONS
Read files
using container ("vector" in C++ std) to store point set data of different frames
Make each triangle a boundingbox and build an octree of all of the bounding box.
Step 1: Point cloud registration to get F_A,k and F_B,k
Point3D_Sets_Registration
Step 2: Calculate d_k
Search Octree
BoundingBoxTree Node::functions
Step 3: Give an initial value of F_reg as T_0
Step 4: FInd closest point c_k on the surface to point T*d_k
Select effective corresponding point pairs d_k_prime and c_k_prime by controling ||d_k_prime-c_k_prime||<yita
Step 5: Update frame transformation T(n+1) by doing point cloud registration of T*d_k_prime=c_k_prime
Calculate sigma and epsilon
Step 6: Calculate indeces for convergence judgement (sigma, epsilon)
Update yita=3*epsilon
no
Convergence criteria (sigma<0.005, or sigma(n+1)==sigma(n))
yes
Step 7: Get F_reg as converged T, and output s_k=F_reg*d_k, closest point c_k

Figure 2 Flowchart of the main program for PA4-Octree

- The steps taken to verify that the program is working correctly. Typically, this would take the form of a discussion of the results using the debugging examples.

1. This time we wrote ICP with two approaches. The first one is searching triangles by BoundingBox, and the other one is searching triangles by the octree of the BoundingBox.
2. First we used the BoundingBox approach.
   1) We wrote some "cout"s at several important points of our programs, mainly to check if the intermediate result and size of containers are correct.
   2) Read the debug data from A to F to get their computed output file separately.
   3) Compare the output data we get from our code with that of the given data by looking through these two files. At the beginning, our outputs is far from the given output. By analyzing the data step by step, we found we didn't multiplate the frame T, which makes the frame not update during the searching in each iteration. After fix this problem, our results are close to the given outputs.
3. And then we built octree to make the ICP approach fast.
   1) We wrote some "cout"s at several important points of our programs, mainly to check if the intermediate result and size of containers are correct.
   2) Read the debug data from A, and set breakpoint in each step of the searching the closest point using octree. Compare the data in the breakpoint with that of the previous approach. For example, the closest point vector in each iteration.
   3) After making sure the searching performance of the octree, set a stricter converge criteria of ICP process and read the debug data from A to F to get their computed output file separately.
   4) The time used to finish the output decreased sharply in debug configurations of visual studio 2019. For example, computing 6 PA4-X-Debug files with the BoundingBox approach with converge criteria of 0.005 used 3h, while computing them with the octree approach with converge criteria of 0.001 used about 0.5h. When compute these files in the application, the speed is much higher and the octree approach with converge criteria of 0.001 used 48s.
4. The iterations of each debug file with converge criteria of 0.001 and relative sigma value 0.95.

Table 1 Output analysis of debug dataset A

Filename	PA4-A-Debug	PA4-B-Debug	PA4-C-Debug	PA4-D-Debug	PA4-E-Debug	PA4-F-Debug
Iterations	1	14	21	18	33	17
Sigma	9.78273e-06	0.00086962	0.00087131	0.000758199	0.00354616	0.00581178
Epsilon	0.00267388	0.0259575	0.0257116	0.0237161	0.0483674	0.0610226

Here for dataset E and F, relative sigma value was the final convergence criterion. It's possible that in these cases there are quite some noise or outliers, so the value of sigma is decreasing very slowly at a magnitude of 0.04~0.06.

5. Analyze the specific data of each debug output with the first 15 points. The PA3-X-Debug row is our outputs, and the Ref row is from the given data.

Table 2 Output analysis of debug dataset A

Filename	N_{samps}	$\|\|T * d_k - c_k\|\|$														
PA4-A-Debug	75	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Ref	75	0.002	0.002	0.003	0.007	0.002	0.002	0.003	0.001	0.001	0.002	0.005	0.001	0.003	0.006	0.004

Table 3 Output analysis of debug dataset B

Filename	N_{samps}	$\|\|T * d_k - c_k\|\|$														
PA4-B-Debug	200	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Ref	200	0.042	0.033	0.023	0.022	0.044	0.019	0.009	0.029	0.014	0.031	0.019	0.033	0.054	0.030	0.037

Table 4 Output analysis of debug dataset C

Filename	N_{samps}	$\|\|T * d_k - c_k\|\|$														
PA4-C-Debug	200	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Ref	200	0.056	0.021	0.013	0.008	0.020	0.027	0.063	0.017	0.053	0.011	0.064	0.035	0.013	0.047	0.010

Table 5 Output analysis of debug dataset D

Filename	N_{samps}	$\|\|T * d_k - c_k\|\|$														
PA4-D-Debug	200	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Ref	200	0.015	0.012	0.014	0.020	0.057	0.015	0.024	0.038	0.018	0.014	0.049	0.018	0.007	0.029	0.024

Table 6 Output analysis of debug dataset E

Filename	N_{samps}	$\|\|T * d_k - c_k\|\|$														
PA4-E-Debug	200	0.158	0.064	0.060	0.009	0.035	0.048	0.015	0.109	0.043	0.005	0.011	0.088	0.073	0.074	0.077
Ref	200	0.143	0.055	0.040	0.011	0.006	0.024	0.019	0.093	0.014	0.017	0.018	0.097	0.095	0.050	0.092

Table 7 Output analysis of debug dataset F

Filename	N_{samps}	$\|\|T * d_k - c_k\|\|$														
PA4-F-Debug	200	0.028	0.029	0.022	0.011	0.032	0.038	0.020	0.165	0.018	0.122	0.118	0.026	0.056	0.015	0.115
Ref	200	0.21	0.031	0.034	0.049	0.040	0.035	0.020	0.171	0.037	0.102	0.115	0.029	0.066	0.017	0.099

Data of point coordinates are in the output file, and they match well with the reference data.

For most of the distance output data of the debug datasets, deviations are found at the 2nd or 3rd digit after the decimal point. Compared to the scale of the distance data, these deviations are at 0.1%-1% magnitude, so we can recognize it as numerical errors and recognize our method and codes as correct. This numerical error most possibly comes from the iteration criteria and the process of solving the least squares problem when calculating interpolation parameters lambda and mu of the projected point. Sometimes different numerical methods (QR decomposition, SVD decomposition) may lead to inconsistence at the trivial digits.

Basically, from these data, we can find one way to improve the results is to decrease the absolute convergence criterion (the sigma cutoff value). For fast convergence we pick 0.001 for now, which is corresponding to an average error of the magnitude of sqrt(0.001). For better precision, a sigma cutoff value of 1e-6 and relative sigma decrease value of 0.95 is recommended.

Our algorithm presents quite good speed with one dataset done in tens of seconds on a common laptop.

6. Also, we draw a graph to visualization the meshes bone and the ICP results. (The red circles in the graph are the results of ICP and the blue dots are surface meshes.)

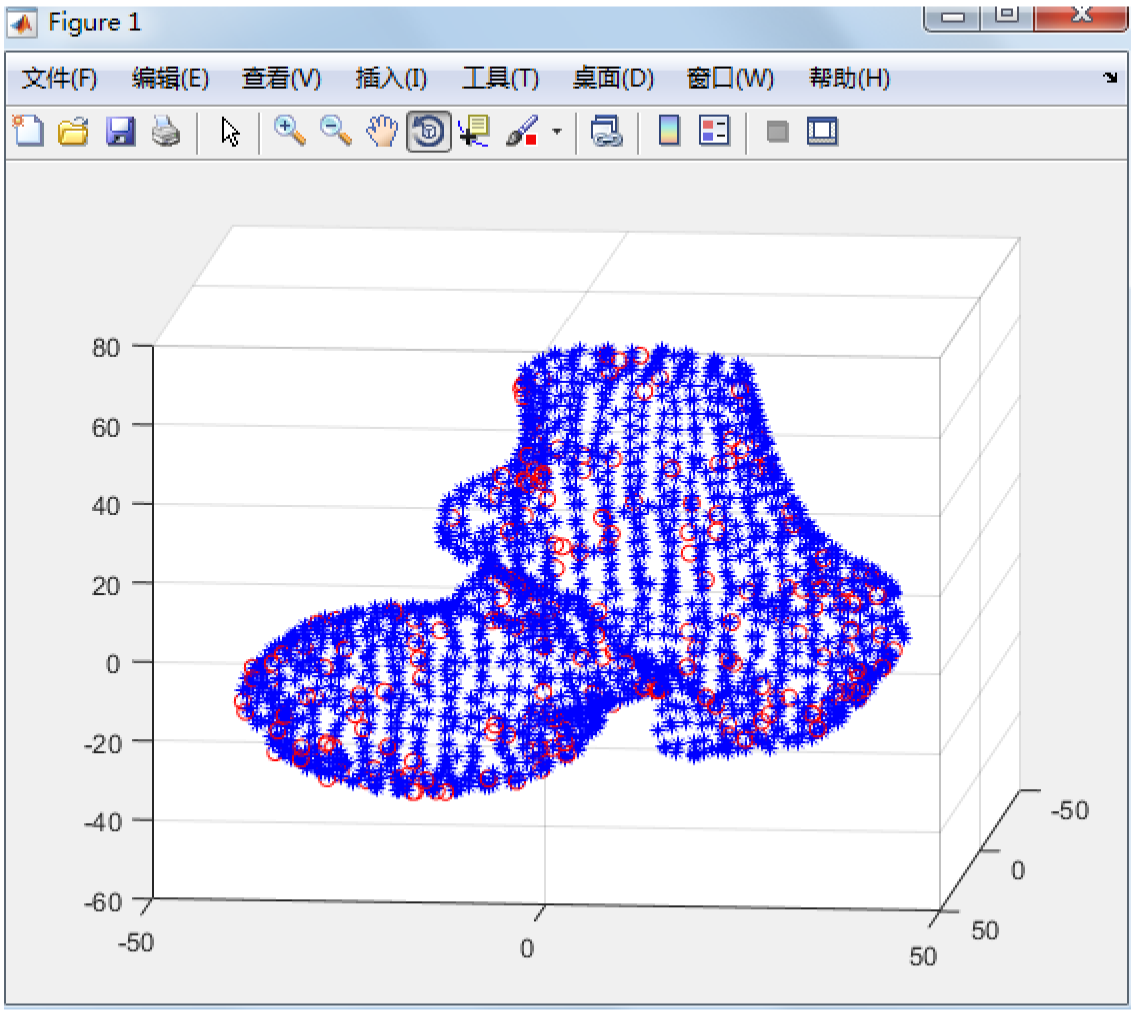

Figure 3 Visualization Graph

- A tabular summary of the results obtained for unknown data

We have too many points that cannot be shown in one tabular, so we are only going to show the iteration results here. The point coordinates can be found in our output file.

Table 8 Results obtained for unknown data

Filename	PA4-G-Unknown	PA4-H-Unknown	PA4-J-Unknown	PA4-K-Unknown
Iterations	17	21	17	14
Sigma	0.000780695	0.000789507	0.00694352	0.00661326
Epsilon	0.0243114	0.0246403	0.0669934	0.0655116

- A short discussion for the results of running your program. This certainly includes the tabular summary above, but may also include a discussion of convergence if you adopt an iterative process or of difficulties if you suspect that your answer is wrong.

  Each of these cases takes around tens of seconds on a common laptop, which is quite a short time. And convergence situation is shown as the above table. Most of the unknown file have the similar iterations with the debug file and their numbers of points in each file is just the same as that of the debug file. Hence based on the data in the table, we can see the results of ICP in the table looks fairly plausible,

- A short statement of who did what.

1. Both: Analyzed the scenario and figured out the Pseudo-code of each part of the assignment. Debugged and modified the whole code. Discussed the result and analyzed the reasons.
2. Qiyuan Wu: Finished the ICP iteration and outputted the right output file.
3. Shuran Zhang: Built up octree.