Report for HW1: KNN Matting

Name：陳政彥

Student ID：310552006

Part1:Explanation of All parts of my code:

My Explanation will be comprised of two parts:

P1. Some announcements

P2.Simple explanation about each function

P3. Flow of my code

P1. Some announcements:

About References:

I’ve referenced some pieces of github source code to complete my code, you can find the reference link on the top of HW1.py.

About Comments:

I’ve written a lot of comments in my code HW1.py to explain the details about my implementation. Therefore, I won’t explain each line of my code in this report, instead, I will only briefly introduce each function and the flow of my code in the following part. If you are really interested in the details, or have doubts after reading this report, feel free to check all the comments in the HW1.py.

P2.Simple explanation about each function

I split up my code to several functions, now I will explain each one of them separately

1. **find_knn()**:

   This function is used to find the k-nearest-neighbors of a pixel i, to measure the distance, first we should have all pixels' feature vectors, which by definition is [R,G,B,x,y] with color info RGB and spatial coordinate xy. After getting the feature vectors, we simply take advantage of sklearn to find the k-nearest-neighbors of each pixels.
2. **find_sparse_A():**

   This function is used to compute the sparse matrix A. To compute A, we need to first compute the distances between every pixel i and their nearest-neighbors. Then, throw these distances into kernel function k(i,j) to get the values to be put into A. After getting these values, we can construct the sparse A, and then we can manually following the paper's definition to construct the D, Delta, Laplacian, v matrix, and finally get H and c, and that’s what we need for solving Hx = c.

3. **solve_linear_system():**

   We have now H and c, and we can call scipy.sparse to help us solving the sparse linear system. However, we should be aware that if there’s no solution to Hx = c, while H is singular, we should take the least-square solution instead. The solution here is the alphas we are looking for

4. **knn_matting**():

   This function first processes the data to the right format, and then calls the above three function in order to get alphas.

5. **composite():**

   After having alphas, we can composite two images together, while one image serves as foreground F and the other serves for background B. I simply composite using following step, first, resize the B to be same size as F, then, apply C = alpha * F + (1 - alpha) * B on each pixel, then I get the composited images.

6. **grab_file_name()**:

   I use this function for pulling out the filename from a path, for indicating the output image path

7. **get_file_paths()**:

   I use this function to get path of all the images, trimaps and background images.

P3. Flow of my code

This part explain the flow when executing HW1.py When execution, we will first get all the image paths we need for the composition, including background, images and trimaps. Then we will start calculating alphas for each pair of (image, trimap), everytime we find alphas for one pair, we composite this image with all the background image, and store the results. This process will continue until each image is composited.

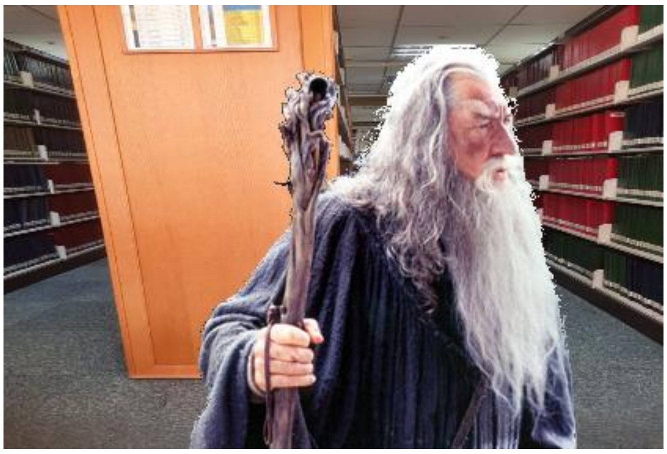
For how to execute the HW1.py and how the results will be stored, please refer to README.txt for details.

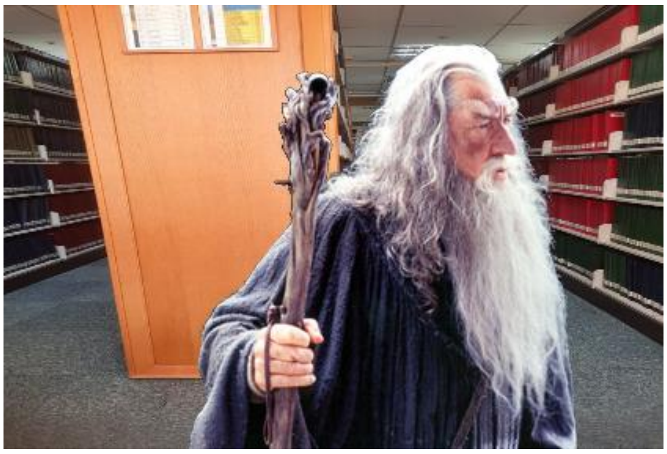
Part2: Explanation on what I have done in experiments and show the results:

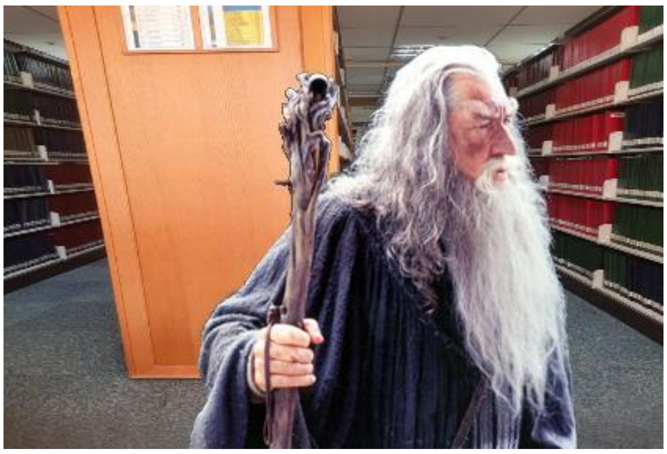
I have done mainly two expriments trying to find a better composition result, they will be separately explained.

exp1. different K number while finding alphas:

In the paper, it’s using k = 10, and I wonder how if k goes up to 15 and down to 3 as the paper has mentioned, thus I perform them on the gandalf.png to see the result.


Result k = 3


Result k = 10

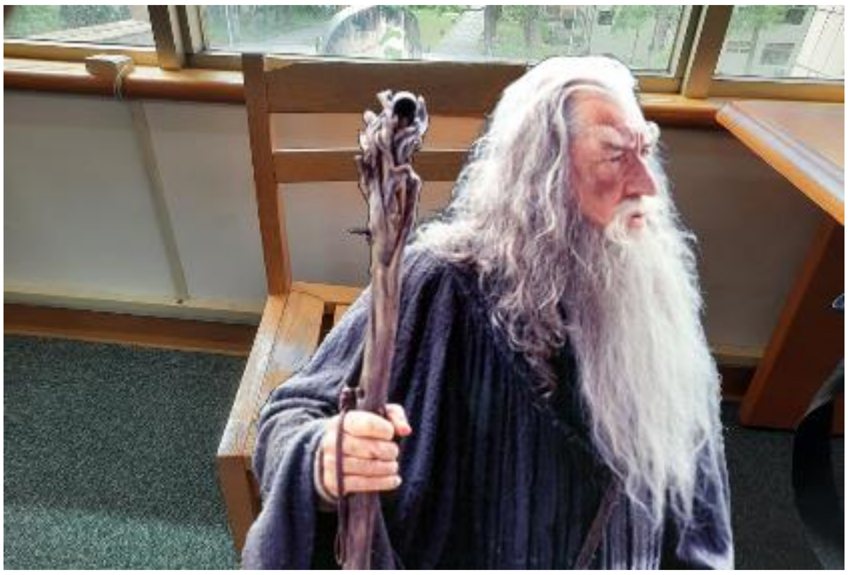

Result k = 15

Apparently, k = 3 has a bad result comparing to k =10 and k = 15. However, it seems not that different between k = 10 and k = 15. Therefore, I think k should not be too small or large, too small k = 3 we got not enough information for solving a good alpha, while too large k = 15 takes a great amount of calculation time. In this case, k = 10 in the middle is an adequate number for coding, and I use k = 10 in my implementation.

exp2. Find the best background for each images:

For the given three images, bear, gandolf, and woman, I tried to find some background image to perform a good composition, however, when I randomly take photos in the campus, I find out not every image would be a good background image,

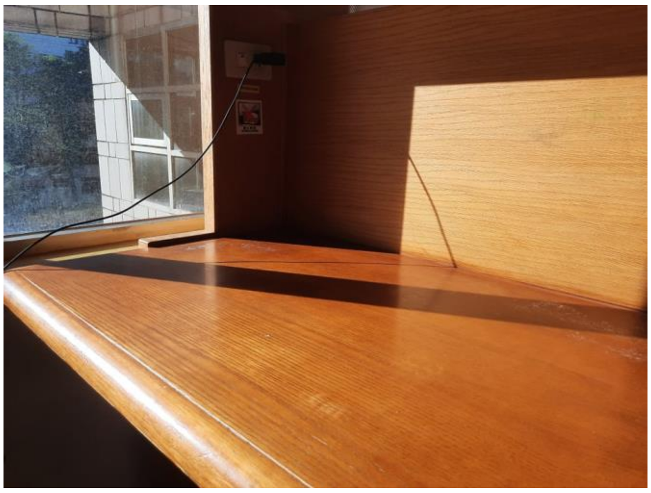
for example, gandalf with a chair:



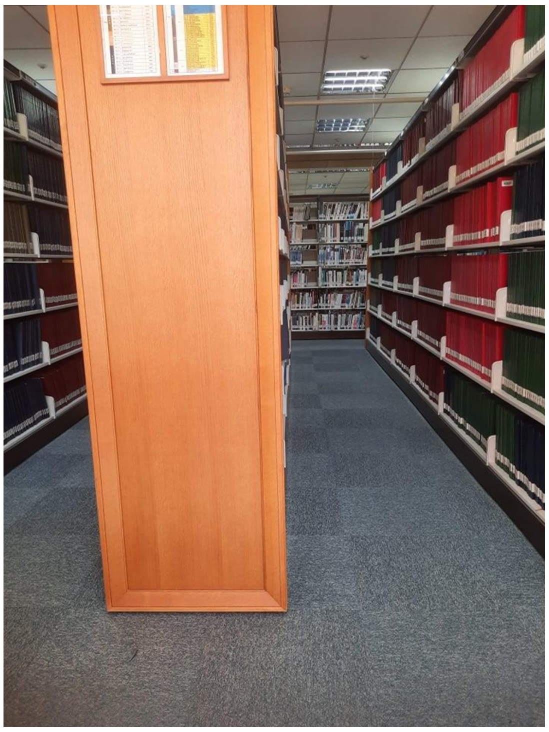
This one looks really weird, gandalf is standing, but the floor is tilted, that because I take the shot of the background from 45-degree above. Also there is a chair behind gandalf and it looks like that they have the same height, even more weird.

To solve the above issue, we need to focus on two points: angles and the objects, for examples, if Gandalf is standing, then we should take a photo with a parallel angle to the floor, and also, there had better not be any weird objects, such as chair, chocolate cake or something else on the position of the composited Gandalf.

And also, light is a issue, since we need to handle the shadow while composition, but we don’t have shadows, therefore, we shall avoid strong and straight sunlight in the background image such as



Instead, we shall take photos in a bright environment, without obvious shadows, such as:



Based on these pricipals, I find the backgrounds and generate my best composite result using the provided three pictures . I will show them in the next part

Part3: Composition Result

Since I don’t see that we need to upload the composite pictures, I think the Composition part of (9% + 6%) is evaluated in this report, and therefore I put the composite pictures generated by my code here for evaluation

On next page are my best three results with different backgrounds.

Bear:

Gandalf:

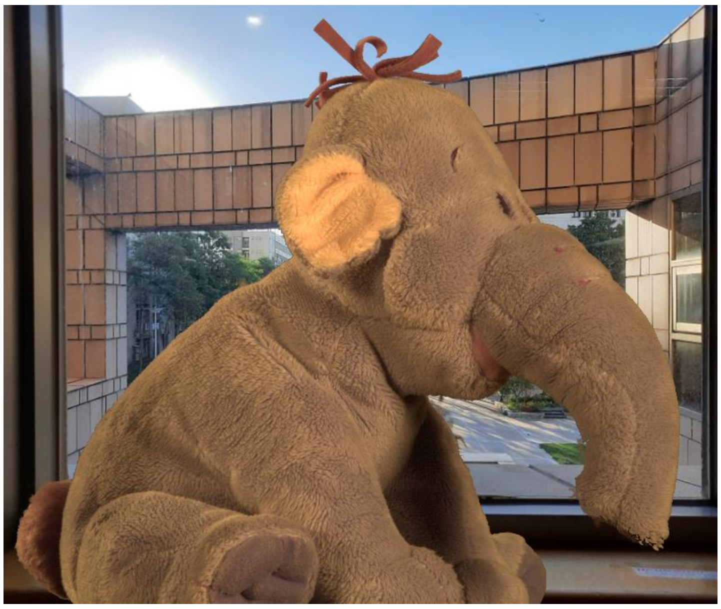
Woman:

Last Part: Description on what I have done for bonus

Basically, I download 2-more photos from the alpha matting low-resolution dataset and make more composite pictures for my bonus part

On next page is the result:

1. elephant



2. Dunk