Programming Assignment 3 Report

Jessica Soong & Andrew Cornelio

November 23, 2020

1 Introduction

The purpose of this assignment really is to implement an efficient search algorithm so that ICP can be done quickly. This is done through a linear search as well as a covariance tree. The purpose of ICP is to register two different coordinate systems to one common one so that each tool, robot arm, or imaging system operate in the same frame.

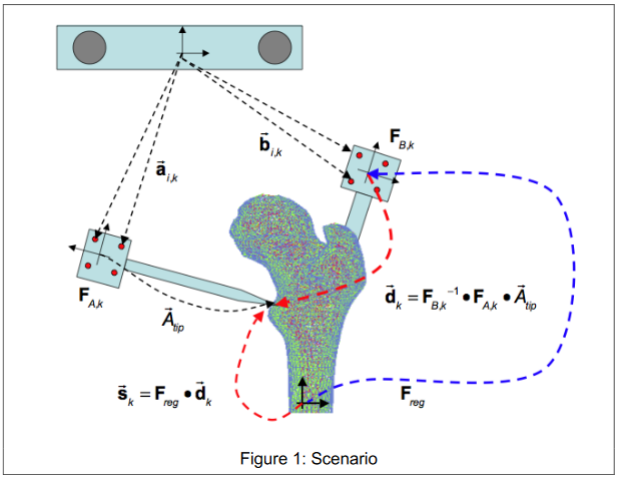

Figure 1: Problem Description

The relevant variables from Fig. 1 are:

- $F_{A,k}$: The local coordinate frame of the calibration object A .
- $F_{B,k}$: The local coordinate frame of the calibration object B.
- $a_{i,k}$: The position of the LED marker i on the calibration object A.
- $b_{i,k}$: The position of the LED marker i on the calibration object B.
- A_{tip}: The transformation $F_{A,k}$ to the tip of the tool for calibration object A.

- B_{tip}: The transformation $F_{B,k}$ to the tip of the tool for calibration object B.
- d_k: The position of the pointer tip with respect to rigid body B.
- F_{reg}: The local coordinate frame of the EM tool (defined with respect to the EM tracker base).
- s_k: Sample point defined in the registration frame F_{reg}.
- c_k: The point on the mesh closest to the point s_k.

2 Mathematical Approach

We describe how to find matches between a test point, **a**, and a triangular mesh, as well as the general mathematic steps to set up the registration problem posed in the problem statement.

2.1 Finding c_k and d_k

From the provided data, we have $a_{i,k}$ and $b_{i,k}$. We do cloud-to-point cloud registration to find the poses $F_{A,k}$ and $F_{B,k}$ respectively. The mathematical steps behind this subroutine is outlined in PA #1. After this is found, we calculate the position d_k of the pointer tip with respect to the rigid body B,

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

For PA #3, we assume that F_{reg} is the identity matrix. Then, we compute s_k as:

$$s_k = F_{reg} d_k$$

And then we do a search for the point on the mesh closest to s_k using the selected algorithm, which returns c_k.

2.2 Closest Point on Triangle

We use the method outlined in slides 8-11 on the "Finding point-pairs" presentation. We have a test point **a** and a triangle $(\mathbf{p}, \mathbf{q}, \mathbf{r})$ and we wish to find the point **c** that lies on the triangle and which is closest to **a**. This can be done by projecting **a** to a point $\mathbf{c}^*$ that lies on the plane defined by the triangle. If the new point lies outside the triangle, we can then project it onto an edge of the triangle. We can find the point projection onto the plane using barycentric coordinates and least squares. First note that any point on the plane, c^*, can be defined in barycentric coordinates as:

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

Where (μ, λ) are the barycentric coordinates. To find the closest point on a plane to a test point we must set up a least square problem as follows, where $\mathbf{c}^*$ is a point on the plane.

$$\min_{\mathbf{c}^*} (\mathbf{c}^* - \mathbf{a})^2$$

Now we can combine these two expression to find closest point in barycentric coordinates:

$$\min_{\mu,\lambda}[\mu(\mathbf{q}-\mathbf{p})+\lambda(\mathbf{r}-\mathbf{p})-(\mathbf{a}-\mathbf{p})]^2$$

We can set up the least squares problem as such:

$$\begin{bmatrix}\mathbf{q}-\mathbf{p} & \mathbf{q}-\mathbf{p}\end{bmatrix}\begin{bmatrix}\mu\\ \lambda\end{bmatrix}=\begin{bmatrix}\mathbf{a}-\mathbf{p}\end{bmatrix}$$

Once we find the (μ, λ), we can convert from barycentric coordinates to cartesian coordinates and find c^* using the first equation. Then we need to make sure c^* lies within the triangle. We do this by ensuring that $0 \leq \mu, \lambda \leq 1$. If this is not true, then we project the point on the plane to one of the segments on the triangle. If we have a line segment $(\mathbf{p}, \mathbf{q})$ and a point $\mathbf{a}$, we can find the projection, by creating the vectors $(\mathbf{a}-\mathbf{p})$ and $(\mathbf{q}-\mathbf{p})$. We can find the normalized projection by taking the dot products:

$$\rho^* = \frac{(\mathbf{c}^*-\mathbf{p})\cdot(\mathbf{q}-\mathbf{p})}{(\mathbf{q}-\mathbf{p})\cdot(\mathbf{q}-\mathbf{p})}$$

We then make sure that ρ is between 0 and 1:

$$\rho = max(0, min(1, \rho^*))$$

Finally, we find the projection onto the line:

$$c = \mathbf{p} + \rho(\mathbf{q}-\mathbf{p})$$

2.3 Closest Point on Mesh

There are two ways of finding the closest point, c, on a mesh to test point, a. The first is by linearly searching all the triangles on the mesh. For each triangle, we can compute the closest point on that triangle the test point, and iteratively refine our guess for the closest point by comparing the minimum distance from each triangle to the test point. The mathematical details section covers how to find the closest point on the mesh. The second method is to use a tree structure, like the covariance tree. We will describe the covariance tree in more detail in the next section.

3 Algorithmic Steps

3.1 Creating the Covariance Tree

The construction of the covariance tree is summarized in the following steps that were outlined in slides 64-71 and 75-80 of the "Finding point-pairs" presentation:

1. Take in a list of of triangles and the length of the list. Use the centroids of the triangles as the sort points.

2. Perform PCA on the sort points to find the directions of greatest variance. These vectors will necessarily be orthogonal. Convert these directions into a rotation matrix and use it to define a local coordinate frame for the points.

3. Transform all the points including the vertices of the triangles into the the local coordinate frame and generate upper and lower bounds based on all the vertices of all the triangles.

4. Create a node containing the references to the triangles, the length of the list, the upper and lower bounds, and the local reference frame.

5. Check if the number of triangles is less than or equal to a minimum number, or if the upper bound minus the lower bound (the bounding box diagonal) is less than a minimum length. If either is true, stop. Otherwise continue.

6. Split the triangles into two groups based on whether the first principle component of the point (the first coordinate in the local coordinate system) is positive or negative. Repeat step 1 with both lists of triangles and the corresponding sizes of each list. Set the two resulting trees as left and right children of the current node.

3.2 Searching the Tree

Searching a convariance tree to find the closest point on the mesh to a test point can be summarized as a depth first search through the covariance tree that visits the leaf nodes of the tree, keeping track of the current closest point from each leaf node it visits. The covariance tree speeds up the search because it can quickly check if the current minimum distance from the test point will reach into a bounding box of a node. This can quickly eliminate a lot of triangles, especially if a node higher up in the tree is skipped because it's bounding box is too far away. Here are the steps of the recursive algorithm that were outlined in slides 72-74 of the "Finding point-pairs" presentation:

1. Take a test point **v**, a current minimum distance, and a current closest point.

2. Transform the point v to the local coordinate frame of the current node. Test if the test point minus the minimum distance is less than the lower bounds. Test if the test point plus the minimum distance is greater than the upper bounds. Do this for each dimension, and if any are true, then return/quit.

3. If the current node has children, get the closest point and minimum distance from the left subtree, use these as the current closest point and minimum distance when searching the right subtree.

4. If the tree does not have children, search all the triangles in the node and check if there is any point closer than the current closest point. Update the closest point and minimum distance if one is found. Return the closest point and minimum distance and continue or finish the recursive search.

4 Software Overview / Program Structure

Description of files in the programs directory. Subsections represent directories. Detailed explanations of what every function or subroutine does can be found inline in comments of code, including input and output arguments.

PA_3

- `main.m`: Function that executes the algorithmic steps in §2 for a single text file. Edit the inputs near the top to change the files used. `problemNum` for PA #3 will always be 3, `sampleReadingLetter` corresponds to the different scenarios posed by the text files, and `fileType` can be either "debug" or "unknown." The output will be saved in the **output** folder with the designated filename.
- `init.m`: This file adds all files and files in sub-directories to the path. It is run at the beginning of main to initialize the paths needed.

4.1 programs

- `generate_outputs.m`: runs the main file multiple times for all text files given.
- `Project_On_Segment.m`: This function takes a point c and projects it on the line formed by the two points p and q, and returns the projection.
- `Find_Closest_Point_Triangle.m`: This function takes a matrix of triangle vertices, and a point a, and calculates the closest point on the triangle to the point.
- `Find_Closest_Point_Mesh.m`: This function implements brute force search for closest point on mesh via a linear search that keeps track of the closest point and minimum distance.

4.1.1 data_structures

Folder for classes used for different search methods.

- `CovTreeNode.m`: Covariance tree node class. Implements the functions for the search as well as for tree building.
- `Triangle.m`: Triangle class. Implements functions to find the closest point on a triangle to a test point, and to create and enlarge the bounding boxes of a triangle.

4.1.2 test_data

Folder of data used for unit testing

- `generate_cube.m`: makes a cube in 3D space ranging from (+/-1,+/-1,+/-1) and saves it to cube.mat
- `generate_rectangular_prism_mat.m`: makes a rectangular box in 3D space ranging from (+/-1,+/-2,+/-3) and saves it to rect_prism.mat
- `cube.mat`: resulting file from `generate_cube.m`, containing vertices and the corresponding corner indices.
- `rectangular_prism_mat.mat`: resulting file from `generate_rectangular_prism_mat.m`, containing vertices and the corresponding corner indices.

4.1.3 unit_testing

Files here are for unit testing different functions. We will go into more depth in the testing section.

- `CovTreeSearch_Unit_Testing.m`: Tests the search functions for the covariance tree implementation.
- `Find_Closest_Point_Mesh_Unit_Testing.m`: Tests the function that finds the closest point on a mesh to a given point.
- `Find_Closest_Point_Unit_Testing.m`: Tests the function that finds the closest point on a triangle to a given point.
- `Project_On_Segment_Unit_Testing.m`: Tests the function that projects a point in 3D space onto a mesh.
- `Triangle_Unit_Testing.m`: Unit tests functions in the Triangle class.

4.1.4 util

Files here are general usage functions used throughout `main.m` as well as the unit tests, generally from previous programming assignments.

- **kinematic**
  - `homoify.m`: Takes in a homogeneous vector and makes it non-homogenous (removes the 1). Alternatively, takes in a non-homogeneous vector, and makes it homogeneous.
  - `randomSE3.m`: Generates a random valid rotation and translation and returns the transform.
  - `rot_rpy.m`: Given an input vector q = [roll, pitch, yaw], rotates about the correct axes and returns the resultant rotation matrix.
  - `RX.m`: Given a scalar angle, returns the rotation about the x-axis by this angle.
  - `RY.m`: Given a scalar angle, returns the rotation about the y-axis by this angle.
  - `RZ.m`: Given a scalar angle, returns the rotation about the z-axis by this angle
  - `SE3_INV.m`: Uses SE3 properties to quickly calculate the inverse of a given transformation.
  - `SE3.m`: Takes in a rotation and translation, and quickly appends them to the right format
  - `randomSE3.m`: Generates a random valid member of SE3.
- **registrations**: These are the registration/calibration functions from PA # 1
  - `Point_Cloud_Registration.m`: Takes in two sets of points (A, B, with points in row format), finds the transformation from one set of points to the other by performing point cloud calibration/registration.

- **read**: These are the files read in the text data given into a desired format.
  - `read_ProblemX_BodyY.m`: Reads a `ProblemX_BodyY.txt` file and parses the data
  - `read_SampleReadings.m`: Reads a `SampleReadings.txt` file and parse the data.
  - `read_ProblemXMesh.m`: Reads a `ProblemXMesh.txt` file and parses the data.

5 Verification

5.1 Unknown Data Results

This is the output files of all the unknown data. See the appendix for the output data for the debugging data.

Listing 1: Program Generated PA3-G-Unknown-SampleReadingsTest.txt

```
20, pa3-G-Output.txt
   41.91     5.19    -3.93    40.55     5.20    -4.04     1.365
  -21.46   -30.82   -12.92   -21.20   -29.10   -14.09     2.093
   37.40    11.95     4.41    37.66    12.08     4.41     0.287
    7.53     2.31    61.81     7.69     2.43    63.23     1.429
    2.24    -7.26    61.28     2.24    -7.26    61.28     0.001
   32.72   -11.48     2.19    30.56    -8.29     1.71     3.883
  -29.72   -20.22   -46.59   -29.41   -19.77   -45.73     1.014
  -12.15   -34.29   -32.72   -13.38   -30.69   -31.92     3.891
   16.00     5.82   -17.70    14.57     6.32   -18.62     1.777
  -19.39    -4.28   -47.42   -19.23    -4.07   -47.86     0.517
    9.55    -9.06    57.73     9.17    -7.05    57.76     2.045
    7.95   -20.01   -14.75     7.27   -16.31   -14.71     3.765
   -8.28   -23.31    -9.68    -8.78   -21.97   -10.80     1.810
   26.39   -13.71    -1.02    25.94   -11.24    -1.86     2.648
  -35.68   -21.69   -40.86   -35.12   -21.28   -40.30     0.901
   12.32    22.88     7.00    11.63    26.55     7.36     3.753
  -11.07   -29.41   -13.64   -11.76   -27.40   -15.56     2.864
   18.38   -11.85     9.69    17.18    -9.54     8.51     2.855
   13.81    21.57    16.72    14.20    24.36    17.28     2.869
    2.10    -8.11    41.06     2.16    -6.89    41.04     1.222
```

Listing 2: Program Generated PA3-H-Unknown-SampleReadingsTest.txt

```
20, pa3-H-Output.txt
   15.20    22.30    -9.93    13.80    24.67   -10.80     2.889
   26.46    20.39    -5.81    28.08    22.41    -5.70     2.597
   22.69   -16.30    -8.43    22.31   -15.85    -8.50     0.593
```

```
 27.90    20.32   -16.46    28.83    21.62   -16.86    1.647
  3.15    17.79     1.56     2.82    18.02     1.42    0.419
-39.48   -25.34   -25.68   -38.43   -24.37   -25.73    1.431
 -0.43    19.27     9.37    -0.93    19.80     8.98    0.821
-34.89    -9.84    -7.71   -34.82    -9.83    -7.87    0.176
  4.93   -11.54    18.23     4.58    -9.60    17.63    2.067
  5.18    15.29    41.93     4.42    20.41    42.73    5.245
 -1.22   -12.54    21.40    -1.14    -9.41    20.94    3.171
 18.57    18.43    32.51    19.95    19.86    32.58    1.997
  3.52   -10.23    35.71     3.66    -7.16    35.51    3.077
 -5.81    10.17     8.61    -7.41    11.66     8.01    2.271
 -6.57   -22.90   -36.73    -5.98   -23.37   -37.13    0.852
-12.19   -28.08   -15.99   -12.20   -27.79   -16.17    0.351
-41.46     1.25   -28.65   -40.69     0.73   -28.61    0.925
-20.67   -13.51    -3.94   -20.48   -13.02    -4.84    1.041
  0.42    19.76    17.36    -0.39    23.73    17.46    4.060
  0.22   -11.09    55.55     1.53    -7.05    55.50    4.245
```

Listing 3: Program Generated PA3-J-Unknown-SampleReadingsTest.txt

```
20, pa3-J-Output.txt
 20.72    10.33    45.62    22.80    11.07    46.05    2.253
 31.23     7.64    14.20    32.34     7.90    14.95    1.362
  3.86    20.61    57.10     4.34    19.07    57.01    1.609
-19.88     0.56   -49.69   -19.49    -1.21   -46.92    3.312
 -8.10    17.83     9.18    -5.51    16.12    10.64    3.430
-22.35    15.78   -35.39   -21.26    11.19   -33.84    4.963
 16.20    21.17    39.04    16.32    21.48    39.12    0.340
-23.02   -10.83   -49.98   -22.95   -10.69   -49.68    0.344
 19.14    -8.53     1.25    20.40   -10.75     2.48    2.830
 -6.26   -11.74    -3.88    -6.63   -14.51    -2.15    3.283
-16.75   -22.48   -42.83   -16.35   -23.16   -44.12    1.514
-17.57    13.90   -38.90   -17.33    10.13   -36.62    4.410
-40.79    -9.24   -34.40   -40.29    -9.19   -34.39    0.508
-11.36    -1.68    13.40   -11.16    -1.62    13.29    0.232
 30.38    18.69   -21.85    29.85    18.27   -21.36    0.835
-11.73    -1.31    18.62    -9.73    -0.93    18.17    2.080
  1.36     1.49   -25.24     0.73     0.96   -24.60    1.046
-23.87    -2.22   -49.68   -23.94    -2.10   -47.68    2.002
 -1.39    11.88   -18.31    -1.59     8.07   -17.32    3.941
-11.46    14.83    10.75    -8.84    13.77    12.48    3.307
```

5.2 Testing

Here we go into more depth into the unit tests.

5.2.1 `Find_Closest_Point_Mesh_Unit_Testing.m`:

Load cube.m file and ensure that the linear closest point search that iterates through all the triangles will compute the correct closest point on the mesh for three points. We know the result based on the geometry of the cube and the location of the test point so we can easily text against these cases.

5.2.2 `CovTreeSearch_Unit_Testing.m`:

Load rect_prism.m file and ensure that the covariance tree point search will find the correct closest point on the mesh for three points, similarly to the brute force search test. We compare this to hard-coded known values as well as the result from the brute force search. This was done with a prism so that we could tell how the points were being divided, since a cube is regular and symmetric. We used a prism to break the symmetry of using a cube. This ensures that the tree is sorting the points correctly. We also tested with several random points to make sure that the covariance tree search matched the brute force search.

5.2.3 `Find_Closest_Point_Unit_Testing.m`:

Create test points on vertices of a triangle. Ensure that the computed closest points are the vertices. Then average the vertices together to get the midpoints of the edges for the test points. Ensure the computed closest points are the midpoints. Then offset the triangle by a fixed amount in the direction normal to the surface and repeat the tests.

5.2.4 `Project_On_Segment_Unit_Testing.m`:

We created line segments of different lengths and orientations and tested to ensure that the test point was always projected onto the correct point on the line segment which we calculated by hand. This includes 1D examples as well as 2D examples.

5.2.5 `Triangle_Unit_Testing.m`:

This script tests that the default constructor is working correctly by testing that the center of the triangle is correctly computed. It also tests some of the class subroutines. `EnlargeBounds` is tested by making sure that the updated the lower bound is the minimum over all dimensions of all vertices, and the upper bound is the the maximum of all vertices. This is done with the identity as the local frame transformation, as well as with a Z-rotation with hand calculated-values. `BoundingBox` is tested by making sure that the returned upper and lower bounds are $+\infty$ and $-\infty$ respectively, no matter the input values.

5.2.6 util:

All of the functions in the util folder have been tested before in previous programming assignments. They are verified to work correctly. Please refer to those reports for the respective unit tests.

6 Discussion

Based on our unit tests, the closest point on triangle function and the brute force closest point on mesh work as expected. That means that the covariance tree search must also work, since it matches both hand calculated results from a simple mesh case, as well as the output of the brute force search for all our random test cases.

There is some slight difference between the "Answer" files provided and the program output. A summary of the statistics can be found in Table 1. The complete result of differences can be found in a separate excel file turned in with the zip file, called "Error Analysis.xlsx". Generally, there is some very small error for some of the calculated files. It is likely that this is due to some storage or type issues with MATLAB rather than an algorithmic fault since the differences are so small and also because the program passed every unit test implemented. This most likely occurs when the points are being projected on to the mesh/triangle, and not during the tree search itself.

			Absolute Average Difference Between "Answer" File and Tree Search Output						
File Type	Scenario	Stat.	c_x	c_y	c_z	d_x	d_y	d_z	norm
Debug	A	Avg.	0.0020	0.0040	0.0013	0.0013	0.0033	0.0000	0.0015
		Std.	0.0041	0.0051	0.0035	0.0035	0.0049	0.0000	0.0016
Debug	B	Avg.	0.0000	0.0047	0.0000	0.0007	0.0007	0.0013	0.0023
		Std.	0.0000	0.0052	0.0000	0.0026	0.0026	0.0035	0.0027
Debug	C	Avg.	0.0013	0.0040	0.0013	0.0027	0.0040	0.0007	0.0023
		Std.	0.0035	0.0051	0.0035	0.0046	0.0051	0.0026	0.0019
Debug	D	Avg.	0.0013	0.0040	0.0013	0.0027	0.0040	0.0007	0.0023
		Std.	0.0035	0.0051	0.0035	0.0046	0.0051	0.0026	0.0019
Debug	E	Avg.	0.0013	0.0040	0.0013	0.0027	0.0040	0.0007	0.0023
		Std.	0.0035	0.0051	0.0035	0.0046	0.0051	0.0026	0.0019
Debug	F	Avg.	0.0013	0.0040	0.0013	0.0027	0.0040	0.0007	0.0023
		Std.	0.0035	0.0051	0.0035	0.0046	0.0051	0.0026	0.0019
Unknown	G	Avg.	0.0030	0.0040	0.0005	0.0040	0.0005	0.0015	0.0026
		Std.	0.0047	0.0050	0.0022	0.0050	0.0022	0.0037	0.0020
Unknown	H	Avg.	0.0030	0.0040	0.0005	0.0040	0.0005	0.0015	0.0026
		Std.	0.0047	0.0050	0.0022	0.0050	0.0022	0.0037	0.0020
Unknown	J	Avg.	0.0030	0.0040	0.0005	0.0040	0.0005	0.0015	0.0026
		Std.	0.0047	0.0050	0.0022	0.0050	0.0022	0.0037	0.0020

Table 1: Average Difference Between "Answer" File and Tree Search Output

FileType	Scenario	Brute Force	Tree Building	Tree Search
Debug	A	0.356	2.479	0.124
Debug	B	0.276	2.402	0.069
Debug	C	0.284	2.412	0.094
Debug	D	0.320	2.432	0.079
Debug	E	0.280	2.406	0.082
Debug	F	0.274	2.265	0.061
Unknown	G	0.380	2.495	0.107
Unknown	H	0.381	2.412	0.104
Unknown	J	0.414	2.538	0.109
Mean		0.329	2.427	0.092
Std. Dev		0.051	0.073	0.019

Table 2: Runtimes for Search Algorithms (in seconds)

The brute force search can be considered faster than the covariance tree search only when there is a limited number of searches done and the time it takes to sort the data (build the tree) is considered as well. For most purposes where a large amount of data needs to be queried multiple times, a tree search will be faster. The run times for each file and algorithm is outlined in Table 2.

7 Contributions

We originally were working on two separate projects. Jessica was working in Matlab and Andrew was working in Python. When we decided to work together, we chose the the MATLAB implementation over the Python. Vestigial python files are in the PA3-Python folder.

Jessica: Worked created the find closest point on triangle and mesh functions in MATLAB. Worked on covariance tree search and unit tests. Worked on code documentation. contributed to report.

Andrew: Worked on find closest point on triangle and mesh functions in Python. Contributed to unit tests. Worked on report.

8 Citation

Taylor, Russel. "Finding point-pairs." (2020). pgs 8-11, 64-80

9 Appendix

9.1 Debugging Data Results

Listing 4: Program Generated PA3-A-Debug-SampleReadingsTest.txt

```
15, pa3-A-Output.txt
  -11.24      0.67     15.07     -11.24      0.67     15.07     0.000
  -10.17    -18.47    -44.87     -10.17    -18.47    -44.87     0.004
   -9.09     17.46     38.95      -9.09     17.46     38.95     0.000
   -8.68    -24.66    -14.32      -8.68    -24.66    -14.33     0.004
   27.77     16.75     16.42      27.77     16.75     16.42     0.000
   -5.36     16.56     44.30      -5.35     16.56     44.30     0.004
   31.65     -8.90    -16.83      31.65     -8.90    -16.83     0.001
  -36.07     -4.49    -42.48     -36.07     -4.49    -42.48     0.000
    2.34     -8.12     26.83       2.34     -8.12     26.83     0.001
    5.85      6.61    -16.38       5.85      6.61    -16.38     0.001
   -7.14      2.27     33.68      -7.14      2.27     33.68     0.001
   11.35     -0.75    -21.57      11.35     -0.75    -21.57     0.001
   28.08     -0.35     21.09      28.08     -0.35     21.09     0.000
    1.29    -17.86     -9.68       1.28    -17.86     -9.69     0.002
   11.39     18.21    -25.64      11.38     18.20    -25.64     0.003
```

Listing 5: Program Generated PA3-B-Debug-SampleReadingsTest.txt

```
15, pa3-B-Output.txt
   19.99    -13.91     -3.15      20.14    -14.18     -2.94     0.368
  -18.41     -3.42    -50.11     -19.14     -4.41    -47.99     2.450
  -17.71    -28.07    -14.58     -17.56    -28.56    -14.25     0.606
   24.66     21.75     -0.06      25.78     23.94      0.13     2.477
   -6.95      8.78     56.62      -6.64      8.68     56.62     0.328
  -37.11     -1.92    -43.08     -36.04     -2.25    -41.89     1.635
  -27.27    -20.85    -10.01     -27.80    -22.48     -8.47     2.305
  -35.41    -28.01    -20.36     -34.69    -27.08    -20.64     1.199
  -19.41     -0.06    -47.88     -19.24     -0.81    -46.70     1.409
    0.80     18.76      3.23       1.65     18.24      3.80     1.141
  -43.01     -3.17    -30.31     -42.57     -3.28    -30.16     0.482
    6.27     18.66     46.43       5.99     20.26     46.54     1.625
   32.45     -9.01    -15.63      32.12     -8.67    -15.57     0.476
   20.68     23.44    -10.35      21.38     25.44    -10.31     2.117
  -23.47    -34.62    -26.77     -23.36    -32.82    -26.81     1.801
```

Listing 6: Program Generated PA3-C-Debug-SampleReadingsTest.txt

```
15, pa3-C-Output.txt
   18.26     24.06    -26.02      18.15     23.77    -25.75     0.409
  -30.60     -8.23     -9.38     -30.64     -7.37     -7.77     1.830
```

```
 23.69   -13.92    -7.10    23.85   -14.11    -7.02   0.253
 34.94    -3.86     6.60    35.28    -4.21     6.71   0.504
-15.36    22.46    25.64   -14.91    21.96    25.65   0.679
-12.29    15.83    34.20   -12.95    15.43    34.86   1.008
 25.05    -8.65     5.61    25.05    -8.45     5.54   0.211
 32.34     2.52   -26.02    31.95     2.57   -25.76   0.473
-13.28     4.16    10.85   -12.57     4.21    10.75   0.718
 32.29     6.90    12.13    33.72     7.23    13.13   1.774
 -4.08    -2.12   -34.63    -3.93    -2.08   -34.66   0.156
  6.16   -13.48     1.58     6.14   -13.57     1.60   0.090
-19.96   -16.48    -8.12   -20.12   -17.09    -6.25   1.979
-36.80   -12.75   -11.70   -37.49   -12.92   -11.01   0.993
 16.42    -3.45    43.94    17.12    -4.63    44.07   1.378
```

Listing 7: Program Generated PA3-D-Debug-SampleReadingsTest.txt

```
15, pa3-D-Output.txt
-44.17   -15.58   -27.09   -43.36   -15.37   -27.15   0.841
-30.77     4.55   -40.74   -30.59     4.24   -40.29   0.574
-29.91     2.30   -16.28   -29.74     2.88   -13.62   2.723
-39.49     2.03   -30.50   -39.51     2.04   -30.50   0.021
 -1.62     9.49    62.18    -1.62     9.55    63.36   1.172
-21.55   -33.73   -33.87   -21.42   -32.08   -33.35   1.739
 25.83    21.23     4.78    26.73    23.03     4.94   2.016
-22.65     1.12   -46.49   -22.68     0.68   -45.99   0.670
  3.45   -10.56    14.62     3.45   -10.57    14.63   0.006
-13.18   -14.14    -2.95   -13.15   -14.01    -3.08   0.182
  9.17   -17.66    -4.43     9.68   -16.18    -4.52   1.566
-34.56     4.97   -35.62   -34.54     4.94   -35.61   0.034
 20.54     3.74    37.80    24.10     3.19    38.41   3.658
 14.93    -7.14    21.00    14.89    -6.90    20.98   0.243
 21.49    23.54   -10.13    22.06    25.20   -10.10   1.750
```

Listing 8: Program Generated PA3-E-Debug-SampleReadingsTest.txt

```
15, pa3-E-Output.txt
-18.28   -35.06   -29.92   -18.67   -32.52   -29.70   2.572
 -1.34     4.25    61.50    -1.37     4.29    63.38   1.885
 22.75    20.71     8.81    23.77    23.39     9.40   2.920
  6.86    24.59    11.60     6.73    25.07    11.60   0.488
 30.31     3.20   -27.16    30.92     3.13   -27.52   0.710
 -9.94   -14.46     0.35    -9.73   -13.05    -0.58   1.693
 15.95    -0.56   -25.54    17.57    -0.29   -22.69   3.288
 17.25     5.44    53.64    21.69     4.57    54.25   4.569
 -3.09   -10.71   -30.63    -0.92    -9.50   -31.80   2.752
  7.58   -15.50     2.05     7.96   -14.01     1.67   1.574
-18.77   -17.45   -49.76   -19.14   -16.95   -47.87   1.986
```

```
-12.58    -29.17    -37.31      -12.84    -28.56    -37.00     0.724
 -7.33     -4.16     45.71       -4.90     -2.73     45.39     2.843
 -5.38    -11.50     10.15       -5.02    -10.03      9.75     1.568
-11.15     -1.74     25.38       -8.27     -1.31     24.75     2.974
```

Listing 9: Program Generated PA3-F-Debug-SampleReadingsTest.txt

```
15, pa3-F-Output.txt
 -2.16    -13.52    -41.44      -3.66    -13.30    -40.23     1.935
 25.93    -13.44     -0.85      25.62    -11.27     -1.79     2.378
-20.74      2.35    -43.11     -20.61      3.01    -43.87     1.013
-19.61    -14.27    -49.98     -19.73    -13.94    -48.92     1.117
  1.97    -18.64    -27.57       0.53    -18.09    -27.40     1.547
 29.71      6.26    -28.52      29.74      6.25    -28.58     0.068
 -5.41    -14.96      1.45      -5.40    -13.04      0.80     2.021
  1.63     22.08     13.70       1.14     23.02     13.09     1.222
-21.62     -3.67     -8.83     -22.93     -2.77     -7.45     2.101
  2.73     -7.24     63.18       2.77     -6.98     63.09     0.280
 -3.53    -14.36      3.79      -3.63    -12.36      2.76     2.251
 24.63     -7.11     16.25      24.11     -5.71     15.87     1.542
 -6.35     -3.80    -42.61      -6.68     -4.03    -42.23     0.554
 21.14     21.30     13.65      21.82     23.10     14.04     1.968
-38.51    -13.17    -37.26     -39.78    -13.29    -37.50     1.290
```