ECE558 Project 03-Final

Ashwini Muralidharan
Purushothaman Saravanan
Shahil Manoj Dhotre

Pyramid blending refers to seamlessly blending an area/region from a source(foreground) image into a target (background) image. This project develops functions to implement Gaussian/Laplacian pyramid:

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

where input_image is an *input_image* in grayscale or RGB and *num_layers* is the number of Gaussian/Laplacian pyramid layers to be computed.
In order to implement the Gaussian/Laplacian pyramid, a smoothing function, a down-sampling function, and an up-sampling function were developed. The project is divided into 2 major parts:

- A GUI to create a mask
- Gaussian/Laplacian blending

1. **A GUI to create mask:**
   A simple GUI is used to create a black and white binary mask image. The GUI opens the foreground image and enables the user to select the region of interest (ROI) using either a rectangle, ellipse or a free-form. The GUI can generate a black/white image, called the mask, of the same size as the opened image, in which the selected region(s) are white and the remaining black.
   In special cases where the source region does not match the target region, the two are aligned manually by placing the "actual" source image in a black background. A function, called $align()$ was created to implement this. The function creates a black background image of the same size as the target image, and then "actual" small source image is placed at a proper location. This location is manually aligned and tuned.

   Steps for marking the Region of Interest(ROI).
   1) Press 'e' for ellipse
   2) Press 'r' for rectangle
   3) By default it is free hand
   4) After drawing the RoI, press escape

2. **Gaussian/Laplacian blending:**
   In the Gaussian pyramid, the original image is downsampled by a factor of ½ at every step. To do so, the image is first smoothed using a Gaussian kernel, and then downsampled. For eg:

Figure 1: Gaussian Pyramid Sample

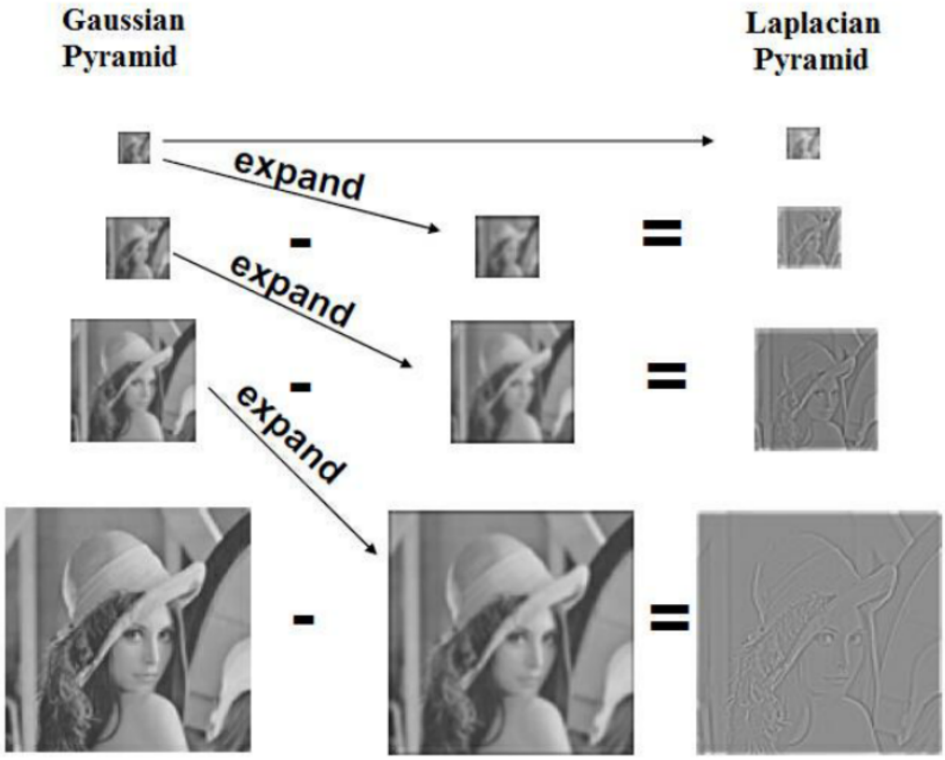

Figure 2: Gaussian/Laplacian pyramid

A Gaussian filter of dimension 5×5 is taken as the kernel for this task. We use a 5x5 Gaussian kernel so that we won't encounter the aliasing caused due to subsampling. In a Gaussian pyramid the subsampling is done at every level so the aliasing effect is more. Hence smoothing using a Gaussian filter helps to reduce aliasing by effectively reducing the higher image frequency. This smoothing is done by convolving the Gaussian kernel with the image. Since the Gaussian kernel is separable, we tried to convolve using this kernel having time complexity MN(P+Q). Where M, N are image size and P, Q are kernel size. (**Note:** Also, the optimization for convolution was tried by taking the FFT of the image and the kernel. Convolution using FFT (for a grayscale image) produced a 10% reduction in running time.Unfortunately, we were not able to complete it on time for RGB image with padding functionality.)

The number of layers is computed such that the pyramid sequence of images can be downsampled by a factor of ½ until the dimension of image is so low that it cannot be filtered using a 5×5 kernel. Hence, the actual possible layers computed can be less than the desired number of layers entered, and this depends on the images' sizes and the kernel's size respectively.

The Laplacian pyramid builds from the Gaussian pyramid. Compute the difference between upsampled Gaussian pyramid level $k + 1$ and Gaussian pyramid level k. This gives the final blended image.

The Upsampler and the Downsampler use the nearest neighbor interpolation method. Nearest neighbor interpolation is a simplest approach to interpolation, where instead of calculating an average value by some weighting criteria or generating an intermediate value based on complicated rules. This method simply determines the "nearest" neighboring pixel, and assumes the intensity value of it.

3. **The Code:**

Two images - foreground and background images are opened and are checked if manual alignment is needed. The GUI for selecting the region of interest is opened and the area is selected. The corresponding mask is generated. The $computePyr()$ function is run on the foreground and background images to generate the Gaussian and Laplacian pyramids respectively. The Gaussian pyramid of the mask image is also generated. Finally, the Laplacian of the foreground and background are combined using the Gaussian pyramid of the mask image, given by the formula:

$$L = Gaussian\ Pyramid\ of\ Mask\ *\ Laplacian\ Pyramid\ of\ Foreground\ +\ (1\ -\ Gaussian\ Pyramid\ of\ Mask)\ *\ Laplacian\ of\ Background$$

The blended image is reconstructed by adding the combined Laplacian to the original Gaussian-resized images multiplied by their respective masks. This is performed repeatedly by

upsampling the result and adding the result to the combined Laplacian until the fully blended image is at original scale. The images are displayed and saved.

Please find 3 python files:

1) run.py: the main file, run this file to get the blended image
2) conv.py: find our convolution code and padding code in this folder
3) pyramid _functions.py: function for computing Gaussian and Laplacian pyramid

4. Steps to execute the program:

- Set the path for foreground and background images and execute the program.
- Press ‘e’ to enable Ellipse shape for marking ROI (or) Press ‘r’ to enable Rectangle shape for marking ROI (or) Start marking lines in free-style.
- Press the ‘Esc’ button after marking your ROI.
- The background image and mask will pop up, press escape once you are okay with mask
- Blended image will be ready in a few minutes.

5. Input Images and Output:

The following 3 pairs of images are considered in this project, and their respective blended results are also presented.

Pair 1:

The two images shown below were the first pair. The Mona Lisa image is taken as background and the face of the man is the foreground.

Figure 3: (a) Background Image (b) Foreground Image

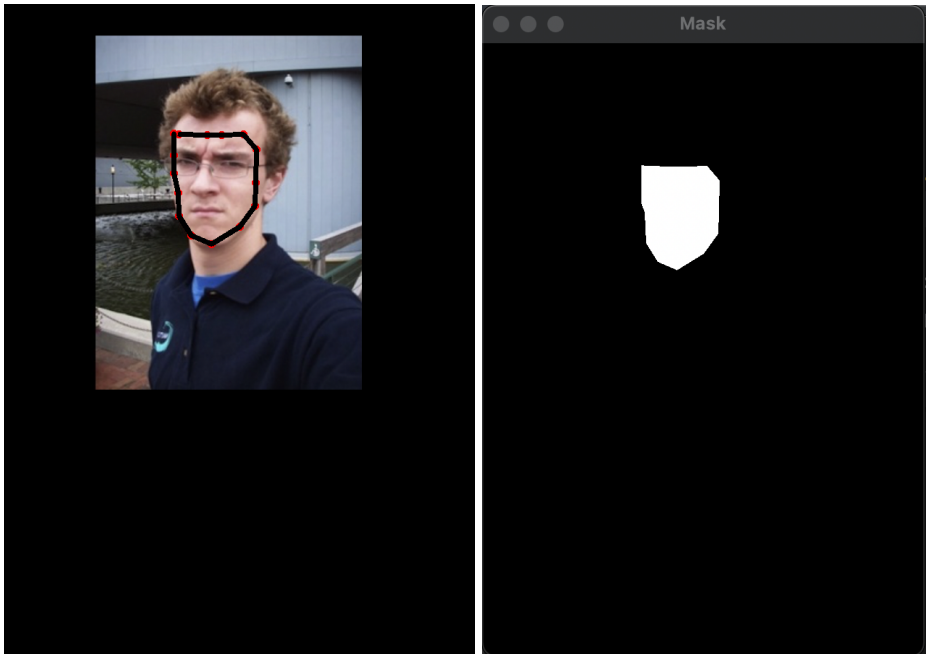

Figure 4: (a) Selected free-style ROI (b) Generated Mask

x

Figure 5: Blended Image

Pair 2:

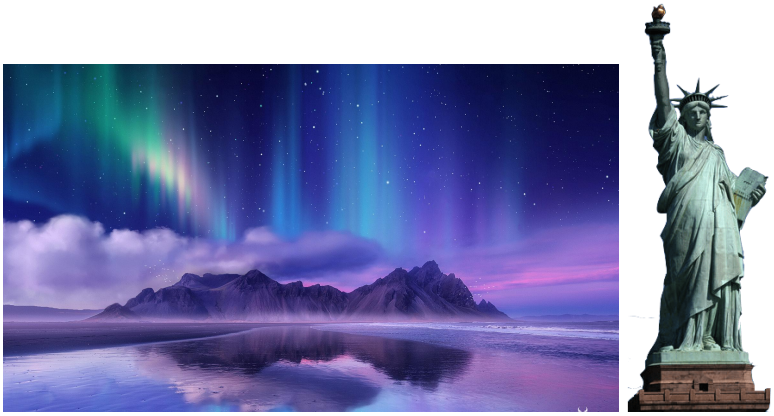

Figure 6: (a) Background Image (b) Foreground Image

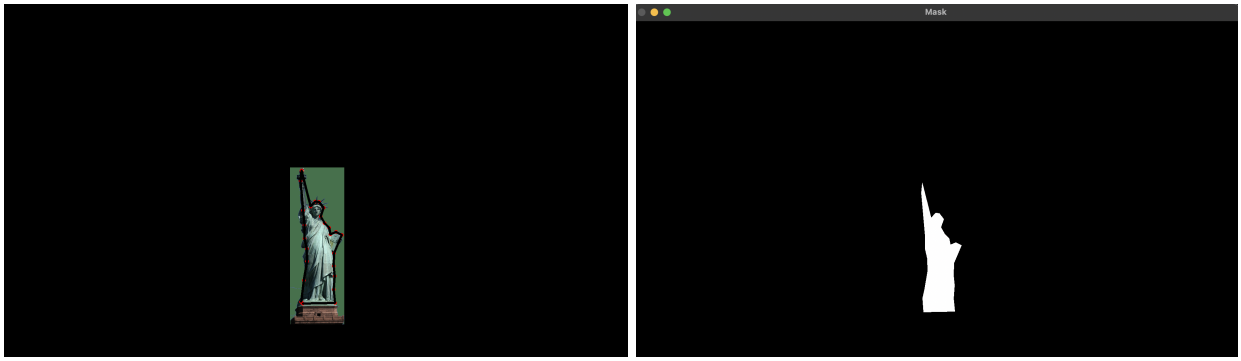

Figure 7: (a) Selected free-style ROI (b) Generated Mask

Figure 8: Blended Image

Pair 3:

Figure 9: (a) Background Image (b) Foreground Image

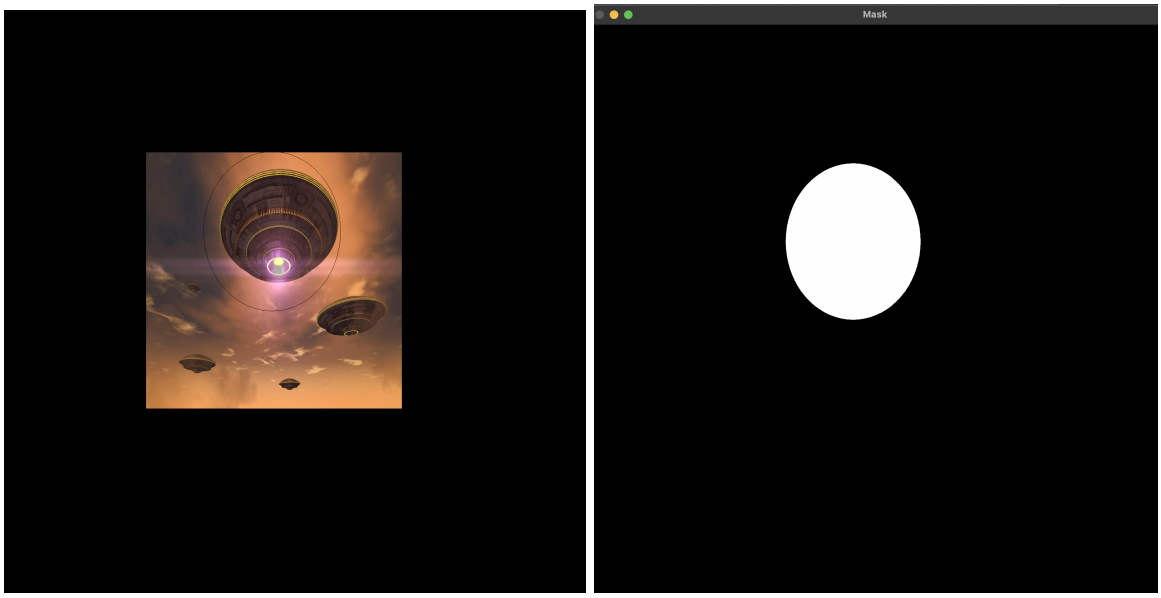

Figure 10: (a) Selected elliptical ROI (b) Generated Mask

Figure 11: Blended Image

Pair 4:

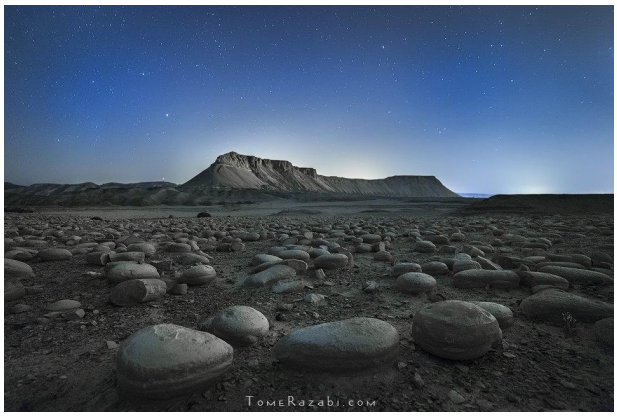

Figure 12: (a) Background Image (b) Foreground Image

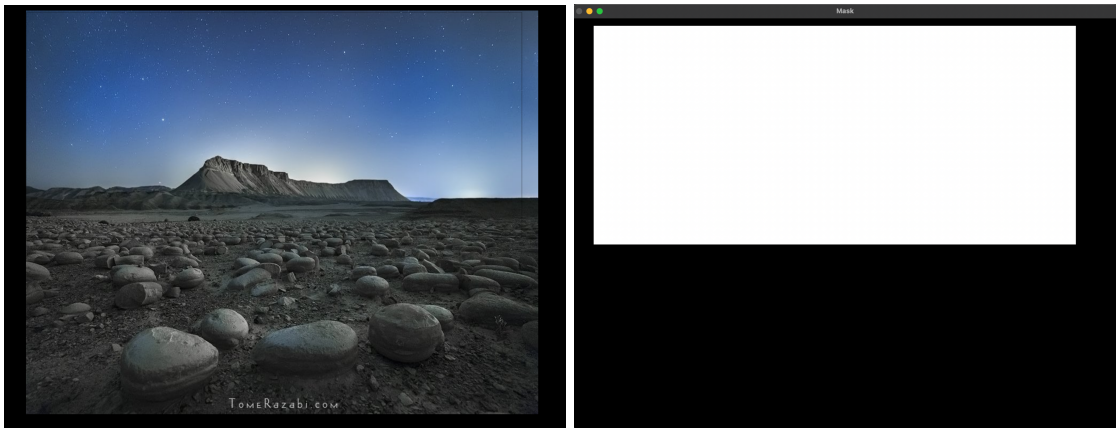

Figure 13: (a) Selected rectangle ROI (b) Generated Mask

Figure 14: Blended Image

Contribution:

Ashwini Muralidharan:

- Creating rectangle and elliptical mask
- Aligning foreground image with background image
- Blending of Laplacian/Gaussian and image reconstruction
- Testing on pairs of images

Shahil Manoj Dhotre

- Creating free-form mask
- Convolution function, Optimizing convolution function using FFT
- Computing Gaussian/Laplacian pyramid

Purushothaman Saravanan

- Convolution function, Optimizing Convolution function using FFT
- Upsampler and Downsampler functions
- Computing Gaussian pyramid of mask