Report

Part1:

Basic Algorithm:

The overall process of my basic algorithm is: read a greenscreen image of myself and another tourist attraction scene, extract a binary alpha map that separates me from the greenscreen background, and composite me(foreground) with the tourist attraction scene(background). The script name is: assignment1Part1.m

To do the initial manipulation with the images, I used imread to import the images, scaled the tourist attraction scene according to the dimensions of my greenscreen image, and turned both of them to double.

To extract the binary map, I first create 3 alphamap matrices that represent the alpha values of R, G, and B channels. These matrices have the same size as each channel of the image. The initial values of every element in the matrices are 1.
Next, I used this as my threshold:
$origin\_image(i, j, 2) > 80 \;\&\&\; (origin\_image(i, j, 2) > origin\_image(i, j, 1))$

Origin_image(i,j,2) represents green channel values, and origin_image(i,j,2) > rigin_image(i,j,1) compares the green channel values to the red channel values. If this threshold condition is met, it means that the current pixel at position (i,j) belongs to the greenscreen background, so I set the value at the same position in the alpha map to 0. After every pixel is reached, the elements with value of 0 represent greenscreen background pixels, and the elements with value of 1 represent foreground pixels. Last, I combine the three matrices together to form the binary alpha map image.

To composite the foreground and the tourist attraction scene (background) together, I used this formula:
$result = alpha\_map.* image + (1 - alpha\_map) .* background$

This is the standard composition formula.

How to run my code:
The script is assignment1Part1.m
First, you have to put my script and the images that you want to test in the same folder. Then, click on the run button, it will show you the results. To change images, change the photo names at line 6 and line 7 of assignment1Part1.m

References for basic algorithm:
https://www.geeksforgeeks.org/how-to-change-the-background-of-an-image-with-green-screen-background-in-matlab/

Advanced algorithm:

The flow of my advanced algorithm is really similar to the basic one, except that I used a non-binary alpha map and used median filter to smooth the map. The following explaination will focus on how I extracted the non-binary map, and how did I smooth the map.

To extract a non-binary alpha map, I first extracted the foreground with by using this threshold:
$im(w, h, 2) < 0.3 \,||\, (im(w, h, 2) < im(w, h, 1)) \,||\, im(w, h, 2) < im(w, h, 3)$

Every pixel that meets this threshold belongs to the foreground. The same positions in the alpha map matrices will have the value 1. If not, then the pixels belong to the greenscreen background.

To create different alpha values for the greenscreen pixels, I made some adjustments to the formula created by Petro Vlahos,, the innovator of bluescreen and greenscreen. It looks like this:

$alpha(w, h) = threshold - a1 * (im(w, h, 2) - a2 * im(w, h, 3))$, where threshold is a value between 0 and 0.5 of your choice, and a1, a2 equals 1 by default. If the result value is less than 0, scale it to 0. If the result value is bigger than 1, scale to 1. By this way, there can exist many alpha values that are between 0 and 1, which provides a non-binary alpha map. The pattern is more obvious when the threshold is closer to 0.5, and becomes a binary map when the threshold equals 0.

After extracting the true alpha map, I applied a median filter to the mask. This will remove the noise on the map, which makes it smoother. The difference will be shown later when I display the images.

The rest of the steps are similar to the ones in the basic algorithm.

How to run:
The script is a1p1_2.m
You still have to put the images and the script in the same folder. After clicking on run, the command line will ask you to enter a value between 0 and 0.5, I recommand entering 0.5 to see a obvious patterns of the non-binary map, or 0.3 to see a normal map and result image.
To change images, change the image names at line 6 and line 10 of a1p1_2.m

References for advanced algorithm:
https://graphics.stanford.edu/wikis/cs148-07/Assignment6
https://en.wikipedia.org/wiki/Chroma_key

Basic image results:

Easy: (do not have to deal with hair strands)

Parts of the mask. The edge is not smooth, and there are some noise in the white (foreground area).

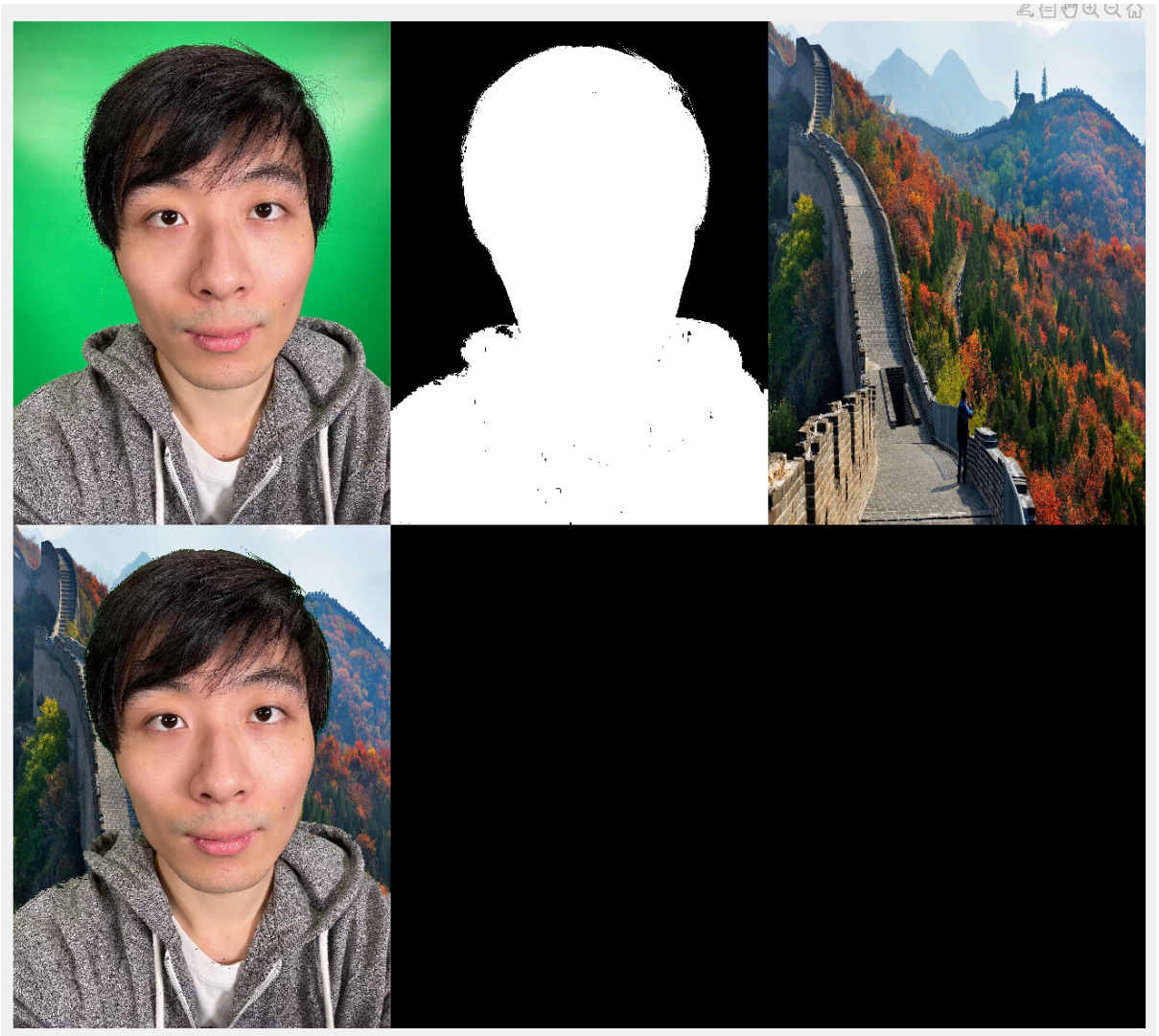
Parts of the result image. The edges are not smooth.

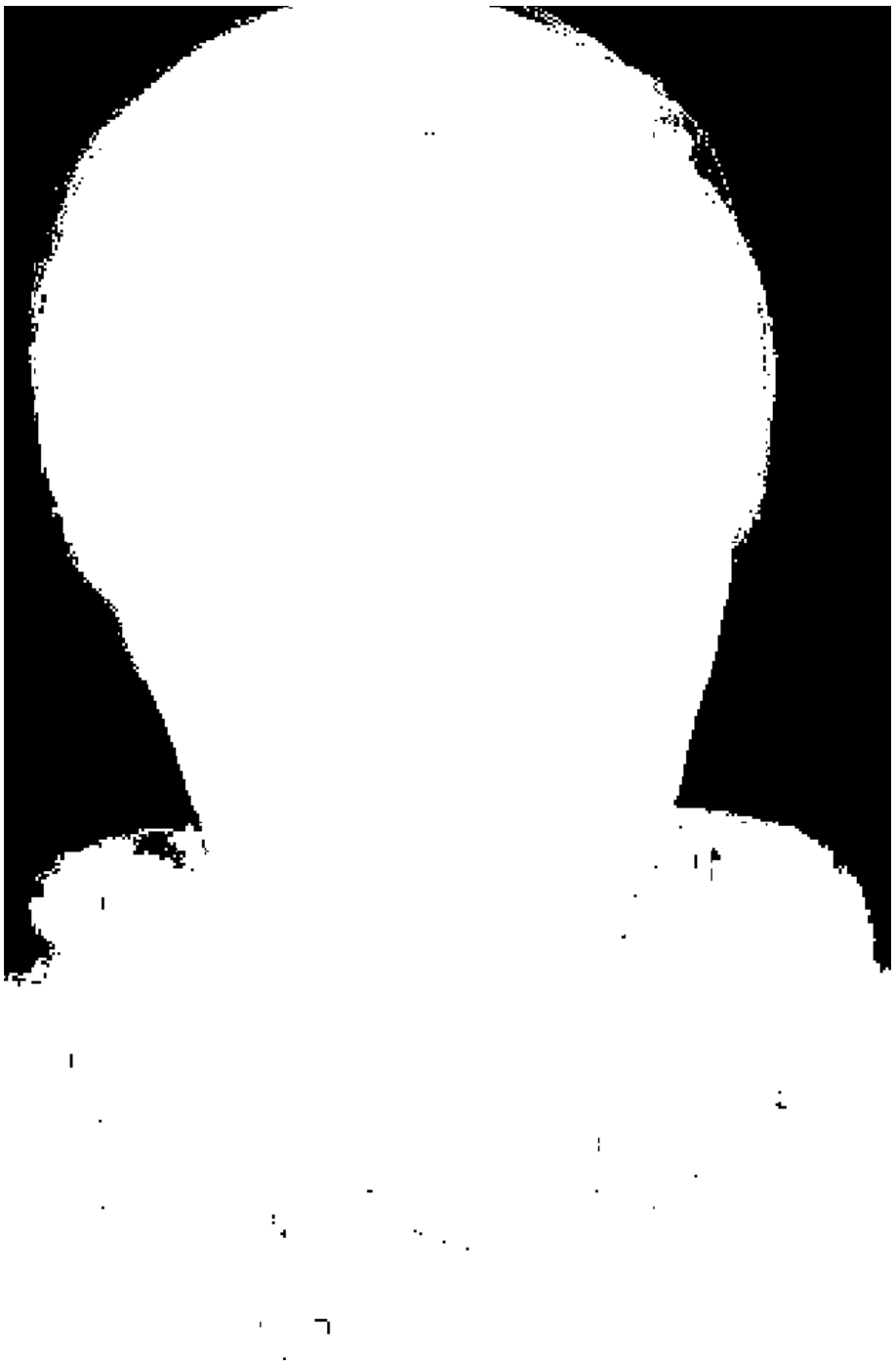
Medius: (has some hair, but the hair is tidy)

Parts of the mask: Obviously not smooth, more dots inside the foreground area.

Parts of the result image: the edges of the hair are green.

Difficult: (has to deal with more messy hair)





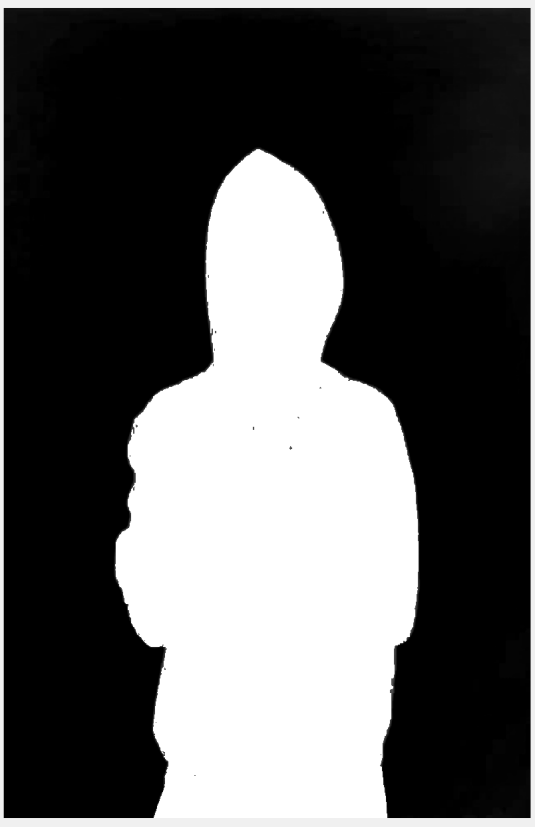
Parts of the map: edges are very rough and there are a lot of dots inside the foreground.

Parts of the result image: hair messed up with green color, hoody messed up with the background.

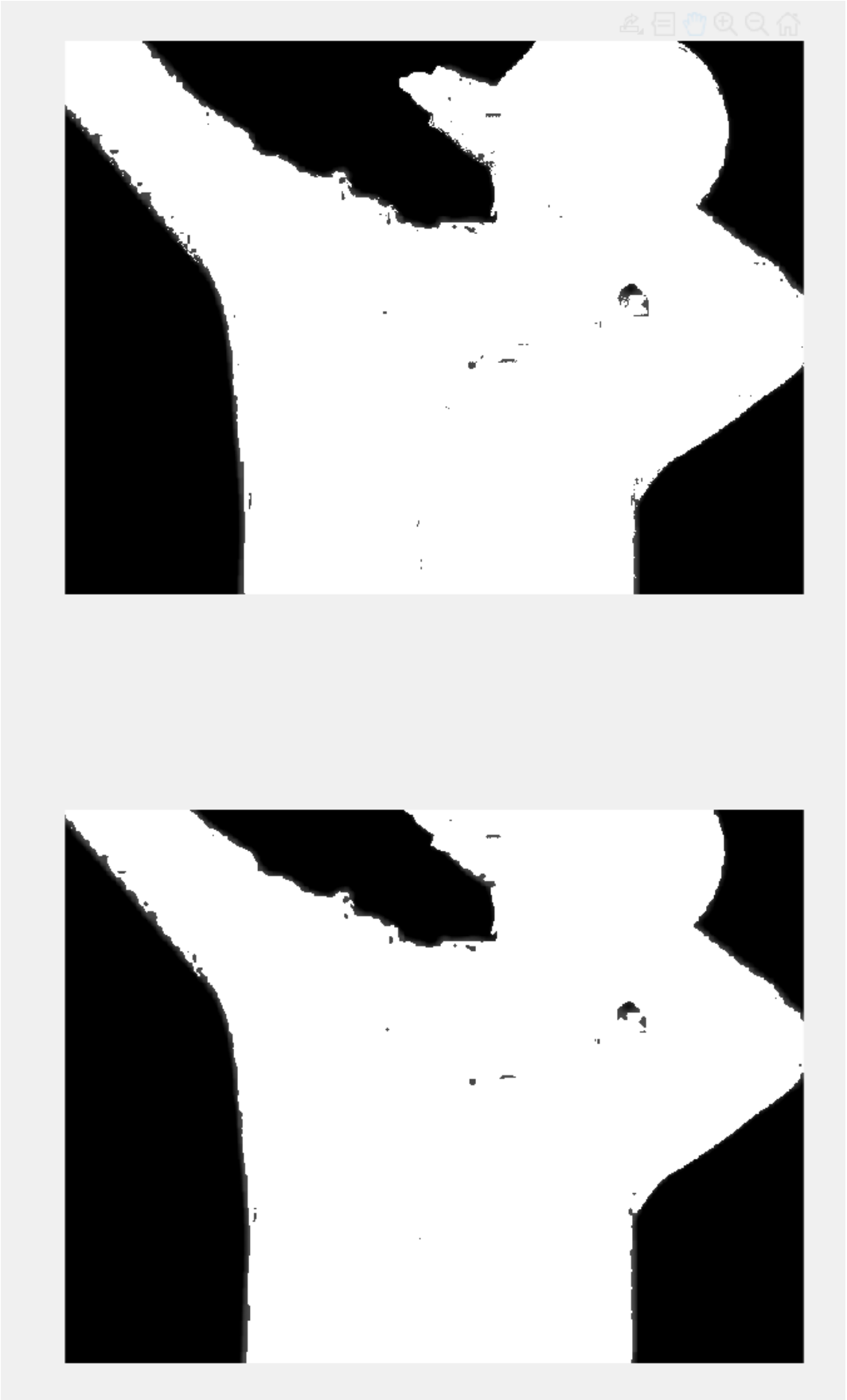
Advanced alogithm:

Mask when threshold = 0.5:

threshold=0.3:



You can see a obvious transparency difference when threshold = 0.5

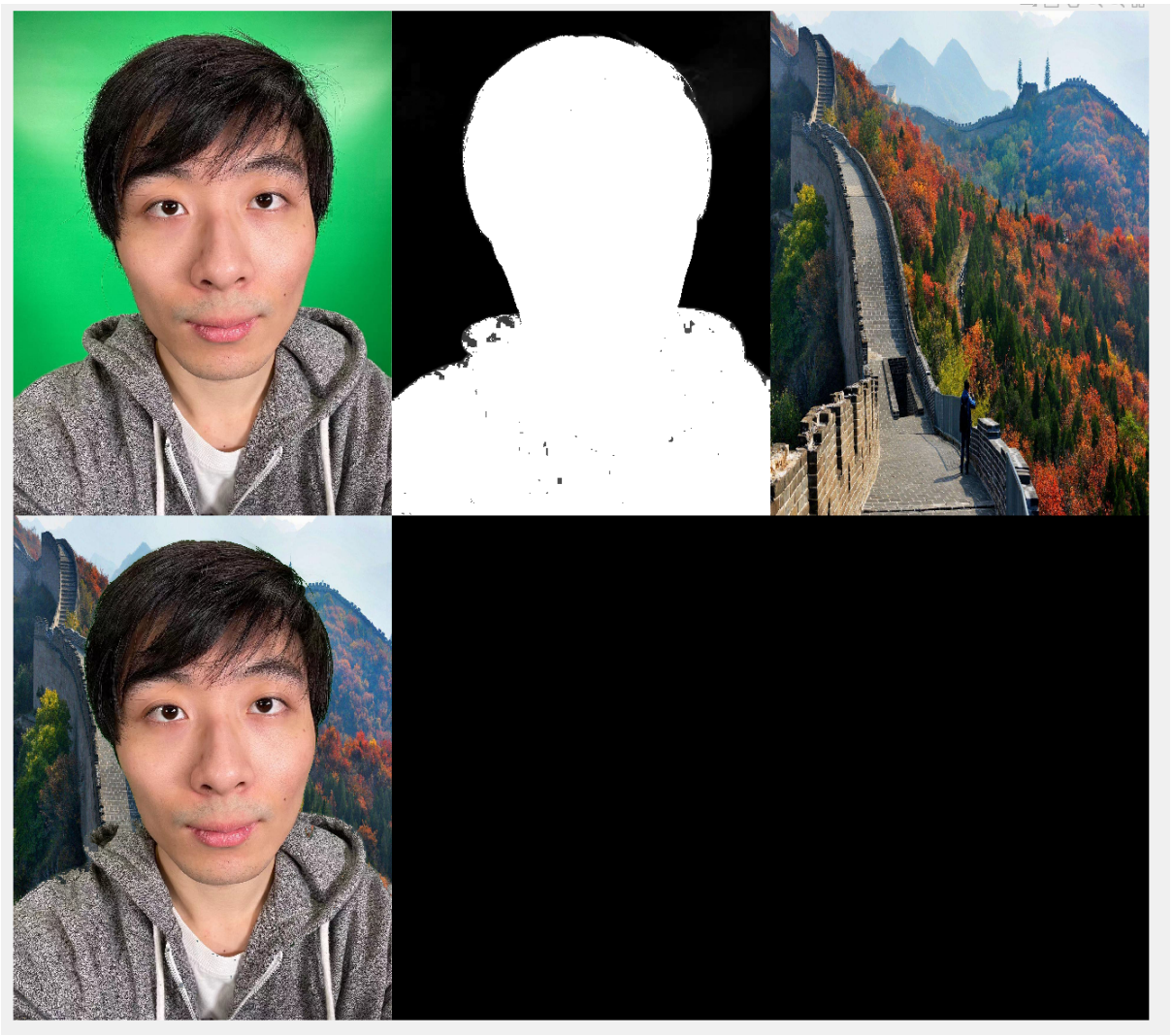


Top is map without median filter, bottom is map with median filter. The filtered mp has less dots in the foreground, and the edges are smoother.

Easy: (no hair)

Medium: (have tidy hair):

Difficult (messy hair)

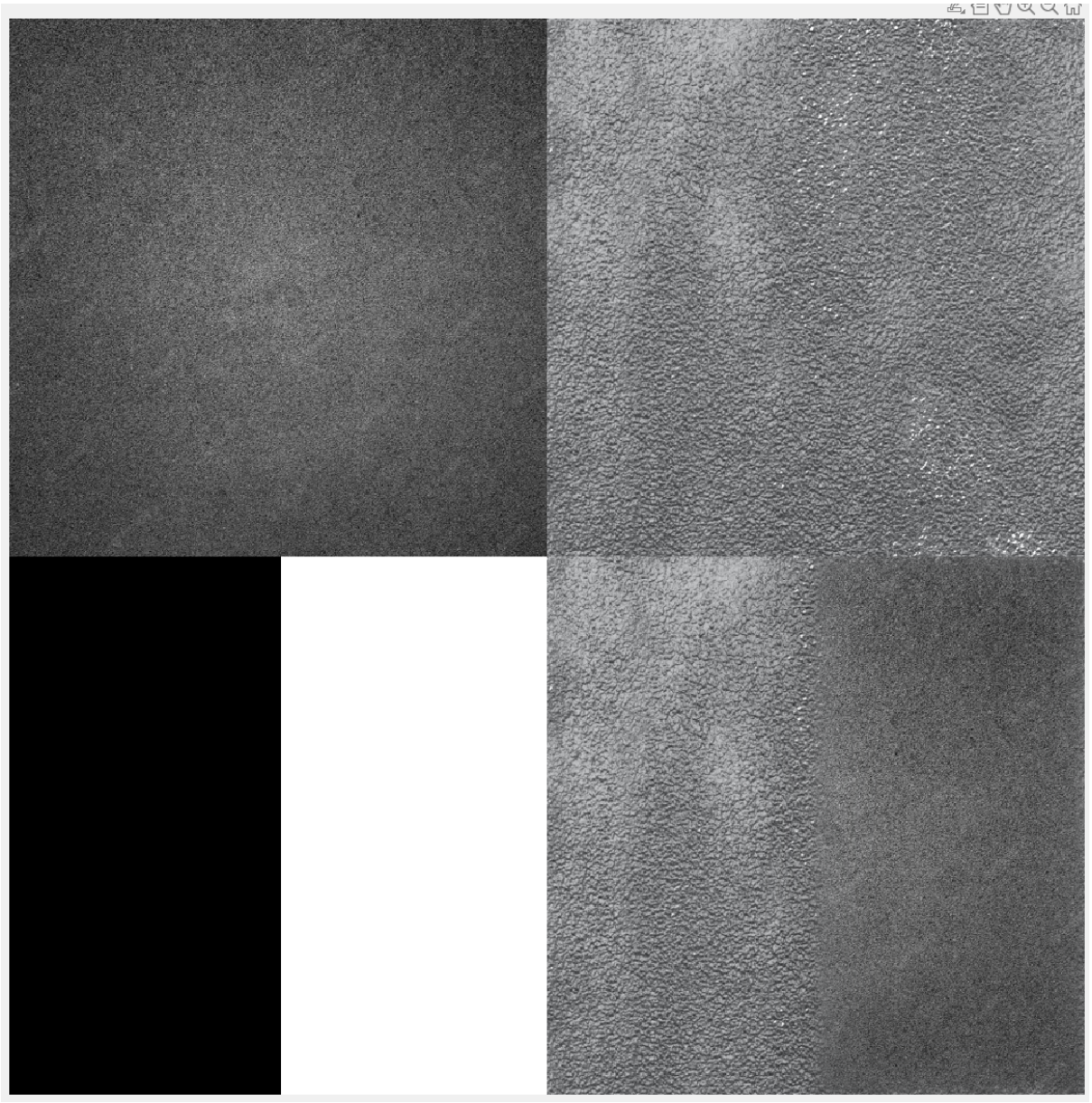


Overall: The advanced algorithm is better than the basic algorithm in both easy and medium cases, because the edges are smoother and there are less dots in the foreground. However, the basic algorithm is better than the advanced algorithm for the difficult set of images. The advanced algorithm has more dots on the edges, and the dots are gray. This is because it has a non-binary map, which makes the ambiguous part of the mask grayish.

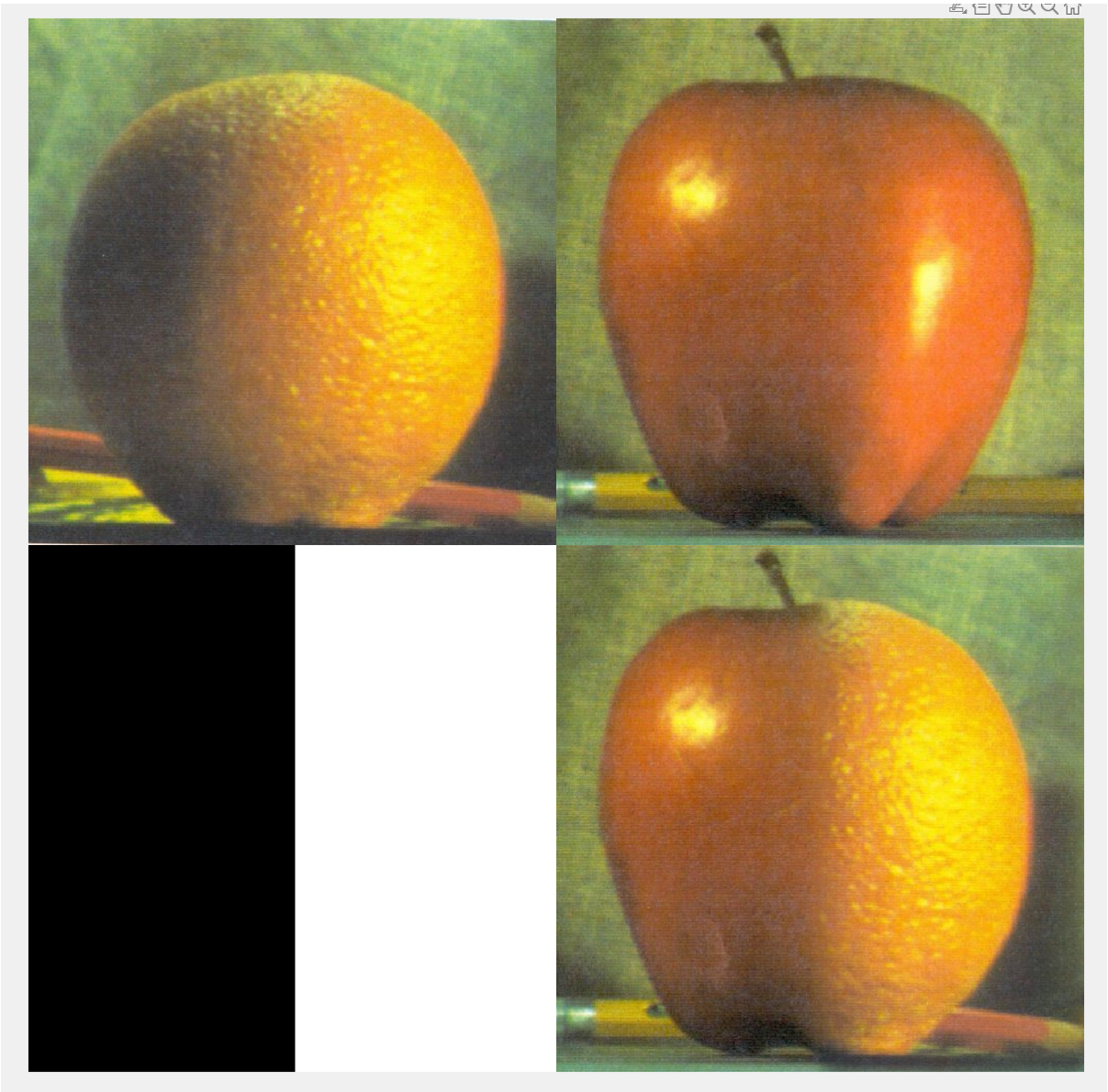
Part 2:

How to run:
The script is a1p2.m
Save all the images you want to test and the scripts in the same folder. Recommend using square images because it looks better. Just click on run and everything is there. To change images, change line 6, line 28, line 50 of a1p2_1.m To show the pyramids, uncomment the lines at line 91.
I arranged the order of images in the opposite direction, and I commented this out at those three lines.

Easy texture: line6 is text2.jpg, line 28 is text1.jpg, line 50 is mask.jpg

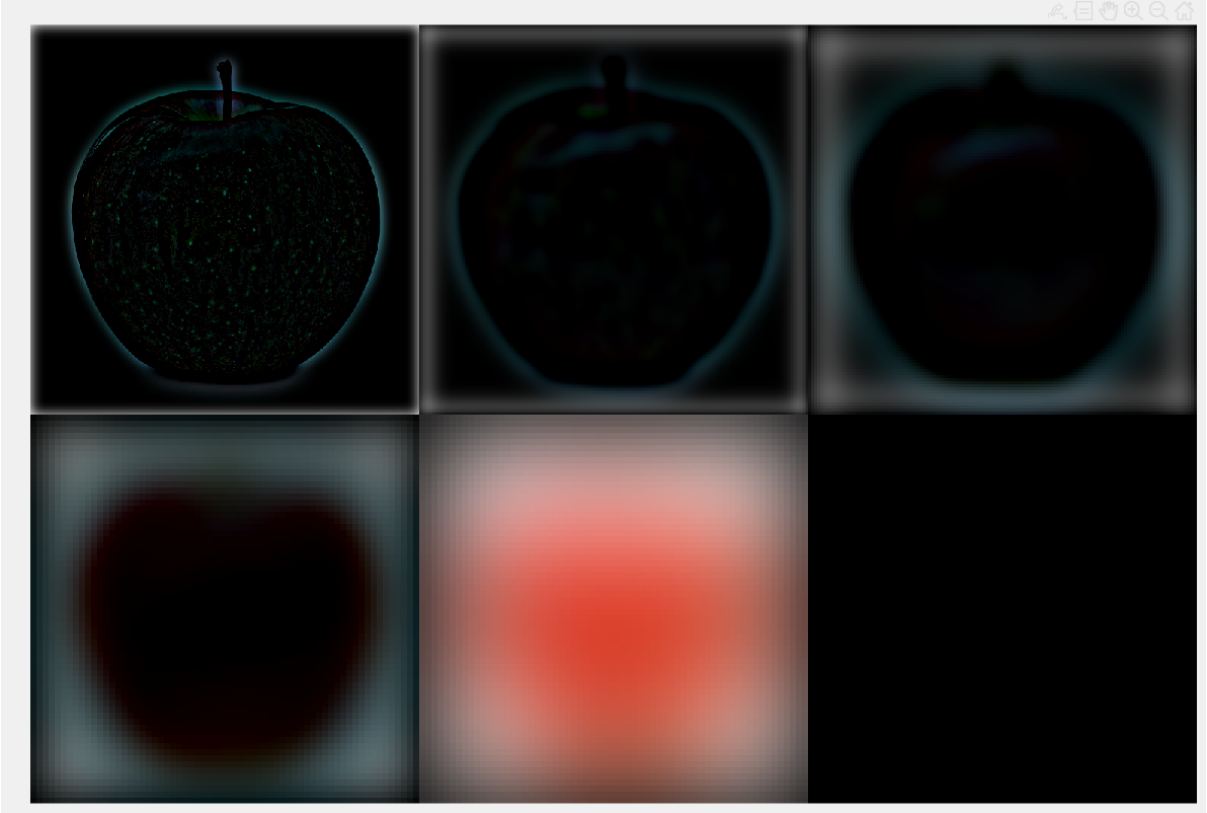


Medium fruit: line6 is orange.jpg, line 28 is apple.jpg, line 50 is mask.jpg

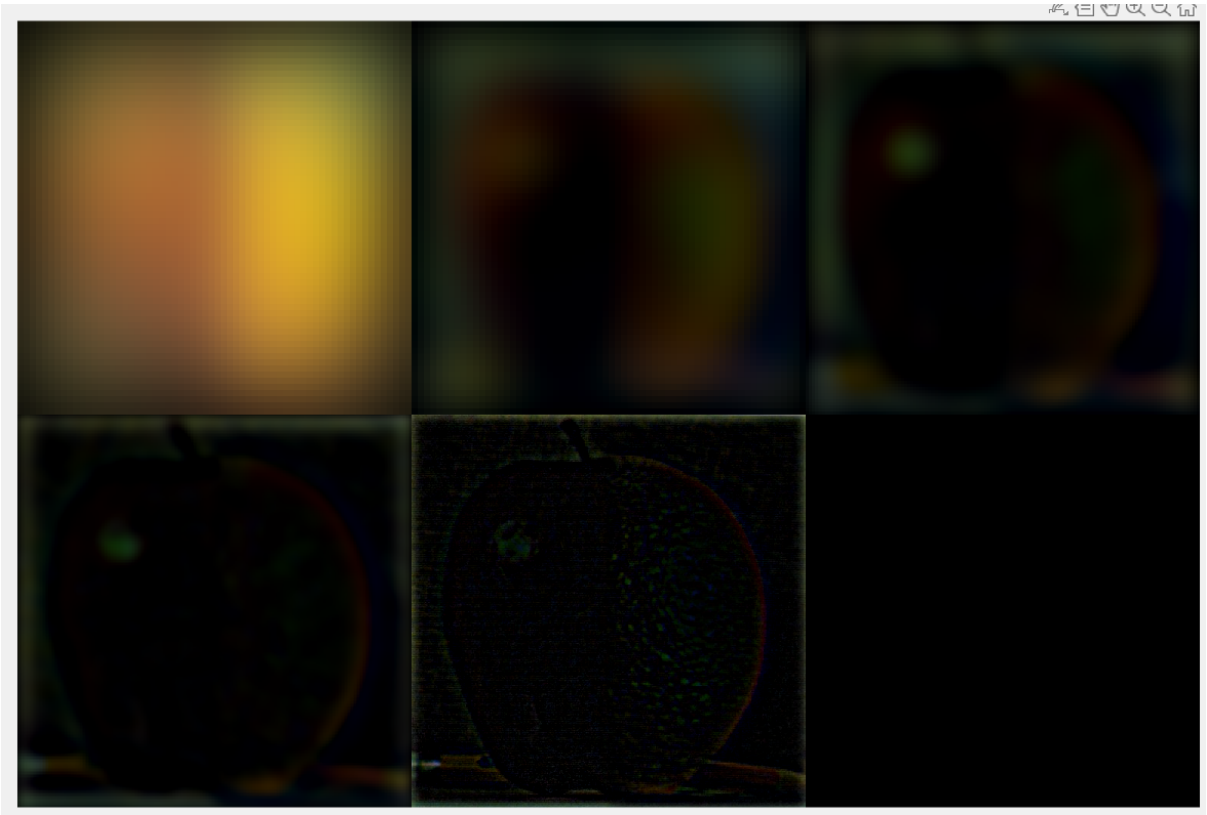


Difficult human face: line6 is face2.jpg, line 28 is face1.jfif, line 50 is mask6.png

Laplacian image for line 6=apple2.jpg



LC pyramid: for line6 is orange.jpg, line 28 is apple.jpg, line 50 is mask.jpg



Overall: The gaussian filter, up sampling and downsampling heavily affects the results. The gaussian filter blurs the images to different degrees, and the subsampling process loses a lot of data (lossy data transmission). and also due to these reasons, reconstructing images from laplacian pyramids will be slightly different from the original images.