CMU Computer Vision HW4

Jiaqi Geng

November 3rd 2020

1 Keypoint Detector

Q1.1

This part has already been implemented.

Q1.2

See the code for my implementation.

Q1.3

Dissimilarity:

Harris Corner method calculate $R = det(M) - k * Trace(M)^2$
This method calculate $R = Trace(H)^2/det(H)$

H is the Hessian of the Difference of Gaussian function
M is the Hessian of weighted sums of nearby pixels values around each pixel.

Similarity:

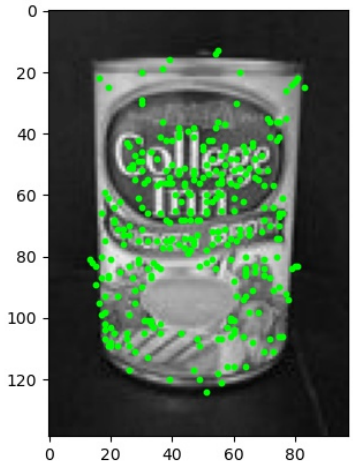
The general idea is the same.
They both compute the covariance matrix,
then compute eigenvectors and eigenvalues
and finally use threshold on eigenvalues to detect corners

Q1.4

See the code for my implementation.

Q1.5

Key points for model_chickenbroth.jpg



2 BRIEF Descriptor

Q2.1

See the code for my implementation.

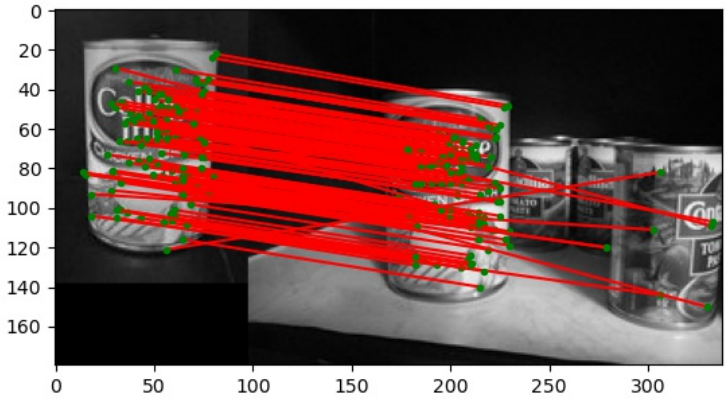
Q2.2

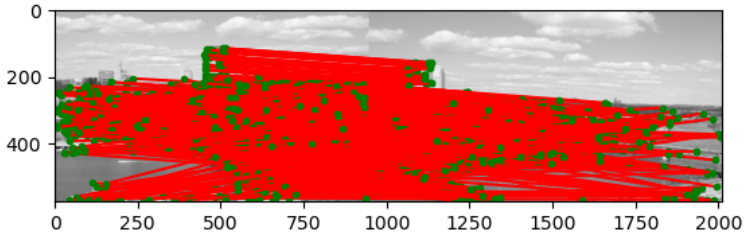
See the code for my implementation.

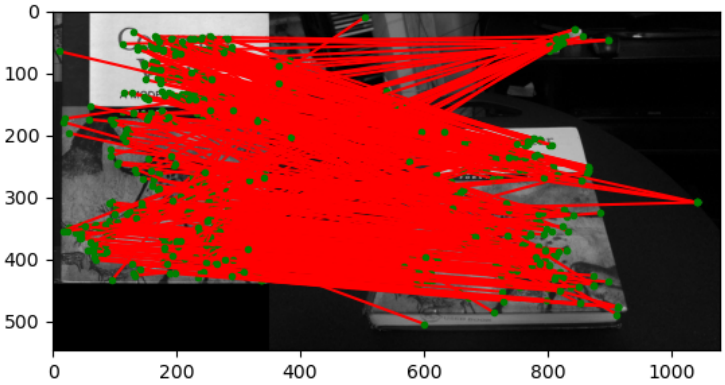
Q2.3

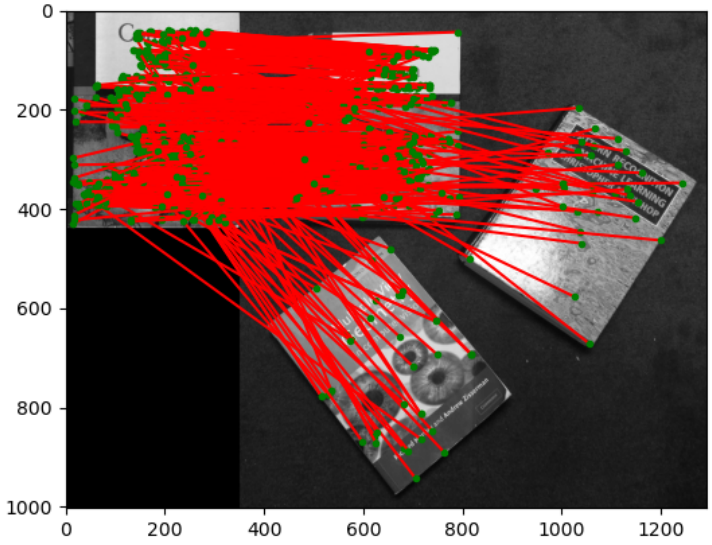
See the code for my implementation.

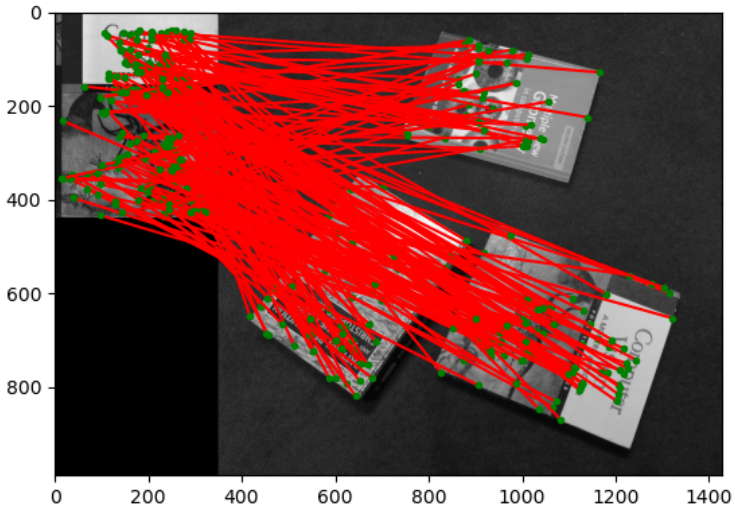
Q2.4

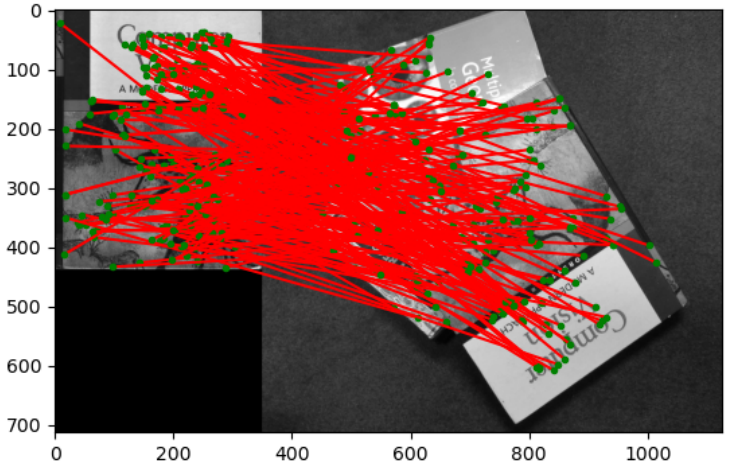

0
20
40
60
80
100
120
140
160
0
50
100
150
200
250
300

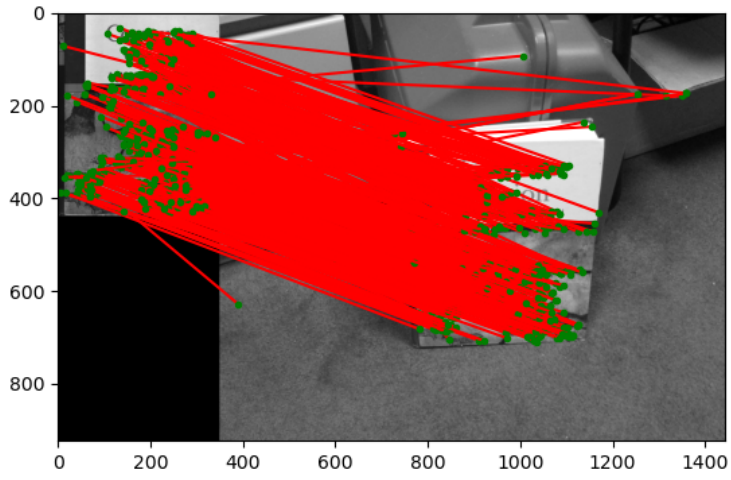

0
200
400
0
250
500
750
1000
1250
1500
1750
2000


0
100
200
300
400
500
0
200
400
600
800
1000


0
200
400
600
800
1000
0
200
400
600
800
1000
1200


0
200
400
600
800
0
200
400
600
800
1000
1200
1400

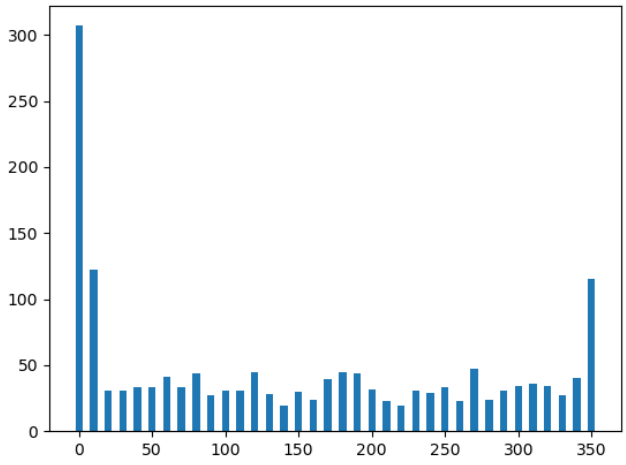

0
100
200
300
400
500
600
700
0
200
400
600
800
1000



Discussion:

My program performs well when the images are not being rotated.
We can see that the last CV cover image is really good,
while previous ones (with rotation) contain many incorrect matches.
See reason in Q2.5

Q2.5



I think this is because when we rotate the image,
the patch for each pixel is not rotated.
Therefore, after rotation,
we will get different patches around corresponding pixels from two images.

3 Planar Homographies: Theory

Q3.1

(a)

We are given that

$$\lambda_n \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = H \begin{bmatrix} u_n \\ v_n \\ 1 \end{bmatrix}$$

We can represent $H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$

Therefore, $\lambda_n x_n = h_1 u_n + h_2 v_n + h_3$
$\lambda_n y_n = h_4 u_n + h_5 v_n + h_6$
$\lambda_n = h_7 u_n + h_8 v_n + h_9$

We divide the first equation and second equation by the third equation
We can get
$x_n(h_7 u_n + h_8 v_n + h_9) = h_1 u_n + h_2 v_n + h_3$
$y_n(h_7 u_n + h_8 v_n + h_9) = h_4 u_n + h_5 v_n + h_6$

Then we can expand and rearrange these equations:
$-h_1 u_n - h_2 v_n - h_3 + h_7 u_n x_n + h_8 v_n x_n + h_9 x_n = 0$
$-h_4 u_n - h_5 v_n - h_6 + h_7 u_n y_n + h_8 v_n y_n + h_9 y_n = 0$

Therefore, we define $h = [h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9]^T$
We define $A_n = \begin{bmatrix} -u_n & -v_n & -1 & 0 & 0 & 0 & u_n x_n & v_n x_n & x_n \\ 0 & 0 & 0 & -u_n & -v_n & -1 & u_n y_n & v_n y_n & y_n \end{bmatrix}$
We want to have $A_n h = 0$ for $n = 1 : N$,
so we can stack all A_n vertically to form A
and have $Ah = 0$

(b)

Since H is a $3 * 3$ matrix,
there will be 9 elements in h.

(c)

We can rewrite $H = h_9 \begin{bmatrix} \frac{h_1}{h_9} & \frac{h_2}{h_9} & \frac{h_3}{h_9} \\ \frac{h_4}{h_9} & \frac{h_5}{h_9} & \frac{h_6}{h_9} \\ \frac{h_7}{h_9} & \frac{h_9}{h_9} & 1 \end{bmatrix}$
We know that $\lambda_n \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = H \begin{bmatrix} u_n \\ v_n \\ 1 \end{bmatrix}$ and λ_n can be arbitrary.
Therefore, we can let $\lambda_n = \frac{\lambda_n}{h_9}$ (still arbitrary) and $H = \begin{bmatrix} \frac{h_1}{h_9} & \frac{h_2}{h_9} & \frac{h_3}{h_9} \\ \frac{h_4}{h_9} & \frac{h_5}{h_9} & \frac{h_6}{h_9} \\ \frac{h_7}{h_9} & \frac{h_9}{h_9} & 1 \end{bmatrix}$
Thus, the degree of freedom of H is 8 (8 unknowns).
Each point pair (correspondence) gives us 2 equations.
Therefore, we need 4 point pairs (8 equations) to solve the system.

(d)

We want to minimize $||Ah||^2$
To make sure h won't a zero vector,
we add a constraint $||h||^2 = 1$
Therefore,
we want to minimize $\frac{||Ah||^2}{||h||^2}$ (Rayleigh quotient)
We can then decompose $A = U\Sigma V^T$
and now we need to minimize $\frac{argmin_h||U\Sigma V^T h||^2}{||h||^2}$
Due to orthonormality of U,
U just only rotates but doesn't scale,
so the magnitude is unchanged
Therefore, $||U\Sigma V^T h||^2 = ||\Sigma V^T h||^2$
Now we can substitute $V^T h$ with y
We know from orthonormality that $||V^T h||^2 = ||h||^2 = 1$
Now we want to find the y that minimize $||\Sigma y||^2$
When singular values are sorted in decreasing order,
we see that $y = [0, 0, 0, ..., 0, 1]^T$ minimize $||\Sigma y||^2$
We convert y back to h, $h = Vy$
Thus,
h is the last column of V

Q3.2

I did not do this part.

Q3.3

I did not do this part.

Q3.4

I did not do this part.

4 Planar Homographies: Implementation

Q4.1

See the code for my implementation.

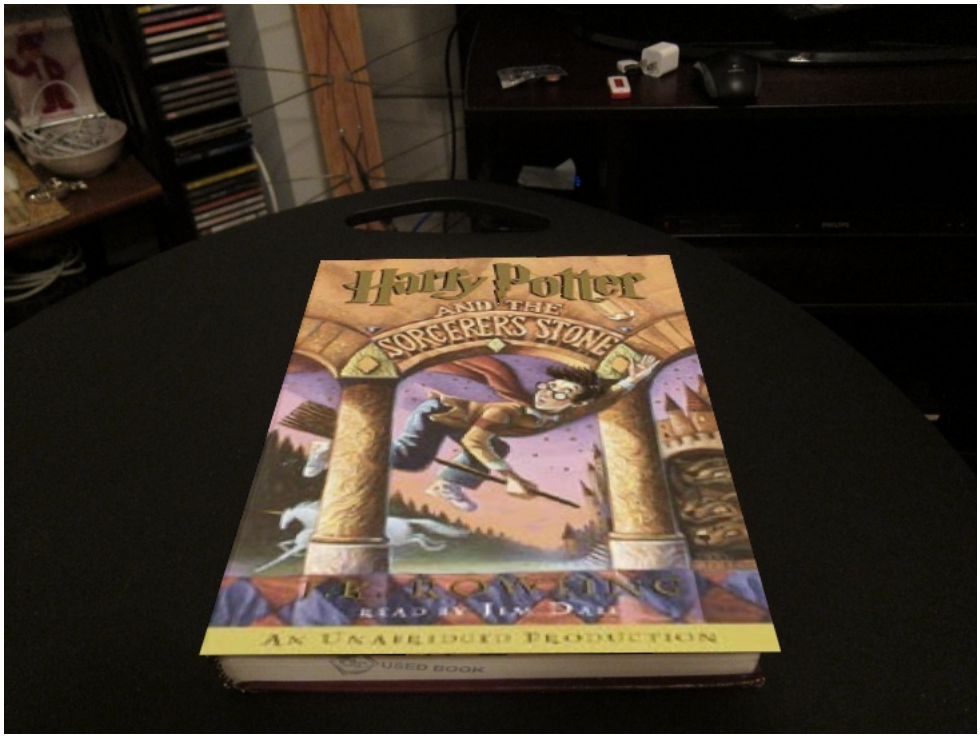
5 RANSAC

Q5.1

See the code for my implementation.

6 Automated Homography Estimation and Warping

Q6.1



The cover is not filling up the same space as the CV cover because it is smaller in size. We need to resize the Harry Potter cover to the size of CV cover.

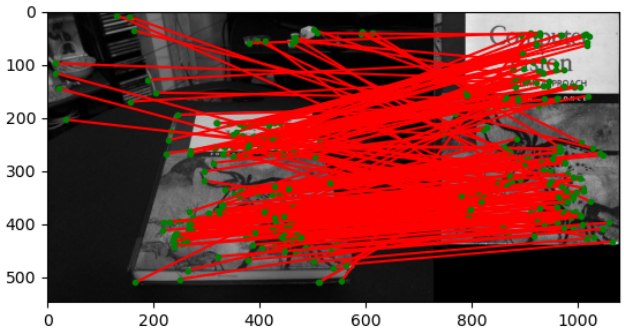
Final H:

$$\begin{bmatrix} 7.30081662e-01 & -3.39936760e-01 & 2.37483817e+02 \\ -1.90857543e-02 & 2.28157155e-01 & 1.92388375e+02 \\ -3.25385180e-05 & -9.08317782e-04 & 1.00000000e+00 \end{bmatrix}$$

Q6.2

Number of iterations:
When the number of iterations is too small,
then it is likely that the program stops before we could find a reasonable homography.
When the number of iterations if very large,
we are more likely to find a reasonable homography,
but it might take a long time to run.

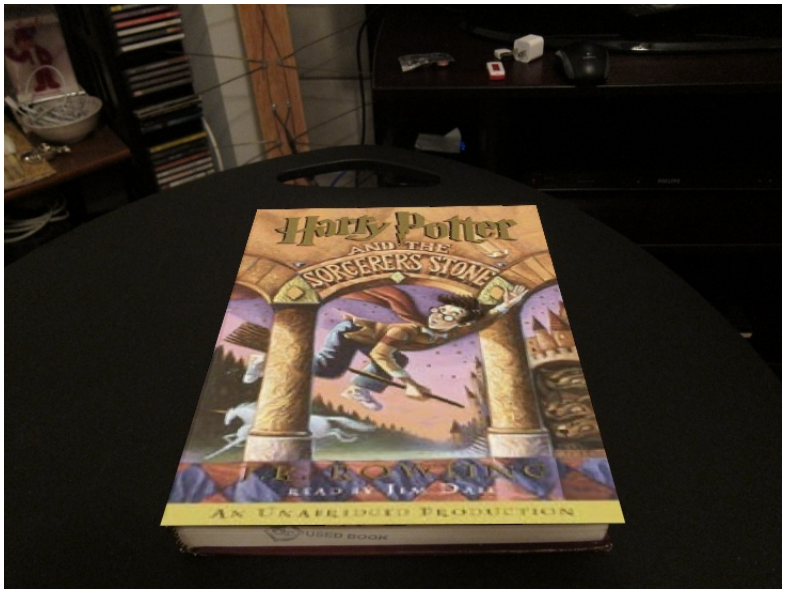
We can use the formula discussed in class to find a suitable number of iteration:
$N = \frac{\log(1-p)}{\log(1-(1-e)^s)}$
In this case, $s = 4$ and we let $p = 0.99$



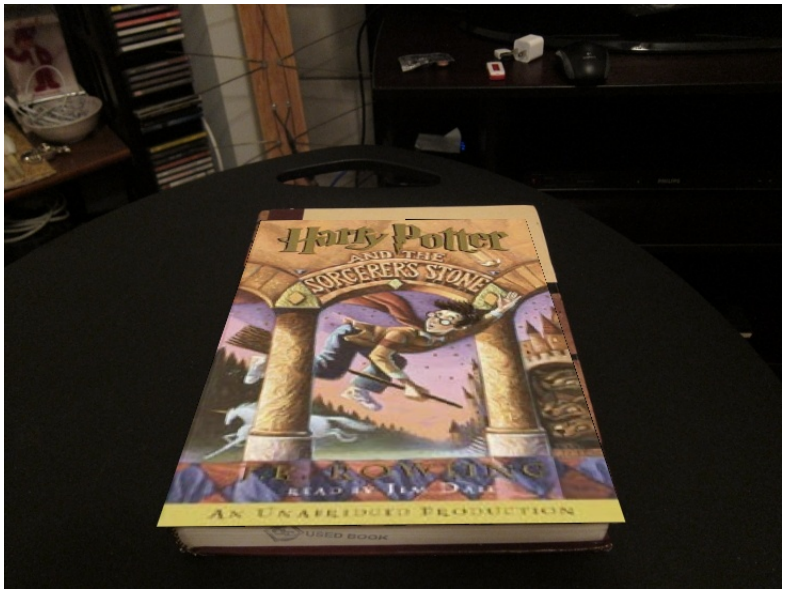
I found roughly 55 inliers (193 total matches),
so $e = 55/193$
Therefore, $N = \frac{\log(1-0.99)}{\log(1-(1-\frac{55}{193})^4)} = 695$
Thus, I choose 700 as my number of iteration.

Tolerance value:
When the tolerance value is too small,
we could not find enough inliers to obtain a reasonable homography.
When the tolerance value is too large,
then we would include too many incorrect matching points as inliers,
giving us bad homography.

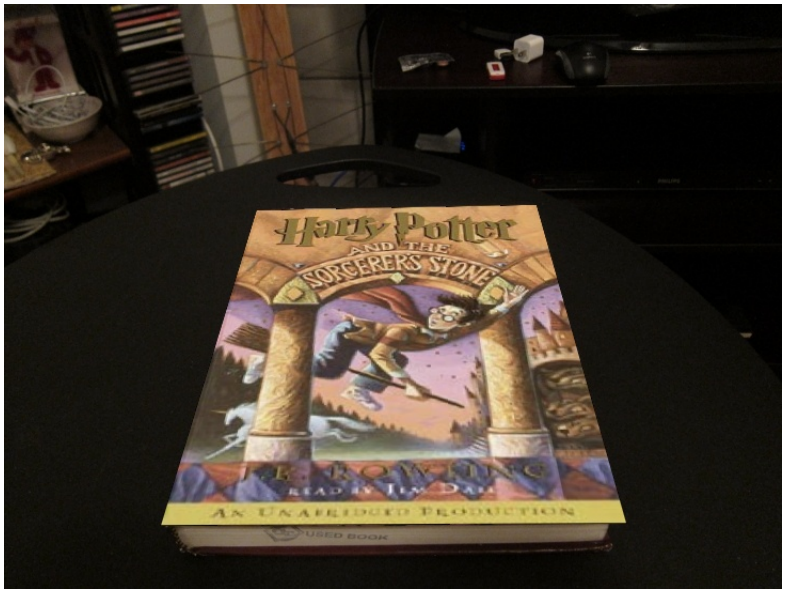
MAX_iter = 700, tol = 2



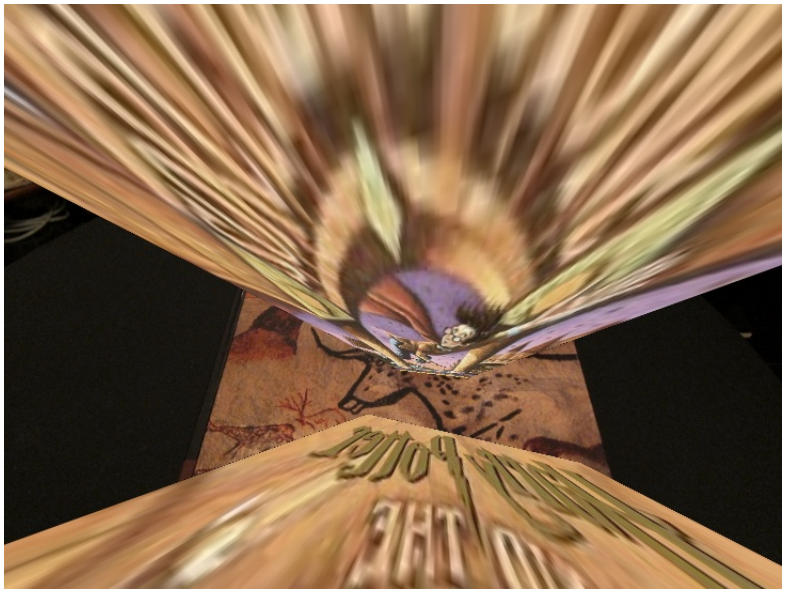
MAX_iter = 100, tol = 2



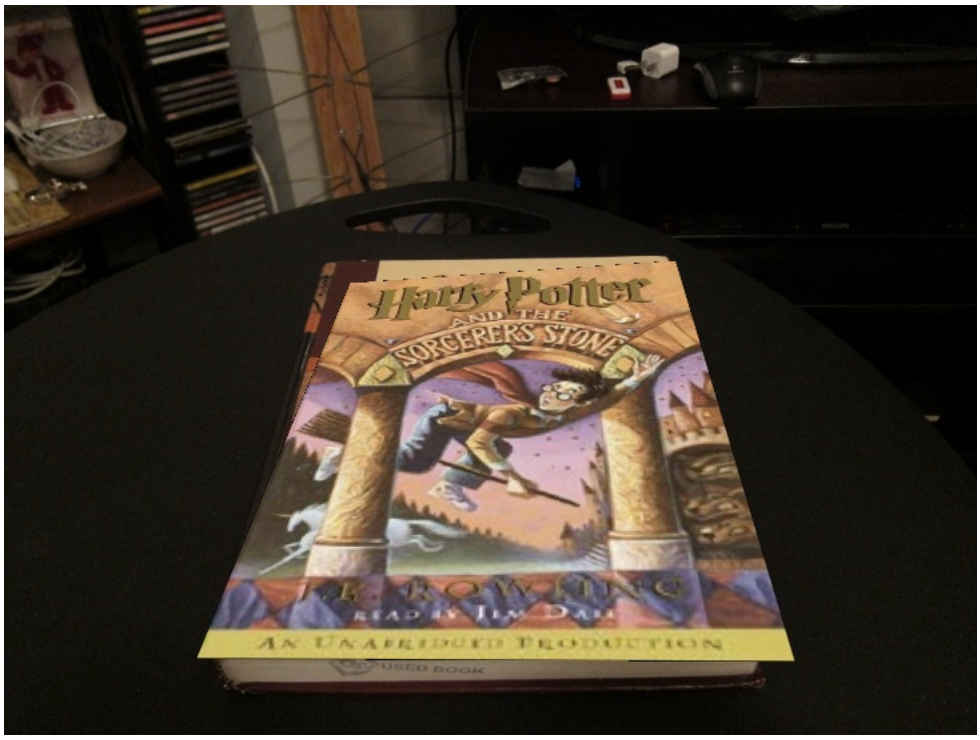
MAX_iter = 100000, tol = 2



MAX_iter = 700, tol = 0.01



MAX_iter = 700, tol = 20



7 Stitching it together: Panoramas

Q7.1

Q7.2

See the code for my implementation.

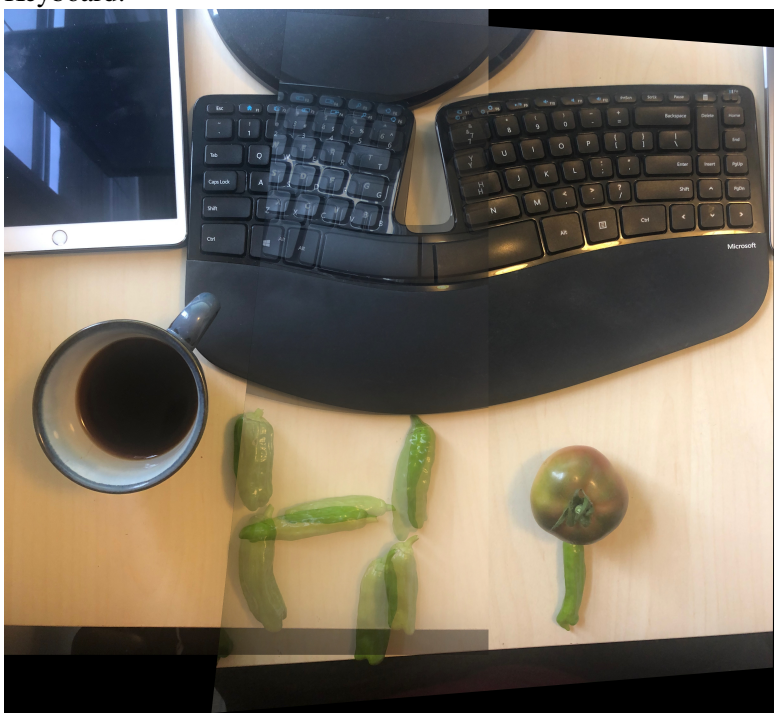
Note:

I included two options for scaling in the code.
You can either set s directly
or you can give a output width and the program will find the correct height.

Q7.3

Pittsburgh City:

Keyboard:

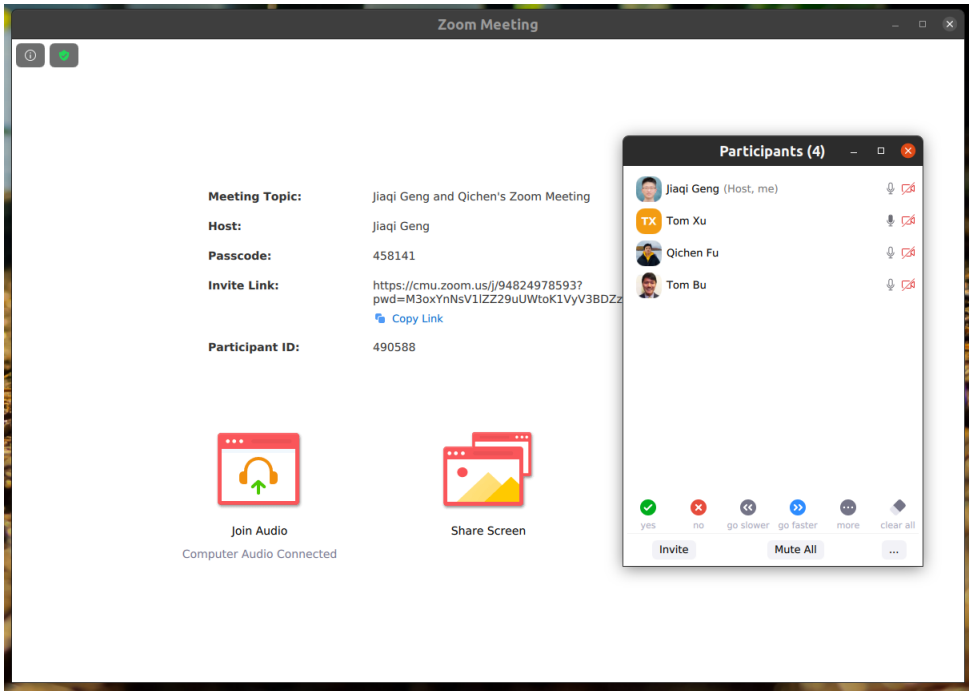


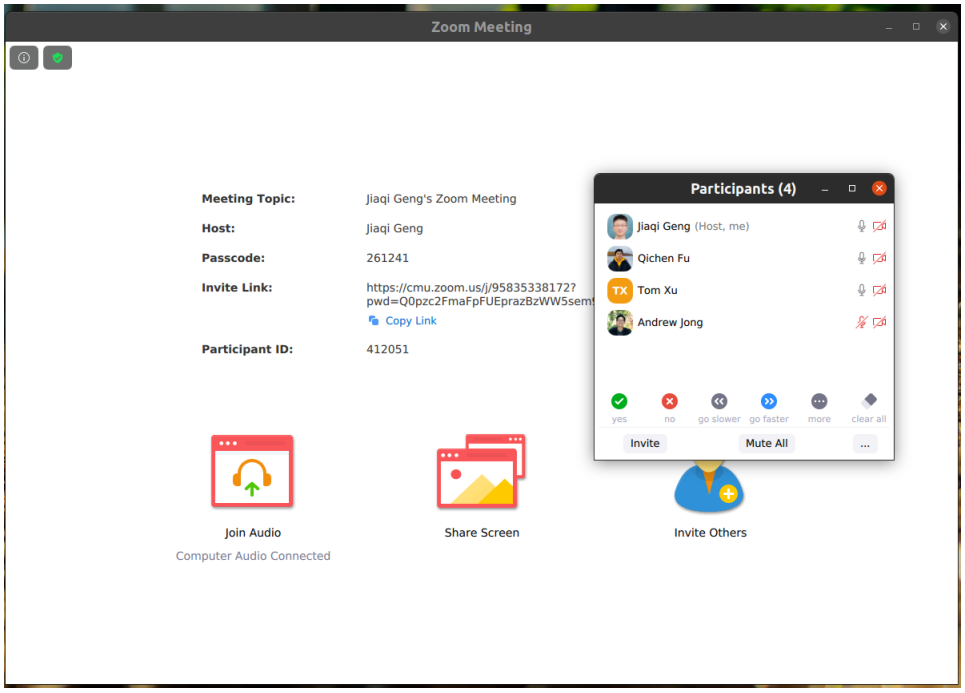
For the keyboard images,
I need to resize both the width and height to $\frac{1}{10}$ of their original values to find decent matches.
I then scale the matching points back to the original dimension to get full-sized panorama.

8 Extra Credit

Q8.1

I (Jiaqi Geng) have hosted two online study sessions with Qichen Fu.





Q8.2

I have attended Ce Liu's VASC seminar on "Advancing the State of the Art of Computer Vision for Billions of Users".

During the seminar, I first learned about glare removal. Interestingly, I found this project was largely built on the same techniques as we used in this homework. The application first requires the users to take multiple pictures of the same scene while moving the camera for a little each time. Then, it will detect feature points and find the matching between them. Using these matches, the application would be able to find the homography. Finally, it could check if regions of the image contain glare by checking if they are brighter than their corresponding areas in other pictures.

I also learned about adding and removing watermarks with computer vision techniques. This technology is quite useful since it could protect the content created by photographers. Ce Liu mentioned that the problem with many current watermarks on the market is that they are generally located at the same locations. The team from Google was able to remove these watermarks easily. They proposed an new method, where the watermarks won't be consistent all the time.

An very interesting topic that I have learned is image extension. Given only a small portion of the image, the researchers from Google were able to use Generative adversarial network (GAN) to synthesize the rest of the image. I was very impressed by their results. They also included some failure cases. For example, the program currently could not correctly predict the legs of the chairs from the upper part of the chairs.

I also learned about a special feature that Google provides to advertisers, called image popout. They used segmentation techniques to locate the area of people or other objects, and then add a frame behind the person or object. This will create a pop out effect, making advertisement more appealing to customers.

Q8.3

I did not do this part.

Q8.4

I did not do this part.