ECE-558 Project 2 Report

Derik Fernando Muñoz Solis

Introduction

This project explored using Gaussian and Laplacian pyramids along with binary masks to blend two images together. In the final product the user can specify the images to use in the terminal and then select the region of interest (ROI) from the foreground image as a free form polygon or even multiple polygons and then this will create a mask that will be used for the blending. The final output will be saved to the disk along with the mask that was used to create them

1 Pyramid Blending

1.a Gaussian and Laplacian Pyramids

In order to do the image blending the first task was to create a function $ComputePyr(input\_image, num\_layers)$ that would take in an image and the number of layers specified by the user and return the Gaussian and Laplacian pyramids as two python list. The basic structure of this function is as follows:

1. Compute the Gaussian Pyramid:

   - The bottom layer is the same as the source image

   - The next layers are made by convolving the previous layer with a 5×5 Gaussian kernel and then resizing the image by 50% using nearest neighbor interpolation

2. Compute the Laplacian pyramid using the Gaussian pyramid:

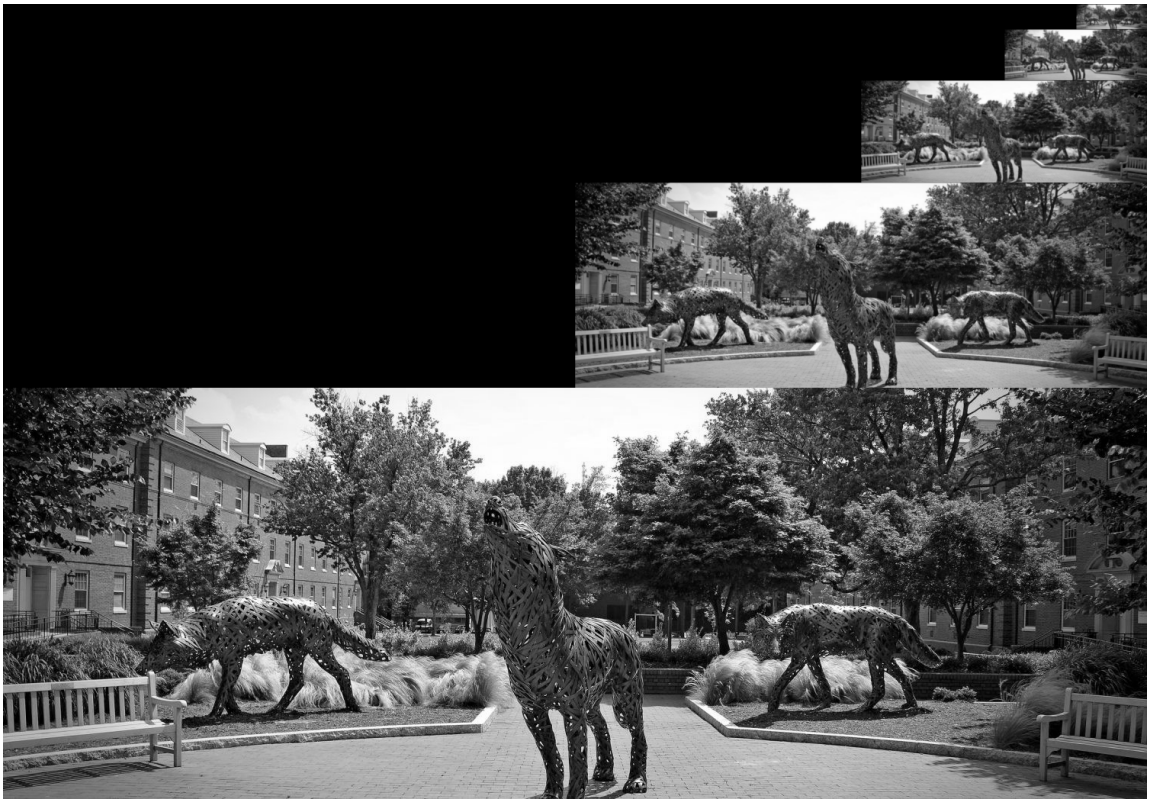
   -The top layer is the same as the last layer for the Gaussian pyramid

   -The next layers are then calculated by resizing the previous layer by making it 50% larger and then convolving with the Gaussian kernel this expanded Gaussian layer is then subtracted from the Gaussian layer in the Gaussian pyramid with the same size to produce the Laplacian of the image at that layer

3. Once all the layers for the Gaussian and Laplacian pyramids have been created they are then returned by the function so that they can be used for blending

The Gaussian kernel that was used was created by using the scipy python module, The convolution function was the one created for the previous project and this convolution was done using reflect-across-edge padding. The number of layers for the pyramids can be specified when the function is called and this value is checked to make sure that the number is valid and if it is not valid then the maximum number of layers that can be computed for the image is used instead.

The results from the $ComputePyr$ function can be seen in Figure 1 below



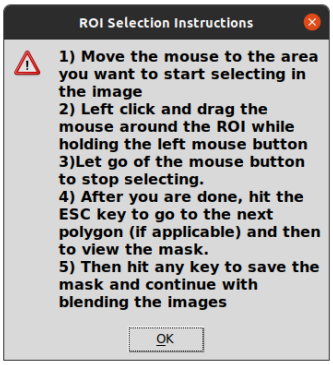
(a) Gaussian Pyramid with 5 layers for *wolves.png*

(b) Laplacian Pyramid with 5 layers for *wolves.png*

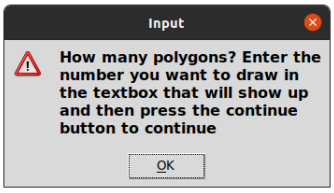
Figure 1: Gaussian and Laplacian Pyramids for *wolves.png*

1.b GUI to Select the ROI for the Mask

The next step was to create a simple GUI to select the region to use as a mask for the image blending this was done using the *tkinter* Python module and the *OpenCV* module to display messages and collect the coordinate for the free form polygon that was used to create the mask



(a) Message 1



(b) Message 2

(c) Selecting the number of polygons to draw on the image

Figure 2: Instructions for the GUI as well as selecting how many polygons to draw on the image that will be used as a mask in the blending process

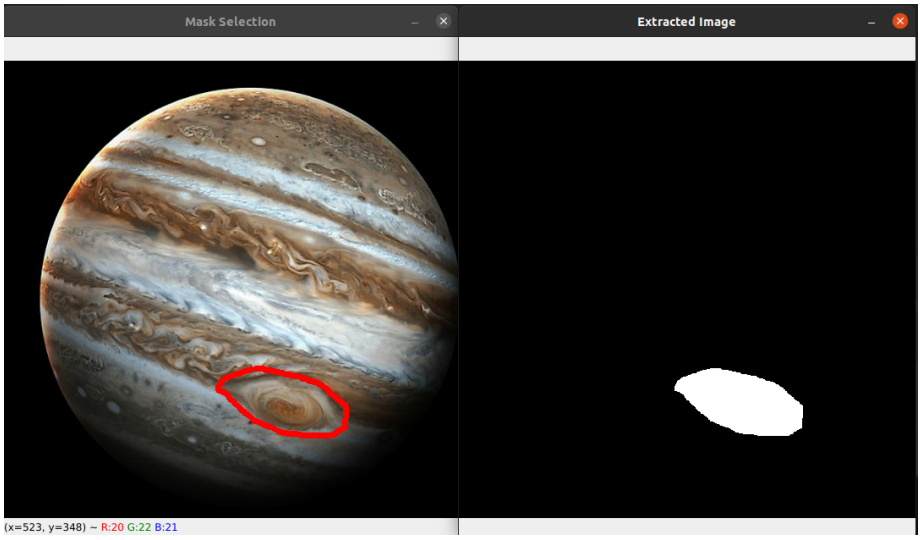


Figure 3: Example of using the GUI to create the mask as well as the resulting mask

The GUI collects the coordinates of each of the polygons for the mask and then creates a new

black image that is the same dimensions as the foreground background image and the region that was selected is filled with a value of 1 for each of the channels

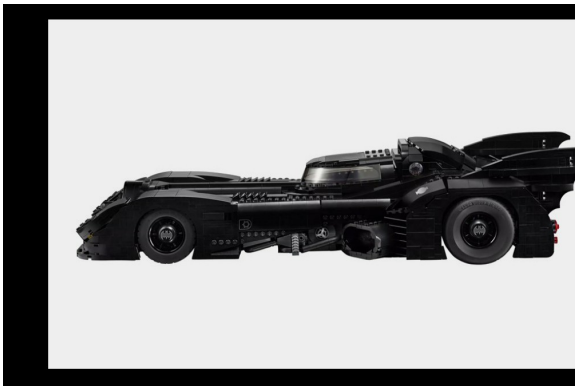
1.c Image Blending using the image pyramids and the mask

To achieve the final image the program first loads the images in and then uses the function implemented in Section 1 to generate both the Laplacian and Gaussian Pyramids for the two source images and the mask and then another function is called to do the blending of the images by following the equation

$$L_1 * G + L_2 * 1 - G = L$$

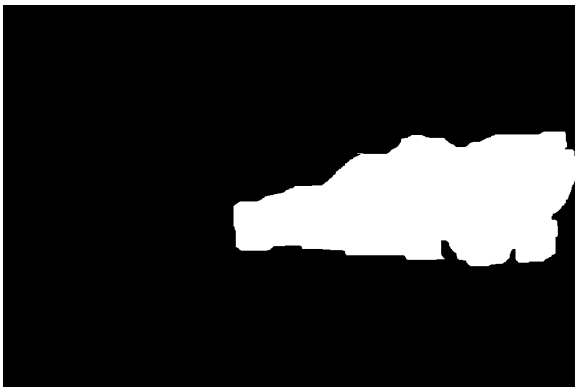
Where L_1 is the Laplacian of image 1
G is the Gaussian of the mask
L_2 is the Laplacian of image 2
$1 - G$ is the flipped mask and
L is the Laplacian blend

The Final image is then reconstructed by collapsing the final Laplacian pyramid and the image is saved to the disk The following figures show the three image pairs that were chosen to demonstrate the pyramid image blending.



(a) Source image for the foreground

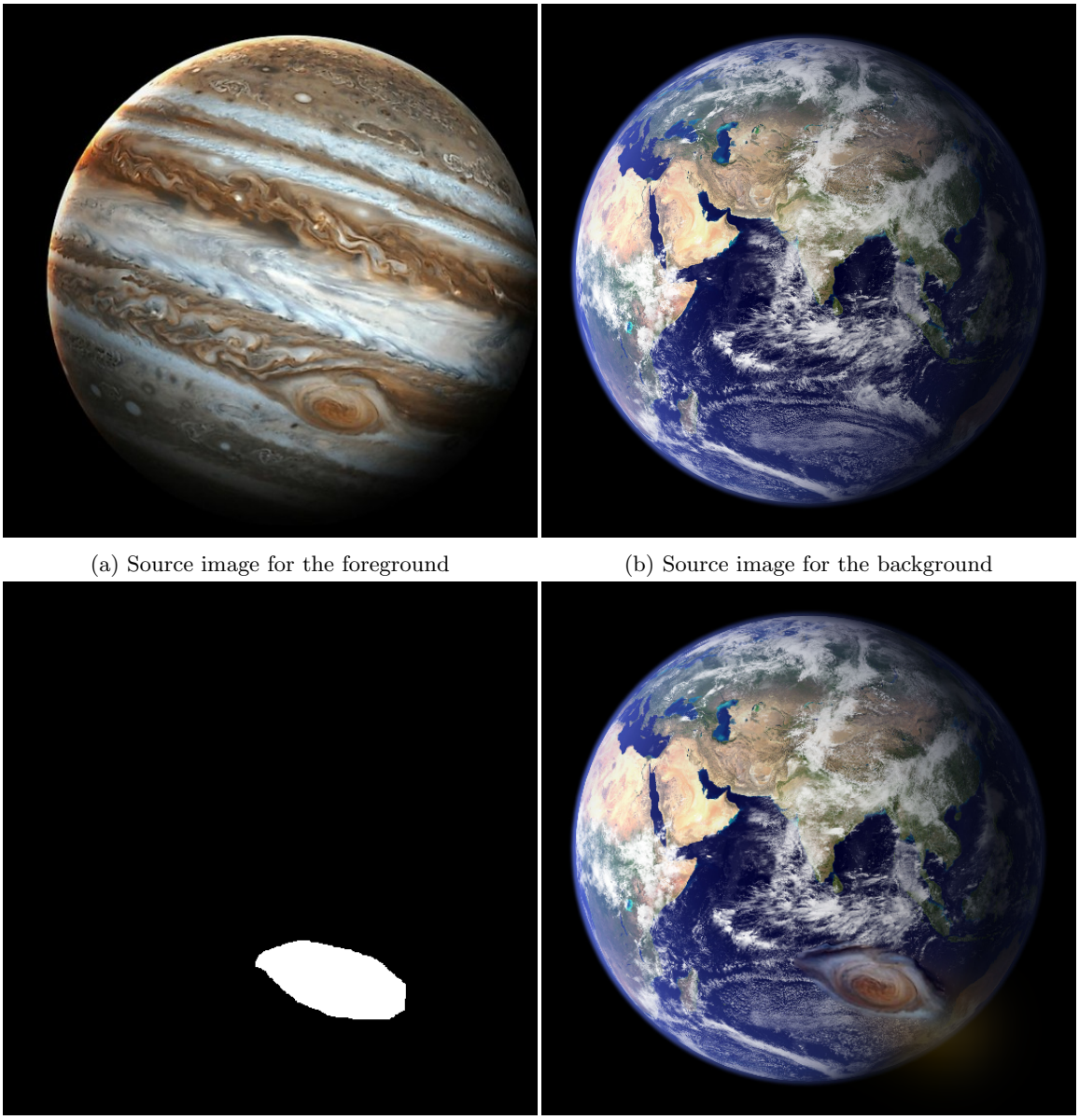
(b) Source image for the background



(c) Binary mask image

(d) Final blended image

Figure 4: Example 1 demonstrating the results of the Gaussian/Laplacian pyramid blending



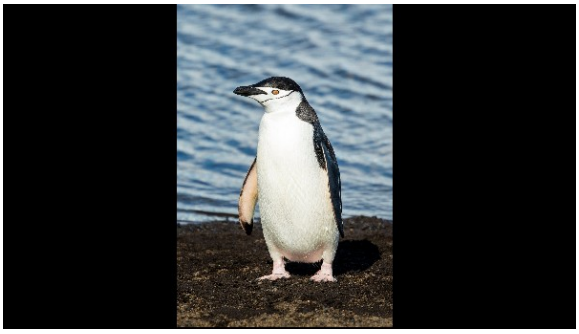
(a) Source image for the foreground

(b) Source image for the background

(c) Binary mask image

(d) Final blended image

Figure 5: Example 2 demonstrating the results of the Gaussian/Laplacian pyramid blending



(a) Source image for the foreground

(b) Source image for the background

(c) Binary mask image

(d) Final blended image

Figure 6: Example 3 demonstrating the results of the Gaussian/Laplacian pyramid blending