Numerical Methods Term Project

Melike Beşparmak
150220061 - besparmak22@itu.edu.tr

***Abstract*—This document is a report of project 5.**

I. Introduction

This report aims to introduce an implementation of Kabsch-Umeyama algorithm with Python.

II. Methods

A. Kabsch-Umeyama Algorithm

The Kabsch-Umeyama algorithm is used to tranform a dataset to another one using the Singular Value Decomposition (SVD) method. A pseudo algorithm is as follows:

Compute $d \times d$ matrix $M = QP^T$.
Compute SVD of M, i.e., identify $d \times d$ matrices V, S, W, so that $M = VSW^T$ in the SVD sense.
Set $s_1 = \ldots = s_{d-1} = 1$.
If $\det(VW) > 0$, then set $s_d = 1$, else set $s_d = -1$.
Set $\tilde{S} = \text{diag}\{s_1, \ldots, s_d\}$.
Return $d \times d$ rotation matrix $U = W\tilde{S}V^T$.

Fig. 1. Algorithm [1]

B. Approaching to the Problem

The algorithm uses SVD to obtain a rotation matrix and some other computations. So what is needed to be done is, implementing an SVD function and following the procedure presented in (*figure 1*).

Following a step-step approach, I used the "numpy.linalg.svd" function to see if the algorithm was correctly implemented. In the implementation, a covariance matrix is found by computing a dxd matrix QP^T. "d" indicates the dimensions we are working with therefore it is 3. To obtain this 3x3 matrix, two compatible datasets are needed. Correspondences matrix is used to calculate two matrices with the same size and the matrices should be centered by subtracting the centroid. Since we are working in 3D, the covariance matrix is $M = QP^T$. The SVD of this matrix will be $M = VSW^T$ and rotation matrix $U = WS'V^T$. The new S' matrix is an identity matrix where the third diagonal entry is replaced with -1 if there is a reflection. This is decided according to the determinant of VW. After finding the rotation matrix, the translation vector is found by solving the equation:
$T = CentroidP - MCentroidQ$

This translation vector is transformed into a matrix and now we have the equation:
$P_{dxn} = R_{dxd}Q_{dxn} + T_{dxn}$

By inverting the operation we just did, the P matrix transformed to Q is found as:
$Q' = M^{-1}X(P' - T)$
Where P' represents the original matrix before centering.

The last step is to merge the two sets. Since there are corresponding points, the duplicates are removed and the merged set which covers both is obtained. After implementing and verifying the algorithm, the next step is implementing an SVD function. Most of the methods fail when the dataset is large, therefore I chose to use a QR Method to find an orthogonal matrix and eigenvalues. By using the Gram-Schmidt Process, the QR factorization is obtained. Q matrix can be used as $'V'$ matrix directly which represents the left-singular eigenvectors of the covariance matrix. It is important to note that the SVD function decomposes the covariance matrix M and the QR method factorizes MM^T. The eigenvalues are found by iterating the QR method, since the matrices are similar the diagonal entries will converge to eigenvalues. Now, both $'V'$ and $'S'$ matrices are found and the only matrix left is W^T.
$M = VSW$ so $S^{-1}V^TM = W$
Finally, we obtained the SVD. The rest is as stated previously.

C. Results

The plotted merged sets:

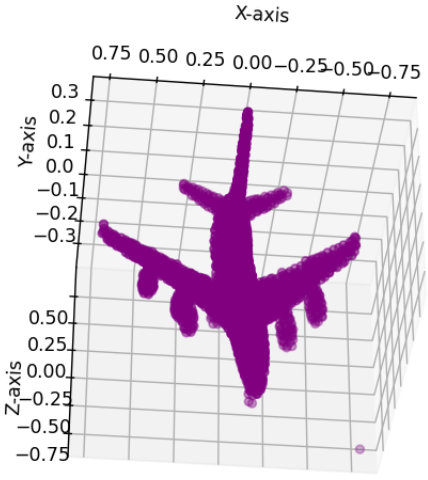

Fig. 2. Plane

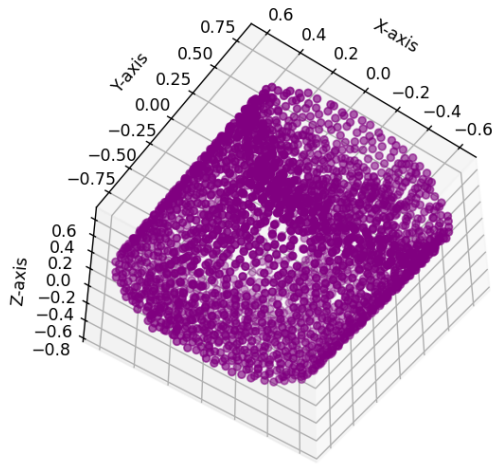

Fig. 3. Cup

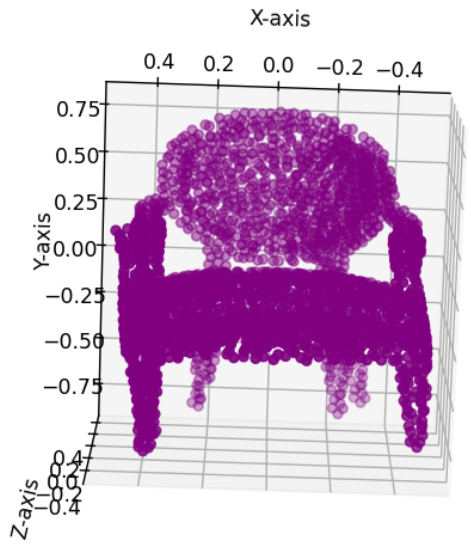

Fig. 4. Chair

D. SVD Function

```python
def manual_qr_and_svd(A):
    Q, _ = qr(A @ A.T)
    V = Q # left-singular vectors

    sorted_eigvals = eigvals_qr(A @ A.T)
    Sigma = np.sqrt(sorted_eigvals)
    Sigma = np.diag(Sigma) # singular values in descending order

    inv_s = np.where(Sigma != 0, 1 / Sigma, 0)  # inverse of a diagonal matrix
    Vt = inv_s @ V.T @ A # compute V -> right-singular vectors
    Vt_normalized = np.apply_along_axis(lambda v: v / np.linalg.norm(v), axis=0, arr=Vt) # normalize v

    return V, Sigma, Vt_normalized
```

Fig. 6. SVD

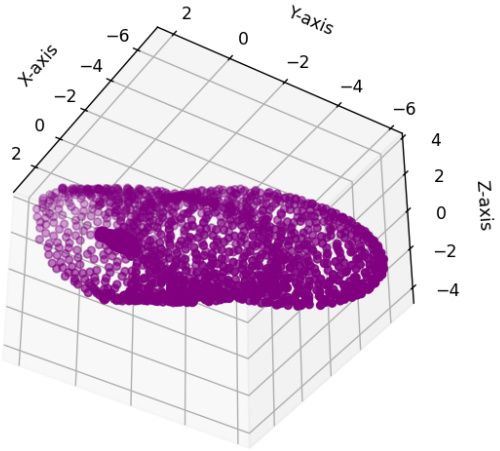

Fig. 5. Bottle

REFERENCES

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.