BLG 202E Term Project 5

Berk Özcan
Computer Engineering
BLG 202E
Istanbul, Türkiye
ozcanbe22@itu.edu.tr

***Abstract*—This document about is a report for implementation of Kabsch-Umeyama alghoritm and its examples. It also contains implementations of SVD and QR algorithms.**

***Index Terms*—kabsch,umeyama,svd,qr**

I. Introduction

In this project, it is asked to implement Kabsch-Umeyama algorithm to transform a dataset based on another dataset. To implement this algorithm, there is also used singular value decomposition for using on the algorithm and QR algorithms for finding eigenvalues and eigenvectors.

II. Ease of Use

The main use of this algorithm is transforming one data set with rotation and translation matrices. This algorithm helps us to align and compare similarities of two different datasets. In this project, it is used to align two parts of an object on common points and bringing them into one single object.

III. Implementation Details

A. Implementation of QR Decomposition and Algorithm

Since it is not allowed to use built-in methods for finding eigenvalues and eigenvectors. It is needed to implement QR algorithm for finding these values and vectors. QR algorithm is a iterative method that converge to the real eigenvalues in each iteration.

$$A(k+1) = R(k)Q(k) \tag{1}$$

Every next A matrix is a multiplication of previous matrices' upper triangle matrix and orthogonal matrix. End of this iterations, more iteration is more closer to the result, A matrix's diagonal becomes eigenvalue of the A matrix.

To find eigenvectors in this algorithm, in each iteration base unit matrix is multipicated with Q(k). Therefore the final **nxn** matrix has columns of the eigenvectors

B. Implementation of SV Decomposition

Since we can not use built-in methods to find singular value decomposition, it is implemented the method of this decomposition based on the linear algebra rules.

$$A = UEVt \tag{2}$$

First, it is calculated that E matrix which is a diagonal matrix that contains singular values. To find singular values it is used:

$$\sigma = \sqrt{\lambda} \tag{3}$$

Each diagonal index of the E matrix has singular value in descending order.

Therefore, when trying to find Vt matrix, eigenvectors are comes in handy. The matrix created as each eigenvector is creating column of this matrix. Of course, the vectors needed to be normalized by their size. Afterwards, it is needed to be transposed for creating Vt matrix.

Finally, to get U matrix, we use an equation that:

$$u1 = (1/\sigma1)Av1 \tag{4}$$

A matrix is multipicated with row of a Vt matrix. Then, it is needed to multipicated with a scalar of corresponding singular value. Therefore, each *u* vector creates columns of the U matrix.

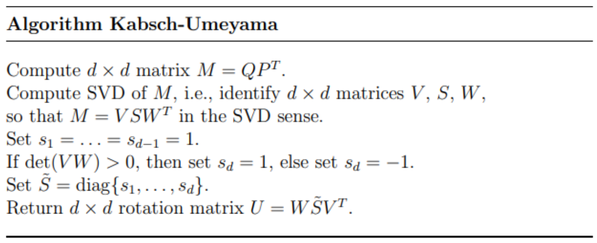
For better contrast it is given below the matrices of the decomposition:

$$\mathbf{A}_{m\times n} = \underbrace{\begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_r \\ | & | & & | \end{bmatrix}}_{\hat{\mathbf{U}}_{m\times r}} \underbrace{\begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r \end{bmatrix}}_{\hat{\mathbf{\Sigma}}_{r\times r}} \underbrace{\begin{bmatrix} — & \mathbf{v}_1^T & — \\ — & \mathbf{v}_2^T & — \\ & \vdots & \\ — & \mathbf{v}_r^T & — \end{bmatrix}}_{\hat{\mathbf{V}}^T_{r\times n}}$$

Fig. 1. Singular Value Decomposition Matrices

C. Implementation of Kabsch-Umeyama Algorithm

It is a strong algorithm that creates rotation and translation matrix for transforming a dataset based on the another.



```
Algorithm Kabsch-Umeyama

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 $\tilde{S} = \mathrm{diag}\{s_1, \ldots, s_d\}$.
Return $d \times d$ rotation matrix $U = W\tilde{S}V^T$.
```

Fig. 2. Psuedocode of Kabsch-Umeyama Algorithm

We have a two matrix that is Q and P from the datasets given. They multiplied into a dxd sized M matrix which is 3x3 in this project. After that, we do singular value decomposition which we implemented earlier. We get U, E and Vt matrices from the decomposition. Therefore, we changed the diagonal of the E matrix to the 1s except the last one. On last one, it needed to be checked that determinant of multipication is positive or negative. If it is positive we set that value also to 1, if not we set -1. Therefore we get rotation matrix with multipication of UEVt with the new E matrix. To obtain translation vector we used the equation:

$$V = RQ + T \tag{5}$$

T is translation matrix we get mean value of the P and Q matrix and apply.

$$t = v - Rq \tag{6}$$

We know that rotation matrix R is symmetric that means inverse of the R is equals to transpose of R. We get equation that:

$$Q = Rt(P - t) \tag{7}$$

Therefore we found the transformed points from dataset P based on dataset Q.

IV. Results

My implementation of Kabsch-Umeyama algorithm was not successful. However, since I get numerical results on the project I wanted to share these results. The translation of the dataset was correct, but rotation matrix was wrong.

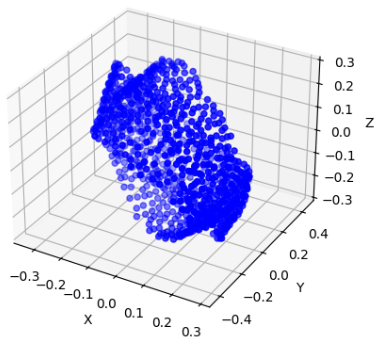


Fig. 3. Bottle Plot

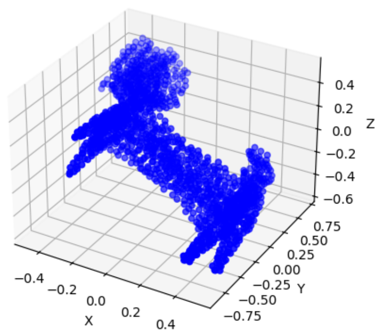


Fig. 4. Chair Plot

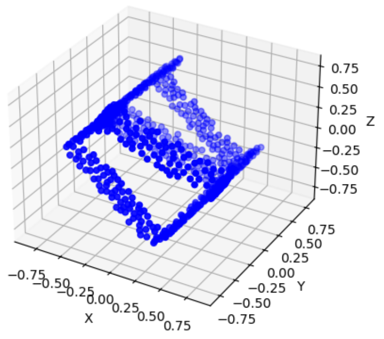


Fig. 5. Cup Plot

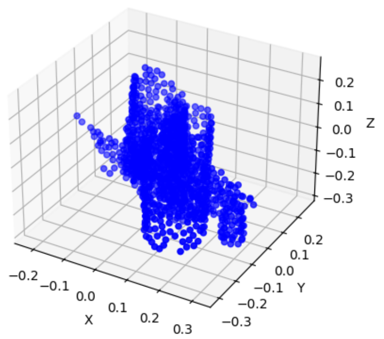


Fig. 6. Plane Plot

TABLE I
ROTATION MATRIX OF BOTTLE

-8.412293993124217995e-01	4.275937811542989547e-02	5.389849101001683573e-01
-4.950758621688140515e-01	3.397846824645224251e-01	7.996538377699864419e-01
-2.173315173420929391e-01	-9.395307366688419037e-01	2.646677283388308433e-01

TABLE II
TRANSLATION VECTOR OF BOTTLE

1.531320133074785650e+00
1.232078284360718312e+00
1.011217362716900769e+00

TABLE III
ROTATION MATRIX OF CHAIR

3.156507473608904935e-01	-4.093778147192854711e-01	-8.560224357493083636e-01
1.341905425838450239e-01	-8.738182078559928456e-01	4.673699154849386139e-01
-9.393388653577354752e-01	-2.623957782339575440e-01	-2.208867392883302472e-01

TABLE IV
TRANSLATION VECTOR OF CHAIR

1.441208404972887269e+00
1.417653531160504743e+00
1.330610024513311451e+00

TABLE V
ROTATION MATRIX OF CUP

-1.051451442082954185e-01	-5.458793444186533783e-01	-8.312394966887093339e-01
4.014262667595860229e-02	8.328638722659919003e-01	-5.520226360127831189e-01
9.936458044631756303e-01	-9.140909236881072930e-02	-6.565730243687303846e-02

TABLE VI
TRANSLATION VECTOR OF CUP

1.518438409013072921e+00
1.893759011516416901e+00
1.893540809912989120e+00

TABLE VII
ROTATION MATRIX OF PLANE

4.828187786937457870e-01	4.213060088058520614e-01	-7.677156196891598006e-01
8.755137969442988721e-01	-2.512645433339316670e-01	4.127247533396747881e-01
-1.901629696122041774e-02	-8.714168773964477976e-01	-4.901744648770765544e-01

TABLE VIII
TRANSLATION VECTOR OF PLANE

4.706703317265338526e+00
2.837501724988887641e+00
3.547352549815219636e+00