Overview

Saturday, November 13, 2021 7:13 PM

Programming Assignment 5 – 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	HongYi Fan
Email	hfan15@jhu.edu.
Other contact information (optional)	
Name 2	Yuxin Chen
Email	ychen506@jhu.edu
Other contact information (optional)	
Signature (required)	I (we) have followed the rules in completing this assignment HongYi Fan Yuxin Chen

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	

NOTE: This is an optional assignment.

If you hand it in, I will use the grade to replace the lowest other programming assignment or written homework assignment with one exception:

You may not drop **both** HW#3 and HW#4. If these are your two lowest grades, then I will drop the lower of those two under the drop 1 homework scenario and replace the next lowest grade (other than the other of HW#3-4) with this score

Overview

Contents of this document:

- Program Structure
- Algorithm of Deformable Registration Calculation
- Algorithm of Iterative Closest Point
- Algorithm of find closest point
- Validation and Results
- A tabular summary of the results obtained for unknown data

A short statement of who did what:

Yuxin Chen:

We went through the whole assignment together and discussed the algorithm and mathematical approach we decided to use. Then I contributed to continue to write the rest ICP algorithm. I contributed the majority of report.

HongYi Fan:

- Contributed in majority of CIS math package
- Contributed in closest point on mesh with KD-Tree
- Contributed in ICP optimization
- Contributed in Deformation Weight Calculation functions, and PA5 utility functions
- Contributed in report writing and validating

External Resources:

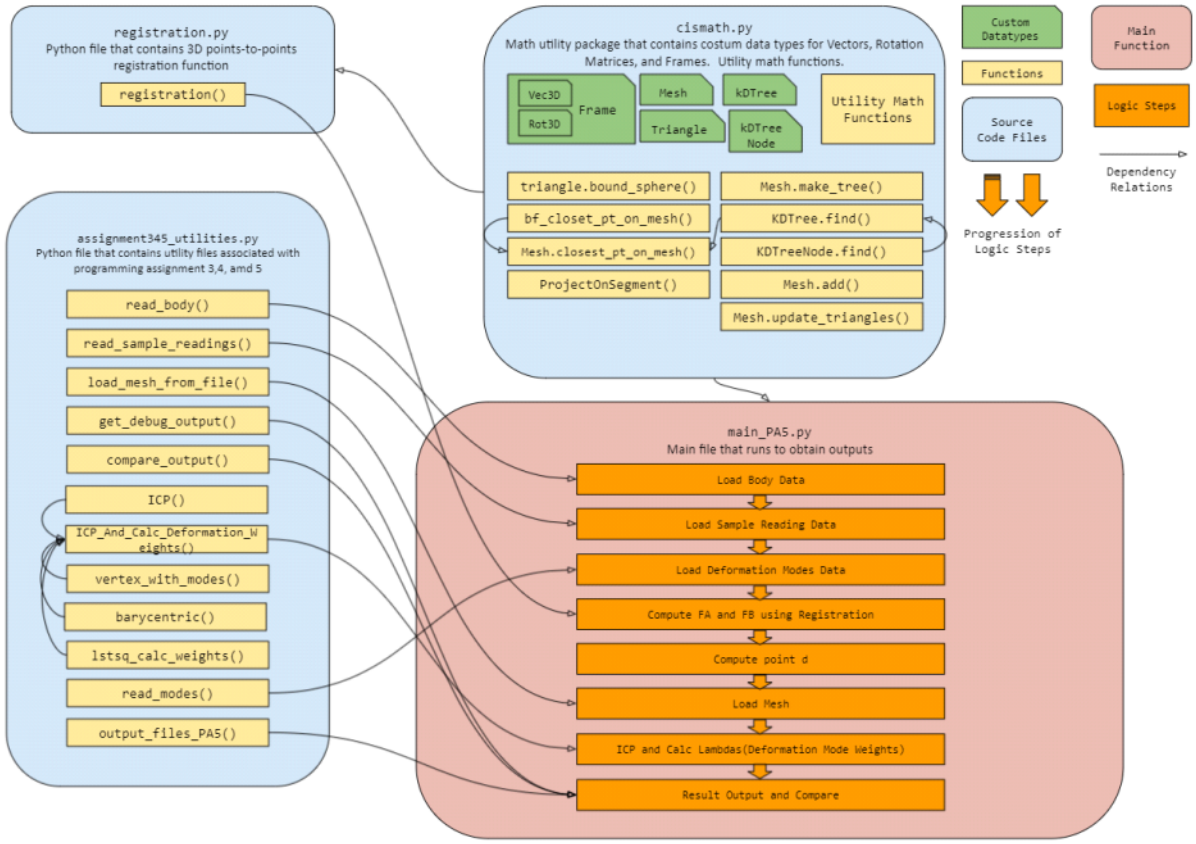
- *Numpy (https://numpy.org/)*

- *Figure 1[GPL, https://commons.wikimedia.org/w/index.php?curid=589909]*

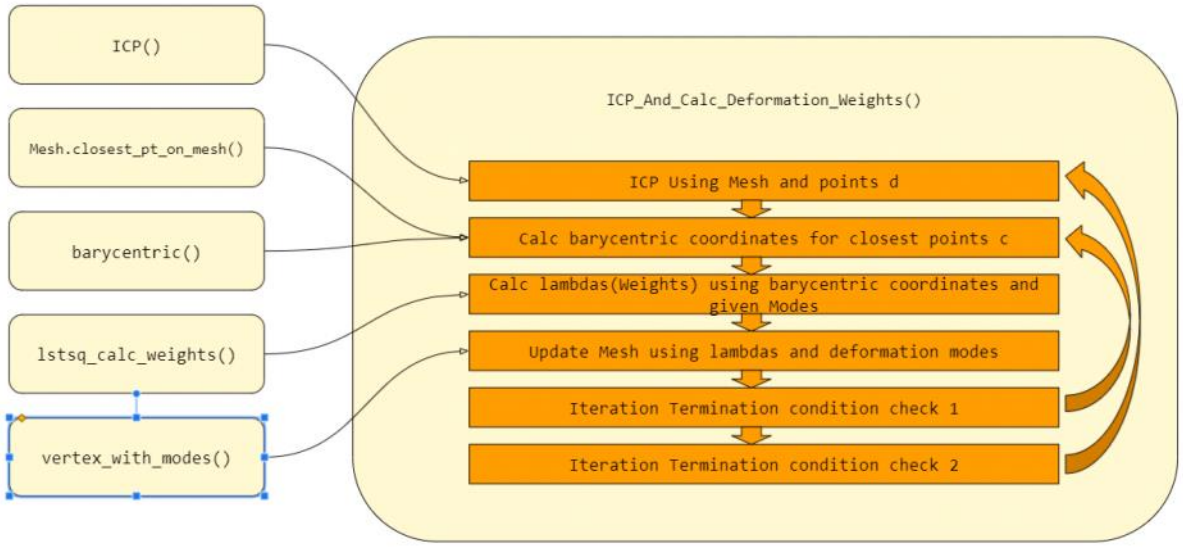
Program Structure

Saturday, November 13, 2021 7:16 PM

Overall Program Structure for Programing Assignment 5:



Detailed Program Structure for Programing Assignment 5 Task:



Algorithm of Iterative Closest Point (ICP)

Saturday, November 13, 2021 7:17 PM

Algorithm of computing $\overrightarrow{d_k}$ has been explained in PA3 report.
Algorithm of ICP used in this PA5 is same as it in PA4.

Iterative Closest Point :

Known:

Input:

- A 3D Surface model M, represented as a mesh of triangles read from the mesh file (the coordinates of the vertices of this mesh in CT coordinates
- Set of points $\overrightarrow{d_k}$ (the position of the pointer tip with respect to rigid body B) that known to be on the 3D surface model M.
- Initial guess F_0 for transformation F_{reg} such that the points $F_{reg} * \overrightarrow{d_k}$, lie on surfece model M. In PA3, we assume the F_0 to be [I | 0].
- Initial threshold η_0 for the match closeness: Initial threshold η_0 will be firstly set as fairly large value based on the data we collected. The threshold will then decrease after a couple of iterations.

Goal:

- Given a set of triangles (mesh) of the patient's bone model, and a set of points obtained on the surface of the mesh, obtain the rigid transformation that describes the pose of the patient's bone relative to the tracker.

Steps:

Step 0: Initialization

n ← 0

η_0 ← large number

$\vec{d}_{k_{Sample}}$ ← an uniform random sample of $\vec{d}_k$, this group of vector used to shorten ICP time cost in early iterations. Here choose $\vec{d}_{k_{Sample}}$ to be $1/3$ or $1/4$ of the size of $\vec{d}_k$

C ← closest points on M

$e_k = \left\| F_n \cdot \overrightarrow{d_k} - \overrightarrow{c_k} \right\|$

Step 1: matching

A ← ∅, B ← ∅

For k ← 1 step 1 to N do:

$bne_k = \left\| F_n \cdot \vec{d}_{k_{Sample}} - \overrightarrow{c_k} \right\|$

[$\overrightarrow{c_k}$, i, e_k] ← FindClosestPoint ($F_n \cdot \vec{d}_{k_{Sample}}$, $\overrightarrow{c_k}$, i_k, bne_k, T)

- Where T is the method we choose to use
- Two method are used in this programming method:
  1. Simple brutal force method
  2. Advanced method with kD-Tree

If ($e_k < \eta_k$) then {put $\overrightarrow{d_k}$ into A; put $\overrightarrow{c_k}$ in to B};

Step 2: transformation update

n = n + 1

$F_n \leftarrow FindBestRigidTransformation(A, B)$

$\sigma_n \leftarrow \frac{\sqrt{\sum_k \overrightarrow{e_k} \cdot \overrightarrow{e_k}}}{NumElts(E)}$

$(\varepsilon_{max})_n \leftarrow \max_k \sqrt{\overrightarrow{e_k} \cdot \overrightarrow{e_k}}$

$\overline{\varepsilon_n} \leftarrow \frac{\sum_k \sqrt{\overrightarrow{e_k} \cdot \overrightarrow{e_k}}}{NumElts(E)}$

Algorithm of Iterative Closest Point (ICP)

Monday, November 22, 2021 9:02 PM

Step 3: adjustment

Compute η_n from $\{\eta_0, \cdots, \eta_{n-1}\}$:

η is the thereshold parameter that determines wether a pair of closest points are considered as matched. If a pair of closest points has ruclidean distance less than η, then they are considered as a matcher pair. η is set to be a large number at the beginning of ICP to accomodate large euclidean distance between points, then η is adaptively decreased as the ICP iterates.

The estimation of η in Programming Assignment 4 is calculated as follow:

$$\eta = 3 \times \bar{\epsilon}_n$$

Step 4: Termination Check With Sample $\vec{d}_{k_{Sample}}$ Group

The termination condition is determined using parameters $\{\sigma_0, \cdots, \sigma_n\}$, $\{(\varepsilon_{max})_0, \cdots, (\varepsilon_{max})_n\}$, $\{\bar{\varepsilon_0}, \cdots, \bar{\varepsilon_n}\}$) .

The program checks if the ICP procedure using $\vec{d}_{k_{Sample}}$ returns promissing results by checking following condition:

$$1 \geq \frac{\bar{\varepsilon}_n}{\bar{\varepsilon}_{n-1}} \geq 0.98 \quad \text{for 7 continuous } n$$

OR

$$\sigma_n < 0.0008 \; and \; \epsilon_{\max} < 0.0008 \; and \; \bar{\epsilon_0} < 0.0008$$

- If so, the program proceeds to Step 5 and continues ICP using the full group of $\vec{d}_k$.
- If not so, the program continues ICP from step 1.

Step 5: Termination Check With $\vec{d}_k$ Group

If a desired condition is reached by ICP algortithm when using $\vec{d}_k$ group, then exit iterations. Otherwise, continue ICP.

The termination condition is determined using parameters $\{\sigma_0, \cdots, \sigma_n\}$, $\{(\varepsilon_{max})_0, \cdots, (\varepsilon_{max})_n\}$, $\{\bar{\varepsilon_0}, \cdots, \bar{\varepsilon_n}\}$) .

Termination Condition:

$$1 \geq \frac{\bar{\varepsilon}_n}{\bar{\varepsilon}_{n-1}} \geq 0.98 \quad \text{for 7 continuous } n$$

OR

$$\sigma_n < 0.0008 \; and \; \epsilon_{\max} < 0.0008 \; and \; \bar{\epsilon_0} < 0.0008$$

Otherwise, continues ICP back from step 1(still using $\vec{d}_k$)

Algorithm of find closest point

Wednesday, November 17, 2021 20:47

Optimized Search Using Bounding Spheres and KD-Tree Structure

Since a high definition mesh structure may have an overwhelming quantity of triangles, calculating closest point to each triangle is not time efficient. Therefore, a KD-Tree Data Structure can be used to optimize the time efficiency by selectively choosing portion of triangles that is closest to the given point in space.

1) Add a bounding sphere attribute to each triangle

Doing so helps the program to keep track of triangles' locations and bounding using one point instead of three

Mathematical Approach:

For each triangle with vertices a, b, c, where (a, b) is the long edge:

1. Calculate possible center $\vec{q} = \frac{\vec{a}+\vec{b}}{2}$
2. Check $\vec{q}$ with counding sphere inequalities:
   a. $\left(\vec{b}-\vec{q}\right)\cdot\left(\vec{b}-\vec{q}\right) = \left(\vec{a}-\vec{q}\right)\cdot\left(\vec{a}-\vec{q}\right)$
   b. $\left(\vec{c}-\vec{q}\right)\cdot\left(\vec{c}-\vec{q}\right) \leq \left(\vec{a}-\vec{q}\right)\cdot\left(\vec{a}-\vec{q}\right)$
   c. $\left(\vec{b}-\vec{a}\right)\times\left(\vec{c}-\vec{a}\right)\cdot\left(\vec{q}-\vec{a}\right) = 0$
3. If above inequalities hold, then $\vec{q}$ is the center of the sphere. Continue otherwise
4. Calculate $\vec{f} = \frac{\vec{a}+\vec{b}}{2}$
5. Define $\vec{u} = \vec{a}-\vec{f}, \vec{v} = \vec{c}-\vec{f}, \vec{d} = \left(\vec{u}\times\vec{v}\right)\times\vec{u}$, now center $\vec{q}$ lies in $\vec{q} = \vec{f}+\lambda\vec{d}$
6. Solve $\lambda \geq \frac{\vec{v}^2-\vec{u}^2}{2\vec{d}\cdot\left(\vec{v}-\vec{u}\right)} = \gamma$, if $\gamma \leq 0$ *then* $\lambda = 0$. *Otherwise* $\lambda = \gamma$
7. With center $\vec{q}$, calculate the radius by calculate the 2-norm of $\left(\vec{q}-\vec{a}\right)$

With a known bounding sphere of a triangle, the program can skip some of closest-point-on-triangle calculations based on previously calculated values. The program only proceed with closest-point-on-triangle calculation only when the following condition is satisfied:

$$\left|\left|\vec{p}-\vec{q}\right|\right| - r < \left|\left|\vec{p}-\vec{u}\right|\right|$$

Where $\vec{p}$ is a point in space, $\vec{q}$ is the center of the bounding sphere, r is the radius of the bounding sphere, and $\vec{u}$ is the closest point found on other triangles.

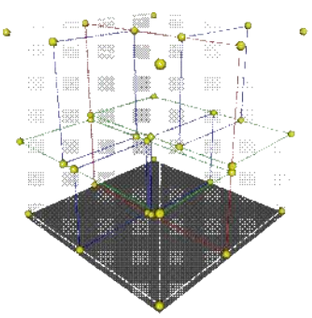


Figure 1

2) Construct kD-Tree (3D-Tree)

1. Separate triangles in the 3D space into two sub-space by the x value of their bounding spheres'. That is, as shown in the figure on the right, pick the median element m among all triangles, for all triangles with x value less than m, is on the leftside of the red plane. All other triangles, including m, is on the right side of the red plane.
2. For each sub space of triangles, divide it again as described in step 1, but uses y values instead of x value (the green planes). And apply again using the z values (the blue plane). And then x values again. Until the subspace contains only one triangle or desired depth is reached.
3. Since triangle shape may reach out of these boxes, like figure on the right shows. Therefore adjust boxes to enclose triangles.
4. Now we have a tree-like structure in which a node (space)
   Has 2 child nodes(subspaces)

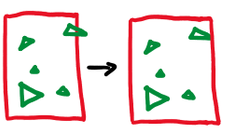


Figure 2

3) Search Using the kD-Tree

1. Given a point $\vec{p}$ in space, locate the closest leaf node
   If the x coordinate of $\vec{p}$ is less than the median value of the current node then go the left node, otherwise the right node, etc.
2. After reaching the leaf node, find closest point on each of the triangles, store the closest point $\vec{c}$ and the distance to the closest point r.
3. Consider a sphere formed on point $\vec{c}$ with radius r. The sphere includes other possible closer triangles in the space. If the sphere touches other node spaces, it means that space may contain closer triangle. Therefore, we also need to look into these nodes.
4. If a closer point on triangle is found during the process, update $\vec{c}$ and r accordingly. Until the sphere on $\vec{c}$ with radius r does not intersect with pther node. We now have the closest point on mesh.

Algorithm of deformable registration

Sunday, December 12, 2021 8:15 PM

Known:

Input:

- A 3D Surface model M, represented as a mesh of triangles read from the mesh file (the coordinates of the vertices of this mesh in CT coordinates
- An atlas "modes" file, giving modes of variation for the model, where "Mode 0" represents the average shape.
- Set of points $\overrightarrow{d_k}$ (the position of the pointer tip with respect to rigid body B) that known to be on the 3D surface model M.

Goal:

- Given a set of triangles (mesh) of the patient's bone model and atlas "modes" file, and a set of points obtained on the surface of the mesh, we aims to extend the ICP program we did in programming 3 and to 4 to perform a simple deformable registration.

Steps:

Step 0: Initialization

Perform an initial "rigid" registration using the same method for Programming 4, which means use ICP to perform a rigid registration by using the mesh file we read which is the same as Mode 0 to get a rigid transformation ${F_{reg}}^{(0)}$.

t ← 0
t is the iteration number

Step 1:

$\overrightarrow{s_k}^{(t)} = {F_{reg}}^{(t)} \cdot \overrightarrow{d_k}$

Where $\overrightarrow{s_k}^{(t)}$ is the current estimate of transformed sample point k at iteration t

${F_{reg}}^{(t)}$ is the current estimate of the rigid transformation

$\overrightarrow{d_k} = {F_{B,k}}^{-1} \cdot F_{A,k} \cdot \overrightarrow{A_{tip}}$ is the corresponding measured sample point value

Algorithm of computing $\overrightarrow{d_k}$ has been explained in PA3 report.

$\overrightarrow{c_k}^{(t)}$ is current estimate of the cloest point on the deformed surface to the transformed sample point we got from ICP. At the first iteration. $\overrightarrow{c_k}^{(0)}$ represents the estimate of the closest point on the average shape of surface (Mode 0).

Suppose the vertex indices of those triangles where $\overrightarrow{c_k}^{(t)}$ are on are {s, t, u}.
Then the coordinates of the corresponding deformed mesh will be

$$\overrightarrow{m_s} = \overrightarrow{m_{0,s}} + \sum_{m=1}^{Nmodes} {\lambda_m}^{(t)} \overrightarrow{m_{m,s}}$$

$$\overrightarrow{m_t} = \overrightarrow{m_{0,t}} + \sum_{m=1}^{Nmodes} {\lambda_m}^{(t)} \overrightarrow{m_{m,t}}$$

$$\overrightarrow{m_u} = \overrightarrow{m_{0,u}} + \sum_{m=1}^{Nmodes} {\lambda_m}^{(t)} \overrightarrow{m_{m,u}}$$

Compute the barycentric coordinates of $\overrightarrow{c_k}^{(t)}$

$$\overrightarrow{c_k}^{(t)} = \zeta_k \overrightarrow{m_s} + \xi_k \overrightarrow{m_t} + \psi_k \overrightarrow{m_u}$$

Get an expression in terms of mode coordinates $\overrightarrow{c_k}^{(t)}$

$$\overrightarrow{c_k}^{(t)} = \overrightarrow{q_{0,k}} + \sum_{m=1}^{Nmodes} {\lambda_m}^{(t)} \overrightarrow{q_{m,k}}$$

Where

$$\overrightarrow{q_{m,k}} = \zeta_k \overrightarrow{m_{m,s}} + \xi_k \overrightarrow{m_{m,t}} + \psi_k \overrightarrow{m_{m,u}}$$

Mathematical Approach for barycentric coordinates calculation:

given $\overrightarrow{c_k}^{(t)} = \zeta_k \overrightarrow{m_s} + \xi_k \overrightarrow{m_t} + \psi_k \overrightarrow{m_u}$
We have:

$$[\vec{m}_s \quad \vec{m}_t \quad \vec{m}_u] \begin{bmatrix} \zeta_k \\ \xi_k \\ \psi_k \end{bmatrix} = \vec{c}_k^{(t)}$$

$$\begin{bmatrix} \zeta_k \\ \xi_k \\ \psi_k \end{bmatrix} = [\vec{m}_s \quad \vec{m}_t \quad \vec{m}_u]^{-1} \vec{c}_k^{(t)}$$

And $\Lambda^{(t)} = \{\lambda_1, \cdots, \lambda_{Nmodes}\}^{(t)}$ represents the current estimate of the mode weights.

Thus, we should have

$$\overrightarrow{s_k}^{(t)} \approx \overrightarrow{c_k}^{(t)}$$

$${F_{reg}}^{(t)} \cdot \overrightarrow{d_k} \approx \overrightarrow{q_{0,k}} + \sum_{m=1}^{Nmodes} {\lambda_m}^{(t)} \overrightarrow{q_{m,k}}$$

Step 2: Iterative Adjustment and update

Keep $\overrightarrow{s_k}^{(t)}$ fixed and find the corresponding $\overrightarrow{q_{m,k}}$ to solve the following least squares problem for $\Lambda^{(t+1)} = \{\lambda_1, \cdots, \lambda_{Nmodes}\}^{(t+1)}$

$$\overrightarrow{s_k}^{(t)} \approx \overrightarrow{q_{0,k}} + \sum_{m=1}^{Nmodes} {\lambda_m}^{(t+1)} \overrightarrow{q_{m,k}}$$

Use $\Lambda^{(t+1)}$ to update the surface mesh model to find the new estimate for the vertices of the deformed mode.

Update the kd-tree structure with new deformed triangles and bounding spheres

Find the new matching points $\overrightarrow{c_k}^{(t+1)}$ on new deformed mode by using the method of finding cloest point explained in PA3&4.

Iterate until the magnitude of least square residual in early step 2 reaches a lowest stable value. In the program, this condition is quantified as:

$0.98 < residual < 1.2$ for 5 continueous iterations

Step 3: Keep the model vertices fixed and use the same method of PA4 to re-estimate ${F_{reg}}^{(t)}$

Step 4: Termination

If a desired condition is reached then exit iterations. Otherwise, return to step 1
The termination condition:

0.8 < New ICP Final Error Mean/old ICP Final Error Mean < 1.2 for 3 continuous iteration
Or
Number of ICP > 10

Validation and Result

Saturday, November 13, 2021 7:17 PM

To verify the result, we have the following testing:

1) List the $\lambda_1, \cdots, \lambda_{Nmodes}$ we got for debug dataset A-F
2) Verify $\lambda_1, \cdots, \lambda_{Nmodes}$ with output data
3) Verify the $\vec{s}$ with the output data
4) Verify the $\vec{c}$ with the output data
5) Verify the $\left\|\overrightarrow{s_k} - \overrightarrow{c_k}\right\|$ with the output data
6) Verify the termination condition
7) Compare the $\sigma_n, (\varepsilon_{max})_n, \bar{\varepsilon}_n$ at the beginning and termination

The result of weight $\{\lambda_1, \cdots, \lambda_{Nmodes}\}$ we got for the debug dataset A-F

Dataset	$\{\lambda_1, \cdots, \lambda_{Nmodes}\}$
A	{76.5996, -40.5849, -9.1862, 157.9178, -34.9335, 101.9813}
B	{74.4831, -192.7506, -94.6029, 136.6587, 52.4793, 43.0591}
C	{34.6571, -107.3067, 77.9070, 46.0204, 121.0507, 54.7387}
D	{-43.8354, 158.2890, 29.5098, 85.5818, -115.9699, 31.8378}
E	{-68.4756, -157.4146, 74.8571, 105.5475, 48.7390, -19.7716}
F	{74.2664 -82.3316 -99.8412 61.3151 -49.7688 81.7338}

Average error between our output with the output file provided by professor

Dataset	Average error in $\{\lambda_1, \cdots, \lambda_{Nmodes}\}$	Average error in $\vec{s}\ (s_x, s_y, s_z)$	Average error in $\vec{c}\ (c_x, c_y, c_z)$	Average error in $\left\|\overrightarrow{s_k} - \overrightarrow{c_k}\right\|$
A	1.223	[0.075, 0.072, 0.165]	[0.179, 0.190, 0.189]	0.297
B	2.102	[0.107, 0.168, 0.138]	[0.235, 0.301, 0.222]	0.408
C	3.811	[0.124, 0.187, 0.091]	[0.189, 0.277, 0.166]	0.318
D	2.471	[0.026, 0.081, 0.048]	[0.186, 0.233, 0.126]	0.339
E	1.636	[0.083, 0.116, 0.125]	[0.194, 0.252, 0.186]	0.304
F	1.954	[0.090, 0.144, 0.049]	[0.178, 0.225, 0.124]	0.238

When terminate:

Unknown Dataset	σ_n	$(\varepsilon_{max})_n$	$\bar{\varepsilon}_n$
A	0.029	0.835	0.282
B	0.037	1.065	0.365
C	0.028	0.791	0.278
D	0.032	0.870	0.319
E	0.021	0.984	0.331
F	0.016	0.789	0.264

Comparation the $\sigma_n, (\varepsilon_{max})_n, \bar{\varepsilon}_n$ at the beginning and termination

Unknown Dataset	σ_n *at beginning*	σ_n *at termination*	Reduced σ_n	$(\varepsilon_{max})_n$ *at beginning*	$(\varepsilon_{max})_n$ *at termination*	*Reduced* $(\varepsilon_{max})_n$	$\bar{\varepsilon}_n$ *at beginning*	$\bar{\varepsilon}_n$ *at termination*	Reduced $\bar{\varepsilon}_n$
A	0.243	0.029	0.214	6.558	0.835	5.723	2.540	0.282	2.258
B	0.112	0.037	0.075	2.677	1.065	1.612	0.930	0.365	0.565
C	0.175	0.028	0.147	4.858	0.791	4.067	1.727	0.278	1.449
D	0.156	0.032	0.124	4.306	0.870	3.436	1.478	0.319	1.159
E	0.064	0.021	0.043	2.600	0.984	1.616	0.876	0.331	0.545
F	0.123	0.016	0.107	5.827	0.789	5.038	2.001	0.264	1.737

Discussion of the result:

As presented in the table above, the average error between our output with the output file provided by professor are small. Since the mode 1-6 represent small deform. The average error between our weight result and the weight output from professor are reasonable. The average error in $\vec{s}, \vec{c}, and\ \left\|\overrightarrow{s_k} - \overrightarrow{c_k}\right\|$ respresent our algorithm can not perform as well as professor's, but the error are small enough to show that our deformable registration gives acceptable and reasonable result. By comparing the $\sigma_n, (\varepsilon_{max})_n, \bar{\varepsilon}_n$ at the beginning and termination, all those values reduced a lot, which can proves that our deformable registration works well. Thus, the table could prove that our deformable registration works properly and give the correct result.

A tabular summary of the results obtained for unknown data

Saturday, November 13, 2021 7:17 PM

Results obtained for unknown data

Unknown Dataset	$\{\lambda_1, \cdots, \lambda_{Nmodes}\}$	Average $\left\|\overrightarrow{s_k} - \overrightarrow{c_k}\right\|$
G	{-3.5332, -44.5134, -98.5065, 142.1668, -55.9859, 38.5880}	0.303
H	{-37.4043, 82.9882, -11.3605, -113.6581, 106.2856, -134.1986}	0.324
J	{51.7074, -75.5610, -96.0240, -94.6479, 105.0154, -49.8621}	0.294
K	{133.4928, -143.6449, 65.4409, -29.5374, -7.0692, -12.8455}	0.341

When terminate:

Unknown Dataset	σ_n	$(\varepsilon_{max})_n$	$\bar{\varepsilon}_n$
G	0.026	0.703	0.262
H	0.019	0.881	0.295
J	0.017	0.809	0.271
K	0.032	0.887	0.314

Discussion of result:

According to the validation and result proved with debug files, we believe that the result for the unknown dataset should be correct. As shown in previous sections, the average $\left\|\overrightarrow{s_k} - \overrightarrow{c_k}\right\|$, $\sigma_n, (\varepsilon_{max})_n, \bar{\epsilon}_n$ are all small and are in same range with those from debug dataset , which indicates that our deformable registration works properly. Thus, we believe that our result for unknown dataset should be correct.