CIS PROGRAMMING ASSIGNMENT 3

REPORT

Work Done By:

1. Samer Aslan (saslan1@jh.edu)
2. Ajay Gawade (agawade1@jh.edu)

1) Overview of Program Structure

The purpose of this assignment was to find the closest point step of an iterative-closest point registration algorithm. The problem involved a 3D triangular surface mesh of a bone found in CT coordinates and two rigid bodies (LED markers).

One rigid body is rigidly attached to the bone and one to be used as a pointer. LED markers were attached to the two rigid bodies so that the coordinates could be determined in optical coordinates. The closest point on the triangular mesh to several points where the tip of the pointer contacted the bone was found using our find closest point algorithm.

First, each point of tracker data was parsed into Eigenvectors of ((x,y,z) coordinates which corresponded to the position of the trackers attached to the rigid bodies, A and B, in optical coordinates. Next, another set of tracker data was parsed into Eigen vectors of (x,y,z) coordinates which corresponded to the position of the trackers attached to the rigid bodies in their body coordinates. The transformation matrix from the body frame to the optical tracker frame was then computed using the Horn Registration function described above. Then the coordinates tip of the rigid body A with respect to rigid body B was found by multiplying the vector of the tip in body A coordinates by the transformations previously found.

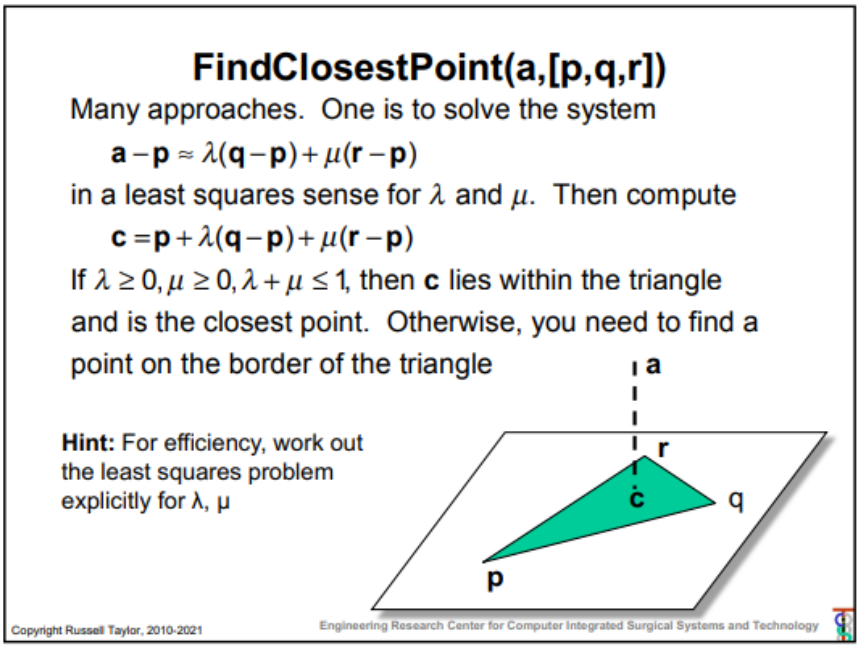
The mesh data was parsed to get the vertices of each known triangle. Then ICP registration could be used to find the point on the mesh that was closest to the tip of rigid body A. First, the transformation from the CT mesh coordinates to the rigid body B coordinates was assumed to be the identity matrix. Then, sample points were found by multiplying the transformation from CT mesh coordinates to rigid body B coordinates by the tip of the pointer A in rigid body coordinates. Now, these sample points were used where points on the CT mesh were closest to the given transformation.

The **find_closest_point** function was implemented in which the nearest point to the sample points on the CT mesh was calculated for every triangle in the mesh. The error between the two points for each triangle was calculated by taking the **2-point norm** (which is used to find the Euclidean distance)) between the points and the smallest error corresponded to the nearest point on the mesh to the pointer tip A.

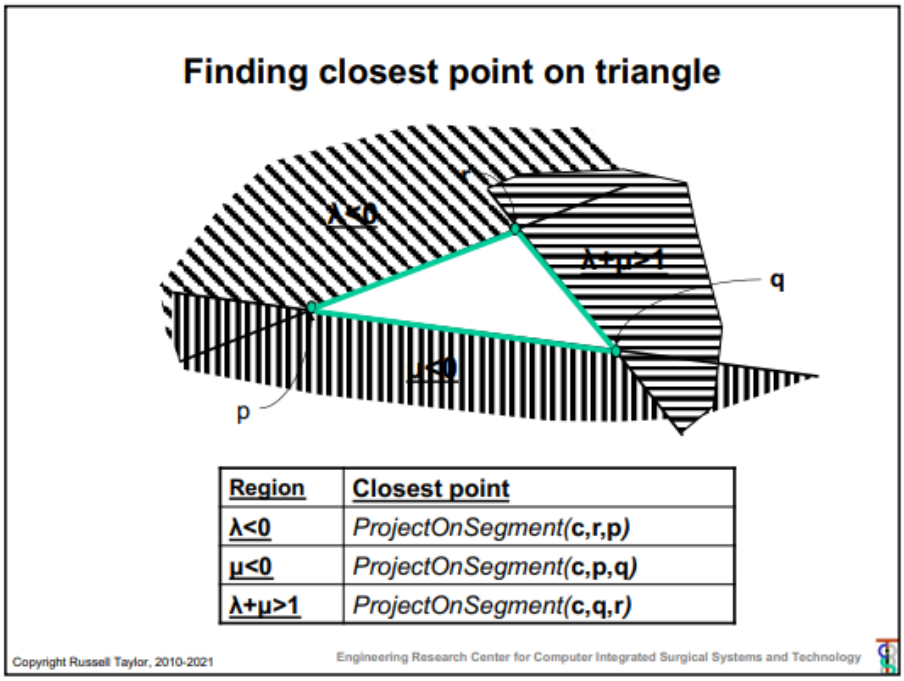
The driver function calls all the helper functions. For every file A-J, it first reads in all the data into the necessary structures. Then, for every frame k, it does the necessary frame transformations to get s_k, and then finds the closest points to the mesh by calling find_closest_point_mesh_linear. The mesh linear function calls the find_closest_point function for each triangle. Finally, this is all outputted into the corresponding file.

2) Mathematical and Algorithmic Approach:

Find nearest Point:



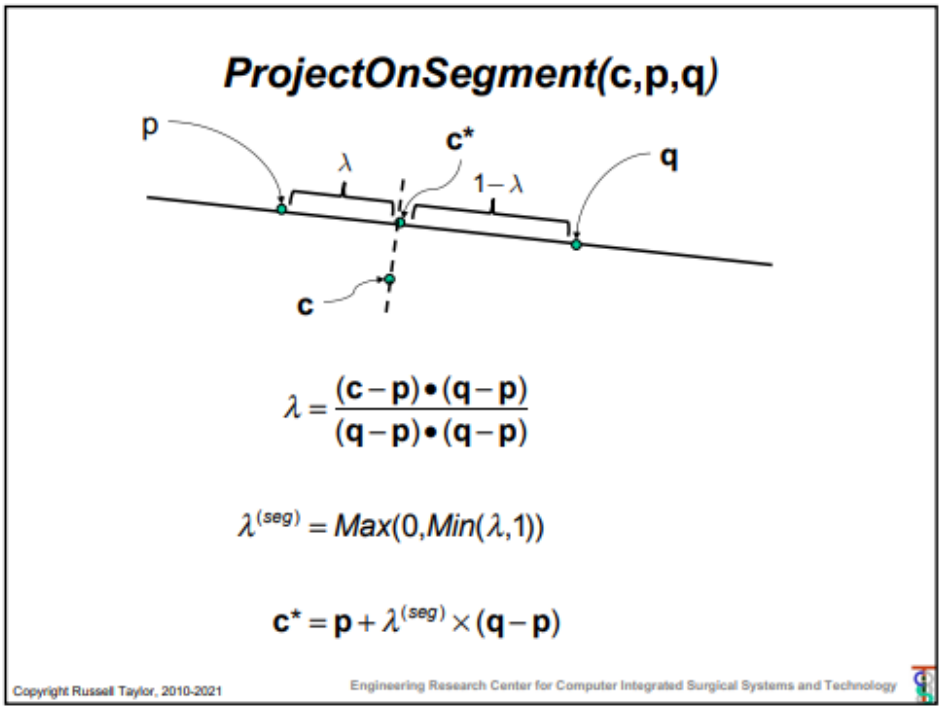
11



Finding Shortest (Euclidean Distance) Distance:

Using 2-point norm to find the Euclidean Distance

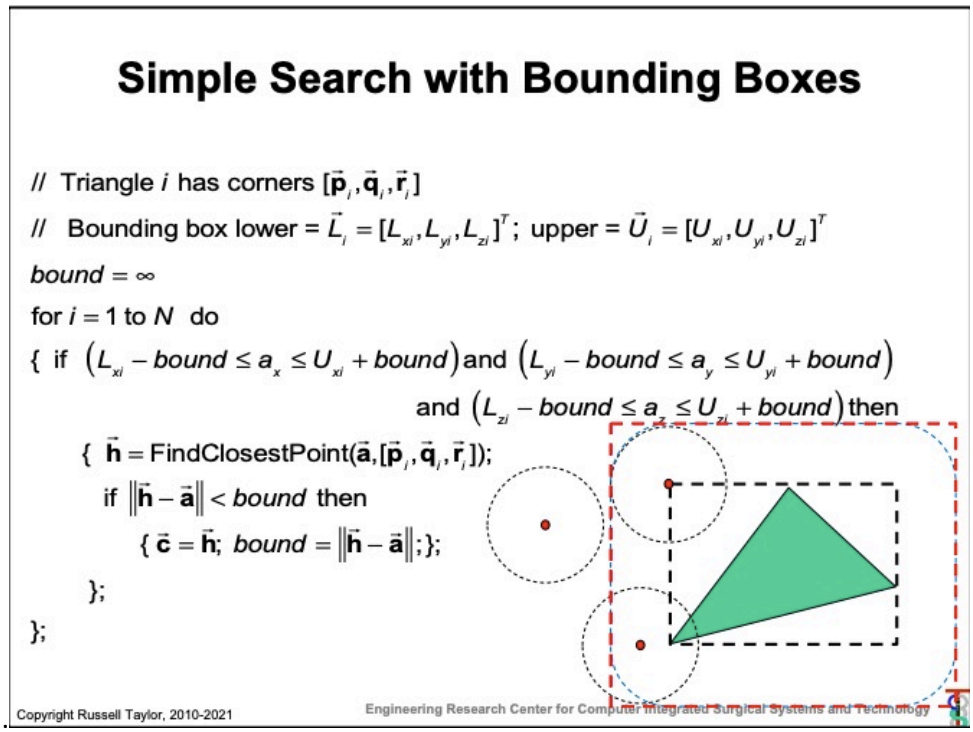
To find the point on the line joining the Euclidean Distance between 2 points.



(

Taking reference from the slides, we used a similar approach to find the nearest point in the triangle. We solved the system in the first slide using the simple least squares method and then checked if certain bounds of mu and lambda were satistfied. If so, we could take the point c to be the closest point and inside the triangle. Otherwise, we need to look at the triangle edges, which we used the second slide above as well as the slide below for determining.

Search For Bounding Box:



We followed the above slide's simple search algorithm to find the closest point to point a. After implementing this, for some reason our output was quite different from the answer output. We then tweaked the algorithm by removing bounding box lower and upper instantiation and initial if statement check altogether. Although this is less efficient, it allowed us to have much more accuracy, and we will focus on efficiency in the next program assignment.

Registration Frame:

Frame registration is an output of a Transformation function of one frame to another frame. To register a Frame, we need 2 co-ordinate systems and a Transformation function (2D-3D rigid transformation).

We have used Rotation and Translation Matrix for the Frame Composition.

The transformation function is to compute the registration frame transformation from the coordinates of the mesh triangles and Rigid bodies.

We are using point cloud registration for rigid transformation of 3D point-to-surface model.

3) Results:

Debugging:

It is worth mentioning that throughout the algorithmic development process, we used unit tests to validate whether the algorithms were working as intended. This was done thoroughly on both find closest point functions, as they are the core of our project. For example, we gave sample triangle coordinates and gave point locations that we knew the correct triangle mapping to and used assert statements to ensure that our functions gave this correct output. Additionally, print statements were used throughout in order to ensure that input/output wascorrect.

Validation of ck using debugging data:

Each debug sample readings test file was inputted, eventually generating output as described in the assignment guidelines. We compared the (dk-ck) norm from the answer text files to our generated output text files in order to determine the accuracy of our algorithms.

Debug A:

Answer.txt (dk-ck) norm	Our Output.txt (dk-ck) norm
0.000	0.002
0.000	0.004
0.000	0.004
0.000	0.0
0.000	0.004
0.000	0.002
0.000	0.004
0.000	0.005
0.000	0.005
0.000	0.002
0.000	0.244
0.000	0.001
0.000	0.001
0.000	0.001
0.000	0.001

Debug B:

Answer.txt (dk-ck) norm	Our Output.txt (dk-ck) norm

0.083	0.085
2.520	2.516
0.719	0.717
0.082	0.083
3.366	3.365
2.784	2.78
2.355	2.351
2.538	2.529
1.460	1.462
1.662	1.66
0.721	0.72
1.554	14.615
0.157	0.159
1.630	1.632
2.869	2.868

Debug C:

Answer.txt (dk-ck) norm	Our Output.txt (dk-ck) norm
1.207	1.202
0.063	0.061
0.322	0.324
0.090	0.089
1.426	1.423
0.030	0.029
1.823	8.158
0.298	0.3
1.888	1.886
0.594	0.596
0.835	0.833
0.620	14.305
0.085	0.087
1.243	1.244
1.379	13.449

Debug D:

Answer.txt (dk-ck) norm	Our Output.txt (dk-ck) norm
2.205	2.204
1.823	1.821
0.667	0.672
0.063	0.056
0.506	0.507
0.097	0.101
1.111	1.109
0.426	0.424
1.406	7.029
4.396	7.136
0.265	0.267
0.334	0.334
1.922	1.92
0.470	0.47
0.902	0.897

Debug E:

Answer.txt (dk-ck) norm	Our Output.txt (dk-ck) norm

3.697	3.69
3.848	3.847
1.997	1.998
0.558	0.557
0.177	0.172
0.481	0.48
3.258	2.123
0.829	0.831
3.991	3.997
1.199	1.194
0.994	0.994
1.976	1.973
3.296	11.039
3.816	3.823
0.240	0.232

Debug F:

Answer.txt (dk-ck) norm	Our Output.txt (dk-ck) norm
0.859	0.855
2.383	2.388
1.741	1.746
2.856	2.859
1.924	1.917
3.195	3.201
2.538	2.532
2.890	2.89
0.382	0.383
1.680	1.678
0.626	0.629
0.591	0.587
1.933	1.94
1.248	1.252
2.896	2.119

Discussion of Debug Results:

When we found the error values between our norm(dk – ck) and answer norm(dk - ck), the %error was < 1% for almost all sample frames. There were a couple of sample frames in some cases, however, in which our output values were different. This could be due to a multitude of factors. Our best guess is that it is due to an edge case that we did not account for in the find closest point algorithm.

Generated Output for Unknown Data:

Unknown G:

Our Output.txt (dk-ck) norm

```
1.505
2.708
0.222
2.636
0.838
4.185
0.934
3.549
1.857
0.775
2.957
1.267
3.185
1.834
1.049
2.001
1.04
1.224
1.47
3.23
```

Unknown H:

Our Output.txt (dk-ck) norm

```
2.467
1.922
0.556
8.955
1.483
9.216
1.26
3.646
3.255
2.468
1.274
3.584
0.909
0.874
0.285
0.877
6.31
0.653
3.912
1.737
```

Unknown J:

Our Output.txt (dk-ck) norm

```
2.164
1.933
1.525
0.807
9.505
5.827
10.844
0.405
0.891
1.097
2.136
4.702
3.39
0.126
1.812
0.129
0.551
0.781
2.029
0.458
```

4) Individual Contributions:

Samer: Primarily Coding, Algorithm development (Coding, testing and Validation)

Ajay: Primarily Mathematical approach behind the algorithm, results comparison (the Math Part)

Citations:

1. InterpolationReview.pdf [https://ciis.lcsr.jhu.edu/lib/exe/fetch.php?media=courses:455-655:lectures:interpolationreview.pdf]
2. Registration.pdf [https://ciis.lcsr.jhu.edu/lib/exe/fetch.php?media=courses:455-655:lectures:registration_part_1.pdf]
3. Finding point pairs for Iterated Closest Point algorithms.pdf [https://ciis.lcsr.jhu.edu/lib/exe/fetch.php?media=courses:455-655:lectures:finding_point-pairs.pdf]
4. Registration Techniques [https://roboticsknowledgebase.com/wiki/math/registration-techniques]