BLG 202E - Numerical Methods in Comp. Eng. Spring 2024 - Term project

Işıl Altınışık - 150220308

May, 2024

1 INTRODUCTION

Point set registration is a fundamental task in many fields like shape analysis, object recognition, image processing and more. The Kabsch-Umeyama algorithm provides an efficient method to align two different point sets with corresponding points. This project presents an implementation of Kabsch-Umeyama algorithm in python. This report details the steps involved. Additionally, an alternative way for SVD (singular value decomposition) is introduced.

2 IMPLEMENTATION

The Algorithm starts by finding the centroids of the points sets (which in this case, also called their arithmetic means). Centroids are denoted by μ. Then these points sets are centered. After that, the covariance matrix is calculated using the formula below.

$$H = \frac{1}{n}\sum_{i=1}^{n}(a_i - \mu_A)(b_i - \mu_B)^T$$

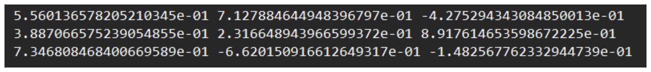
Next, singular value decomposition (SVD) is applied to covariance matrix to extract the rotation matrix (R). Then the translation vector (t) is calculated using the difference of found centroid of A and the rotation matrix (R)

In SVD function (which computes singular value decomposition of given matrix) classical procedure is followed and "Eigen-decomposition method" is used. It involves finding the eigenvectors and eigenvalues of the matrix A$^T A$, then using them to compute the singular vectors and singular values of A. Also to find the eigenvalues, in the "eigenvalues()" function, QR decomposition method is used. Finally, formula below is used to compute the merge matrix.

$$B_{d\times n} = R_{d\times d}A_{d\times n} + T_{d\times n}$$

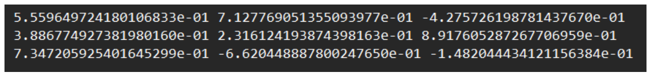
3 DATA

To test the code, ”plane-example” data that comes with the dataset is used. Using the ”plane-example” dataset, all the steps applied to ”mat1.txt” and ”mat2.txt” files and rotation matrix (R) and translation vector (t) is found with a considerably small error. According to output of the code, both matrices are given below with the real values in Figure 1 and Figure 2.



```
5.560136578205210345e-01 7.127884644948396797e-01 -4.275294343084850013e-01
3.887066575239054855e-01 2.316648943966599372e-01 8.917614653598672225e-01
7.346808468400669589e-01 -6.620150916612649317e-01 -1.482567762332944739e-01
```

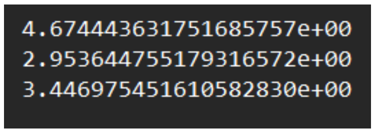
Figure 1: Real Rotation Matrix Given in the Dataset



```
5.559649724180106833e-01 7.127769051355093977e-01 -4.275726198781437670e-01
3.886774927381980160e-01 2.316124193874398163e-01 8.917605287267706959e-01
7.347205925401645299e-01 -6.620448887800247650e-01 -1.482044434121156384e-01
```

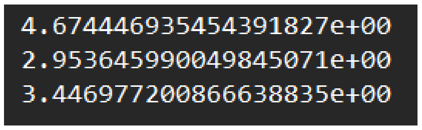
Figure 2: Calculated Rotation Matrix

Also calculated and real translation vectors are given in the Figure 3.1 and Figure 3.2



```
4.674443631751685757e+00
2.953644755179316572e+00
3.446975451610582830e+00
```

(a) Figure 3.1



```
4.674446935454391827e+00
2.953645990049845071e+00
3.446977200866638835e+00
```

(b) Figure 3.2

4 EXPERIMENTS

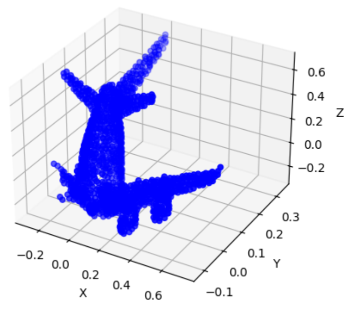
Using the ”plane example” data first points set, second points set, and the results of Kabsch-Umeyama algorithm is shown in the Figure 4.1, Figure 4.2 and Figure 5.

5 REFERENCES

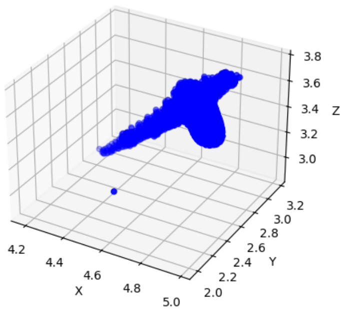
[1] Jim Lawrence, Javier Bernal, and Christoph Witzgall. A purely algebraic justification of the kabsch-umeyama algorithm. Journal of Research of the National Institute of Stan dards and Technology, 124, October 2019.

[2] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. IEEE Transactions on Pattern Analysis Machine Intelligence, 13(04):376–380, 1991.

[3] Tuomas Siipola. Aligning point patterns with Kabsch–Umeyama algorithm. March 2021. https://zpl.fi/aligning-point-patterns-with-kabsch-umeyama-algorithm/



(a) Figure 4.1: 3D PLot of First Point Set



(b) Figure 4.2: 3D Plot of Second Point Set

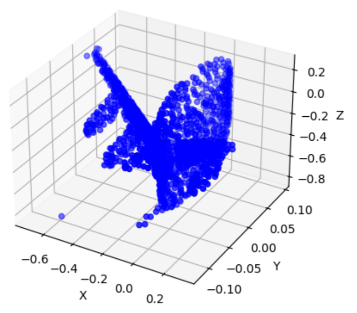


Figure 5: 3D Plot of Merged Point Set