Assignment 1

Q1 Correspondences

1. The column vector **h**, which is the reshaped form of the matrix **H**, has 8 degrees of freedom.
2. We need at least 4 point pairs to solve for the **h** column vector which represents parameters.
3. **x1 =** $[x_1^{\,i}\ y_1^{\,i}\ 1]^T$, **x2 =** $[x_2^{\,i}\ y_2^{\,i}\ 1]^T$ We will remove the subscript i for brevity.

   x1 = Hx2

   We can write the above transformation into the following equations:

   $x_1 = h_1x_2 + h_2y_2 + h_3$

   $y_1 = h_4x_2 + h_5y_2 + h_6$

   $1 = h_7x_2 + h_8y_2 + h_9$

   We can plug the third equation into the other equations and get the following expressions:

   $x_1(h_7x_2 + h_8y_2 + h_9) = h_1x_2 + h_2y_2 + h_3$

   $y_1(h_7x_2 + h_8y_2 + h_9) = h_4x_2 + h_5y_2 + h_6$

   Now we can rearrange the terms within the equations and write the following:

   $h_7x_1x_2 + h_8y_2x_1 + h_9x_1 - h_1x_2 - h_2y_2 - h_3 = 0$

   $h_7y_1x_2 + h_8y_1y_2 + h_9y_1 - h_4x_2 - h_5y_2 - h_6 = 0$

   We can write the following in matrix form Ah = 0, where A will be the following matrix:

   $$A = \begin{bmatrix} -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_1x_2 & x_1y_2 & x_1 \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & y_1x_2 & y_1y_2 & y_1 \end{bmatrix}$$

   $$h = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9]^T$$

4. The trivial solution for **h** would be the **zero vector** $0 \in R^9$. The matrix **A** with dimension R^{2Nx9} is <u>not</u> full rank because the solution (null) space of the matrix **A** has a dimension of 1, corresponding to the 8 degrees of freedom in the homography matrix **H.** Because of the fact that the matrix **A** is not full rank, it will have at least one eigenvalue that is very close to 0. The eigenvector associated with this near-zero eigenvalue will be the least-squares solution for the parameter vector **h** (in other words, homography matrix **H**).

Q2 FAST Detector

1. The Harris corner detector, which we studied in our lectures, is based on the “cornerness” measure, which is a function of the eigenvalues of the 2x2 covariance matrix formed from the derivatives from the horizontal and vertical directions computed at every pixel over a small region. The computation of eigenvalues can be skipped by using the determinant and the trace of the covariance matrix to determine the “cornerness” measure.

   The FAST detector operates by examining a circular patch around a candidate corner point and checking if there is a set of **n** (hyperparameter) contiguous pixels that are either all brighter or all darker than the central pixel by a certain threshold. If such a set of n contiguous pixels is found, then the candidate pixel is considered a corner.

2. Compared to the Harris corner detector, which involves computationally expensive operations like calculating the covariance matrix for every pixel over a small region, the FAST detector's simple examination of a circular patch around a pixel can be performed much faster.

Q3 BRIEF Descriptor

1. As we saw during the lecture, the filterbanks are used for extracting features instead of describing them or comparing them across images. However, the BRIEF is a feature descriptor which represents the image patch around a keypoint as a binary string. Each bit is the result of a simple brightness comparison between a pair of pixels in the patch, where the pixels were selected according to some distribution like uniform or Gaussian distribution.
2. Yes, it is possible to use the filterbanks to encode the local image structure around a keypoint and use it as a descriptor. What we can do is to choose a set of filters that capture local image properties like edges and orientations, and then use the filter responses as the descriptor for the keypoint. **However**, the filterbank descriptors may not be as discriminative as existing advanced methods like SIFT or BRIEF. Thus, it may not be effective to use filterbanks as descriptors.

Q4 Matching Methods

1. The Hamming distance between two binary strings can be computed very efficiently using bitwise operations, such as the XOR operation followed by counting the number of set bits. This is much faster than computing the Euclidean distance between two high-dimensional vectors, which involves more computationally expensive floating-point operations.

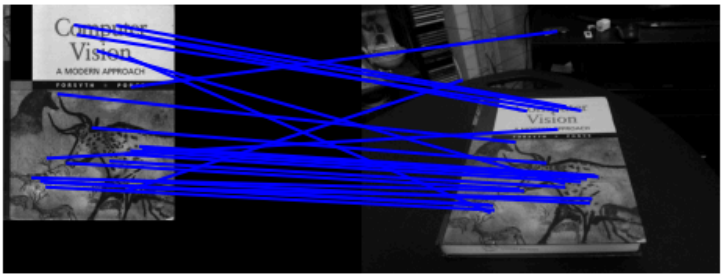
2. The Hamming distance between two binary strings is the number of positions where the corresponding bits are different. It can be computed efficiently using the XOR operation

and counting the number of set bits for two binary BRIEF descriptors. The Hamming distance ranges from 0 (identical descriptors) to the length of the BRIEF descriptor (completely different descriptors).

When it comes to the Nearest Neighbor method, which is based on the idea of finding the nearest neighbor using some distance measure (in our case Hamming distance), we can match feature points between two images using BRIEF descriptor. For each BRIEF descriptor in the first image, we find the descriptor in the second image that has the smallest Hamming distance.

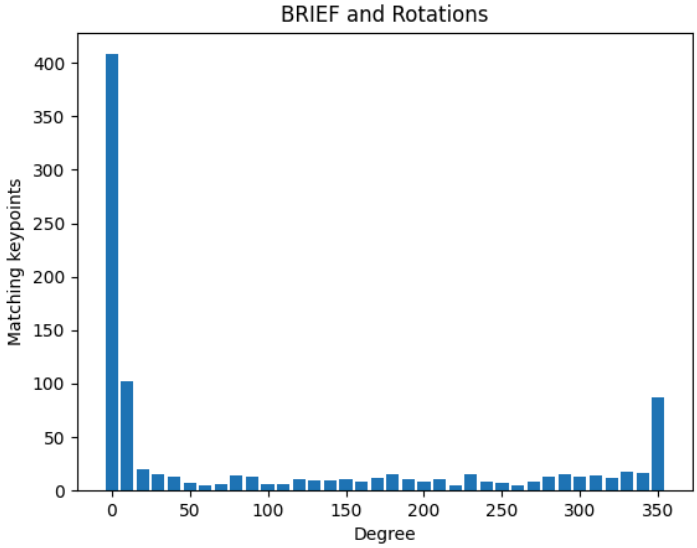
Q5 Feature Matching

1. We tried to change the value of the **ratio** variable to 0.7 and 0.8, but the best result seemed to come from the default 0.8 value, which is shown below.



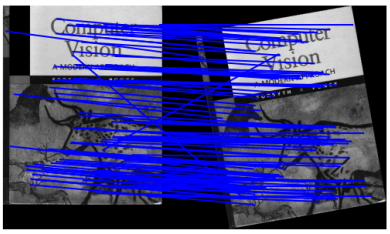
Q6 BRIEF and Rotations

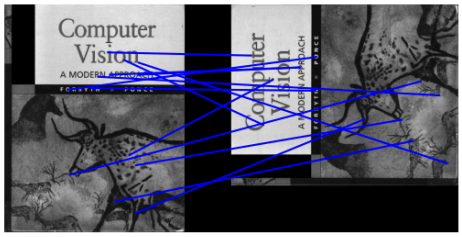
1. Here is the histogram of the number of matching keypoints (hyperparameters: n = 12, thres = 0.15, ratio = 0.8) for different rotations of the image.



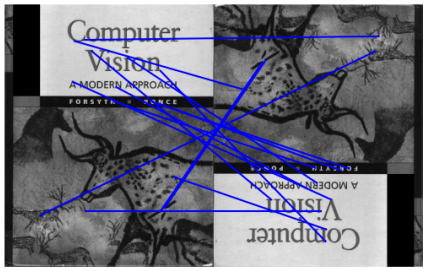
We can see that when there is no rotation (zero degree in the above figure), as expected, all found feature points were matched. This is still a considerable number of matched feature points when the angle of rotation is small in the positive or negative direction (10 and 350 degrees). However, as the rotation angle diverges more from zero, then the number of matched feature points drastically decreases to a very small number. This is because of the fact that the BRIEF descriptor is not inherently invariant to rotations and scaling as the binary representation coming from the intensity comparisons between the different pairs in an image patch are not robust to those transformations.

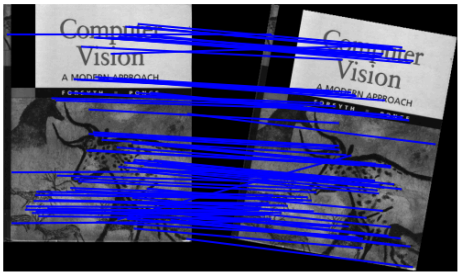
2. Below are the figures for matched features for rotations of 10 (left) and 90 (right) degrees.





Below are the figures for matched features for rotations of 180 (left) and 350 (right) degrees.





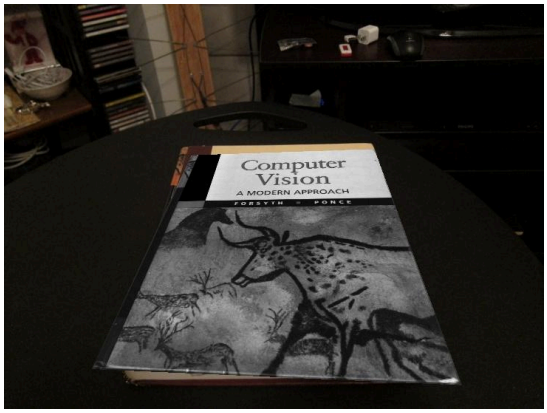
Q7 ORB Descriptor

1. The ORB descriptor combines the FAST keypoint detector and the BRIEF descriptor, with additional modifications to improve the descriptor's ability to be rotation invariant. It starts by detecting keypoints using the FAST detector, similar to BRIEF, then computes the orientation of each keypoint using the intensity centroid method, which approximates the keypoint's dominant orientation. The BRIEF descriptor is then constructed, but with a modification: the pixel location pairs used for the brightness comparisons are rotated based on the computed keypoint orientation. This makes the ORB descriptor more robust to image rotation compared to the BRIEF descriptor, as the brightness comparisons are performed in a rotationally-aligned manner.

2. Above, we discussed why the ORB descriptor method is robust to rotations. However, ORB is not fully scale-invariant, as it does not incorporate a scale-invariant keypoint detection mechanism like the scale-space search used in SIFT.

Q8 Computing the Homography

1. The result of the warping the **cv_cover** image onto the **cv_desk** image using unnormalized homography is shown on the right. We used **backward wrapping** and the **top 8 closest matching points** extracted from the **ORB descriptor** (twice the number of required matching points to calculate the homography matrix).

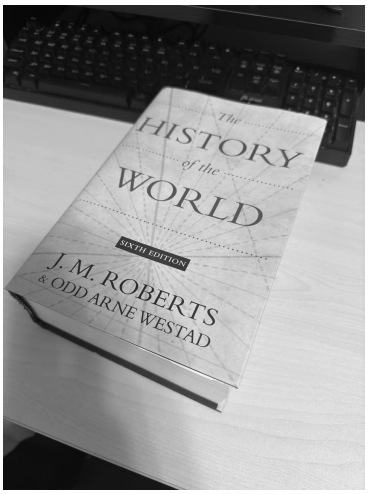


Q9 Homography with Normalization

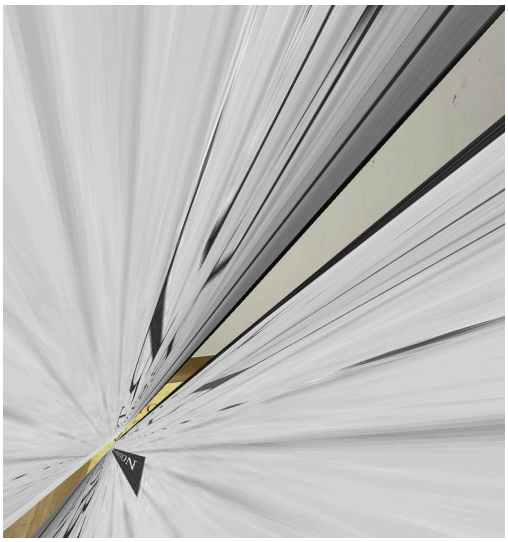
1. Here is the implementation of homography with normalization

```python
def computeH_norm(x1, x2):
    # Q9
    # Compute the centroid of the points
    x1_mean = np.mean(x1, axis=0)
    x2_mean = np.mean(x2, axis=0)
    # Shift the origin of the points to the centroid
    x1_norm = x1 - x1_mean
    x2_norm = x2 - x2_mean
    # Normalize the points so that the largest distance from the origin is equal to sqrt(2)
    x1_max = np.max(np.linalg.norm(x1_norm, axis=1))
    x2_max = np.max(np.linalg.norm(x2_norm, axis=1))

    x1_scale = np.sqrt(2) / x1_max  # set the largest distance to sqrt(2)
    x2_scale = np.sqrt(2) / x2_max  # set the largest distance to sqrt(2)

    x1_norm = x1_norm * x1_scale
    x2_norm = x2_norm * x2_scale

    # Similarity transform 1
    T_1 = np.array([[x1_scale, 0, -x1_scale * x1_mean[0]],
                    [0, x1_scale, -x1_scale * x1_mean[1]],
                    [0, 0, 1]])
    # Similarity transform 2
    T_2 = np.array([[x2_scale, 0, -x2_scale * x2_mean[0]],
                    [0, x2_scale, -x2_scale * x2_mean[1]],
                    [0, 0, 1]])
    # Compute homography
    H2to1 = computeH(x1_norm, x2_norm)

    # Denormalization
    H2to1 = np.dot(np.linalg.inv(T_1) @ H2to1, T_2)

    return H2to1
```

2. We will use the two images, which are shown below to illustrate why normalization helps. The first image (on the left side) will be warped into the second image (on the right side).



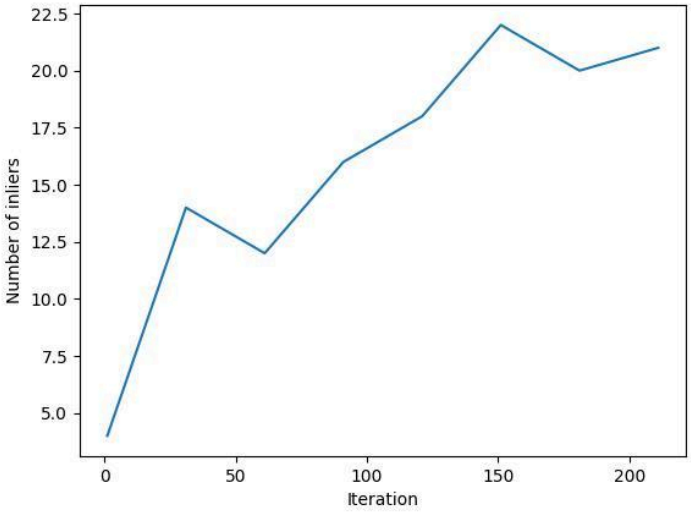
Below, the **left image** is the result of applying the homography matrix **without normalization**, and the **right image** is the result of applying the homography matrix **with normalization**.



We can see that If the scale of the matching points differs significantly, the computed homography may not accurately align the images, leading to distortions in the stitched result. Homography normalization helped to make the homography estimation more robust to differences in scale.

Q10 Implement RANSAC for computing a homography

1. Here is the plot of the number of inliners for different iterations. We used **the threshold** equal to 2. The total number of matches was 120.



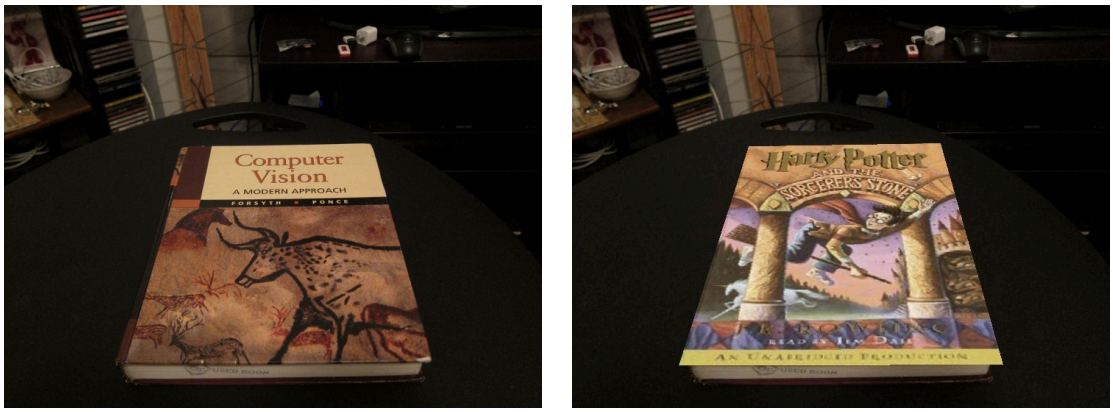
2. As we learned during the lecture, we can use the following formula to determine the number of iterations:

$$N = \frac{log(1-p)}{log(1-(1-e)^{s})}$$

N - number of iterations
p - the probability that at least one random sample is free from outliers (in other words, the desired accuracy level, for example 0.99)
e - outlier ratio
s - minimum number of sampled points needed to fit the model

Q11 Putting it Together

1. On the **right image**, we can see the result of warping the **hp_cover image** onto the **cv_desk image**



Q12 Incorporating Video

1. We created a video in which we warped the frames from the ar_source.mov video to the book.mov video. Here is one of the frames of the composite video.

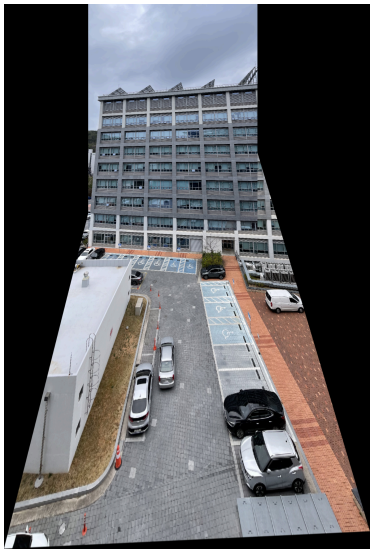
Q13 Creating a Simple Panorama

1. Here are the two images we chose for building a panorama image.

The generated panorama image is shown below.

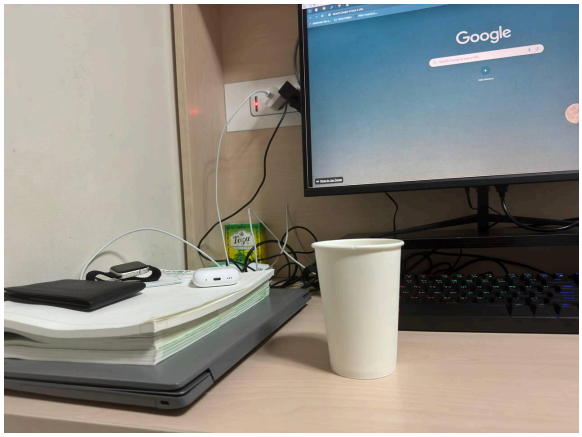
Q14 Create a Panorama Image from Unordered Multi-images

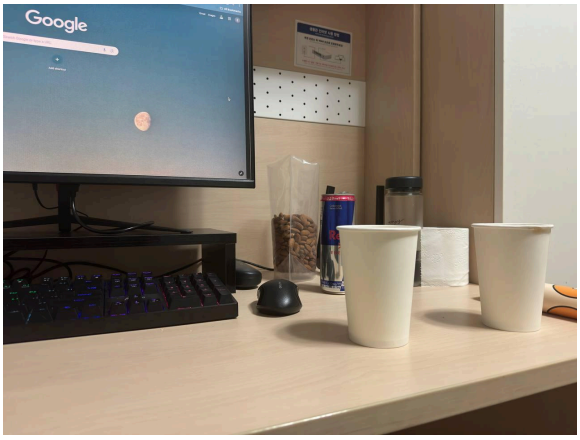
1. We determine the order of images and build a panorama in the following way.
    1) We first randomly choose an image out of the provided images and treat that image as a reference image.
    2) Then, from the pool of other images, we select the best matching image based on the maximum number of inliers that come from computing the homography matrix with RANSAC.
    3) After we choose the best match, we remove that matched image from the pool of candidate images, and extract the corner coordinates from the reference and the matched image.
    4) We compute the inverse homography matrix from the reference image to the matched image, and then project the calculated corner coordinates onto the matched image coordinate system using the inverse homography matrix.
    5) Then we determine the bounding box of the warped image by finding the minimum and maximum coordinates of the projected and original corner points from the reference and the matched image, respectively.
    6) We create a translation matrix to translate the image coordinates such that the top-left corner of the bounding box aligns with the origin (0,0). This matrix is used to adjust the coordinates of the warped image.
    7) We warp the reference image on the matched image using the transformation matrix, which comes from the matrix multiplication of the constructed translation matrix and the inverse homography matrix.
    8) The warped image will be an intermediate panorama output and will be used as a reference image in the next iteration. The iteration will stop when there are no more images in the candidate pool.
    9) To ensure that we have the best panorama, we iterate through all the images by treating them as a reference image one by one. In the end, we choose the constructed panoramas that have the highest total number of inliers coming from the RANSAC.
2. Here are the panoramas that we constructed from the given images. The panorama on the **left comes from stage2**, the panorama on the **middle comes from stage3**, and the panorama on the **right comes from the final stage 4**.



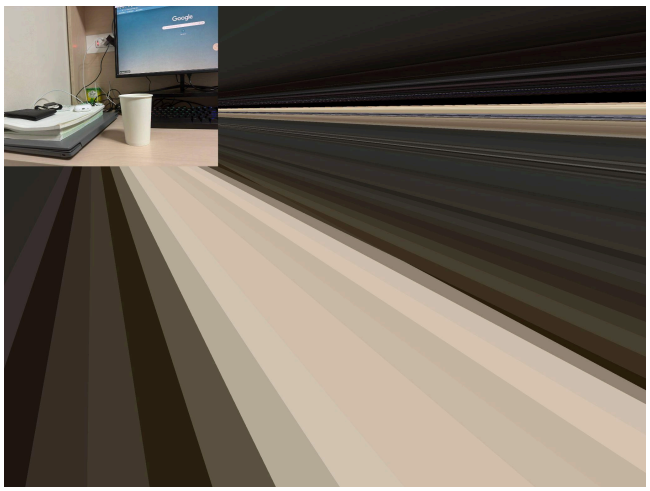
Q15 Discussion: Limitations of Homography Warping (degenerated case)

1. Here are the two images that we chose to show the degenerate case when applying homography transformation.





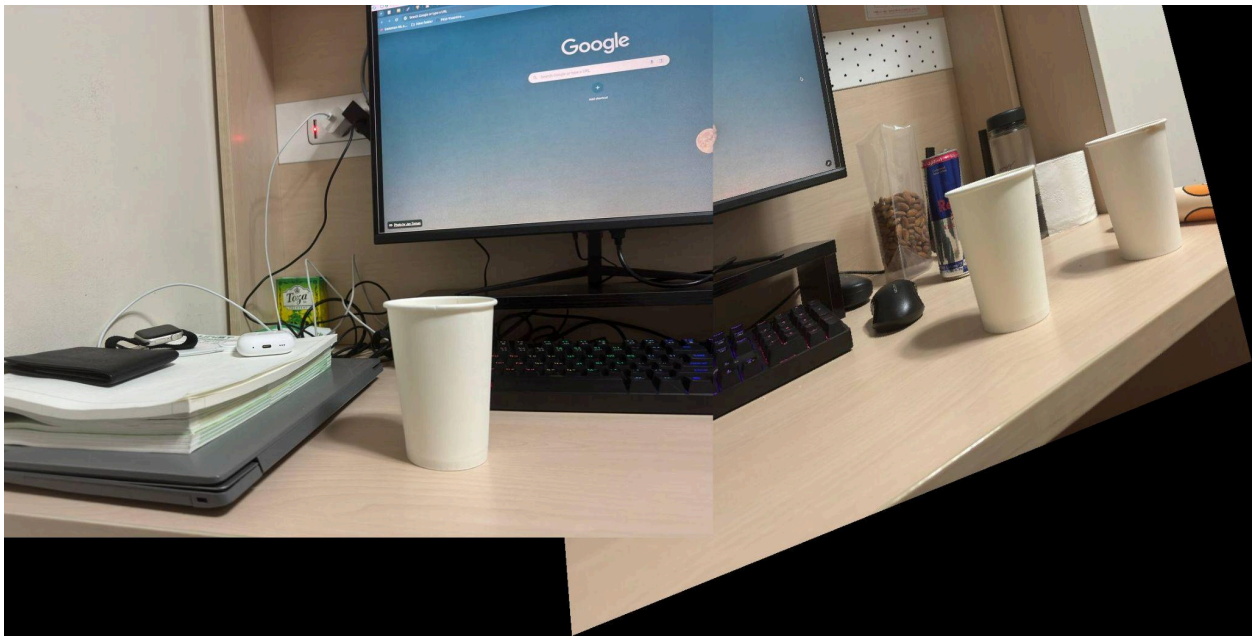
After stitching the right image onto the left image using the code for Q13, we got the following result:



We can see that the image stitching was unsuccessful. The reason is that we have multiple instances of the same object (white cup) in the original left and right images. When we find matching point pairs from those images, there will be a lot of matching pairs corresponding to the same objects. However, if those pairs are used for homography matrix computation, then they will alter it, leading to undesired image stitching results.

2. To address a subset of such cases, in which the different instances of the same object are located far from the image border shared by the two images, we will only compute the homography matrix with the top k (we used 12) closest matching points coming from the ORB detector. By doing that, we can avoid including the matched points corresponding to the same object located far from the shared image border.

Here is the result of applying the above solution.



As we can see, the generated panorama is much better than what we had before.