ECE558 Project 02

Due 11/05/2021

How to submit your solutions: put source code folder [your_unityid_code], your report (word or pdf) and results images (.png) if had in a folder named [your_unityid_project02] (e.g., twu19_project02), and then compress it as a zip file (e.g., twu19_project02.zip). Submit the zip file through **moodle**.

If you miss the deadline and still have unused late days, please send your zip file to TAs and me.

Important Note: We will **NOT** accept any replacement of submission after deadline, even if you can show the time stamp of the replacement is earlier than the deadline. So, **please double-check if you submit correct files**.

Problem 1 (100 points): Pyramid Blending.

(a) [40 points] Write a function to implement Gaussian and Laplacian pyramid,

$$gPyr,\ lPyr = ComputePyr(input\_image,\ num\_layers),$$

Input arguments (for your reference): $input\_image$ is an input image (grey, or RGB), num_layers is the number of layers of the pyramid to be computed. Depending on the size of $input\_image$, num_layers needs to be checked if valid. If not, use the maximum value allowed in terms of the size of $input\_image$.
Outputs: $gPyr,\ lPyr$ are the Gaussian pyramid and Laplacian pyramid respectively.

1) The smoothing function: you can use built-in functions to generate the Gaussian kernel (think about and research what the kernel size should be, as well as [optional] ablation studies in your experiments), but you need to use your own conv function or FFT function implemented in Project 1. It's a good time to improve your conv and/or FFT implementation as you see fit.
2) The downsampler (for GaussianPyr) and upsampler (for LaplacianPyr) use the simplest nearest neighbor interpolation.

(b) [20 points] Write a simple GUI to create a black/white binary mask image. The GUI can open an image (e.g. the foreground image that you will use in blending); On the image, you can select a region of interest

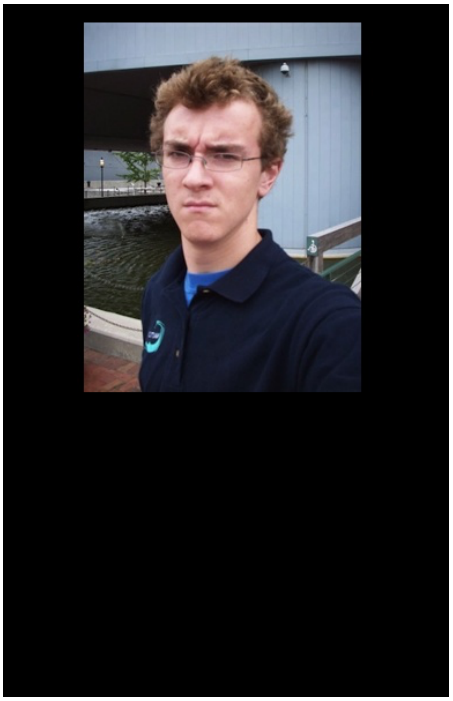
using either a rectangle or an eclipse, [optional] even some free-form region. Based on the opened image and the selected regions, the GUI can generate a black/white mask image of the same size as the opened image, in which the selected region(s) are white and the remaining black.

Note: *For this GUI, you can search online and reuse whatever functions you find useful and can be put together as* **a single self-contained module to realize the aforementioned mask generation functionality. But, you need to finish this on your own.**

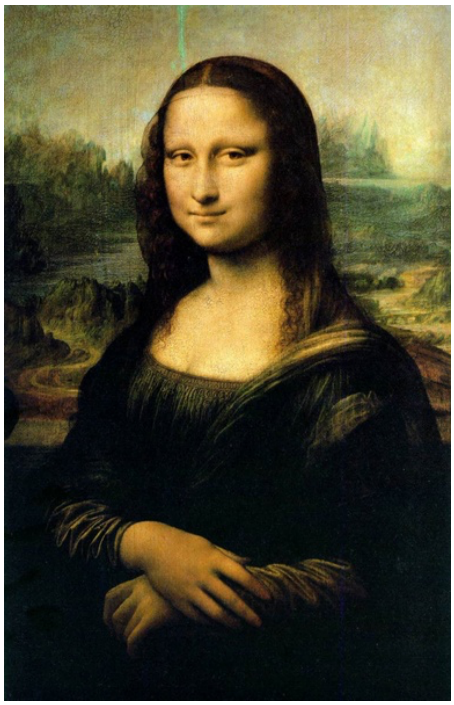
(c) [40 points] On top of the functions in (a) and (b), write a function to implement Laplacian pyramid blending (see lecture note 19).

1) Provide at least 3 pairs of images. You can search images online to find interesting pairs that will generate fun blending results. Be creative and have fun! **Note: it will be of low probabilities that the same image pairs will be used by many of you, even you have a same pair, you will have different region of interest generated by the GUI, so we will expect to see different pairs of images and their blending results in your results.**

Example:



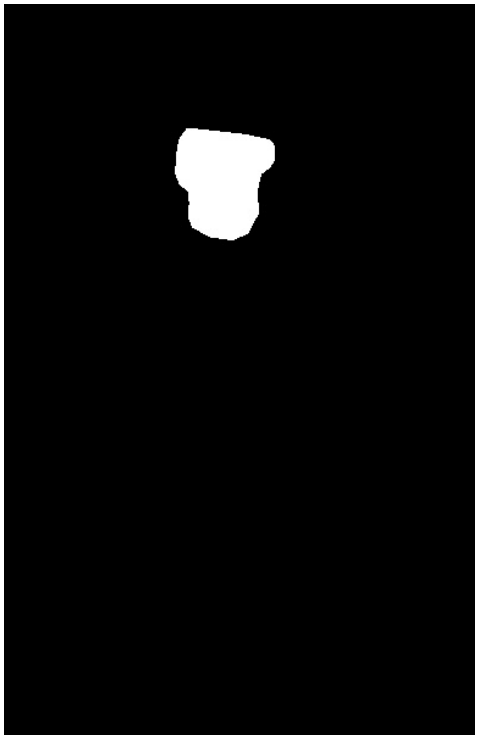
Foreground/source image



Background/target image

The foreground image can be your own photo if you do not mind. Here we want to blending the source face into the target face, so we align these two by placing the “actual” source image in a black background. For scenarios like this, you need to write a function to implement this: first generate a black background image of the same size as the target image, and then place the “actual” small source image at a proper location. The location can be manually aligned and tuned and then use as input argument for the function.

With the foreground image, use the GUI to generate a mask, and the do the blending.



Mask

Blending result (illustration)