ECE 5460
Image Processing
(Project 3)

Zhenyu Bu

November 12, 2024

1 Capture Images

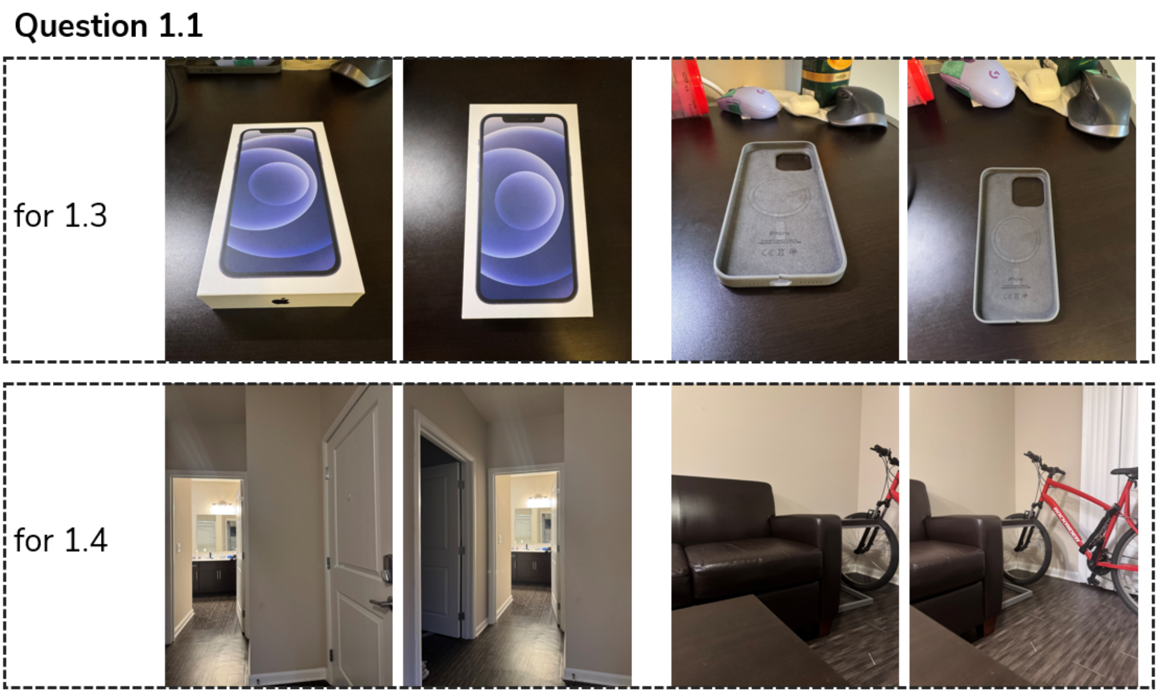

Figure 1: The input image.

The first task is to obtain suitable images in preparation for the subsequent tasks, specifically for Section 1.3. I obtained two pairs of images: one of mobile phone boxes and one of phone cases. For Section 1.4, I also obtained two pairs of images, depicting my living room and bedroom. There is an overlap between each pair of corresponding images. The specific images are shown below.

2 Estimate the Homography Matrix

To estimate the Homography Matrix, I begin by selecting four points from image A and their corresponding points from image B at the matched locations, resulting in four pairs of data points. This forms a system of equations in the form AX = B. To solve the total least squares problem, I use Singular Value Decomposition (SVD). I then select the eigenvector corresponding to the smallest eigenvalue, reshape it, and normalize the result to obtain the Homography Matrix.

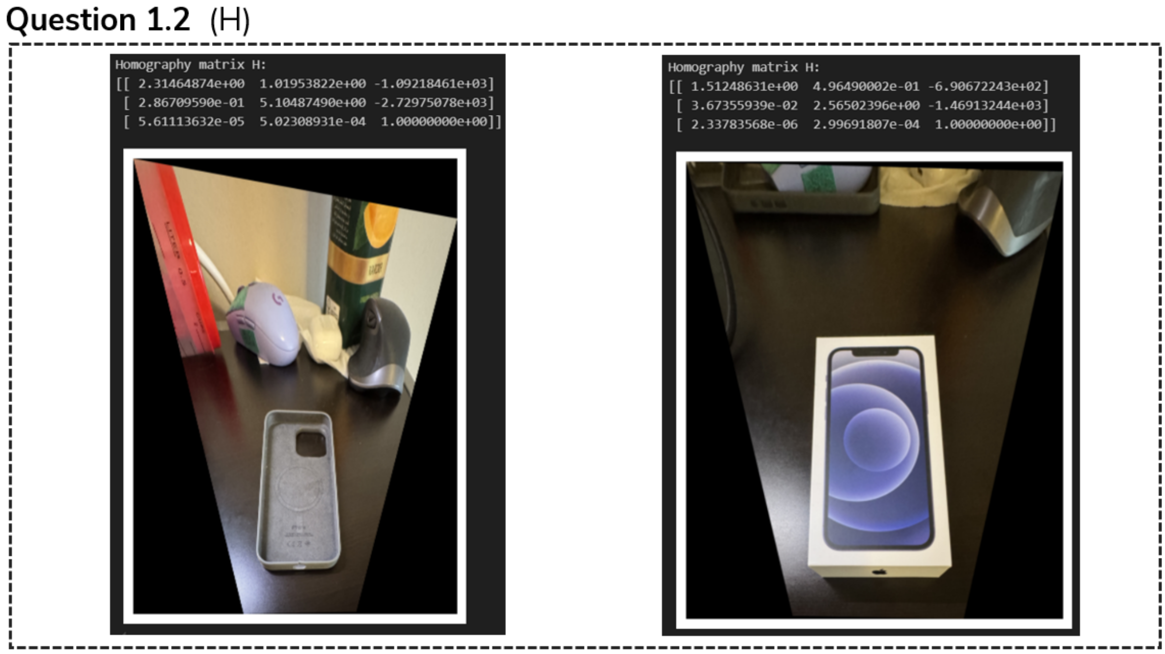

Figure 2: Code for Computing H.

3 Image Warping and Rectification

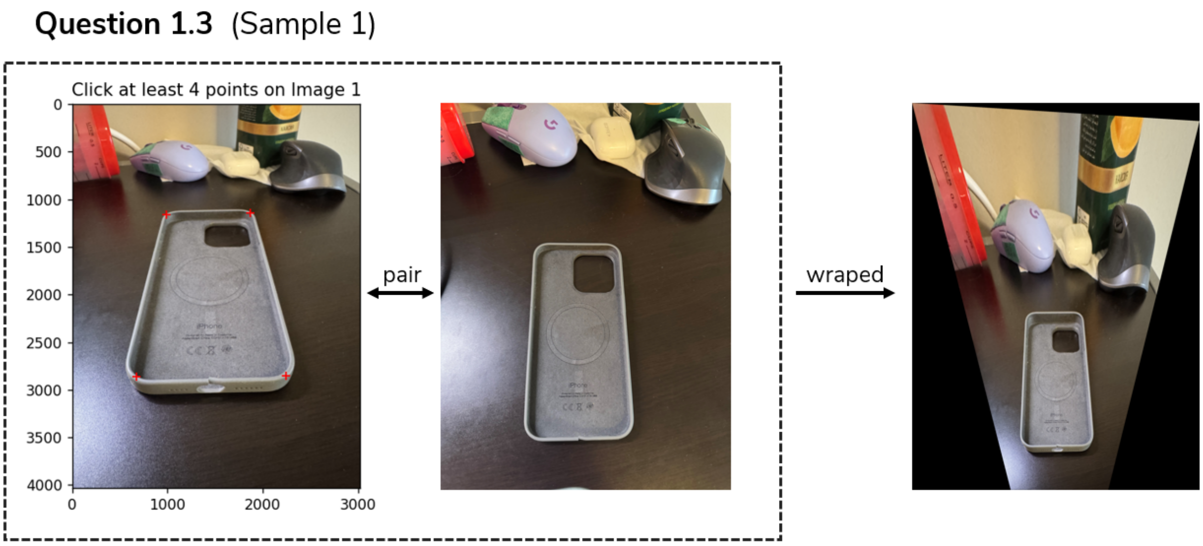

Figure 3: Sample 1 in Question 1.3.

Once I get the H matrix. I can do the warping. I have several parts in the warping process:

Image Dimensions and Corner Points: This specifies the four corner points of the image to establish a reference for the transformation.
Transforming Corners with Homography: Using the provided homography matrix H, the function applies a perspective transformation to these corner points, obtaining their transformed positions in the output space. Also, based on these transformed corners, it calculates the minimum and maximum x and y values to determine the output width and height.
Translation Matrix:A translation matrix is constructed to ensure all transformed coordinates are positive, shifting the entire image if necessary.
Output Grid Coordinates and Homogeneous Coordinates: A mesh grid is created over the dimensions of the output image, covering all pixel locations.
Inverse Homography Mapping: The inverse of the adjusted homography (H_translated) is computed to map each pixel in the output space back to a corresponding location in the input image.
Interpolation: To create a smooth warp, my function uses interpolation on the mapped coordinates for each RGB channel.
Final Output: The function returns warped_image

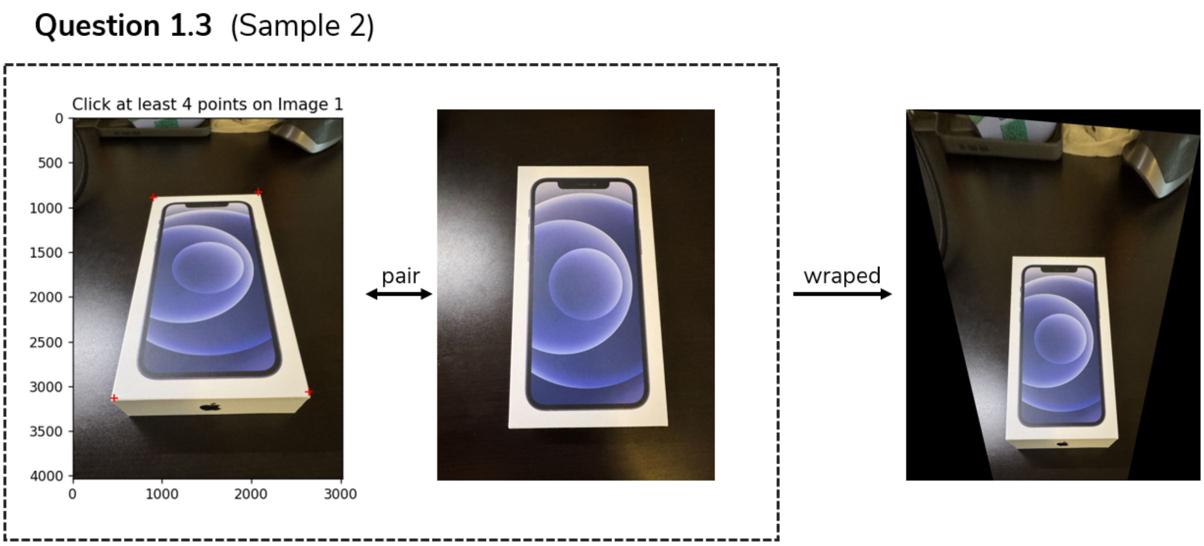

Figure 4: Sample 2 in Question 1.3.

Figure 2 and Figure 3 are two examples. I have achieved the fixed points in my 4question1_3.py (Fig. **??**), from line 95 to line 116.

4 Forming Mosaics

In this section, I first marked several pairs of points in the image, warped one of the images, and placed it onto a larger canvas. Next, I positioned the other image on the canvas at the corresponding location, resulting in both images being aligned on the canvas. I then experimented with comparing the results using **weighted averaging** and **Laplacian blending**. As shown in Figures 7 and 8, the results from the Laplacian method are superior to those from weighted averaging.

Regarding the code, in this section, I have three .py files. The first is mosic.py, which helps me generate wrapped images on new canvas, non-wrapped images on new canvas. I have fixed points on it. The second is blend_weight_averaging.py, which performs weight averaging. The third is blend_Laplacian.py, refers to the Laplacian method.

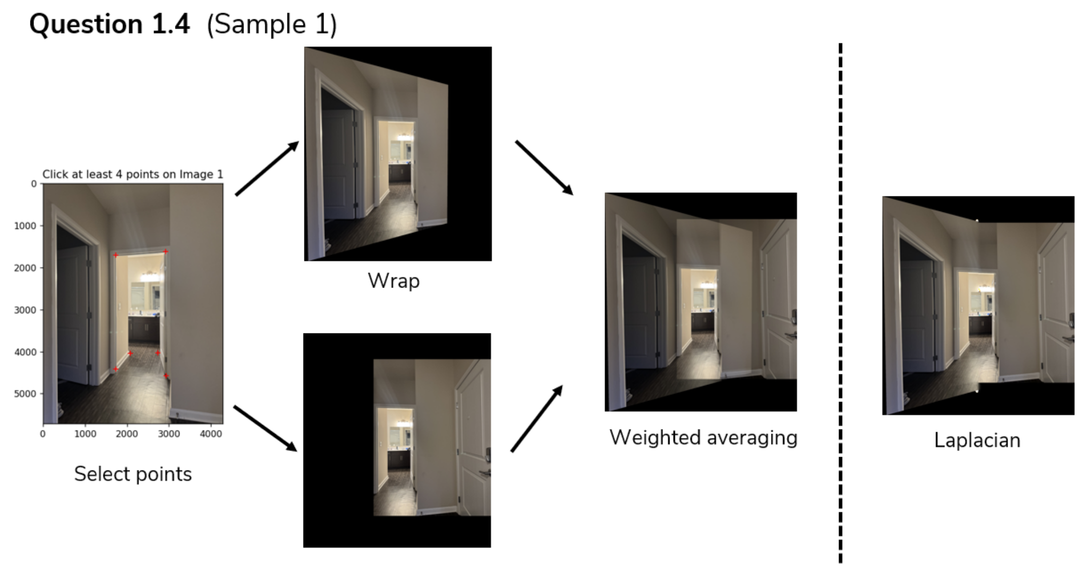

Figure 5: Image blending example one.

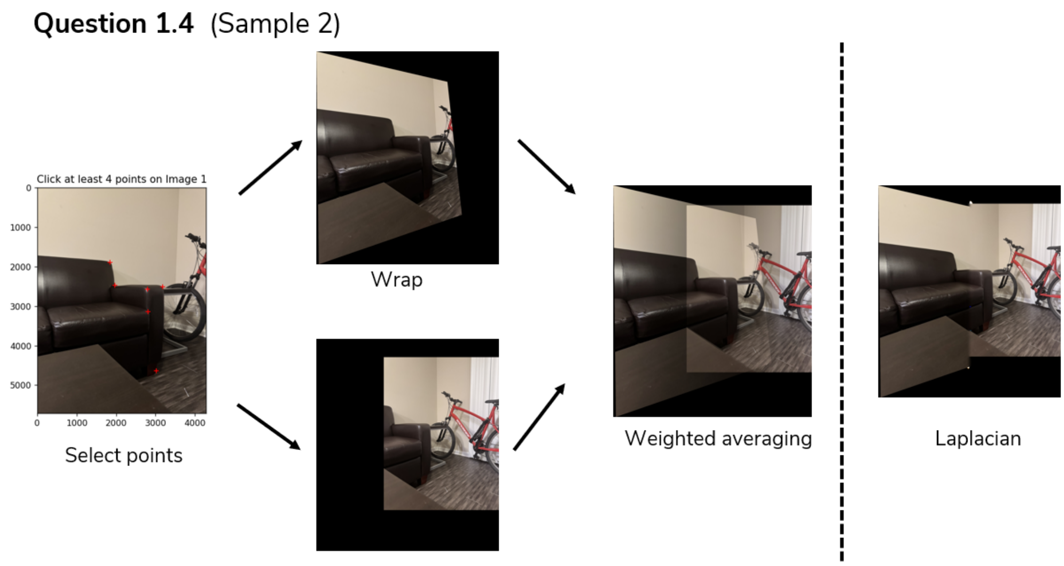

Figure 6: Image blending example two.

5 Bonus Section

For this bonus section, I selected Vincent van Gogh's The Starry Night, painted in 1889, and recognized as one of the most iconic works in Western art. I referenced its original dimensions from Wikipedia (73.7 cm x 92.1 cm) and applied a scale factor of 1.5. Using the method outlined in Section 1.3, I achieved the final result by choosing four points

Question Bonus

Figure 7: Bonus part.

from the original image and warping it to fit a rectangular grid.

In my bonus.py file, I have fixed the four corner points. If you want to select other points, just remove the # from line 88 to line 91 and delete the fixed image1points.

Note: I've combined all the code into a single Jupyter notebook. However, it currently relies on fixed points rather than allowing for user interaction.