Programming Assignments 1 & 2 601.455 and 601/655 Fall 2021

Please also indicate which section(s) you are in (one of each is OK)

Score Sheet

Name 1	Suryansh Shukla
Email	sshukla8@jhu.edu
Other contact information (optional)	
Name 2	
Email	
Other contact information (optional)	
Signature (required)	I (we) have followed the rules in completing this assignment suryansh _____

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	

Report CIS

Mathematical Approach

1. Calculation of d_k

$\vec{d_k} = F_{B,k}^{-1} \bullet F_{A,k} \bullet \vec{A_{tip}}$

$F_{A,k} \bullet \vec{A_{tip}} = R_{A,k} \bullet \vec{A_{tip}} + P_{A,k}$

$F_{B,k}^{-1} = [R_{B,k}^T, -R_{B,k}^T \bullet P_{B,k}]$

Algorithmic Approach

Algorithm for find closest point on a triangle

To get the closest point of the face of triangle I first get the vertex of the frace, the normal of the face and store them in local variable. I get the vector that joints the query point to the first vertex of the face and compute the component of that vector to normal of the face. That gives us the projection vector that projects the query point on the plane of the triangle.

If the length of this projection vector is longer than the best guess that we have so far, that means the closest point doesn't lie on that triangle. It means that the point on triangle is too far away from the query point. At this point we can break.

But if the distance to the plane is not greater than the best given so far we project the query point on to the plane of triangle and loop over the edge of the triangle.

For each edge the projected point might be on eigher the outer side of the edge or inner side of the edge. So to find out which side of the edge the projected point is are we compute the normal of the triangle formed by the edge and the projected point and calculate the dot product of the normal with the face normal. So if the point is in the inner side of the edge the two vectors the two normals would point the same way.

But if the point is on the outer side of the edge the two normals wll point in opposite direction so the dot product will end up being negative. So we store the result of the test

in a boolean variable name outside.

And if the value of that boolean varialbe is true we update the counter we created before to keep track of how many edges we are having to process that is how many edges have the projected point on their outer side.

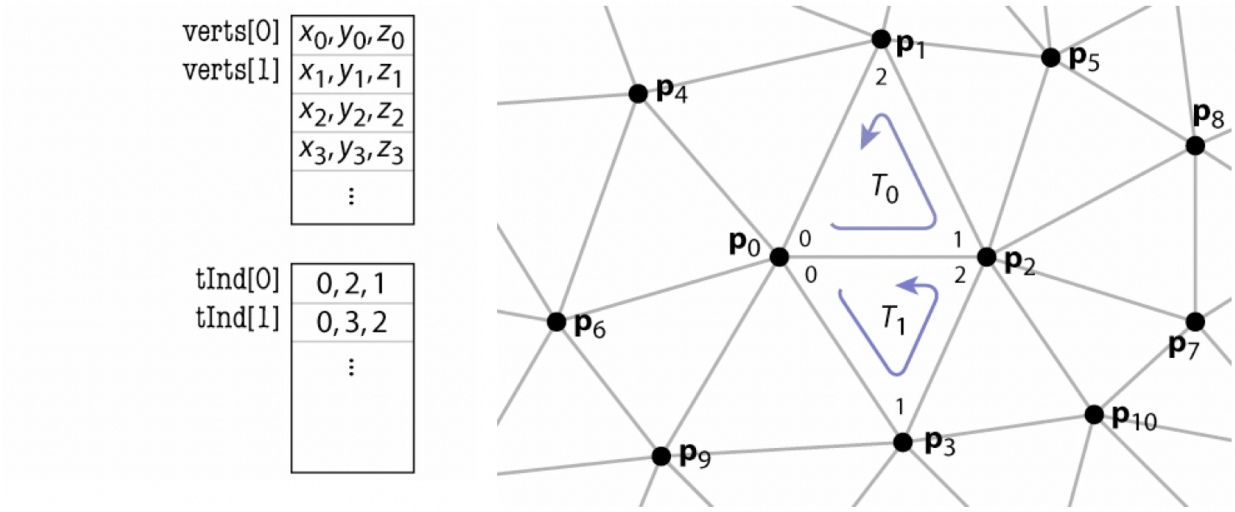
We then compute the closest point between the edge of the triangle and the projected point by pulling the point onto the edge and clamping it to be between the two vertices.

If this closest point is the closer than the best guess we have so far we update out best guess and after processing each edge like this if we realize that our outside edge counter is great than one, then we can exit(). And we don't have to processs all the edges this is because a point in the plane of a triangle can only be on the outside of atmost two edges, it can't be on the outer side of all three edges.

And finally after we are done looping over the edges if it turns out that the projected point was not on the outer side of any of the edge ie. our counter is still set to zero, that means projected point is inside the triangle which means the closest point is the projected point itself.

1. Body Definition File

Indexed triangle set



To load the surface triangle mesh I created two separate array. One array contains the vertex and the second array contained the indices of each of the vertices for each triangle.

Each of the indices array represent one triangle. By taking the number represented by each indice array I found the corresponding index in the vertice and constructed V1, V2, V3 which representes the vertex of the coordinate. V1,V2,V3 are each 1x3 vector containing the xyz coordinate of each vertex.

I didn't used the neighbourhood relation for the vertex of the triangle.

Steps for C_k

We assumed

$$F_{reg} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I$$

therefore $\vec{c_k} = F_{reg} \bullet \vec{d_k}$ is equal to $\vec{c_k} = \vec{d_k} = \vec{s_k}$

That's why we get the output $||\vec{d_k} - \vec{c_k}|| = 0$

Overview of program

Modules and Function

1. **pa1Function.py**
    a. def points_registeration_rotation(A, B):
    b. def points_registeration(A, B):
2. **pointtriangleDistance2.py**
    a. def centroid(V1, V2, V3):
    b. def pointTriangleDistance(TRI, P):
3. **readers.py**
4. **triangleMesh.py**
    a. def getTriVertice(verts, tInd, i):
5. **writers.py**
    a. PA3(Writer)

The first step in our program was to load the vertices into local variable **Vertices** which is a numpy array. And consequently we load the indices of the triangle matrix into **Indices** local variable. It's also a numpy array.

After that we load the "Problem3-RigidBodyA.txt" file into local variable RigidBodyA. At this point RigidBodyA contains both the xyzcoordinates of marker LED in the ad xyz coordinate of tip. This data is further used for the point cloud registration.

Similarly I load the "Problem3-RigidBodyB.txt" file into local variable RigidBodyB, containing both the xyz coordinate of the marker LED and xyz coordinates of tip.

Then we load the sampleRedading.txt file. We created a dictionary for NA record and NB records. This dictionary contains 15 frame data. So each key in the dictionary is

having 6 led coordinates. This is also used in point cloud registration.

At this point I used the cloud-to-point cloud registration, which is performed by using the function **points_registration** this function is present in PA1.py module.

Using the maths explained above we find the **F_Ak and F_Bk,** and then calculate the **d_k, s_k.**

I then used the triangle mesh to find the closest point on the triangle surface. The closest point is represented by **closest_point** variable. To compute the closest point, I used a for loop iteration #3135 (total number of triangles) then using **findTriangleVertices.getTriangleVertice** I get the three vertices, stored in **V**. To finding the closest point on the triangle we used the function **pointTriangleDistance** this gives the output the distance and the closest point on the triangle with input coordinates of c. I run over all the triangle mesh and stored the distance from the point and the closest point on the triangle mesh surface. Then find the minimum of all these distance and that was the closest point chosen.

Results for unknown data

PA3-G-Unknown-Output.txt

PA3-G-Unknown-Output.txt

dx	dy	dz	cx	cy	cz	‖dk-ck‖
-12.965	-6.915	-47.734	-12.965	-6.915	-47.734	0.000
-22.598	-10.716	-49.743	-22.598	-10.716	-49.743	0.000
-14.793	-13.124	-48.546	-14.793	-13.124	-48.546	0.000
-16.630	-1.900	-47.324	-16.630	-1.900	-47.324	0.000
24.551	11.926	-31.016	24.551	-11.926	31.016	0.000
-23.982	-0.238	-46.865	-23.982	-0.238	-46.865	0.000
-13.069	-2.470	-46.644	-13.069	-2.470	-46.644	0.000
-12.977	-6.091	-47.706	-12.977	-6.091	-47.706	0.000
-20.022	-5.170	-48.656	-20.022	-5.170	-48.656	0.000
-3.983	-5.417	-39.923	-3.983	-5.417	-39.923	0.000
-9.494	-9.525	-46.753	-9.494	-9.525	-46.753	0.000
-23.783	-9.690	-49.930	-23.783	-9.690	-49.930	0.000
-23.783	-9.690	-49.930	-23.783	-9.690	-49.930	0.000
-23.783	-9.690	-49.930	-23.783	-9.690	-49.930	0.000
-21.125	-10.361	-49.626	-21.125	-10.361	-49.626	0.000

PA3-H-Unknown-Output.txt

PA3-H-Unknown-Output.txt

dx	dy	dz	cx	cy	cz	‖dk-ck‖
-12.977	-6.091	-47.706	-12.977	-6.091	-47.706	0.000
-10.121	-3.134	-44.609	-10.121	-3.134	-44.609	0.000
27.651	2.384	-30.494	27.651	2.384	-30.494	0.000
-5.383	-6.112	-42.017	-5.383	-6.112	-42.017	0.000
24.551	11.926	-31.016	24.551	11.926	-31.016	0.000
-20.022	-5.170	-48.656	-20.022	-5.170	-48.656	0.000
-9.494	-9.525	-46.753	-9.494	-9.525	-46.753	0.000
-12.977	-6.091	-47.706	-12.977	-6.091	-47.706	0.000
-23.783	-9.690	-49.930	-23.783	-9.690	-49.930	0.000
-23.783	-9.690	-49.930	-23.783	-9.690	-49.930	0.000
-21.889	-11.330	-49.630	-21.889	-11.330	-49.630	0.000
-20.397	-12.622	-49.395	-20.397	-12.622	-49.395	0.000
19.645	16.185	-31.222	19.645	16.185	-31.222	0.000
-23.783	-9.690	-49.930	-23.783	-9.690	-49.930	0.000
24.551	11.926	-31.016	24.551	11.926	-31.016	0.000

PA3-J-Unknown-Output.txt

PA3-J-Unknown--Output.txt

dx	dy	dz	cx	cy	cz	‖dk-ck‖
-23.783	-9.690	-49.930	-23.783	-9.690	-49.930	0.000
-23.783	-9.690	-49.930	-23.783	-9.690	-49.930	0.000
-14.767	-13.202	-48.545	-14.767	-13.202	-48.545	0.000
-10.089	-7.311	-46.749	-10.089	-7.311	-46.749	0.000
27.663	8.859	-30.323	27.663	8.859	-30.323	0.000
-6.082	-12.733	-43.349	-6.082	-12.733	-43.349	0.000
-9.645	-9.523	-46.800	-9.645	-9.523	-46.800	0.000
-23.783	-9.690	-49.930	-23.783	-9.690	-49.930	0.000
-23.783	-9.690	-49.930	-23.783	-9.690	-49.930	0.000
-20.397	-12.622	-49.395	-20.397	-12.622	-49.395	0.000
-12.977	-6.091	-47.706	-12.977	-6.091	-47.706	0.000
-18.956	-19.265	-47.179	-18.956	-19.265	-47.179	0.000
15.530	19.061	-31.255	15.530	19.061	-31.255	0.000
-23.783	-9.690	-49.930	-23.783	-9.690	-49.930	0.000
-15.494	-2.082	-47.107	-15.494	-2.082	-47.107	0.000

Discussion of Result

The discrepency between my result and the answer provided may be due some bug in finding the closest point on the triangle.

In the next assignment, where I will be implementing the ICP, this bug can be fixed. The algorithm is working fine for the closest point but maybe the error arised due to some input of the triangle vertices.

Statement of who did what?

I myself wrote the whole program.