Instituto Superior Técnico

Image Processing and Vision
MEEC

2019/2020 - 5º year, 1º Semestre

Image stitching and 3D point cloud registration

Group 1

David Ribeiro	84027
Pedro Custódio	84169
Laksiya Balasubramaniam	95037

Professor: João Paulo Salgado Arriscado Costeira

October 17, 2021

Contents

1 Introduction

The first problem to be faced in this project is to create a single panoramic image using a given sequence of images. These images can be of the same plane but taken from different positions or they can be images taken by a rotating camera on a fixed position. For this reason, these images have common point in between them, and so it's possible to achieve the single panoramic image by finding these matching points between the different pairs of images and using that to determine the transformations between the pairs. With that information it's then possible to build the final panoramic image.

In the second part of this project the goal is to reconstruct a 3D scene from a sequence of moving and rotating RGB and depth images from a Intel Realsense camera.

The Kinect's Intel Realsense camera is used to give devices depth perception capabilities to "see" the world in 3D. It is implemented with two cameras – a RGB color camera and a depth camera. Each image from the camera returns both an RGB and depth image as shown in figure 1. By combining the RGB values of each pixel with the 3D position from each pixel in the depth image, it is possible to construct a local, single view point cloud from the camera.

To create this 3D reconstruction, the rigid body transformation between the image pairs must be calculated. By constructing point clouds for all the image pairs, the transformation between the point clouds may be calculated by using several techniques such as registration, RANSAC and ICP. The transformations and the images may be used to reconstruct the 3D scene.

In this report the theoretical problem, the procedure and techniques used in the final algorithm will be described, as well as a discussion of the result.

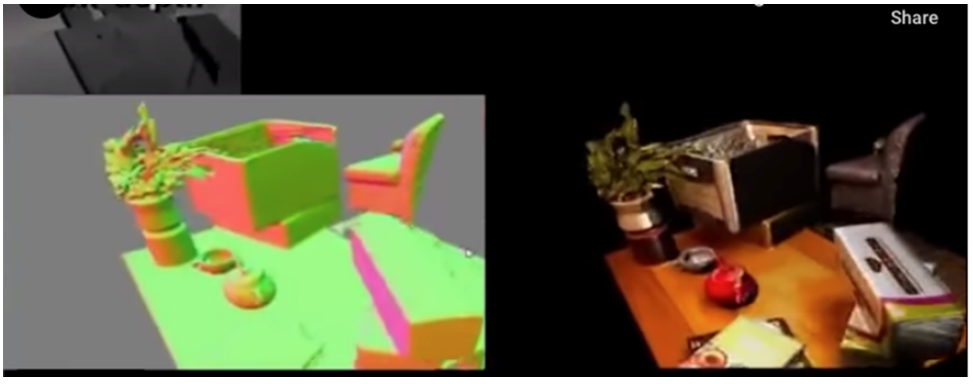

Figure 1: RGB and Depth image from Kinect

2 Theoretic Background

2.1 Homographies

In order to stitch multiple images in one, there is need to transform their coordinates. Assuming one first image that sets the referential, it's possible to compute a transformation in plan that places the others in the way that fits the parts that they have in common.

Projective transformations (homography) are the ones with more degrees of freedom, and are described by

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{1}$$

It could vary the length, the angles, and the parallelism between lines.

To determine the transformation between two images the Least squares are used to find out the h coefficients of the matrix in (1). Assuming $h_9 = 1$, it is possible to get a closed form solution

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -xx' & -xy' \\ 0 & 0 & 0 & x & y & 1 & -yx' & -yy' \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \end{bmatrix} <=> Y = X\beta. \tag{2}$$

where the coefficients could be obtained by

$$\beta = (X^T X)^{-1}(X^T Y). \tag{3}$$

This computation requires, at least, 4 points (each point has 2 coordinates) to determine the 8 parameters ($h_9 = 1$).

2.2 Camera Model

The cameras, such as the ones in the Kinect camera, project 3D points ($\mathbf{X} = [X, Y, Z]^T$) into the image plane ($\mathbf{x} = [x, y]^T$). It is composed by intrinsic parameters, extrinsic parameters, and the pinhole model (figure 2).

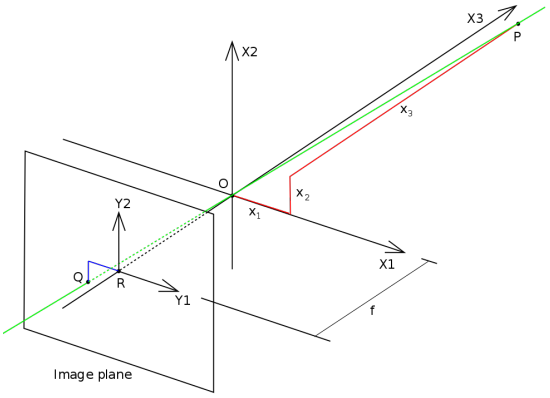

Figure 2: Pinhole Camera model

In the pinhole model, with a normalized camera, where the distance between projection point and optical center is f = 1, and a non inverted image referential (image plan is between the optical center and the points in observation), the homogeneous coordinates of the perspective projection are

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{4}$$

It converts the 3D points in a 2D, in metric coordinates.

The full camera model also takes in account the intrinsic and extrinsic parameters. The intrinsic parameters describe camera parameters, such as focal length f, scale factor f_{s_x} and f_{s_y}, and principal point c_x and c_y. This transforms the metric image point, x, to a pixel in the image, x', as shown in equation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & 0 & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{5}$$

The extrinsic parameters describe how the world coordinates X' is rotated, with the rotation matrix R, and translated , with the translation vector T, into the camera coordinates X as shown in equation 6.

$$\mathbf{X} = R\mathbf{X'} + \mathbf{T} \tag{6}$$

Equation 7 shows how the homogeneous 3D coordinates are first transformed with the extrinsic parameters and then computed in metric pixels (as shown in equation 4),

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_1 & R_2 & R_3 & T_1 \\ R_4 & R_5 & R_6 & T_2 \\ R_7 & R_8 & R_9 & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{7}$$

Combining the results form equations 5 and 7 results the full camera model, as shown in the equation

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & 0 & c_x \\ 0 & fs_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_1 & R_2 & R_3 & T_1 \\ R_4 & R_5 & R_6 & T_2 \\ R_7 & R_8 & R_9 & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{8}$$

2.3 Rigid body transformation in 3D coordinates

For any two give point clouds, pc_1 and pc_2, the rigid body transformation between them is given by

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} + \mathbf{T}, \tag{9}$$

where $\mathbf{R}$ is a 3-by-3 rotation matrix, and $\mathbf{T}$ a 3-by-1 translation vector.

This transformation may be estimated according the following minimization problem

$$\begin{aligned} \min_{R,T} \quad & \|PC_1 - \mathbf{R}PC_2 - \mathbf{T}\|_2^2 \\ \text{s.t.} \quad & \mathbf{R}^T\mathbf{R} = \mathbf{I} \\ & det(\mathbf{R}) = 1 \end{aligned} \quad , \tag{10}$$

where PC_1 and PC_2 are the match points between pc_1 and pc_2. The constraints don't allow this minimization problem to be solved, because it's not convex.

To simplify this problem, the translation vector may be written as a function of the rotation matrix R and the centre of mass of the point clouds. The coordinates of the centre of mass (COM) in a rigid body is not transformed by rotation matrix, only the translation vector. This fact may be used to express the translation vector as

$$\mathbf{T} = COM_1 - \mathbf{R}COM_2. \tag{11}$$

Thus the equation 9 becomes

$$A = \mathbf{R}B, \tag{12}$$

where

$$\begin{cases} A = PC_1 - COM_1 \\ B = PC_2 - COM_2 \end{cases} \quad , \tag{13}$$

and the new minimization problem is

$$\begin{aligned} \min_{R} \quad & \|A - \mathbf{R}B\|_2^2 \\ \text{s.t.} \quad & \mathbf{R}^T\mathbf{R} = \mathbf{I} \\ & det(\mathbf{R}) = 1 \end{aligned} \quad . \tag{14}$$

Knowing that

$$\begin{cases} \|A - \mathbf{R}B\|_2^2 = \text{trace}((A - \mathbf{R}B)(A^T - B^T\mathbf{R}^T)) \\ \text{trace}(\mathbf{M_1}\mathbf{M_2}\mathbf{M_3}) = \text{trace}(\mathbf{M_2}\mathbf{M_3}\mathbf{M_1}) \\ \mathbf{R}^T\mathbf{R} = \mathbf{I} \end{cases} \tag{15}$$

the solution of the minimization problem (14) is the same of

$$\begin{aligned} \max R \quad & \text{trace}(AB^T\mathbf{R}^T) \\ \text{s.t.} \quad & \mathbf{R}^T\mathbf{R} = \mathbf{I} \\ & det(\mathbf{R}) = 1 \end{aligned} \quad . \tag{16}$$

Applying the SVD [3] to the matrix AB^T, it is possible to determine a orthogonal matrix μ, a diagonal matrix with AB^T eigenvalues Σ, and AB^T eigenvectors matrix V, that verifies

$$AB^T = \mu\Sigma V^T. \tag{17}$$

Replacing AB^T by $\mu\Sigma V^T$, and applying the cyclic property of trace of a square matrix which is the product of matrices [4], the function to maximize became $\text{trace}(\Sigma V^T\mathbf{R}^T\mu)$, and the maximum value corresponds to

$$V^T\mathbf{R}^T\mu = \mathbf{I} <=> \mathbf{R} = \mu V^T. \tag{18}$$

This solution doesn't assure the determinant to be equal to 1. Instead, the expression used to compute $\mathbf{R}$ is

$$\mathbf{R} = \mu \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(\mu V^T) \end{bmatrix} V^T. \tag{19}$$

This computation requires, at least, 4 points (each point has 3 coordinates) to determine the 12 parameters (9 for $\mathbf{R}$ and 3 for $\mathbf{T}$).

2.4 RANSAC

The match point dataset to compute the images transformations will have outliers. Ransac is a method that try many combinations of points, with the minimal necessary, and picks the one with the biggest number of inliers. The algorithm works in the following way:

- Iterate a pre-defined number of times;

- Pick randomly a sample of points, the minimal necessary to compute the transformation;

- Compute the transformation for the sample of points picked in the step before;

- Compute the error for the remaining points, compare them with a threshold, and determine the inliers;

- Return the set with more inliers.

3 Methods

3.1 Keypoints and Feature Matching

To construct the 3D scene of several images, the algorithm must know some similarities between each image, to be able to pin point how the image has transformed. For this to be possible, it's important to detect and get a description of the keypoints of each image, these are points that contain interesting information, basically they define what stands out in a picture, and should be easily detected in different images of the same scene even after rotations/translations of the camera.

There are several ways to detect these key points. The technique used in this work is called SURF, Speeded up robust features. This was the feature detector chosen because it works not only as a detector but also as a descriptor and is considered faster than SIFT.

The similarities between each pair of images are called match points and are found through feature matching. When the keypoints of different images are correctly matched, it is possible to calculate the transformation between the images. An example of this matching between the keypoints of two images, in this case for the dataset 'room1', is shown in figure 3.

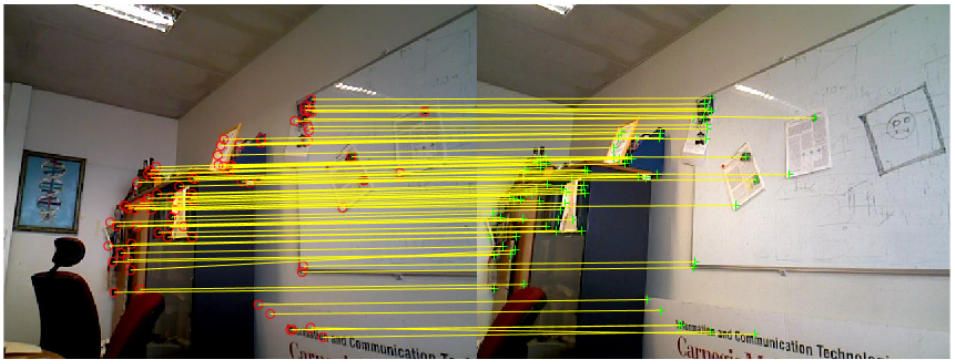

Figure 3: Matching points example from dataset room1

3.2 RANSAC

To reduce the errors when computing the transformations, Ransac is applied to remove the outliers.

For part 1, 4 random different points (2D coordinates, and 8 coefficients to determine) are picked from the match points set, and used to compute a homography. To distinguish between inliers and outliers on this set, a squared error threshold of 9 is used. This process is repeated 100 times, and from all those tries, the biggest set of inliers found will be used after to compute the transformation between images.

For part 2 the process is similar. Instead of an homography, a rigid body transformation is used. The number of points to find the transformation is also 4 (12 coefficients), but now represented in 3D space. The threshold for the error is of 10 cm, and the it iterates 100 as well.

3.3 Using the Camera Model to calculate Point Clouds

The camera model equation is used in reverse in this project to compute the 3D point clouds from the depth array and RGB image. From the depth array and its intrinsic parameters, K_d, it is possible to get the 3D location of each pixel. The metric x and y coordinates may be found using equation 5 and the pixel coordinates from the image. With the depth λ and equation 4, the final 3D position may be calculated,

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \lambda K_d^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{20}$$

To match 3D points from a depth camera to its corresponding RGB value, an RGB image of the same instance is required. Since the depth and RGB camera are located next to each other inside the Kinect, they do not share the same perception center. This means they have individual coordinate system and intrinsic parameters, K_d and K_{rgb}. In other words, a certain pixel coordinate in both images corresponds to different 3D world points. To relate both coordinate systems, the transformation in between the coordinate systems is required.

4 Experimental Results

For the first part of the project the goal was to stitch sets of images in order to create one single panoramic image formed by that sequence of images. The final results obtained for three of the given datasets ('translation', 'vianaPiv' and 'sinteticoprojective', displayed from left to right) can be seen in the figure 4.

Figure 4: Single Image Panoramic of three different datasets

Observing these results it's clear that the objective was accomplished without much error, as the figures show the different images perfectly aligned after the stitching was completed, therefore confirming that the transformations were correctly determined.

To evaluate the results of part 2, the following computation of error is used

$$E = \|PC_1 - \mathbf{R}PC_2 - \mathbf{T}\|_2. \tag{21}$$

It gives the error of one match point. The mean error of all points is given by

$$E_{av} = \frac{1}{N}\sum_{n=1}^{N}\|PC_{1_n} - \mathbf{R}PC_{2_n} - \mathbf{T}\|_2, \tag{22}$$

where N corresponds to the total number of inliars of all transformations.

The dataset 'Room1' is one of the datasets that gets really good results visually. As can be seen in the figure 5, in this 3D reconstruction is possible to visualize an almost complete 3D representation of the room in question. For this dataset, the average error was 6.01 mm.

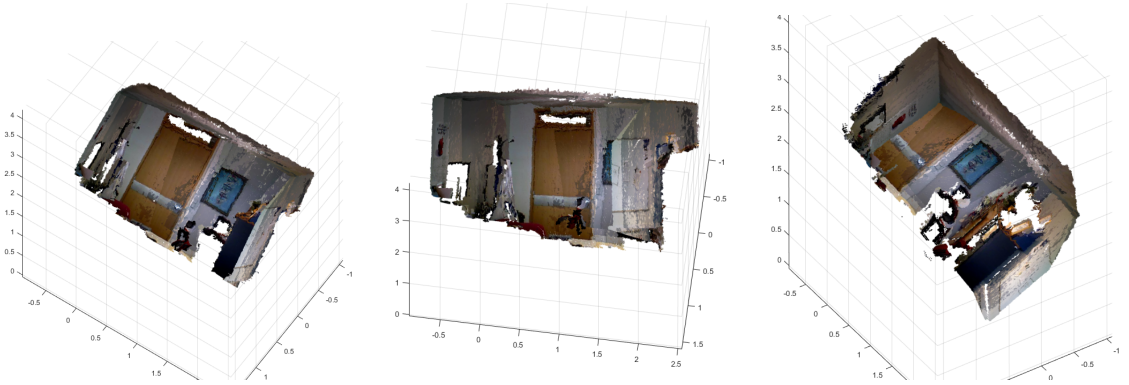

Figure 5: 3D Reconstruction of dataset Room1

The error in the datasets 'newPiv2' and 'data_rgb' was even lower than the one calculated for 'Room1', having averages of 2.03 mm and 4.36 mm respectively. The corresponding 3D reconstructions are presented in figures 6 and 7, both with very satisfying results.

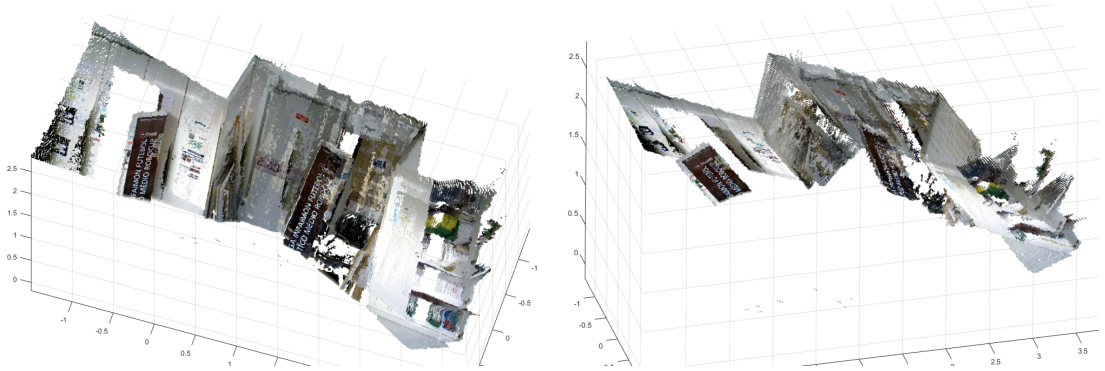

Figure 6: 3D Reconstruction of dataset newpiv2

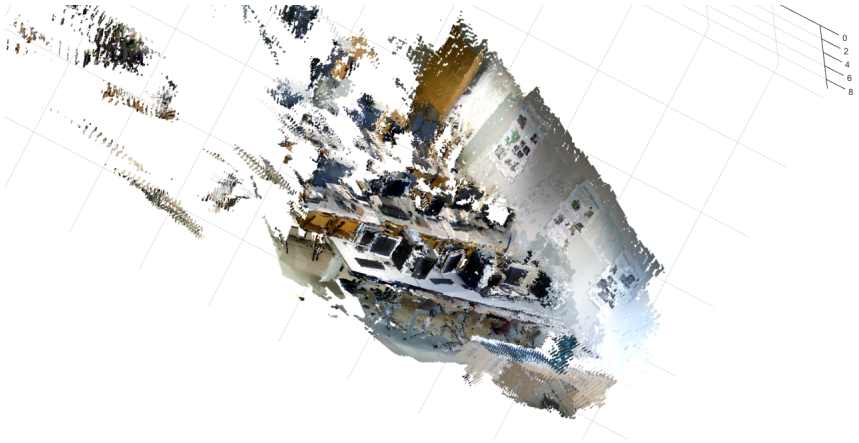

Figure 7: 3D Reconstruction of dataset data_rgb

The two datasets with the smallest error are 'table1' and 'Board1', with error of average 1.96 mm and 0.94 mm. Their 3D representations are presented in the figure 8. It's expected, because these are images where the camera is close to the scene, so the metric values are smaller, which implies lower values of error.

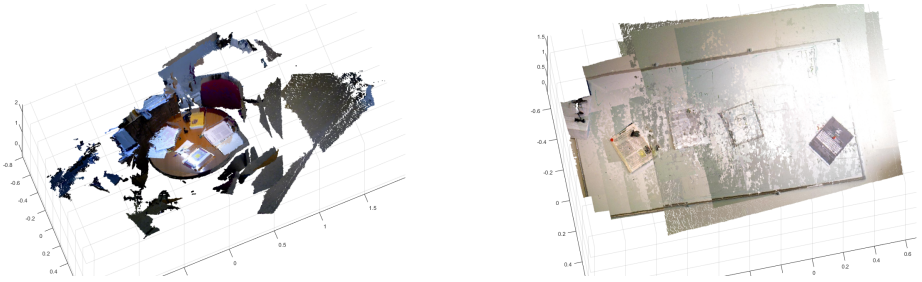

Figure 8: 3D Reconstruction of datasets table1 and Board1

Although it has a small error averages, the 3D reconstruction of the 'data_rgb' dataset doesn't appear to lead to so good results visually. This happens because of the multiplication of transformations to get all the coordinates in the frame of the first one. The frame N will accumulate the error of $N-1$ transformations, leading to worse results. Also, transformations that have a low number of inlears, usually have higher values of error, as could be verified in table 1. These transformations have big errors and have less weight in the total average, because there's less points to be taken into account.

Transformation	Inliers	Mean error (mm)
2 to 1	53	8.06
3 to 2	13	10.91
4 to 3	22	7.32
5 to 4	135	3.87
6 to 5	41	7.12
7 to 6	13	8.42
8 to 7	8	16.34
9 to 8	39	5.16
10 to 9	189	1.98
11 to 10	52	6.05
12 to 11	32	11.06
13 to 12	58	4.09
14 to 13	126	2.83
15 to 14	112	2.53
16 to 15	28	5.97
17 to 16	84	3.11
18 to 17	45	5.50
19 to 18	22	5.98
20 to 19	77	3.21
21 to 20	51	4.03
22 to 21	41	4.66
23 to 22	84	5.12

Table 1: Number of inlears and mean error for all transformations in dataset data_rgb

5 Conclusions

This work describes how to perform image stitching in order to obtain a panoramic, and how to get a 3D point cloud from an image with depth and how to compute a 3D rigid body transformation to merge point clouds.

In the first part, with homographies, it's possible to use pairs of features in images to merge them in one. This is just possible if we take pictures of a plane from different positions, or rotating around in the same place. Otherwise, there will be 2 points in one image that just correspond at 1 in the other image.

In the second part, the rigid body transformation is used to merge point clouds from images. These point clouds can be obtained, because there is information about the depth of the pixels, with the camera model.

The results obtained show a correct matching between the images, with some minor error. It's possible to reduce this error using some tuning tools. One way is to find match points between all the figures, instead of just between pairs, and compute directly the transformation between images, instead of multiply them, reducing math errors.

References

[1] Jorge Marques, José Santos-Victor, Image processing and vision slide sets, 2019

[2] Szeliski, Richard. Computer vision: algorithms and applications. Springer Science & Business Media, 2010.

[3] Wikipedia, Singular value decomposition,
`https://en.wikipedia.org/wiki/Singular_value_decomposition`

[4] Wikipedia, Trace (linear algebra),
`https://en.wikipedia.org/wiki/Trace_(linear_algebra)`