BLG202E Numerical Methods in Comp. Eng.

Spring 2024 - Term Project

Due: May 5, 2023

By turning in this assignment, I agree by the ITU honor code and declare that all of this is my own work.

Important Notes

- Upload your solutions through **Ninova**. Homeworks sent via e-mail and late submissions **will not be accepted**.
- Please make sure that you write your **full name** and student identification **number** to **every file** you submit.
- Cheating is highly discouraged. It will be punished by a negative grade. Also disciplinary actions will be taken. Please do your homework on your own. Team work is not allowed. Pattern of your **solutions must belong only to you**.
- All codes and reports will be run through **plagiarism check**s. Please **do not copy any text or code** from other sources.
- If you have any questions, please contact with **Batuhan Cengiz** (cengiz16@itu.edu.tr).
- Remember, there are only 10 types of people in the world – those who understand binary, and those who don't.

Project No 5: Point Set Registration

Please read the following prerequisites.

- Install a Conda environment if you do not have it already.
- Install Jupyter Notebook (Optional).
- The project should be done in a Python script. You may use Jupyter Notebooks for data inspection and visual generation.
- Do not forget to format your code and leave comments for non-trivial sections.
- You are may use matplotlib, numpy / SciPy, json and pandas libraries.

Problem Description

In this project, you will implement Kabsch-Umeyama Algorithm[1, 2] in python. We use this algorithm on to merge two different point sets based on common points (correspondences). It is a useful algorithm in areas like 2D/3D point-set registration and protein structure comparison. An example use case is shown in this video.

It is an algorithm to estimate the rigid transformation of a point set. For given two point sets P, Q

$$
\begin{aligned}
& P_{d\times n} = R_{d\times d} Q_{d\times n} + T_{d\times n} \\
\text{s.t.} \quad & R^T R = \mathbf{I} \\
& T_{d\times n} = \big(\underbrace{\mathbf{t}_{d\times 1} \quad \mathbf{t}_{d\times 1} \quad \cdots \quad \mathbf{t}_{d\times 1}}_{N \text{ elements}}\big)
\end{aligned}
\tag{1}
$$

Here R correspondences to rotation matrix and t is the translation vector. An example is given in Figure 1. We can estimate values for R and t using Kabsch-Umeyama algorithm.

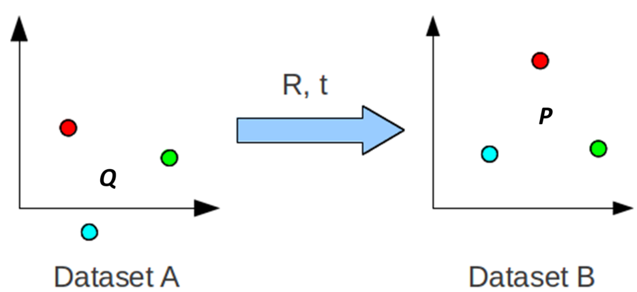

Figure 1: A Rigid Transofmration

Instructions

- Re-implement Kabsch-Umeyama Algorithm from scratch, along with your own Singular Value Decomposition (SVD) implementation.
  - Read the papers by **Umeyama** [2] and **Lawrence et. al.** [1] to understand the algorithm. You do not have to read the full paper for Lawrence et. al. [1], it would be enough to read the introduction section only.
  - Implement the algorithm based on **pseudo-code** given at Lawrence et. al [1].
  - You are also given a few sample point sets from **ModelNet40** [3] dataset Apply the algorithm on shared data to estimate R and t parameters.
  - Apply inverse rotation/translation operation on second point set and merge the two sets. You may refer to **Inputs**, **Outputs** and **Notes** sections for details.
- Report:
  - Write a maximum of 3 pages report using IEEE Latex Template. State the problem, implementation details, available data and the experiments.
  - Try to add visual and numeric results to your report such as two point sets before and after the merge operation, estimated values of R and t parameters.
- Submission
  - Submit a zip file that includes your Python code, plots, and results until the deadline through Ninova.
  - Upload your project report until the deadline through Ninova.

Inputs

- **mat1.txt**: The first set of points. Its corresponding indices create Q matrix.
- **mat2.txt**: The second set of points. Rotation and translation operations are applied to this point set. Its corresponding indices create P matrix.
- **correspondences.txt**: Shows which point is equal to which point between set 1 and set 2. For each row, the first column indicates position(row no) of points in first set, and the second column indicates position(row no) of point in the second set.

Outputs

- **rotation_mat.txt**: Calculated rotation matrix.
- **translation_vec.txt**: Calculated translation vector.
- **merged.txt**: Merged point set of mat1 and inverse rotated-translated mat2.

Notes

- After estimating R and t values with the Kabsch-Umeyama Algorithm, you can apply inverse operations on the second set with R^T and $-t$ using Eq. 1.
- You can merge the two sets after inverse operation to recover full data. You can drop the overlapping points from the second set during merge operation to remove any duplication.
- You may use numpy for matrix operations (e.g. matrix multiplication, determinant).
- Your python script should execute with the following the command.

```
python main.py mat1.txt mat2.txt correspondences.txt
```

- Your python script should output the following files. You may use "numpy.savetxt()" method to save outputs in correct format.

```
rotation_mat.txt
translation_vec.txt
merged.txt
```

- You are expected to write code and get numerical results. Submissions without a working code will not be graded.
- Please do not copy any code from your friends or the internet. Any kind of cheating will be graded negatively.
- You need to implement SVD from scratch. You can't use built-in libraries or methods like 'numpy.linalg.svd', 'scipy.linalg.svd', 'sklearn.decomposition' etc. Please define svd explicitly as a function and add as appendix to your report.

```
import numpy as np

def SVD(Matrix):
    ...
    return U, E, Vt
```

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 Standards 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] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE/CVF CVPR*, pages 1912–1920, 2015.