Q1.1 Homography

Let $\mathbf{x_1}$ and $\mathbf{x_2}$ be the points from two camera views and $\mathbf{x}_\pi$ be the point on plane π.
We know that $\mathbf{x_1} \equiv \mathbf{H_1 x}_\pi$ and $\mathbf{x_2} \equiv \mathbf{H_2 x}_\pi$.
Then we have: $\lambda_1 \mathbf{x_1} = \mathbf{H_1 x}_\pi$ and $\lambda_2 \mathbf{x_2} = \mathbf{H_2 x}_\pi$, where λ_1 and λ_2 are scalars and $\lambda_1 \lambda_2 \neq 0$.
For $\lambda_2 \mathbf{x_2} = \mathbf{H_2 x}_\pi$, we can get $\mathbf{x}_\pi = \lambda_2 \mathbf{H_2^{-1} x_2}$.
Substitute $\mathbf{x}_\pi$ with $\lambda_2 \mathbf{H_2^{-1} x_2}$, we can get $\lambda_1 \mathbf{x_1} = \lambda_2 \mathbf{H_1 H_2^{-1} x_2}$.
Since $\lambda_1 \lambda_2 \neq 0$, we can know $\mathbf{x_1} = \frac{\lambda_2}{\lambda_1} \mathbf{H_1 H_2^{-1} x_2}$
It suggests that $\mathbf{x_1} \equiv \mathbf{H x_2}$, where $\mathbf{H} = \mathbf{H_1 H_2^{-1}}$.

Q1.2 Correspondences

1. **h** has 8 degree of freedom. **h** has 9 elements but the scale does not matter. So $9 - 1 = 8$.

2. 4 points are required to solve **h**.

3.

$$\mathbf{x_1^i} \equiv \mathbf{H}\mathbf{x_2^i} \quad (i \in \{1 \cdots N\})$$
$$\begin{pmatrix} x_1^i \\ y_1^i \\ 1 \end{pmatrix} = \lambda \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_2^i \\ y_2^i \\ 1 \end{pmatrix} \quad (\lambda \neq 0) \tag{1}$$

From (1), we can get:

$$\begin{aligned} x_1^i &= \frac{h_{11}x_2^i + h_{12}y_2^i + h_{13}}{h_{31}x_2^i + h_{32}y_2^i + h_{33}} \\ y_1^i &= \frac{h_{21}x_2^i + h_{22}y_2^i + h_{23}}{h_{31}x_2^i + h_{32}y_2^i + h_{33}} \end{aligned} \tag{2}$$

Linearize (2), we can get:

$$\begin{aligned} h_{11}x_2^i + h_{12}y_2^i + h_{13} - h_{31}x_2^i x_1^i - h_{32}y_2^i x_1^i - h_{33}x_1^i = 0 \\ h_{21}x_2^i + h_{22}y_2^i + h_{23} - h_{31}x_2^i y_1^i - h_{32}y_2^i y_1^i - h_{33}y_1^i = 0 \end{aligned}$$

Rewrite it as $A_i h = 0$ and then we can have: $A_i = \begin{pmatrix} x_2^i & y_2^i & 1 & 0 & 0 & 0 & -x_2^i x_1^i & -y_2^i x_1^i & -x_1^i \\ 0 & 0 & 0 & x_2^i & y_2^i & 1 & -x_2^i y_1^i & -y_2^i y_1^i & -y_1^i \end{pmatrix}$

4. A trivial solution for **h** is **0**.
Suppose **A** is a square matrix and its shape is 3×3 matrix, then **A** is not full rank. The reason is that we know there would be some non-trivial solution for **h**. Thus the Null space of **A** is not empty and **A** is not full rank.
A will have at least one 0 eigenvalue. There is also a non-unique eigenvector corresponding to the 0 eigenvalue.

Q1.3 Homography under rotation

For camera 1, we have $\mathbf{x}_2 = \mathbf{K}_2[\mathbf{R} \quad \mathbf{0}]\mathbf{X}$.
Since K is a 3×3 matrix with full rank and $\mathbf{R}$ is also a 3×3 rotation matrix with full rank, then they are both invertible. We can know that $\mathbf{X} = \begin{bmatrix} \mathbf{R^{-1}} \\ \mathbf{0} \end{bmatrix} \mathbf{K_2}^{-1}\mathbf{x_2}$.
Substitute $\mathbf{X}$ in $\mathbf{x_1} = \mathbf{K_1}[\mathbf{I} \quad \mathbf{0}]\mathbf{X}$, we can have:
$\mathbf{x_1} = \mathbf{K_1}[\mathbf{I} \quad \mathbf{0}] \begin{bmatrix} \mathbf{R^{-1}} \\ \mathbf{0} \end{bmatrix} \mathbf{K_2}^{-1}\mathbf{x_2} = \mathbf{K_1}\mathbf{R^{-1}}\mathbf{K_2}^{-1}\mathbf{x_2}$.
Thus, there exists a homography $\mathbf{H}$ that satisfies $\mathbf{x_1} \equiv \mathbf{H}\mathbf{x_2}$, where $\mathbf{H} = \mathbf{K_1}\mathbf{R^{-1}}\mathbf{K_2}^{-1}\mathbf{x_2}$.

Q1.4 Understanding homographies under rotation

From Q1.3, we know that there is $\mathbf{x_1} \equiv \mathbf{H}\mathbf{x_2}$, where $\mathbf{H} = \mathbf{K_1}\mathbf{R}^{-\mathbf{1}}\mathbf{K_2}^{-1}\mathbf{x_2}$.
In this question, let's first derive the $\mathbf{H}$ such that $\mathbf{x_2} = \mathbf{H}\mathbf{x_1}$, where $\mathbf{x_2}$ is $\mathbf{x_1}$ after rotating by θ. Similarly, we can know that $\mathbf{H} = \mathbf{K_2}\mathbf{R}\mathbf{K_1^{-1}}$.
Here, we know that $\mathbf{K_1} = \mathbf{K_2}$, so $\mathbf{H} = \mathbf{K}\mathbf{R}\mathbf{K}^{-\mathbf{1}}$.
$\mathbf{H^2} = (\mathbf{K}\mathbf{R}\mathbf{K}^{-\mathbf{1}})(\mathbf{K}\mathbf{R}\mathbf{K}^{-\mathbf{1}}) = \mathbf{K}\mathbf{R}(\mathbf{K}^{-\mathbf{1}}\mathbf{K})\mathbf{R}\mathbf{K}^{-\mathbf{1}} = \mathbf{K}\mathbf{R^2}\mathbf{K}^{-\mathbf{1}}$, where $\mathbf{R^2}$ represents rotating by 2θ.
Thus, H^2 is the homography corresponding to a rotation of 2θ.

Q1.5 Limitations of the planar homography

If the corresponding planes of the two viewpoints are not on the same plane in the world coordinate, then the homography no longer holds. That's the reason planar homography is not completely sufficient to map any arbitrary scene image to another viewpoint.

Q1.6 Behavior of lines under perspective projections

A line in the 3D space can be denoted as $\mathbf{X} = \mathbf{X_0} + t\mathbf{D}$, where $\mathbf{X_0}$ is a point on the line, $\mathbf{D}$ is the line travel direction and t is a scalar.
Then, $\mathbf{x} = \mathbf{PX} = \mathbf{PX_0} + t\mathbf{PD}$.
Assume $\mathbf{x_0} = \mathbf{PX_0}$ and $\mathbf{d} = \mathbf{PD}$, then we can have $\mathbf{x} = \mathbf{x_0} + t\mathbf{d}$, where $\mathbf{x_0}$ and $\mathbf{d}$ both represent vectors in the 2D space.
Therefore, the projection $\mathbf{P}$ in $\mathbf{x} = \mathbf{PX}$ preserves lines.

Q2.1.1 FAST Detector

Differences: FAST detector is designed to find key points or corners in an image by comparing the intensity of pixels in a circular neighborhood around a candidate pixel with the intensity of the candidate pixel itself. It looks for a set of contiguous pixels that are all brighter or darker than the candidate pixel by a certain threshold. The Harris corner detector, on the other hand, operates based on the idea that corners have large intensity changes in multiple directions. It uses a score function to decide whether a region is a corner or not.

Computational Performance: FAST detector is faster because it only requires a simple intensity comparison between a few pixels in a circular pattern. It also involves a high-speed test to further accelerate the process. However, the Harris detector can be very slow because it requires the calculation of local gradients as well as eigenvalues of the inertia matrix.

Q2.1.2 BRIEF Descriptor

BRIEF is a binary descriptor. Each element of the binary vector is determined by comparing the intensity of pairs of pixels around a feature point. The feature representation is compact and efficient for storage and matching.
Filter banks are used for feature extractions. They produce continuous-valued feature responses that encode various characteristics of the image. They are not compact and thus not as efficient as BRIEF.

In theory, one of the filter banks can be used as a descriptor. However, they may contain less compact information and are less computationally efficient compared to common feature descriptors such as BRIEF.

Q2.1.3 Matching Methods

When matching two images A and B, we first obtain their interest points and associated descriptors. For each point in image A, we find its nearest neighbor (i.e. smallest Hamming distance) in image B. They are considered as potential matches.

Hamming distance is faster when computing the distance between two binary strings. They only need to apply XOR operations on corresponding positions and then count the appearances of 1s. However, Euclidean distance is required to compute the squared error on each position and sum over all the float numbers, which essentially brings more computational cost.

Q2.1.4 Feature Matching

The result image showing the matched points is shown below (see Figure 1).

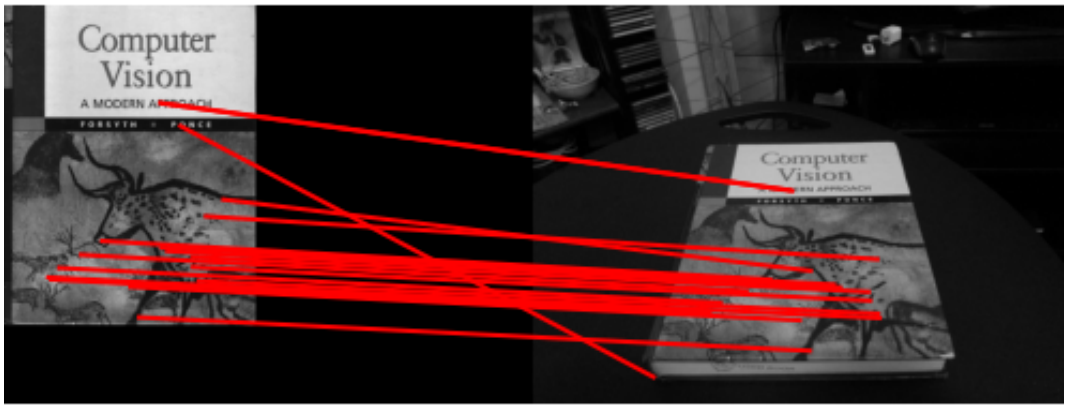


Figure 1: Illustration of matched points.

Q2.1.5 Feature Matching Parameter Tuning

4 ablation studies are conducted and the results are shown in Table 1.

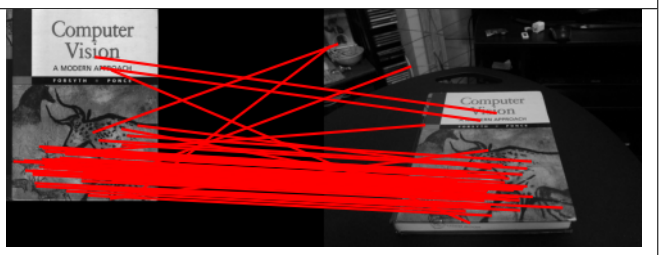
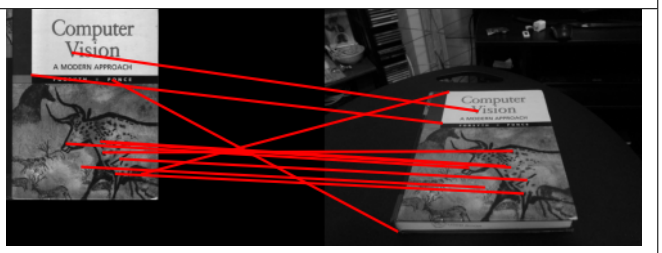
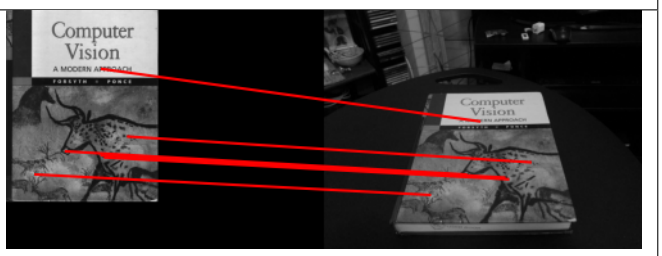
sigma	*ratio*	**Matched Features**
0.1	0.7	
0.2	0.7	
0.15	0.6	
0.15	0.8	

Table 1: Ablation Studies

sigma decides whether the pixels on the circle should be brighter or darker w.r.t. the center. Increasing *sigma* will lead to fewer corners detected.
ratio is the maximum ratio of distances between the first and second closest descriptor in the second set of descriptors. This threshold is useful to filter ambiguous matches between the two descriptor sets. Increasing *ratio* will lead to more feature matches.

Q2.1.6 BRIEF and Rotations

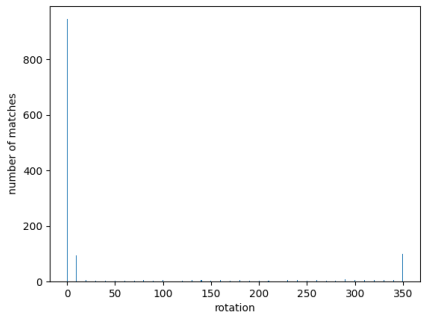


Figure 2: Number of matches w.r.t. different rotation angles.

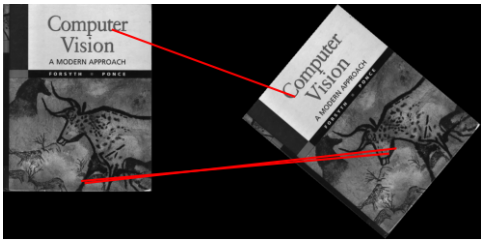


Figure 3: Visualization when rotation angle is 50.

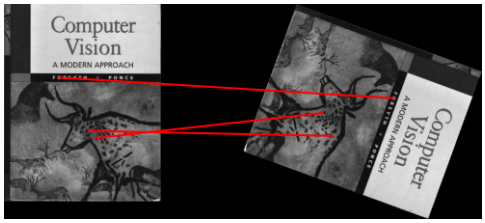


Figure 4: Visualization when rotation angle is 250.

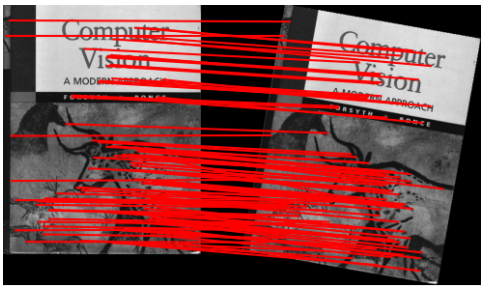


Figure 5: Visualization when rotation angle is 350.

From Figure 2, we can observe that the BRIEF descriptor is not rotation-invariant. The reason could be BRIEF computes its binary descriptor by comparing the intensities of pixel pairs in a fixed pattern around a keypoint. These pixel pairs are defined based on a predetermined sampling pattern. Since the pixel pairs are fixed and do not adapt to the local orientation of the image features, they are not rotation-invariant.

Q2.2.4 Putting it together

4. The reason is that cv_cover.jpg and hp_cover.jpg don't have the same size. Therefore, after warping hp_cover.jpg, it does not fill the same space as the book. The solution is to resize hp_cover.jpg to have the same size as cv_cover.jpg.
6. The result is shown below (see Figure 6).

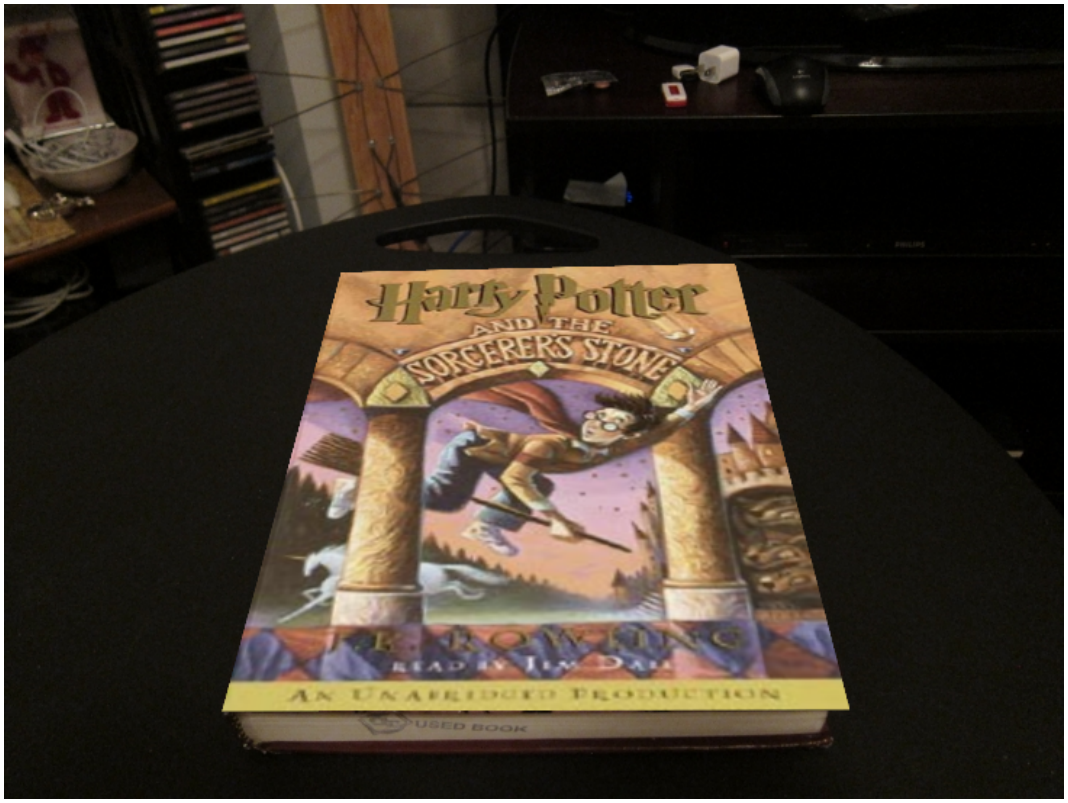


Figure 6: Warped hp cover.

Q2.2.5 RANSAC Parameter Tuning

Four ablation experiments are run, as shown in Table 2.

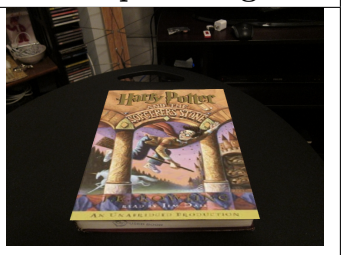
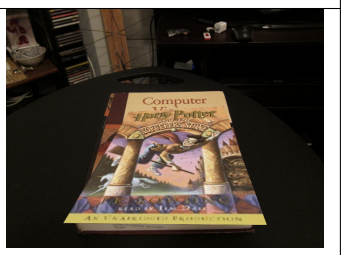
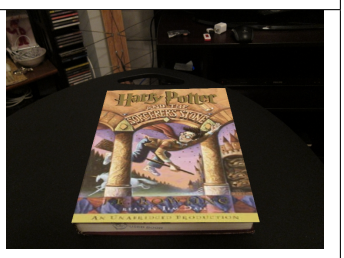
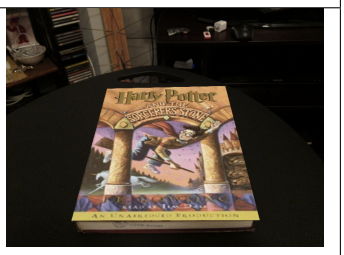
max_iters	inlier tol	Warped Image
500	1	
500	8	
100	2	
1000	2	

Table 2: Ablation Studies on RANSAC Parameter Tuning.

According to the experiment results, **max_iters** is the number of iterations we run the RANSAC algorithm. The more iterations we run, the more likely that we will find a set of correspondences without an outlier.
inlier tol is the threshold whether we consider a correspondence to be an inlier or not. If we increase **inlier tol**, we can find the resulting image becomes worse. That is because the homography fits some noise in the correspondences.

Q3.1 Incorporating video

The video can be accessed through this link: Google Drive Link.

Figure 7: Images from top to bottom: Overlay at left, center and right.

Q4.1x: Make Your AR Real Time

The bottleneck mainly arises in feature extraction and homography computation. In addition, processing video frames can also be paralleled using multi-core processing.

For feature extraction, I replaced my original implementation with OpenCV's cv2.ORB_create(). It computes the keypoints as well as the descriptors.
For feature matching, I used OpenCV's cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True) to replace my original implementation. It can obtain matched points from two sets of descriptors.
For homography computation, I used OpenCV's cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0). It also uses the RANSAC algorithm and is faster than my original implementation.
I observe that the program can be even faster with multi-core processing by using the Python multiprocessing package. I would implement it if I am given more time.

For details of implementation, please refer to ar_ec.py.

Finally, I achieved an FPS of **81.42**, which is real-time.

Q4.2x: Create a Simple Panorama

Figure 8: Original left image.

Figure 9: Original right image.

Figure 10: Panorama result image.