

## Programming Assignment 5 – 601.455/655 Fall 2025

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		
Email		
Other contact information (optional)		
Name 2		
Email		
Other contact information (optional)		
Signature (required)	I (we) have followed the rules in completing this assignment  _____	
<b>Grade Factor</b>		
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#4 and HW#5. 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#4-5) with this score

# Instructions

## Instructions

- You may work alone or in groups of 2 people. I am opposed to groups of three or more.
  - Contact the TA if you are having trouble finding a partner.
  - In the past, a team consisting of one person who is a bit stronger in math with one who is a bit stronger
  - If you work with a partner, only one of you should do the uploading, but both names should be on the cover sheet, so we know who is working together.
- Except in extraordinary circumstances, both members of the group will receive the same grade, but your report should indicate who did what.
- You may use C, C++, Python, or MATLAB for the assignment, or some other language with permission of the TA.
- You may use standard numerical libraries and packages for least-squares problems and Cartesian coordinate transformations.
  - One such library is that associated with Numerical Recipes in C. There is also a numerical library I wrote for the ERC. There is a large collection of C++ libraries for medical robotics and computer-

integrated surgery, described at <https://github.com/jhu-cisst/cisst/wiki>. The code is at <https://github.com/jhu-cisst/cisst>. These libraries are supported by engineering staff in LCSR.

- Generally, you are free to use such external libraries or packages, but you **MUST** provide citations for any libraries or packages that you use.
- If you do use software from our web site, remember that it is copyrighted and part of the CIS ERC infrastructure. It is mostly open source, but respect any rules and copyright for this software and any other software you use.
- You should **not** use software that essentially solves the programming problem. E.g., you can use packages for Cartesian transformation, numerical solvers, and the like, but do not use canned subroutines for solving point-cloud-to-point-cloud registration transformations or for performing calibrations. When in doubt, talk to the TA or instructor.
- You must cite the source of any software that you do not write yourselves.
- You may not use the results of previous years' programming assignments or the work of other students **or the results of any AI code generation systems** on this year's assignments.

## Abstract of the Problems

- In these problems, you will be asked to extend the ICP program of assignments 3 and 4 to perform a simple deformable registration. The input files are similar to the earlier exercises. They include

1. A 3D surface, represented as a mesh of triangles. You will be given the coordinates of the vertices of this mesh in CT coordinates. For the purposes of this exercise, you can think of this object as being a bone. In this case, the surface will represent the “average” shape of the bone.
2. An atlas “modes” file, giving modes of variation for the model. “Mode 0” represents the average shape (i.e., the shape provided in the mesh file). If  $\vec{\mathbf{m}}_{m,k}$  represents a 3D value associated with vertex  $k$  and mode  $m$ , then the actual coordinate of vertex  $k$  will be given by

$$\vec{\mathbf{m}}_k = \vec{\mathbf{m}}_{0,k} + \sum_{m=1}^{N_{\text{modes}}} \lambda_m \vec{\mathbf{m}}_{m,k}$$

3. A pair of definition files for two rigid bodies, “A” and “B”. Each file gives the positions of LED markers  $\vec{\mathbf{A}}_i$  and  $\vec{\mathbf{B}}_i$  in body coordinates, together with the positions of two tips  $\vec{\mathbf{A}}_{tip}$  and  $\vec{\mathbf{B}}_{tip}$  in

body coordinates. As shown in the illustration below, we assume that the “tip” of the B rigid body is rigidly screwed into the bone in some unknown orientation. The A rigid body is used as a pointer. I.e., its tip is placed into contact with a number of points on the surface of the bone.

4. A file of “sample” readings giving the positions and  $\vec{\mathbf{b}}_{i,k}$  of the LED markers relative to an optical tracker when each sample  $k$  is taken.

- You are to output the CT coordinates  $\vec{\mathbf{c}}_k$  corresponding to each sample taken, for the “deformed” atlas, together with the values for the  $\lambda_m$ .
- Note that this assignment is based loosely on the “active appearances” method of Cootes and Taylor and of other authors:
  1. T. F. Cootes and C. J. Taylor, "Combining Elastic and Statistical Models of Appearance Variation," in Proc. European Conference on Computer Vision, vol. 1, 2000, pp. 149-163.
  2. T. F. Cootes and C. J. Taylor, "Statistical Models of Appearance for Computer Vision," 2000.
  3. T. F. Cootes, C. Beeston, G. J. Edwards, and C. J. Taylor, "A Unified Framework for Atlas Matching using Active Appearance Models," in IPMI: Springer, 1999, pp. 322-333.

4. J. Yao and R. H. Taylor, "Non-Rigid Registration and Correspondence in Medical Image Analysis Using Multiple-Layer Flexible Mesh Template Matching," *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, vol. 17(7), pp. in press, 2003.
5. J. Yao and R. H. Taylor, "A Multiple-Layer Flexible Mesh Template Matching Method for Non-rigid Registration between a Pelvis Model and CT Images," in *SPIE Medical Imaging*. San Diego, 2003, pp. 1117-1124.
6. J. Yao and R. H. Taylor, "Deformable registration between a statistical bone density atlas and X-ray images," in *Second International Conference on Computer Assisted Orthopaedic Surgery (CAOS 2002)*. Santa Fe: CAOS International, 2002.
7. M. Fleute and S. Lavalley, "Nonrigid 3-D/2-D Registration of Images Using Statistical Models," in *MICCAI 99, Springer Lecture Notes in Computer Science*. Cambridge, UK,, 1999, pp. 138-147.

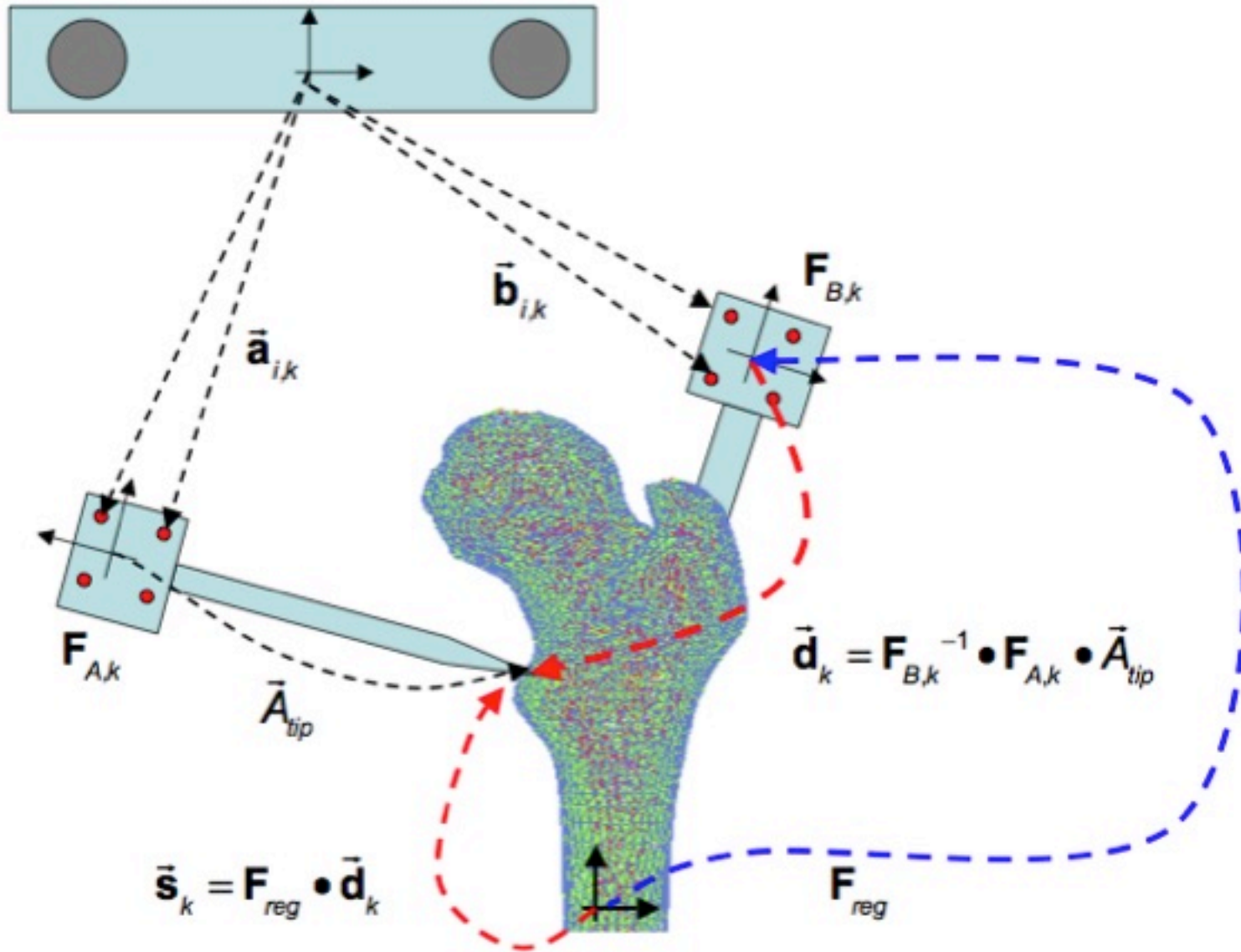


Figure 1: Scenario

## Hints on suggested procedure

- Start with your program for Programming Assignment #4.
- Input the data and verify that the vertices of the mesh you have read in are the same as “Mode 0”.
- Perform an initial “rigid” registration using the method for Programming Assignment #4. This will produce a rigid transformation  $\mathbf{F}_{reg}^{(0)}$
- Let  $\vec{\mathbf{s}}_k^{(t)} = \mathbf{F}_{reg}^{(t)} \bullet \vec{\mathbf{d}}_k$  be your current estimate of transformed sample point  $k$  at iteration  $t$ , where  $\mathbf{F}_{reg}^{(t)}$  is the current estimate of the rigid transformation, and  $\vec{\mathbf{d}}_k = \mathbf{F}_{B,k}^{-1} \bullet \mathbf{F}_{A,k} \bullet \vec{\mathbf{A}}_{tip}$  represents the corresponding measured sample point value.
- Suppose  $\vec{\mathbf{c}}_k^{(t)}$  represents your current estimate of the closest point on the deformed surface to the transformed sample point. Then  $\vec{\mathbf{c}}_k^{(t)}$  will be on some triangle. Suppose the vertex indices of this triangle are  $\{s, t, u\}$ . Then the coordinates of the corresponding deformed mesh will be

$$\begin{aligned}
\vec{\mathbf{m}}_s &= \vec{\mathbf{m}}_{0,s} + \sum_{m=1}^{N_{\text{modes}}} \lambda_m^{(t)} \vec{\mathbf{m}}_{m,s} \\
\vec{\mathbf{m}}_t &= \vec{\mathbf{m}}_{0,t} + \sum_{m=1}^{N_{\text{modes}}} \lambda_m^{(t)} \vec{\mathbf{m}}_{m,t} \\
\vec{\mathbf{m}}_u &= \vec{\mathbf{m}}_{0,u} + \sum_{m=1}^{N_{\text{modes}}} \lambda_m^{(t)} \vec{\mathbf{m}}_{m,u}
\end{aligned} \tag{1}$$

- If we compute the barycentric coordinates of  $\vec{\mathbf{c}}_k^{(t)}$  on the triangle

$$\vec{\mathbf{c}}_k^{(t)} = \zeta_k \vec{\mathbf{m}}_s + \xi_k \vec{\mathbf{m}}_t + \psi_k \vec{\mathbf{m}}_u$$

we can get an expression in terms of mode coordinates  $\vec{\mathbf{c}}_k^{(t)}$

$$\vec{\mathbf{c}}_k^{(t)} = \vec{\mathbf{q}}_{0,k} + \sum_{m=1}^{N_{\text{modes}}} \lambda_m^{(t)} \vec{\mathbf{q}}_{m,k} \tag{2}$$

where

$$\vec{\mathbf{q}}_{m,k} = \zeta_k \vec{\mathbf{m}}_{m,s} + \xi_k \vec{\mathbf{m}}_{m,t} + \psi_k \vec{\mathbf{m}}_{m,u}$$

and  $\Lambda^{(t)} = \left\{ \lambda_1, \dots, \lambda_{N_{\text{modes}}} \right\}^{(t)}$  represents the current estimate of the mode weights. So

$$\begin{aligned}\vec{\mathbf{s}}_k^{(t)} &= \mathbf{F}_{reg}^{(t)} \bullet \vec{\mathbf{d}}_k \\ \vec{\mathbf{c}}_k^{(t)} &= \vec{\mathbf{q}}_{0,k} + \sum_{m=1}^{N_{modes}} \lambda_m^{(t)} \vec{\mathbf{q}}_{m,k}\end{aligned}\tag{3}$$

should represent the same point. I.e.,

$$\mathbf{F}_{reg}^{(t)} \bullet \vec{\mathbf{d}}_k \approx \vec{\mathbf{q}}_{0,k} + \sum_{m=1}^{N_{modes}} \lambda_m^{(t)} \vec{\mathbf{q}}_{m,k}\tag{4}$$

**Note:** In the above, remember that  $s, t, u$  also depend on  $\vec{\mathbf{c}}_k$ .

- Now, you have a few choices. One choice is to iterate the following sequence:
  - Iterate the following sequence some number of times until it seems to be stalled:
    - Keeping  $\vec{\mathbf{s}}_k^{(t)}$  fixed, find the corresponding  $\vec{\mathbf{q}}_{m,k}$  solve the following least squares problem for  $\Lambda^{(t+1)} = [\lambda_1^{(t+1)}, \dots, \lambda_{N_{modes}}^{(t+1)}]$

$$\vec{\mathbf{s}}_k^{(t)} \approx \vec{\mathbf{q}}_{0,k} + \sum_{m=1}^{N_{modes}} \lambda_m^{(t+1)} \vec{\mathbf{q}}_{m,k}\tag{5}$$

- Use  $\Lambda^{(t+1)}$  to update the surface mesh model to find a new estimate for the vertices of the deformed model.
- If you are using a spatial data structure to speed your search, make the appropriate updates. E.g., if you are using some form of bounding box tree, update the bounds. Note that it is not always necessary to completely recompute the tree, since some loss of efficiency is ok.
- Now, find new matching points  $\vec{\mathbf{c}}_k^{(t+1)}$
- Iterate for a while, until you meet some suitable stopping condition
- After this iteration converges, keep the model vertices fixed and use the method of PA#4 to re-estimate  $\mathbf{F}_{reg}^{(t)}$ . Use the result to re-transform the sample points.
- Iterate this sequence (mode matching, rigid body transformation, ...) until done.
- Alternatively, you can combine the two iterations by using a linearized form of Equation (4):

$$\begin{aligned}
\mathbf{F}_{reg}^{(t+1)} \bullet \vec{\mathbf{d}}_k &= \vec{\mathbf{c}}_k^{(t)} + \sum_{m=1}^{N_{modes}} \lambda_m^{(t+1)} \vec{\mathbf{q}}_{m,k} \\
\Delta \mathbf{F}^{(t+1)} \bullet \mathbf{F}_{reg}^{(t)} \bullet \vec{\mathbf{d}}_k &= \vec{\mathbf{c}}_k^{(t)} + \sum_{m=1}^{N_{modes}} \lambda_m^{(t+1)} \vec{\mathbf{q}}_{m,k} \\
\Delta \mathbf{F}^{(t+1)} \bullet \vec{\mathbf{s}}_k^{(t)} &= \vec{\mathbf{c}}_k^{(t)} + \sum_{m=1}^{N_{modes}} \lambda_m^{(t+1)} \vec{\mathbf{q}}_{m,k} \\
\vec{\mathbf{s}}_k^{(t)} + \vec{\alpha}^{(t+1)} \times \vec{\mathbf{s}}_k^{(t)} + \vec{\varepsilon}^{(t+1)} &\approx \vec{\mathbf{c}}_k^{(t)} + \sum_{m=1}^{N_{modes}} \lambda_m^{(t+1)} \vec{\mathbf{q}}_{m,k}
\end{aligned} \tag{6}$$

where  $\Delta \mathbf{F} = [\Delta \mathbf{R}(\vec{\alpha}), \vec{\varepsilon}]$ . This rearranges to

$$\vec{\mathbf{s}}_k^{(t)} \times \vec{\alpha}^{(t+1)} - \vec{\varepsilon}^{(t+1)} + \sum_{m=1}^{N_{modes}} \lambda_m^{(t+1)} \vec{\mathbf{q}}_{m,k} \approx \vec{\mathbf{s}}_k^{(t)} - \vec{\mathbf{c}}_k^{(t)} \tag{7}$$

- In this case, you do the following:
  - Solve (7) to estimate  $\Lambda^{(t+1)}$ ,  $\vec{\alpha}^{(t+1)}$ , and  $\vec{\varepsilon}^{(t+1)}$ . Use  $\vec{\alpha}^{(t+1)}$  and  $\vec{\varepsilon}^{(t+1)}$  to produce a real value for  $\Delta \mathbf{F}^{(t+1)}$ , remembering that you need to make a proper rotation, and cannot just use a skew matrix.
  - Update  $\vec{\mathbf{s}}_k$  and the mesh using  $\Delta \mathbf{F}^{(t+1)}$  and  $\Lambda^{(t+1)}$ . Again, remember that you may have to update bounding boxes or other spatial data structures.

- Now, find new matching points  $\vec{c}_k^{(t+1)}$
- Iterate for until you meet some suitable stopping condition.
- I solved the problems both ways. You can even mix them. In solving the problem, I used the first method, followed by the second method.
- **NOTE:** Again, there are other ways to approach this problem, as well. You are free to develop an alternative algorithm so long as you explain it clearly and explain your reasoning.

## Input data

You will be given a rigid body design files “ProblemX-BodyY” where X is 3 or 4 and Y is A or B. These files have the following format

Record	Data	Comments
0	$N_{\text{markers}}$ “ProblemX-BodyY”	Number of marker LEDs Filename
Next records	$N_{\text{markers}}$ $Y_x, Y_y, Y_z$	xyz coordinates of marker LEDs in body coordinates
Next record	$t_x, t_y, t_z$	xyz coordinates of tip in body coordinates

Each line (record) of the file will terminate with an end of line character.

## The format of the body surface definition file “ProblemXMesh.sur”

<b>Record</b>	<b>Data</b>	<b>Comments</b>
0	$N_{\text{vertices}}$	Number of vertices
Next $N_{\text{vertices}}$ records	$V_x, V_y, V_z$	xyz coordinates of vertices in CT coordinates
Next record	$N_{\text{triangles}}$	Number of triangles
Next $N_{\text{triangles}}$ records	$i_1, i_2, i_3, n_1, n_2, n_3$	Vertex indices of the three vertices for each triangle, followed by triangle indices for the three neighbor triangles opposite to the three vertices (not needed for this problem). “-1” means not a valid neighbor.

The format of the atlas modes file is as follows:

<b>Record</b>	<b>Data</b>	<b>Comments</b>
0	Problem5Modes.txt Nvertices=nnnn Nmodes=mmm	nnnn = Number of vertices mmm = Number of modes
1	Mode 0 :Average Vertex Values	Just a comment
Next $N_{\text{vertices}}$ records	$V_x, V_y, V_z$	xyzcoordinates of vertices in CT coordinates of “mode 0” – I.e., the mean shape
Next record	Mode 1 :Vertex Displacement	Just a comment
Next $N_{\text{vertices}}$ records	$D_x, D_y, D_z$	Vertex displacements for mode 1
Next record	Mode 2 :Vertex Displacement	Just a comment
<i>... et cetera ...</i>	<i>... et cetera ...</i>	<i>... et cetera ...</i>

Note that the number of modes in the atlas file may exceed the number of modes to be used in any individual problem.

Finally, you will be given a file of sample readings “pa5-X-ddddd- $N_{samps}$  SampleReadings.txt”, where X is a letter and dddd is “debug” or “unknown”. This file has the following format.

Record	Data	Comments
0	$N_S = N_A + N_B + N_D, N_{samps}$ , “pa5-X-ddddd- SampleReadings.txt”, $N_{modes}$	Number of LEDs read by the tracker in each sample frame (“A” markers, “B” markers, “Dummy” markers) Number of sample frames File name, Number of modes you are to use from the atlas in solving this problem

Next $N_A$ records	$x, y, z$	xyz coordinates of A body LED markers in tracker coordinates
Next $N_B$ records	$x, y, z$	xyz coordinates of B body LED markers in tracker coordinates
Next $N_D = N_S - N_A - N_B$ records	$x, y, z$	xyz coordinates of other (unneeded) LED markers in tracker coordinates
This pattern of $N_S$ records is repeated for a total of times.	See above	Additional sets of data corresponding to each sample

# Output Data

You should produce an output data file with the format.

Record	Data	Comments
0	$N_{samps}$ , "pa5-X-Output.txt", $N_{modes}$	Number of sample frames File name
1	$\lambda_1, \dots, \lambda_{N_{modes}}$	Mode weights determined (give 4 decimal places)
Next $N_{samps}$ records	$s_x, s_y, s_z$ $c_x, c_y, c_z$ $\ \vec{s}_k - \vec{c}_k\ $	xyzcoordinates of $\vec{s}_k$ , xyzcoordinates of $\vec{c}_k$ magnitude of difference

For debugging purposes, I have provided the output my program got. I have also included an "answer" file that contains data used to generate

the test problems. In some cases the answer data will also have non-zero error due to various simulated noise.