Point Set Registration and Image Stitching*

*Using Kabsch-Umeyama Algorithm

1st Sabyr Bazarymbetov
Department of Computer Engineering
Istanbul Technical University
Istanbul, Turkey
bazarymbetov22@itu.edu.tr

2nd Batuhan Cengiz
Department of Computer Engineering
Istanbul Technical University
Istanbul, Turkey
cengiz16@itu.edu.tr

3rd Abdulkerim Çapar
Department of Computer Engineering
Istanbul Technical University
Istanbul, Turkey
capar@itu.edu.tr

***Abstract*—This work presents a method based on the Kabsch-Umeyama algorithm for 3D point set registration and RGB image stitching. The method was developed as part of a course project for the Numerical Methods (BLG 202E) class.**

***Index Terms*—Kabsch, Umeyama, Point Cloud, Image Stitching, 3D points, RGB, SVD, Rotation Matrix, Translation Vector, Rigid Transformation**

I. Introduction

Point set registration is a fundamental task in computer vision and robotics, involving the alignment of two sets of data points in a common coordinate frame. Accurate and efficient point set registration has applications in 3D reconstruction, object recognition, medical imaging, and more. One of the most widely used methods for solving the rigid registration problem is the Kabsch-Umeyama algorithm.

The Kabsch algorithm provides a solution for finding the optimal rotation matrix that minimizes the root mean square deviation (RMSD) between two paired point sets. Umeyama extended this approach to include translation and optional scaling, allowing for a complete similarity transformation between the datasets. This project implements a version without scaling.

In this work, the Kabsch-Umeyama algorithm was applied to both 3D point cloud registration and 2D image stitching. The implementation demonstrates how the same underlying mathematical principles can be adapted to solve two related but distinct problems in spatial data alignment.

II. Methods

A. Singular Value Decomposition (SVD)

Given a real matrix $A \in \mathbb{R}^{m \times n}$, the Singular Value Decomposition (SVD) factorizes A as:

$$A = USV^T$$

This decomposition is fundamental for computing optimal transformations in least-squares problems.

where:

- $U \in \mathbb{R}^{m \times k}$ is a matrix with orthonormal columns,
- $S \in \mathbb{R}^{k \times k}$ is a diagonal matrix with non-negative real numbers (the singular values),
- $V \in \mathbb{R}^{n \times k}$ is a matrix with orthonormal columns,
- $k = \text{rank}(A)$.

The SVD is computed as follows (also known as economy SVD):

- If $m \geq n$, compute the Gram matrix $B = A^T A$ and its eigendecomposition $B = V \,\text{diag}(w) V^T$. The singular values are $S = \sqrt{\max(w, 0)}$, and $U = AV/S$.
- If $m < n$, compute $C = AA^T$ and its eigendecomposition $C = U \,\text{diag}(w) U^T$. The singular values are $S = \sqrt{\max(w, 0)}$, and $V = A^T U/S$.
- Economy SVD was used to avoid numerical instability similar to division by 0 by using tol parameter in the implementation.

Here, Q represents the source point set and P the target point set. Given two sets of corresponding points $Q, P \in \mathbb{R}^{n \times d}$, the Kabsch-Umeyama algorithm finds the optimal rotation $U \in \mathbb{R}^{d \times d}$ and translation $t \in \mathbb{R}^d$ that minimize the mean squared error between P and the transformed Q:

$$\min_{U,t} \|P - (UQ^T)^T - t\|_F^2$$

subject to U being a proper rotation matrix ($U^T U = I$, $\det(U) = 1$).

The algorithm proceeds as follows:

1) Compute the centroids:

$$\mu_Q = \frac{1}{n} \sum_{i=1}^{n} Q_i, \quad \mu_P = \frac{1}{n} \sum_{i=1}^{n} P_i$$

2) Center the points (centroids are repeated to match the matrix shape):

$$Q' = Q - \mu_Q, \quad P' = P - \mu_P$$

3) Compute the covariance matrix:

$$M = Q'^T P'$$

4) Compute the SVD of M:

$$M = VSW^T$$

5) Construct U:

$$s_1 = s_2 = \cdots = s_{d-1} = 1, \quad s_d = \begin{cases} 1, & \text{if } \det(VW) > 0, \\ -1, & \text{otherwise.} \end{cases}$$

$$S' = \text{diag}(s_1, \ldots, s_d)$$

$$U = WS'V^T$$

6) Compute the translation vector $t_{d\times 1}$:

$$t = \mu_P - U\mu_Q$$

The transformed points are then given by (t can be repeated to match the correct shape):

$$P_{\text{reconstructed}} = (UQ^T)^T + t^T$$

B. Root Mean Square Error (RMSE)

The RMSE between the original and reconstructed points is computed as:

$$\text{RMSE}(P, \hat{P}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|P_i - \hat{P}_i\|^2}$$

where P is the ground truth and $\hat{P}$ is the reconstructed point set. It is a popular algorithm in the ML (Machine Learning) field, hence, was a first choice for checking the accuracy of Kabsch-Umeyama transformation.

C. Application to 3D Point Cloud Data

Given two point sets, P and Q, where each point in P corresponds to a point in Q, the rotation matrix R and translation vector $\mathbf{t}$ were computed using the Kabsch-Umeyama algorithm. All point data were stored in '.txt' files and loaded using `numpy.loadtxt()`.

D. Application to 2D Images

The input images were loaded and converted from RGBA to RGB format, resulting in image matrices of shape $[H, W, 3]$, where H and W denote the height and width of the images, respectively. Correspondences between images were provided as JSON files and parsed into appropriate data structures.

In each sample, the second image was a continuation of the first, forming a left-to-right sequence (`img1` $\rightarrow$ `img2`). In most cases, `img2` was horizontally aligned with `img1`, although in some examples, `img2` appeared slightly above `img1`.

A blank composite image array of shape $[2H, 2W, 3]$ was initialized with zeros to accommodate the merged result. `img1` was placed in the bottom-left corner of the composite image. The Kabsch-Umeyama algorithm was then used to estimate a rigid transformation (rotation and translation) aligning `img2` to `img1`, based on corresponding pixel coordinates (x, y) extracted from the JSON files. The calculated rotation matrix and translation vector was used to correctly place each pixel of `img2` in the composite array.

III. Results

Rotation matrix and translation vectors along with the visualisations are presented for each of the samples.

A. Bottle

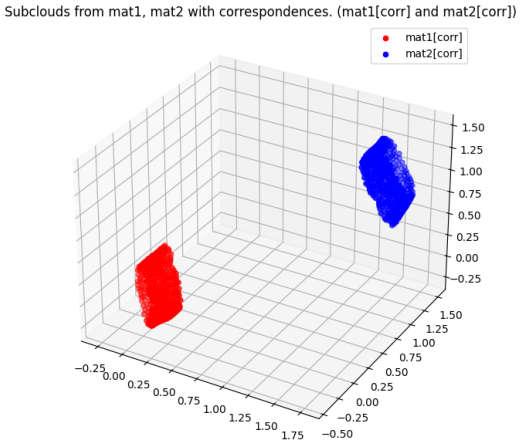
Rotation Matrix R for bottle:

$$\begin{bmatrix} -0.7469 & -0.5665 & 0.3483 \\ -0.3887 & -0.0531 & -0.9198 \\ 0.5396 & -0.8224 & -0.1805 \end{bmatrix}$$

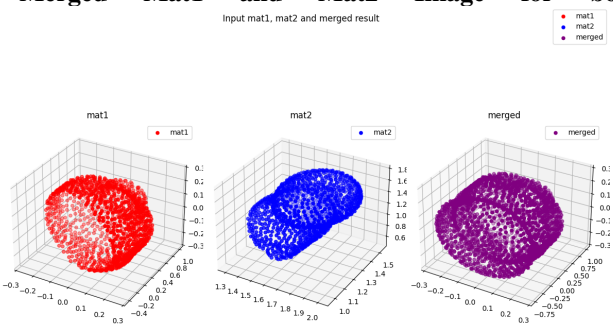
Translation Vector T for bottle:

$$\begin{bmatrix} 1.4893 \\ 1.5909 \\ 0.3703 \end{bmatrix}$$

Correspondences Image for bottle:



Merged Mat1 and Mat2 Image for bottle:



B. Chair

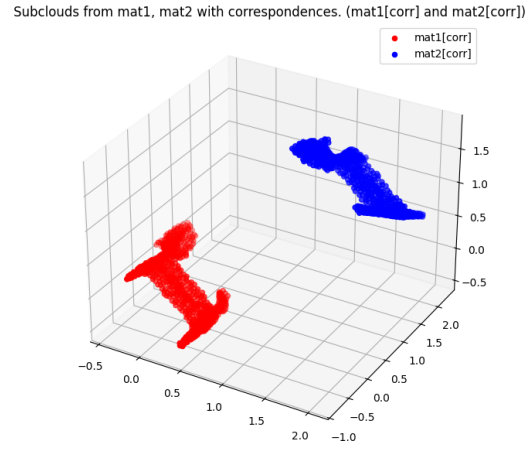
Rotation Matrix R for chair:

$$\begin{bmatrix} 0.1280 & -0.4535 & -0.8820 \\ -0.1474 & -0.8881 & 0.4353 \\ -0.9808 & 0.0743 & -0.1805 \end{bmatrix}$$

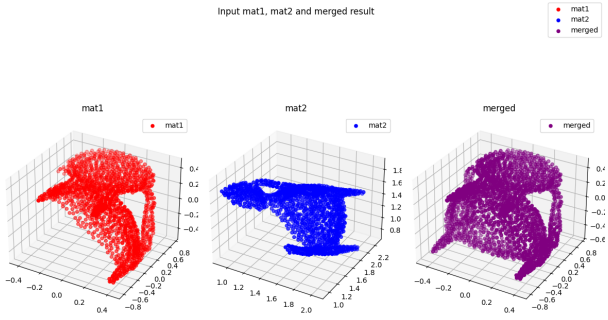
Translation Vector T for chair:

$$\begin{bmatrix} 1.6338 \\ 0.8939 \\ 1.5424 \end{bmatrix}$$

Correspondences Image for chair:



Merged Mat1 and Mat2 Image for chair:



C. *Cup*

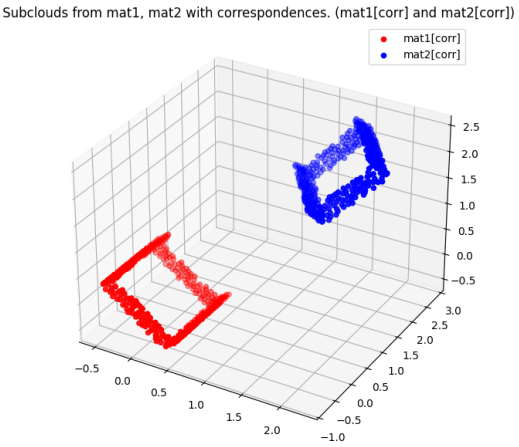
Rotation Matrix R for cup:

$$\begin{bmatrix} -0.1016 & -0.0059 & 0.9948 \\ -0.4943 & 0.8681 & -0.0453 \\ -0.8633 & -0.4963 & -0.0911 \end{bmatrix}$$

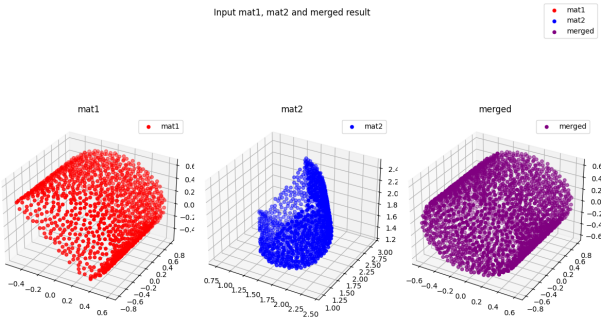
Translation Vector T for cup:

$$\begin{bmatrix} -1.7217 \\ -0.8106 \\ 2.4167 \end{bmatrix}$$

Correspondences Image for cup:



Merged Mat1 and Mat2 Image for cup:



D. *Grid*

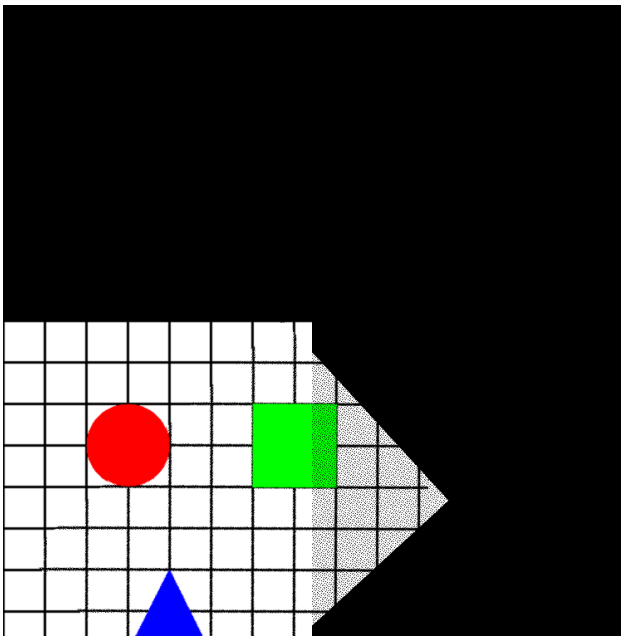
Rotation Matrix R for grid:

$$\begin{bmatrix} 0.5560 & 0.7128 & -0.4275 \\ 0.3887 & 0.2317 & 0.8918 \\ 0.7347 & -0.6620 & -0.1482 \end{bmatrix}$$

Translation Vector T for grid:

$$\begin{bmatrix} 336.7682 \\ 0.2388 \end{bmatrix}$$

Merged Img1 and Img2 Image for grid:



E. Mountain

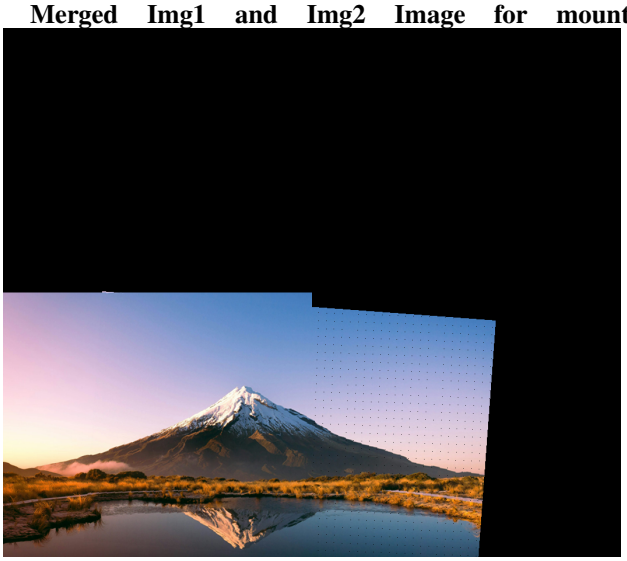
Rotation Matrix R for mountain:

$$\begin{bmatrix} 0.5560 & 0.7128 & -0.4275 \\ 0.3887 & 0.2317 & 0.8918 \\ 0.7347 & -0.6620 & -0.1482 \end{bmatrix}$$

Translation Vector T for mountain:

$$\begin{bmatrix} 170.6503 \\ -1.4710 \end{bmatrix}$$

Merged Img1 and Img2 Image for mountain:



F. Pikachu

Rotation Matrix R for pikachu:

$$\begin{bmatrix} 0.5560 & 0.7128 & -0.4275 \\ 0.3887 & 0.2317 & 0.8918 \\ 0.7347 & -0.6620 & -0.1482 \end{bmatrix}$$

Translation Vector T for pikachu:

$$\begin{bmatrix} 208.0000 \\ 0.0000 \end{bmatrix}$$

Merged Img1 and Img2 Image for pikachu:

Note: the images are extended with zeros (black pixels) in the merged result.

IV. Discussion

A. Point Set Registration

The formulas provided in both referenced research papers implicitly assumed that the point matrices P and Q were transposed, and a similar issue existed with the computation of centroids. Additionally, no normalization procedures were discussed in the project guide or the recommended sources. A correct and clarified implementation was found in [4], which helped solidify understanding of centroids and column-wise operations.

The airplane example provided by the assignment instructors was initially confusing. While the Rotation matrix and translation vector were computed using the transformation (R, t) from `mat1` to `mat2`, the example merged matrix depicted the result of applying the reverse transformation. This discrepancy led to confusion at first; however, it prompted the development and testing of a hypothesis regarding the reverse transformation, which ultimately resolved the issue.

B. Image Stitching

It was evident from the beginning that the rotation R and translation t needed to be applied to the row and column coordinates of `img2`, which was relatively straightforward. However, the subsequent image merging process proved more complex. Initially, I attempted to implement a general solution that would handle arbitrary relative positions between `img1` and `img2`. This approach turned out to be overly complicated, so I opted for a simpler solution: positioning `img2` to the right

of `img1`, with additional padding at the top to accommodate alignment.

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 and Machine Intelligence*, 13(4):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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[4] Tuomas Siipola. Aligning point patterns with Kabsch–Umeyama algorithm. Online article, March 2021. Available at: https://zpl.fi/aligning-point-patterns-with-kabsch-umeyama-algorithm/ [Accessed: May 20, 2025].