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	
Email	
Other contact information (optional)	
Name 2	
Email	
Other contact information (optional)	
Signature (required)	I (we) have followed the rules in completing this assignment _____ _____

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	

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
- 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 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 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 at the CISST ERC, as part of our open-source "Surgical Assistant Workstation (SAW)" project.

- 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 on this year's assignments.
- Your programs (and the reports) should clearly indicate authorship of the code and also should cite any libraries that you use, including source.

- Do not go out and try to find a package that solves this particular problem. Numerical libraries are one thing, but using someone's registration or calibration code is entirely another matter.
- The TA will also post some possibly useful software from the ERC on our web site. You do not have to use this software, but may find it helpful.
- Note that these rules are intended to supplement the general instructions discussed at the opening session of the class, not to replace them. If you have questions, see me or the TA.
- You should hand in a report containing the following:
  - A narrative report summarizing
    - The mathematical approach taken
    - The algorithmic steps followed
    - 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.
    - 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, together with a

discussion of unit tests, creation of your own debugging data, etc.

- A tabular summary of the results obtained for unknown data
- 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.
- A short statement of who did what.

- A well structured and well documented program listing

- You will also upload a directory containing
  - A directory called “PROGRAMS”, containing (at least) all source files for your program, together with a file “README.TXT”, containing the names of all the source files, together with a 1-line description of each file. Optionally, you may also include an executable program and instructions for using it.
    - **Note that the graders will attempt to compile and run your program.**

- Another directory, called “OUTPUT”, containing the program output files in the specified format and with the specified name (see below).

- The TAs will specify how and where these things are to be uploaded.

Abstract of the Problems

- In these problems, you will be asked to implement and use a simplified version of iterative-closest point registration. You will be given:
    - 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.
    - A pair of definition files for two rigid bodies, "A" and "B". Each file gives the positions of LED markers $\vec{A}_i$ and $\vec{B}_i$ in body coordinates, together with the positions of two tips $\vec{A}_{tip}$ and $\vec{B}_{tip}$ 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.
    - A file of "sample" readings giving the positions $\vec{\mathbf{a}}_{i,k}$ and $\vec{\mathbf{b}}_{i,k}$ of the LED markers relative to an optical tracker when each sample *k* is taken.

- For PA #3 you will need to implement the matching part of the ICP algorithm.
- For assignment for PA#4 you will add an iteration to implement a complete ICP algorithm.
- In each case, you are to output the CT coordinates $\vec{\mathbf{c}}_k$ corresponding to each sample taken.

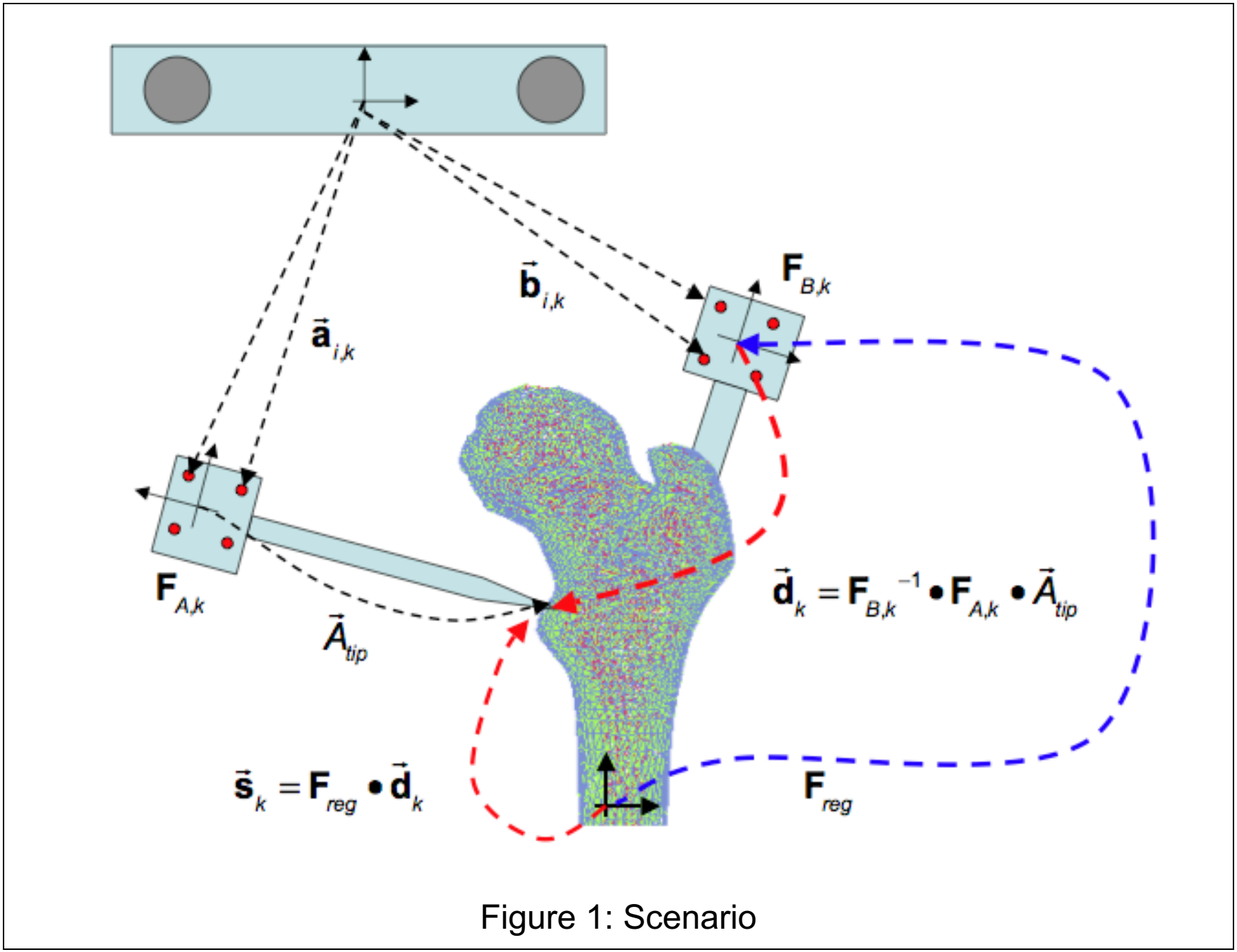

Figure 1: Scenario

Hints on suggested procedure

- Input the surface mesh into an appropriate data structure. For the current problem, this will likely be two arrays, one to hold the vertex coordinates for all of the triangles and another to hold the indices of the vertices for each triangle.
- Implement a "find closest point on triangle" subroutine and rigorously debug it. You can follow the lecture notes in doing this.
- Implement a simple, but possibly slow, "find closest point on mesh" routine that works by linear search through all the triangles.
- After you do this, you can implement a more efficient search using a tree or other clever data structure and can also implement other speedups. Validate any clever methods against the simple brute force search.
- To solve the problem, your first step is to process the sample points input file. Roughly, you need to do the following:
  - Input the body definition files.
  - For each sample frame *k,* input the values of $\vec{\mathbf{a}}_{i,k}$ and $\vec{\mathbf{b}}_{i,k}$.
  - Use your point cloud-to-point cloud registration subroutine to determine the poses $\mathbf{F}_{A,k}$ and $\mathbf{F}_{B,k}$ of the rigid bodies with respect to

the tracker. Then the position $\vec{\mathbf{d}}_k$ of the pointer tip with respect to rigid body B (which moves with the bone) will be $\vec{\mathbf{d}}_k = \mathbf{F}_{B,k}{}^{-1} \bullet \mathbf{F}_{A,k} \bullet \vec{A}_{tip}$.

- Now assume that $\mathbf{F}_{\text{reg}}$ is an unknown transformation such that $\vec{\mathbf{c}}_k = \mathbf{F}_{\text{reg}} \bullet \vec{\mathbf{d}}_k$. For Problem 3, you can assume that $\mathbf{F}_{\text{reg}} = \mathbf{I}$, and for Problem 4 you can use this as an initial guess. Compute sample points $\vec{\mathbf{s}}_k = \mathbf{F}_{\text{reg}} \bullet \vec{\mathbf{d}}_k$. Now find the points $\vec{\mathbf{c}}_k$ on the surface mesh that are closest to the $\vec{\mathbf{s}}_k$. For Problem 3 you can output these values. For Problem 4, you need to use these points to make a new estimate of $\mathbf{F}_{\text{reg}}$ and iterate until done.
- **NOTE** that the "suggested procedure" is just that. You are free (even encouraged) to develop your own method, if you think it may be better. But you should clearly document what you do and why you do it.

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_{markers} “ProblemX-BodyY”	Number of marker LEDs Filename
Next N_{markers} records	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”

Record	**Data**	**Comments**
0	N_{vertices}	Number of vertices
Next N_{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.

Finally, you will be given a file of sample readings “paV-X-ddddd-SampleReadings.txt”, where V is 3 or 4, X is a letter, and ddddd is “debug” or “unknown”. This file has the following format.

Record	**Data**	**Comments**
0	$N_S = N_A + N_B + N_D, N_{samps}$, “paV-X-ddddd-SampleReadings.txt””	Number of LEDs read by the tracker in each sample frame (“A” markers, “B” markers, “Dummy” markers) Number of sample frames File name
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	x, y, z	*xyz*coordinates of

$N_D = N_S - N_A - N_B$ records		other (unneeded) LED markers in tracker coordinates
This pattern of N_s records is repeated for a total of N_{samps} times.	See above	Additional sets of data corresponding to each sample

Output Data

You should produce an output data file with the format.

For assignment 3 the format is

Record	Data	Comments
0	N_{samps},"pa3-X-Output.txt""	Number of sample frames File name
Next N_{samps} records	$d_x, d_y, d_z \quad c_x, c_y, c_z \quad \left\|\vec{\mathbf{d}}_k - \vec{\mathbf{c}}_k\right\|$	*xyz*coordinates of $\vec{\mathbf{d}}_k$, *xyz*coordinates of $\vec{\mathbf{c}}_k$ magnitude of difference

For assignment 4, the format is.

Record	Data	Comments
0	N_{samps},"pa4-X-Output.txt""	Number of sample frames File name
Next N_{samps}	$s_x, s_y, s_z \quad c_x, c_y, c_z \quad \left\|\vec{\mathbf{s}}_k - \vec{\mathbf{c}}_k\right\|$	*xyz*coordinates of $\vec{\mathbf{s}}_k$,

records		*xyz*coordinates of $\vec{\mathbf{c}}_k$ 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 differ slightly from the output file data due to simulated noise.