Programming Assignments 1 Report

Arijit Nukala and Ilana Chalom
Computer Integrated Surgery 1, 601.455

1: Cartesian Math Package

For our program, we used native Python functions and various libraries—namely Numpy, Scipy, and Pandas— to help perform frame transformations, rotations, and 3D point cloud representations. [3], [4], [5]. For questions 4, 5, and 6, we were able to use a combination of these libraries and the mathematical functions we wrote to compute the answers. We relied on the homework instructions and the ways we learned to solve these problems in class, and applied this understanding to these questions, only this time, using our program to do the mathematical computations for us. For instance, by writing more foundational functions, such as composition and transformation, we were able to use these as building blocks, both within our more complex algorithmic functions, and in answering questions 4, 5, and 6 of the homework.

2: Algorithmic Approach

For our **registration algorithm**, we used an approach developed by Arun et al., in which the authors explored a non-iterative algorithm which employs Singular Value Decomposition [1]. Additionally, we compensated for rotation matrices with a determinant of negative one using an approach described by Sorkine-Hornung et al [7]. The algorithmic approach using can be broken down into roughly five or so steps:

1. Center the point sets
2. Calculate a matrix H by multiplying one matrix by the transpose of the other
3. Find the SVD of H
4. Use the results of the SVD to calculate a matrix X
5. Calculate the determinant of X: it should be 1 and correct the matrix if the determinant is negative one

For step one, calculated the mean of each point cloud then subtracted the mean from the points in the cloud to center:

```
a_mean = np.mean(A.points, axis=1, keepdims=True)
b_mean = np.mean(B.points, axis=1, keepdims=True)
centered_a = A.points - a_mean
centered_b = B.points - b_mean
```

For step 2, we computed the 3 x 3 matrix H:

```
H = np.dot(centered_a, centered_b.transpose())
```

For step 3 we found the SVD of H:

```
u, s, vt = np.linalg.svd(H)
```

For step 4, we used U and Vt to find the rotation matrix:

```
u = u.transpose()
vt = vt.transpose()
```

For step 5, we validated the result, and accounted for potential problems in determinant value:

```
comp_size = vt.shape[0]
reflection_comp = np.eye(comp_size)
reflection_comp[comp_size - 1][comp_size - 1] = np.linalg.det(np.dot(vt, u))
```

And with the newly found rotation matrix, we returned a Frame:

```
R = np.dot(vt, np.dot(reflection_comp, u))
 p = b_mean - np.dot(R, a_mean)
return Frame(R, p)
```

For the pivot calibration algorithm, we used Ziv Yaniv's paper titled "Which Pivot Calibration?" [10]. We first got the mean of the points in the first frame and used this to define a coordinate system for the probe. We then subtracted the mean from the rest of the points to convert these points into the frame of the probe. Lastly, we solved the overdetermined system described in the paper using the Algebraic One Step (ATS) method. Using these steps we were able to solve for the pivot.

3: Program Structure

This program was written in Python and uses various numerical libraries for algebraic manipulation and linear algebra. The transforms in our program are represented as frame objects with a rotational and translational components. Our testing is performed through the PyTest library [2] and command line arguments are run using Click.

Our program contains 6 important files which stem from the top level PROGRAMS directory:
cis.Frame.py: this file contains the Frame class, which has the function for transformation, composition, and inversion.
cis.point_set.py: this file contains the pointSet class, which has the function for point set registration, employing Arun's algorithm
cis.file_rw.py: this file contains methods to read in the different type of files and return the appropriate point sets (ie: getDataCalBody, getDataCalReading, getEmpivot, getOptpivot)
cis.pivot_cal.py: this file contains the function for the pivot calibration algorithm
cis.prob456.py: this file contains the functions which take the appropriate steps to incorporate our other functions/algorithms and answer question 4, 5, and 6
pa_one.py: this file contains the main method and a function to write the output.

The structure of our program can be broken down by the different paths and directory in the pa_one directory. The OUTPUT folder contains the output files for the different known and unknown debugging cases. The PROGRAMS folder contains all of the programs. Inside PROGRAMS, is the README.txt and the pa_one.py (containing the main method to run). There are then three other folders, cis, which contains the all the other files mentioned above, data, which contains all of the input files, and test, which contains files for unit testing.

4: Verification

To verify that our program and algorithms worked correctly, we first checked with the debugging examples. Our outputs for the debugging files can be found in pa_one/OUTPUT. Of particular importance are the point_set and pivot_cal files for algorithmic operations on our data. To test pivot calibration, we created a unit test that writes a random rotation and point set, rotates the point set to create a new point set, and checks our method against the written random rotation. Additionally, to test our pivot calibration in pivot_cal, we compared our results from solving question 5 to the given output results.

Our pivot calculation= 200.03, 199.52, 195.07
EM pivot post actual position = 200.02, 199.52, 195.07

As we can see, our values are nearly identical, and offsets can be explained by slightly differences in mathematical calculations attributed to rounding or to the specific methods we used.

5: Results

For our results to file unknown H, we yielded:

	27	8	pa1-unknown-h
Estimated post position with EM probe pivot calibration	185.16	202.74	177.69
Estimated post position with optical probe pivot calibration	395.79	400.4	190.14
Frame 1 Coordinates of C1	210.11	211.73	209.2
.....	210.45	208.21	334.15

For file i, we yielded:

	27	8	pa1-unknown-i
Estimated post position with EM probe pivot calibration	192.24	194.3	198.2
Estimated post position with optical probe pivot calibration	381.97	401.75	190.64
Frame 1 Coordinates of C1	212	211.25	211.33
.....	212.58	212.85	336.32

For file j, we yielded:

	27	8	pa1-unknown-j
Estimated post position with EM probe pivot calibration	196.76	194.55	197.76
Estimated post position with optical probe pivot calibration	408.1	406.76	198.6

Frame 1 Coordinates of C1	211.69	210.55	208.18
.....	208.25	211.22	333.13

For file k, we yielded:

	27	8	pa1-unknown-k
Estimated post position with EM probe pivot calibration	202.69	191.91	206
Estimated post position with optical probe pivot calibration	360.96	411.83	192.13
Frame 1 Coordinates of C1	211.42	208.9	208.9
.....	212.74	211.62	333.86

6: Discussion

Based on unit tests and visual comparisons of our output files with the given outputs, our results suggest we implemented our point cloud calibration and pivot transformation algorithms correctly. For our tests cases, we tested for an ideal case in which noise does not exist. In the future, we can improve our testing by including methods to calculate error, generating noisy data to better test our algorithmic approach, and include unit testing for incorrect data types to ensure our program is only taking the correct data inputs.

Our results for the debugging files with noise included indicate that our program may have some error (visually, errors were up to 5-6mm or more) when running calculations on debug-f and debug-g especially. For all other debugging files, our computed values were reasonably close to the estimated positions in the auxiliary1 data files.

7: Contributions

Ilana: I wrote the Frame.py file containing the frame class, transformation function, composition, function, and inversion function. I also created the point_set.py file, which included the pointSet class, the registration function (I also researched the different types of registration algorithms to find Arun's method), and all of the reading file functions: getDataCalBody, getDataCalReading, getEmpivot, getOptpivot. I also created the prob456.py file, in which I answered question 4. Lastly, I wrote some of the write-up report (sections 1, 2, 3, 5, 7, and the bibliography)..
Arijit: Arijit wrote pa_one.py and the corresponding methods, pivot_cal.py which contains the pivot calibration method, problems 5 and 6 of the prob456.py file, and the README.txt. Additionally, Arijit created the project structure, completed debugging and testing of the program, and analyzed program outputs. Arijit also contributed algorithm, data, program structure, and discussion sections of the write-up report.

Bibliography:
[1] Arun, K. S., et al. *Least Squares Fitting of Two 3-D Point Sets*. IEEE, Sept. 1997, www.ece.queensu.ca/people/S-D-Blostein/papers/PAMI-3DLS-1987.pdf.

[2] Florian, et al. “Pytest-Dev/Pytest.” *GitHub*, 13 Oct. 2022, github.com/pytest-dev/pytest/blob/main/CITATION. Accessed 13 Oct. 2022.

[3] Harris, Charles R., et al. “Array Programming with NumPy.” *Nature*, vol. 585, no. 7825, 16 Sept. 2020, pp. 357–362, 10.1038/s41586-020-2649-2.

[4] “Logging — Logging Facility for Python — Python 3.10.0 Documentation.” *Docs.python.org*, docs.python.org/3/library/logging.html.

[5] McKinney, Wes. “Data Structures for Statistical Computing in Python.” *PROC. OF the 9th PYTHON in SCIENCE CONF*, 2010.

[6] “Point-Set Registration.” *Wikipedia*, 2 July 2022, en.wikipedia.org/wiki/Point-set_registration. Accessed 13 Oct. 2022.

[7] Sorkine-Hornung, Olga, and Michael Rabinovich. *Least-Squares Rigid Motion Using SVD*. 16

Jan. 2017, igl.ethz.ch/projects/ARAP/svd_rot.pdf. Accessed 13 Oct. 2022.

[8] Virtanen, Pauli, et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python.” *Nature Methods*, vol. 17, no. 3, 3 Feb. 2020, pp. 261–272, 10.1038/s41592-019-0686-2.

[9]“Welcome to Click — Click Documentation (8.1.x).” *Click.palletsprojects.com*, click.palletsprojects.com/en/8.1.x/.

[10] Yaniv, Ziv. “Which Pivot Calibration?” *SPIE Proceedings*, 18 Mar. 2015, www.yanivresearch.info/writtenMaterial/pivotCalib-SPIE2015.pdf, 10.1117/12.2081348. Accessed 13 Oct. 2022.