Universiteit van Amsterdam

Image Alignment & Stitching

Lab Assignment 4

October 12, 2021

Students:
Name: Nicole Sang-Ajang
UvAnetID: 10734155

Name: Haohui Zhang
UvAnetID: 14124262

Name: Xiaoyang Sun
UvAnetID: 13825453

Affiliation:
Master Artifical Intelligence

Lecturer:
Dr. Shaodi You

Course:
Computer Vision 1

Course code:
52041COV6Y

Introduction

[**szeliski2010computer**] This assignment we will work on the algorithm to perform image alignment and stitching operation. We will study the theory behind the existing algorithm. In the first section, we implement the SIFT, RANSAC and nearest-neighbor interpolation to find the best transformation between the input images. In the second section, we use the best transforamtion found in the previous section to merge the two images and obtain a real panorama of the scence.

1 Image Alignment

Question 1 This section is focus on the methods for image alignment. Image alignment algorithms can find a large number of (parameter) relationships between images with different degrees of overlap. This algorithm contains multiple steps, first is detection and extraction the feature points from the input images, then is locating the corresponding point set between two images, finally is calculating the transformation matrix from the source image to the target image.

1.1 Keypoints detection

In this section, the algorithm separate in two parts, feature detection and feature description. We used **Scale Invariant Feature Transform (SIFT)** from Lower, which looks for extreme points in the spatial scale, and extracts its position, scale, and rotation invariants from it. The key points found by SIFT are some very prominent points that will not change due to factors such as illumination, affine transformation and noise, such as corner points, edge points, bright spots in dark areas, and dark points in bright areas.

In the paper, Lowe decomposes the SIFT algorithm into the following four steps [3]:

- Scale-space Extrema Detection: Search for image positions on all scales. The difference of Gaussian(DoG) is used to identify potential points of interest that are invariant to scale and rotation.

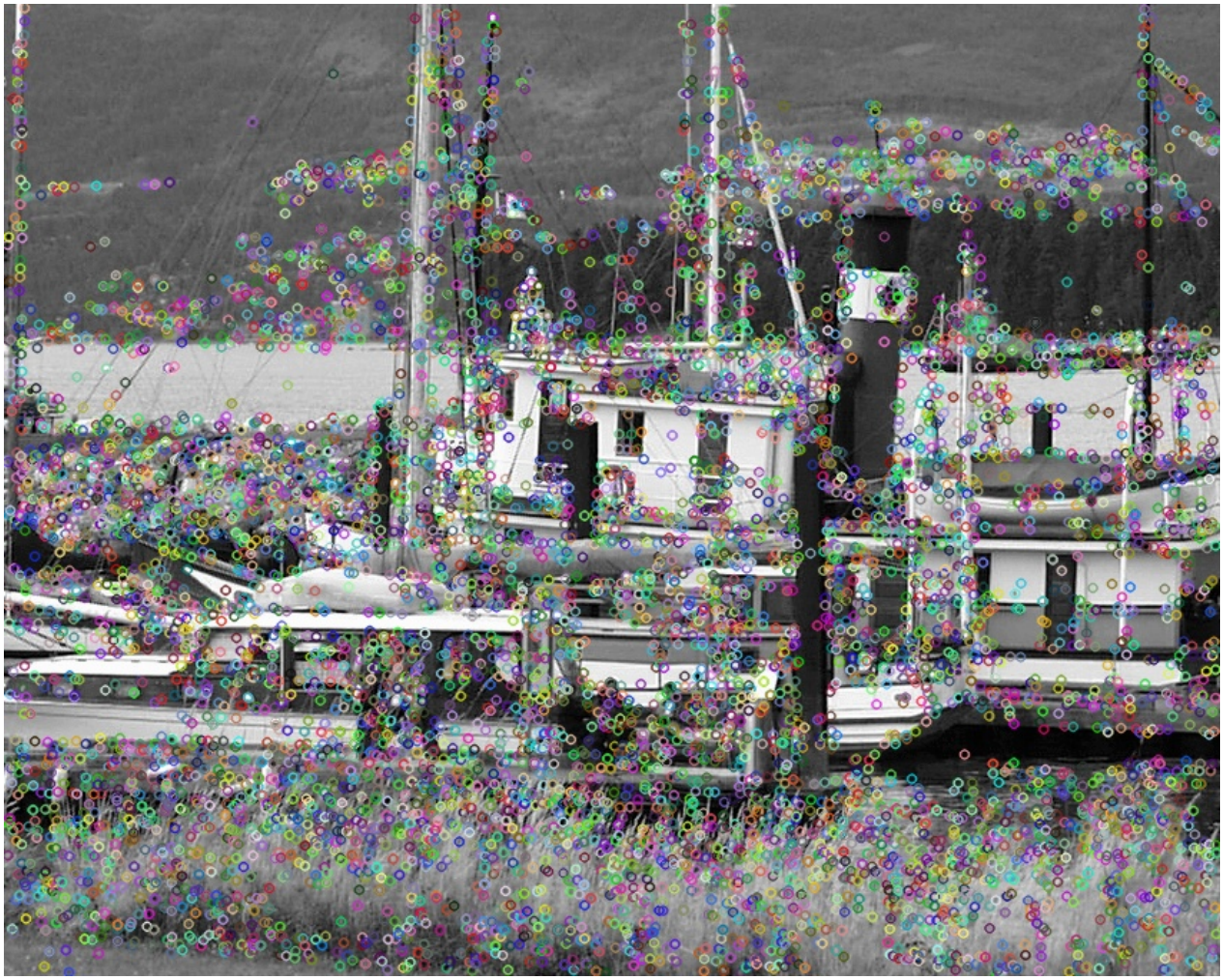

Figure 1: Key points in boat1.pgm

- Keypoint Localization: At each candidate position, a fine-fitting model is used to determine the position and scale.
- Orientation Assignment: Based on the local gradient direction of the image, one or more directions are assigned to each key point position. All subsequent operations on the image data are transformed relative to the direction, scale and position of the key points, thereby providing invariance to these transformations.
- Key point description: For each key point, there are three pieces of information: location, scale, and direction. In the neighborhood around each key point, measure the local gradient of the image on the selected scale. These gradients are used to create a descriptor for each key point, and use a set of vectors to describe the key point so that it does not change with various changes.

Key points in Figure 1 are selected based on their stability and small circles on the locations of keypoints are drawn using cv2.drawKeyPoints function.

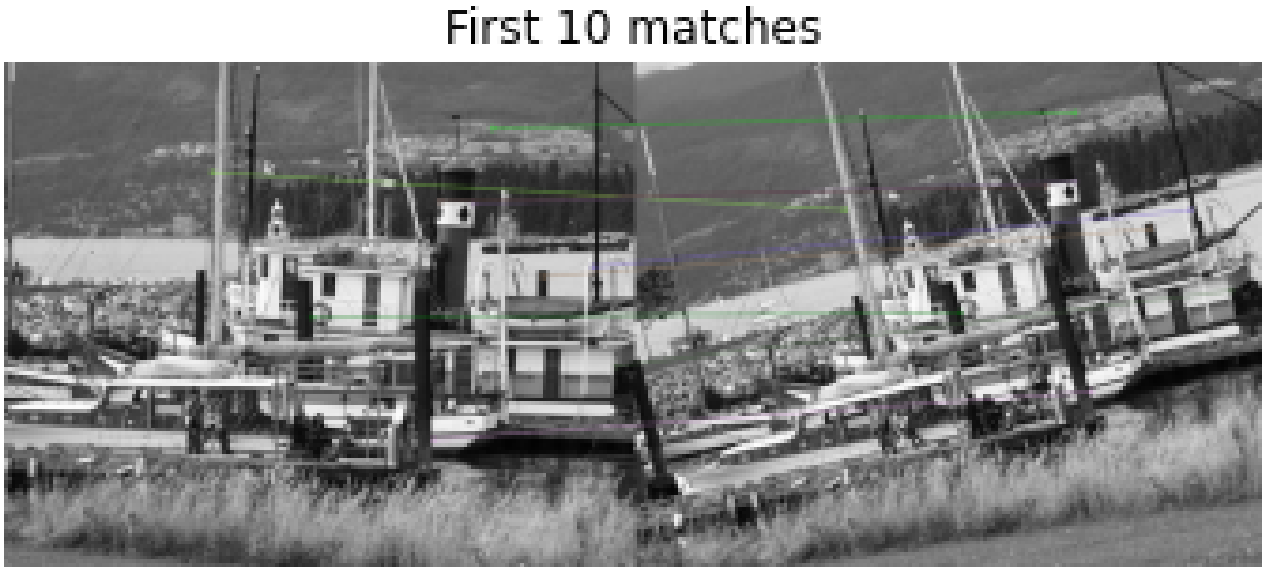

Figure 2: Keypoint matching sorted by distance

1.2 keypoints matching

In this section, we used Brute-Force Matcher to match the keypoints between two images.The Brute-Force Matcher takes the descriptor of one feature in the first group and uses some distance calculations to match all other features in the second group. And the closest one is returned [2]. The Figure 2 shows a sample of the first 10 keypoints matching sorted by distance. The Figure 3 shows random 10 keypoint matches.

1.3 Image alignment

After extracting the invarient key points from the image and matching them correctly, we should consider a method of calculating the parameters from the transformation matrix to transform the source image to the target image and vice versa.

In this section, we use **Random Sample Consensus (RANSAC0** method from Fishler and Bolles, which is an iterative method for estimating the parameters of a mathematical model from a set of observation data containing outliers.

The RANSAC method is split in following steps:

- Repeat N times
- Given a linear function $mA = t = b$ with:

$$A = \begin{bmatrix} x \\ y \end{bmatrix}, m = \begin{bmatrix} m1 & m2 \\ m3 & m4 \end{bmatrix} t = \begin{bmatrix} t1 \\ t2 \end{bmatrix} b = \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{1}$$

- Pick P matches at random from the total set of matches T (from source image (x, y) and target image (x', y'))

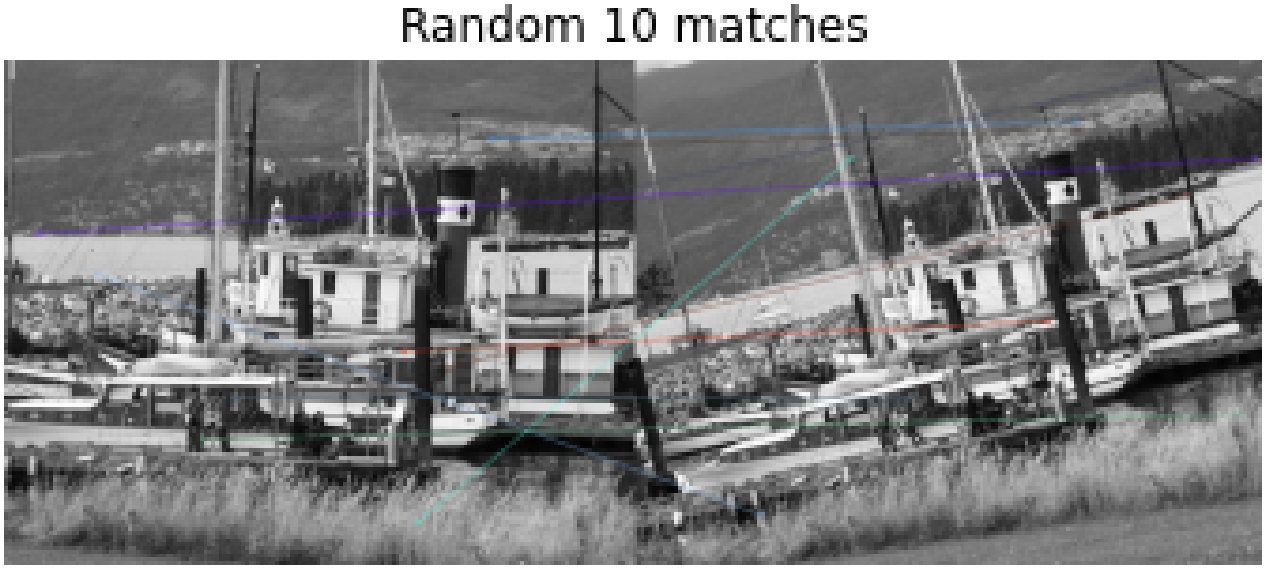

Figure 3: Random 10 keypoints matching

- Use P pairs of key points to resolved the transformed function $x = pinv(A)b$ with:

$$x = \begin{bmatrix} m1 \\ m2 \\ m3 \\ m4 \\ t1 \\ t2 \end{bmatrix} A = \begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \end{bmatrix} b = \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{2}$$

- For each key points in T, compute the transformation with x. If the distance between the point in transformed source image and counterpart point in the target image is smaller than a threshold t, it will be classified as an inliner.

- Iterate for N repeats, and select the parameters which generate the biggest number of inliners.

During the experiment, we plot the source image and target image side by side with a line connecting the original T points in image1 and transformed T points over image2. Figure 4 is one example during the experiment. We can see that the RANSAC method has a lot of errors normally.

Finally, we used two method separately, nearest-neighbor interpolation and cv2.warpAffine to transform the image1 to image2 and image2 to image1. We found that the pose of the object in the scene of image 1 is correspond to its pose in image 2. The Figure 5 and Figure 6 provides the examples of our best transformations generated by the parameters got form RANSAC. In these examples, we iterated $N = 100$ times and selected $P = 10$ matches to resolve the equation.

Also, we meet some problem when using nearest-neighbor interpolation method. There are some regular black holes between pixels. The reason is that because when we are using nearest-neighbor interpolation, this is a forward warping and an affine matrix, and because of the round method, some pixel will lose. Thus, in the image stitching part, we use backward warping and a homography to transform the point from cartesian coordinate system to a homogeneous coordinate system by creating a fictional dimension z.

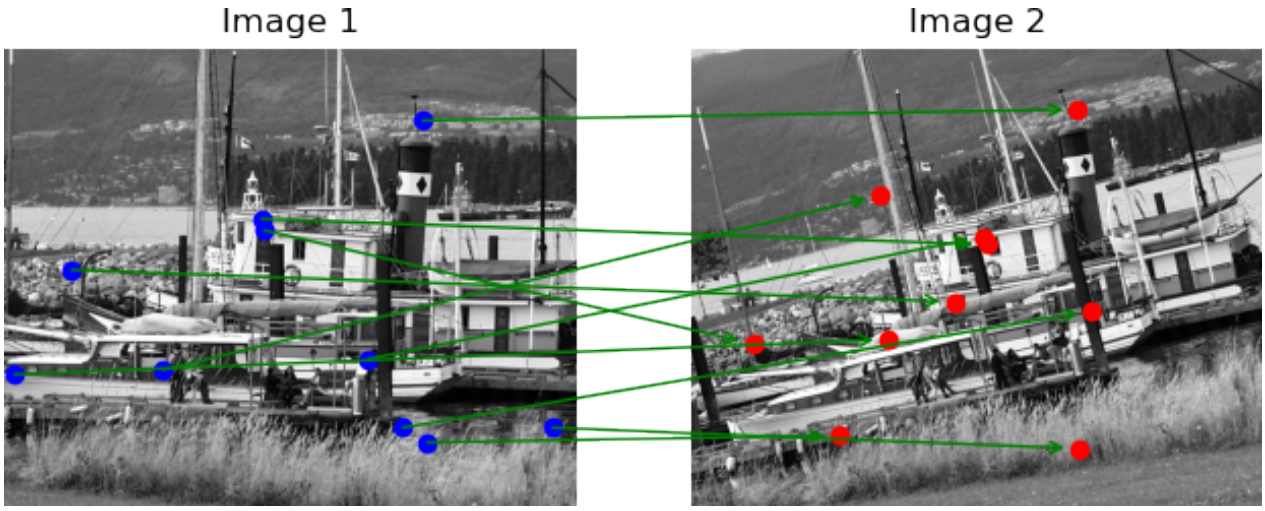

Figure 4: Transformation result

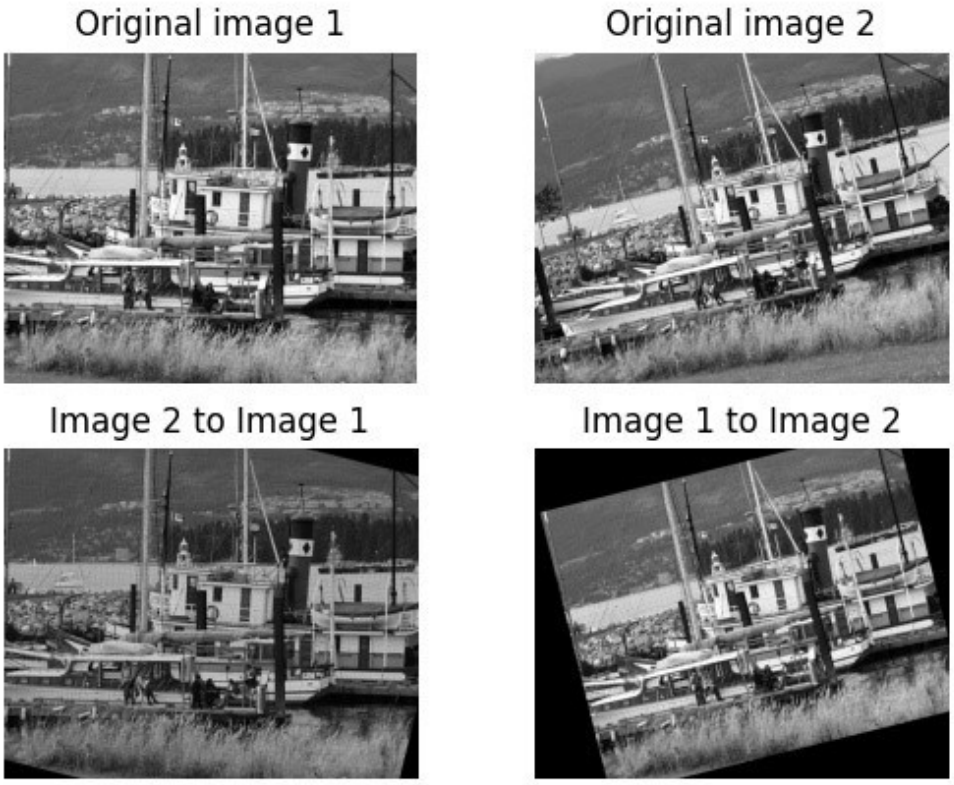

Figure 5: Transformations using nearest-neighbor interpolation

(a) Image 2 to image 1

(b) Image 1 to image 2

Figure 6: Transformations using warp affine provided by openCV

Question 2

1.4

To solve the affine transformation, we need three matching points. Because we have six unknown variables m1, m2, m3, m4, t1, t2, then we need 6 equations to be able to solve the system, indicating that 3 pairs of matching points are needed. Therefore, 3 points are needed to calculate the affine transformation.

1.5

[1]The iterative process is to randomly select n points in the data as the inner group, calculate the model suitable for the inner group, bring the other points that were not selected just now into the model just established, calculate whether they are the inner group points, and write down For the number of inner groups, repeat the above steps, iterate k times, compare which calculation has the largest number of inner groups, and the model built with the largest number of inner groups is the solution we require. In the above steps, there are two parameters that are more important and need to be determined in advance, namely: how many points (n) we want to randomly select at the beginning, and the number of iterations (k). Assume that the probability of each point being a true inner group is w:

- The selected n points are the probability of the inner group: w^n
- The probability that at least one of the selected n points is not an inner group: 1-w^n
- Probability that not all n points are in-groups after repeating k times: $(1 - w^n)^k$
- Suppose the probability of success after the algorithm runs k times is p, then: $p = 1 - (1 - w^n)^k$
- Therefore, $k = log(1 - p)/log(1 - w^n)$

Therefore, the number of matches P should be greater than 3, the minimum number to solve the linear equation. If two small iteration time, sometimes there will be no result and my kernel is broken. So, during our experiment, we set that $P = 10$ matches and $N = 100$ iterations are good enough to get good results.

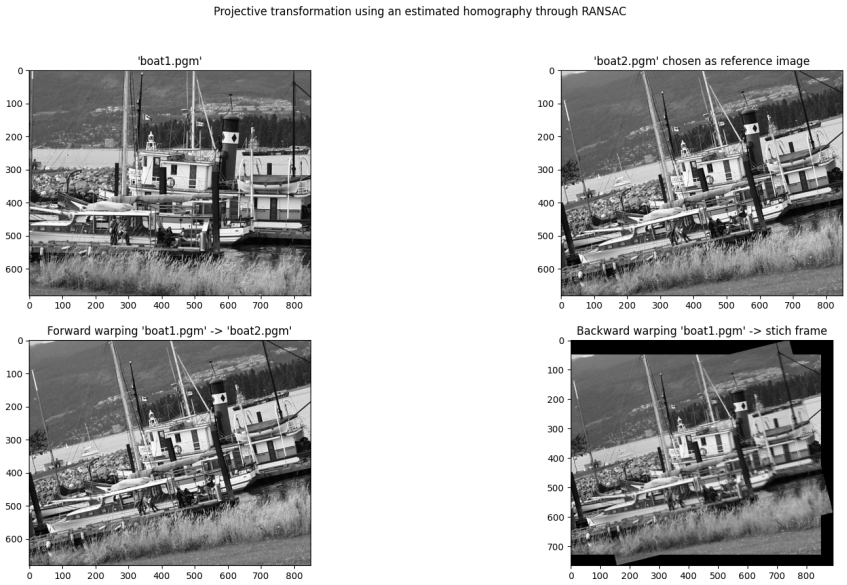

Figure 7: Resulting stitched image of boat1.pgm and boat2.pgm

2 Image Stitching

Assuming a set of images can be said to have the same center of projection, e.g. by having roughly the same viewpoint, it is possible to create a larger image or panorama composed of these images such that it appears as being one whole. This process is called stitching. To be able to stitch images we have to deal with the difference in coordinate systems of each image to be stitched. For that it makes sense to translate the coordinate system from one image to the coordinate system of the other system such that the images can be properly stitched. This requires for us to detect points of interests, i.e. features, which are then matched between two images. However, the difference in coordinate systems inherently allows matching objects to be rotated (i.e. not aligned), scaled or occluded compared across the different images next to inevitable differences in range e.g. due to lighting etc. To extract and match the appearance of such complex objects we require a robust and reliable method where consequently using edges or corners as feature points does not suffice but demands more descriptive unique features. This makes SIFT an appropriate method. In our implementation SIFT is used for image alignment and 2D object recognition. A powerful aspect of SIFT is that it can remove variation in images (orientation, scale etc. as mentioned before) such that feature matching becomes easier. For our implementation we estimate an homography matrix H that projects the image on the coordinate system of another image the so called reference image but inside a third image the so-called frame. In other words, two 2D images are related by a homography $\mathbf{H}$, if both view the same plane from a different angle. The homography relationship does not depend on the scene being viewed. We therefor transform the images from the Cartesian coordinate system to a homogeneous coordinate system by adding a fictional number z.

We apply constrained least squares method to find the best model fit, i.e. finding H that results in the most accurate transformation. Estimation is done through treating the system of linear equations as an eigenvalue problem. I.e. if we solve the equation $\mathbf{A}b = x$ using singular value decomposition with which we can find H. Using the theorem that the eigenvector with the smallest eigenvalue corresponds to H we apply SVD to $\mathbf{A}^\top\mathbf{A}$ and take the last column vector to use as H. As a threshold a projected point is considered accurate when

Figure 8: Resulting stitched image of left.jpg and right.jpg

it coordinates falls between a radius of 10 pixels from the true coordinate values. This process is repeated N times during which we keep track of the best model. At the end the best model is selected as being H.

Our **H** gives us 8 degrees of freedom. As a thumb rule number of constraints must equal or exceed the number of degrees of freedom the transformation. Thus we require at least sample size $P = 4$ since each matching pair gives a solution for two variables x and y, i.e. solves two constraints, hence satisfying the thumb rule.

To properly blend the images a weight function is generated in the y- and x-direction based on the Gaussian distribution but normalized to the opacity scale thus $[0, 1]$. The weight function represents the alpha in the opacity formula which is 1 - alpha.

We use backward warping instead of forward warping due to the limitations of nearest neigborhood rounding causing empty pixels, i.e. black holes to emerge.

At the end the image is converted to the RGB color space for proper presentation. The results are shown as Figure 7 and Figure 8.

References

[1] AIBooster. *Ransac Algorithm*. 2021. URL: `bbs.huaweicloud.com` (visited on 01/16/2021).

[2] Martin A. Fischler Robert C. Bolles. *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*.

[3] Dacid G. Lowe. *Distinctive Image Features from Scale-Invarient Keypoints*.