



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	John Han
Email	jhan62@jh.edu
Other contact information (optional)	
Name 2	UNWAT ANTANI
Email	wantani1@jh.edu
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

Programming Assignment 3

This programming assignment involved the implementation of a basic Iterative Closest Point (ICP) Algorithm. This report analyzes our approach, procedure, structure, and results of the assignment.

Mathematical Approach and Algorithmic Steps

This basic ICP Algorithm uses the method mentioned in the lecture slides as shown:

Finding Closest Pair:

FindClosestPoint(a,[p,q,r])

Many approaches. One is to solve the system

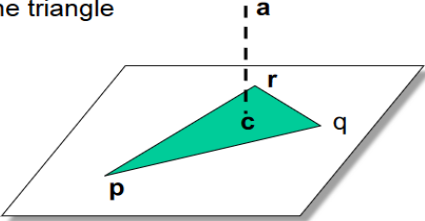
$$\mathbf{a} - \mathbf{p} \approx \lambda(\mathbf{q} - \mathbf{p}) + \mu(\mathbf{r} - \mathbf{p})$$

in a least squares sense for λ and μ . Then compute

$$\mathbf{c} = \mathbf{p} + \lambda(\mathbf{q} - \mathbf{p}) + \mu(\mathbf{r} - \mathbf{p})$$

If $\lambda \geq 0, \mu \geq 0, \lambda + \mu \leq 1$, then \mathbf{c} lies within the triangle and is the closest point. Otherwise, you need to find a point on the border of the triangle

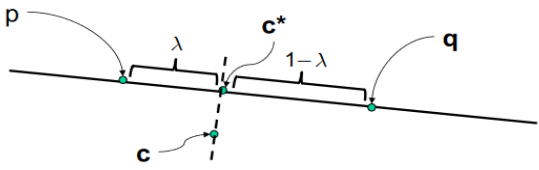
Hint: For efficiency, work out the least squares problem explicitly for λ, μ



Copyright Russell Taylor, 2010-2021

Engineering Research Center for Computer Integrated Surgical Systems and Technology

ProjectOnSegment(c,p,q)


$$\lambda = \frac{(\mathbf{c} - \mathbf{p}) \cdot (\mathbf{q} - \mathbf{p})}{(\mathbf{q} - \mathbf{p}) \cdot (\mathbf{q} - \mathbf{p})}$$
$$\lambda^{(seg)} = \text{Max}(0, \text{Min}(\lambda, 1))$$
$$\mathbf{c}^* = \mathbf{p} + \lambda^{(seg)} \times (\mathbf{q} - \mathbf{p})$$

Copyright Russell Taylor, 2010-2021

Engineering Research Center for Computer Integrated Surgical Systems and Technology

Source: https://cwis.lcsr.jhu.edu/lib/exe/fetch.php?media=courses:455-655:lectures:finding_point-pairs.pdf

Interpolation method is used and set up in such a way that we set up, using the data points we have a least squares problem to obtain the scaling factors μ and λ . After finding these, we check whether $\mu, \lambda \geq 0$ and $\mu + \lambda = 1$. If this condition satisfies, we say that the calculated nearest point lies on the triangle.

Using this closest point obtained on the current triangle, it then loops on other triangles and only keeps the best(closest) point to the mesh found by iterating over all the triangles. This implementation is not the most efficient but it is a good starting point.

Point Cloud Registration

We decided to go with the quaternion method to calculate the rotation matrix between the two clouds. To summarize,

Quaternion method for R

Step 1: Compute

$$\mathbf{H} = \sum_i \begin{bmatrix} \tilde{a}_{i,x} \tilde{b}_{i,x} & \tilde{a}_{i,x} \tilde{b}_{i,y} & \tilde{a}_{i,x} \tilde{b}_{i,z} \\ \tilde{a}_{i,y} \tilde{b}_{i,x} & \tilde{a}_{i,y} \tilde{b}_{i,y} & \tilde{a}_{i,y} \tilde{b}_{i,z} \\ \tilde{a}_{i,z} \tilde{b}_{i,x} & \tilde{a}_{i,z} \tilde{b}_{i,y} & \tilde{a}_{i,z} \tilde{b}_{i,z} \end{bmatrix}$$

Step 2: Compute

$$\mathbf{G} = \begin{bmatrix} \text{trace}(\mathbf{H}) & \Delta^T \\ \Delta & \mathbf{H} + \mathbf{H}^T - \text{trace}(\mathbf{H})\mathbf{I} \end{bmatrix}$$

$$\text{where } \Delta^T = \begin{bmatrix} \mathbf{H}_{2,3} - \mathbf{H}_{3,2} & \mathbf{H}_{3,1} - \mathbf{H}_{1,3} & \mathbf{H}_{1,2} - \mathbf{H}_{2,1} \end{bmatrix}$$

Step 3: Compute eigen value decomposition of G

$$\text{diag}(\tilde{\lambda}) = \mathbf{Q}^T \mathbf{G} \mathbf{Q}$$

Step 4: The eigenvector $\mathbf{Q}_k = [q_0, q_1, q_2, q_3]$ corresponding to the largest eigenvalue λ_k is a unit quaternion corresponding to the rotation.

Figure : Summary of the Quaternion Method for R

We can use the unit quaternion found in step 4 to explicitly solve for the R matrix using the following expression:

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$$

Figure : Closed form solution for a rotation matrix from a unit quaternion

Finally, the translation component is calculated by simply performing

$$\bar{\mathbf{p}} = \bar{\mathbf{b}} - \mathbf{R} \cdot \bar{\mathbf{a}}$$

Figure : Closed form solution for translation vector from R and the mean of two point clouds

Overall Structure of the program:

The code is divided into modules such that each subroutine is its own class.

reader.py

This script gets the data from the relevant text and mesh files. The user has to type in the path of relevant files i.e., LED for Body A, Body B, Mesh file and Sample Readings respectively while running the *pa3.py* script from the command line. The argparser will parse the file path and return an error if the specified file path is not found. The script has 3 classes, each for different input types. BodyLEDs class is used to read the readings of LEDs of Bodies(A and B). SurfaceMesh class is used to read and parse the data of input mesh files. SampleReadings class is used to read the data from the sample reading files. There are separate classes as the structure of data in these files were different from each other.

writer.py

This script is used to write the output data generated i.e., d, c and $||d - c||$ as mentioned in the assignment in the output file, the path and name of which is to be mentioned while executing *pa3.py* as argument.

frameTransform.py

A script which declares another class - Frame. Frame is an object to represent and handle rotation and translation in a better way. It has helper functions declared namely F_{inv} to calculate inverse of a transformation matrix, ComposeTransform to get the resultant transformation matrix after multiplying two transformation matrices and finally, transformVector, to get the resultant vector after applying a transformation on it.

pointsetRegistration.py

Script to implement 3d Point cloud to point cloud registration. It employs the Quaternion method(which in turn uses the eigenvalue decomposition method) to calculate the Rotation matrix. Using this Rotation Matrix, we calculate the position vector. The return type is a Frame type object as mentioned in the frameTransform.py with Rotation and Position as its attribute

pa3.py

This is the main script of the program which controls the flow of the data as well as generates the output text file. The inputs and the output file path and name are to be passed as an argument. The script has all the main functions i.e., data reader, calling the ICP algorithm, calculating F_{A_k} and F_{B_k} and writing the output generated by the algorithm in a file. It also has a `validate_file` function which will check whether the files parsed exist or not.

test.py

This script has simple test cases for all the functions like reader, writer, finding closest point on the triangle and finding closest point on the mesh. One can change the values in the test and check whether the algorithm is able to give out a meaningful output or not.

In order to execute the algorithm, change the contents of `config.ini` to the desired filenames and simply run

```
$ python pa3.py
```

To run the test scripts, we can similarly run

```
$ python test.py
```

Validation Steps:

Once the ICP algorithm is successfully constructed and run, the result is matched with the output file provided with the debug data and the error is noted. Note: There will always be mismatches between the values, but the error obtained in our case was almost negligible i.e., acceptable within limits. There are multiple debug datasets given using which we can tweak and improve our algorithm. One snippet of the comparison is as shown below:

```
15 PA3-B-Debug-Output.txt 0
    11.16    19.12   -26.09    11.24    19.10   -26.08    0.083
   -29.93   -33.23   -31.16   -28.97   -30.94   -30.73    2.520
    -8.33     2.92    31.07    -7.64     3.03    30.92    0.719
     1.15    11.01    -8.30     1.18    10.95    -8.27    0.082
   -31.62    -5.26   -12.85   -32.46    -3.97    -9.86    3.366
   -24.44    -0.76   -14.42   -25.34     0.98   -12.44    2.784
   -16.63    15.47    17.48   -14.83    16.06    18.89    2.355
    12.63    22.04    16.64    12.70    24.55    16.99    2.538
    30.57     4.13   -29.96    29.80     3.90   -28.74    1.460
    17.56    24.28   -10.29    17.57    25.92   -10.54    1.662
   -22.81     3.90   -44.38   -22.83     3.48   -43.79    0.721
   -16.49   -33.90   -28.17   -16.72   -32.37   -28.04    1.554
    27.77    17.56   -24.54    27.67    17.47   -24.45    0.157
    31.59    -2.09    11.52    32.61    -3.00    12.40    1.630
    27.25     5.83   -32.89    27.02     6.17   -30.05    2.869
```

Figure: Debug output(provided)

```
15  ..\data\pa3-B-Output.txt
11.24  19.10  -26.08  11.16  19.12  -26.09  0.08
-28.96  -30.95  -30.73  -29.93  -33.23  -31.16  2.52
-7.63  3.03  30.92  -8.33  2.91  31.07  0.72
1.18  10.95  -8.27  1.15  11.01  -8.30  0.08
-32.46  -3.97  -9.86  -31.62  -5.27  -12.85  3.37
-25.34  0.98  -12.44  -24.44  -0.76  -14.42  2.79
-14.83  16.06  18.89  -16.63  15.47  17.48  2.35
12.70  24.55  16.99  12.63  22.04  16.64  2.54
29.81  3.90  -28.74  30.57  4.12  -29.96  1.46
17.57  25.92  -10.54  17.56  24.27  -10.29  1.66
-22.83  3.48  -43.79  -22.81  3.89  -44.38  0.72
-16.72  -32.37  -28.04  -16.49  -33.90  -28.17  1.56
27.67  17.47  -24.45  27.77  17.55  -24.54  0.16
32.61  -3.00  12.40  31.59  -2.09  11.52  1.63
27.03  6.16  -30.05  27.25  5.82  -32.89  2.87
```

Figure: Output Generated by ICP

Once validation is complete the algorithm is applied to unknown input data and the output file is generated.

Results

A basic ICP algorithm was constructed and implemented. Validation on initial debug datasets showed that the algorithm works well and within acceptable error rates. The output on the unknown dataset G is as follows:

c			d			d - c
-8.7	7	30.61	-9.99	6.45	31.05	1.47
21	24.36	9.76	18.94	18.92	8.56	5.94
10.02	12.51	-10.8	10.37	12.14	-10.22	0.77
8.56	-13.44	3.48	7.62	-17.89	5.33	4.91
-37.78	-21.15	-36.61	-37.5	-20.98	-36.46	0.35
24.66	-14.62	-14.95	27.95	-19.76	-15.69	6.15
-0.53	-10.91	-32.56	0.03	-10.6	-32.86	0.71
-12.79	1.35	10.25	-8.01	1	8.43	5.13
4.94	-6.99	43.17	5.77	-14.59	43.25	7.65
-13.62	-2.99	-46.82	-13.79	-3.19	-45.67	1.18
-11.01	9.46	-24.9	-13.9	5.3	-23.98	5.15
40.22	4.54	-2.05	41.88	4.63	-1.67	1.7

24.15	-5.46	16.75	26.22	-11.89	18.49	6.97
11.74	5.02	62.94	11.76	4.97	61.48	1.46
16.15	7.05	-20.3	17.14	6.76	-19.47	1.33
18.79	14.06	-31.22	18.2	13.61	-32.74	1.69
-0.38	-15.82	-34.89	0.76	-16.02	-35.22	1.2
-7.02	-8.88	10.4	-8.28	-10.34	11.21	2.1
6.41	-7.45	61.75	6.53	-11.5	61.49	4.06
0.65	9.89	-10.72	1.25	4	-9.31	6.09

Contributions

John Han: pointsetRegistration, pa3, reader, IterativeClosestPoint

Unnat Antani: pa3, writer, pointsetRegistration, frameTransform