Assignment 3

Aleyna Alper, 21827024
Department of Computer Engineering
Hacettepe University
Ankara, Turkey
b21827024@cs.hacettepe.edu.tr

December 12, 2022

1 Introduction

Image blending is the process of combining different parts of an image to create a better and smoother image. For example, if a part of an image is blurry and low quality, it can be blended with a clearer and higher quality part of another image to create a smooth image. Image blending is a commonly used technique in image processing and helps to combine images in a seamless manner.

The purpose of this homework is to do Image Blending using Image pyramids and the mask that exists separately for each input. For this process, two pyramid structures, Gaussian and Laplacian pyramid, are used.

2 Experiment

In this section, I will explain each part of the Image blending with using process that is desired to be done using Pyramid, detailing one by one. There are four steps in the homework report that we need to follow. These steps are respectively called Laplacian Pyramid, Gaussian Pyramid, Blending and Collapsing. However, there is a situation here, before performing the operation in Laplacian Pyramid, we have to apply Gaussian Pyramid to two photographs to be blended and mask images suitable for them. That's why I changed the place of the first two titles given to us in the PDF here. While writing the code, I first apply Gaussian Pyramid and then Laplacian Pyramid. I will also explain the collapse part under the heading of reconstructing.

2.1 Gaussian Pyramid

A Gaussian pyramid is a data structure used in image processing. It consists of a series of levels representing different scales of an image. In a Gaussian pyramid, the highest level represents the largest scale of the image and the levels below it are created by reducing the size of the image. The purpose of a Gaussian pyramid is to make it easier to process and scale images at different sizes. It helps to process images in a smooth manner.

The Gaussian pyramid code I wrote in Python to apply in the assignment is as follows;

```
def gaussianPyramid(img, pyramidLevel):
    gaussian = [img]
    resizedImg = cv2.pyrDown(img)
    for i in range(pyramidLevel):
        gaussian.append(np.float32(resizedImg))
        resizedImg = cv2.pyrDown(resizedImg)
    return gaussian
```

I defined a function called gaussianPyramid that takes two arguments: img, which is an image, and pyramidLevel, which is an integer representing the number of levels in the Gaussian pyramid. I created an empty list called gaussian and added the original image img to this list. I then used the pyrDown function from the OpenCV library to reduce the size of the image. This reduced image was added to the gaussian list.

Next, I entered a loop that ran for as many iterations as specified by the pyramidLevel argument. In each iteration, the reduced image was added to the gaussian list, and its size was reduced again using the pyrDown function. This process continued until the desired number of pyramid levels was reached. Finally, I returned the gaussian list, which contains all the levels of the Gaussian pyramid. I used this function for image1, image2 and mask that I created to get result.

2.2 Laplacian Pyramid

A Laplacian pyramid is very similar to a Gaussian pyramid but saves the difference image of the blurred versions between each levels. Only the smallest level is not a difference image to enable reconstruction of the high resolution image using the difference images on higher levels. This technique can be used in image compression.
The Laplacian pyramid code I wrote in Python to apply in the assignment is as follows;

```python
def laplacianPyramid(gaussianIm):
    laplacian=[gaussianIm[-1]]
    for i in range((len(gaussianIm) - 1),0,-1):
        enlargedGaussian=cv2.pyrUp(gaussianIm[i])
        subLaplacian=np.subtract(gaussianIm[i-1], enlargedGaussian)
        laplacian.append(subLaplacian)
    return laplacian
```

I took a sequence gaussianIm and created a Laplacian pyramid using the Gaussian pyramids contained in it. First, I set the top point of the Laplacian pyramid as the last element of the gaussianIm array. Then I created a loop starting from the very end of the array. In each loop round, it enlarges the Gaussian pyramid in element i, I subtracted the enlarged Gaussian pyramid from the Gaussian pyramid in element i-1, and I added it to the resulting Laplacian pyramid. When the loop is finished, I get the Laplacian pyramid as a return value.

2.3 Blending Images

Image blending is the process of creating a new image by combining two or more images together. This process is often used to combine the characteristics of two images. For example, it can be used to combine similar parts of two images into a larger image, or to mix and combine parts of two different images. Image blending is often performed using pyramids, masks, or similar data structures. The Blending Image code I wrote in Python to apply in the assignment is as follows;

```python
def blending(laplacianIm1, laplacianIm2, gaussianMask):
    blendedIm = []
    size=len(laplacianIm1)
    for i in range(size):
        element = laplacianIm2[i] * gaussianMask[i] + laplacianIm1[i] * (1.0 - gaussianMask[i])
        blendedIm.append(element)
    return blendedIm
```

This code performs an image blending operation using two Laplacian pyramids and a Gaussian mask. It assumes that the lengths of the Laplacian pyramids are equal. First, I defined the blended image sequence as an empty array. Then I created a loop of the same length as the Laplacian pyramids. In each loop round, it computes the element i of the blended image by multiplying the element i of the Laplacian pyramid by the Gaussian mask and adding the result to the product of the element i of the Laplacian pyramid and the subtraction of 1.0 from the Gaussian mask as requested in pdf. The resulting element i is then added to the blended image sequence. When the loop is finished, the blended image sequence is returned.

2.4 Reconstructing Images

In the context of image blending, reconstruct refers to the process of recreating the original image from a blended image sequence. This process aims to create an image with the same resolution as the original image using the blended image sequence. It is achieved by enlarging the levels of the Laplacian pyramid and adding them together. The result is a copy of the original image.
The Reconstructing Image code I wrote in Python to apply in the assignment is as follows;

```
def reconstruct(blendedIm):
    laplacian=[blendedIm[0]]
    for i in range(len(blendedIm) - 1):
        enlargedLaplacian=cv2.pyrUp(blendedIm[0])
        blendedIm[0]=cv2.add(blendedIm[i+1],enlargedLaplacian)
        laplacian.append(blendedIm[0])
    return laplacian
```

In this code I defined a function that reconstructs the original image from a Laplacian pyramid. It takes a blended image sequence blendedIm as input. First, it adds the top level of the blended image sequence to a list of the reconstructed Laplacian pyramid. It then creates a loop of the same length as the blended image sequence. In each loop round, it enlarges the top level of the blended image sequence and adds the result to the next level of the blended image sequence. The resulting next level of the Laplacian pyramid is then added to the list of the reconstructed Laplacian pyramid. When the loop is finished, the list of the reconstructed Laplacian pyramid is returned.

3 Results

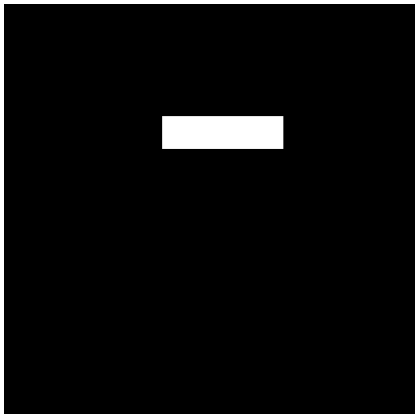
In this section, I will explain the examples I used in the code I created, the results and separate explanations for each.

3.1 Example 1

3.1.1 Inputs

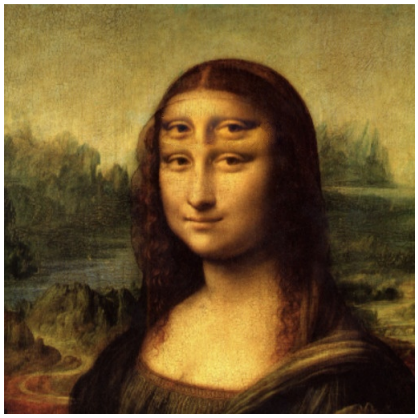

I have specially prepared this example specifically to be suitable for the output in the pdf given to us. Input 1, the first photo that appears above, is the classic Mona Lisa painting that everyone knows. The second photo, Input 2, is a particularly displaced part of the forehead with the eye. I created this image with photoshop.

3.1.2 Mask



I have defined the mask above to get the eyes in Input 2 in this example. I set the black part of this mask to Input 1 and the white part to Input 2.

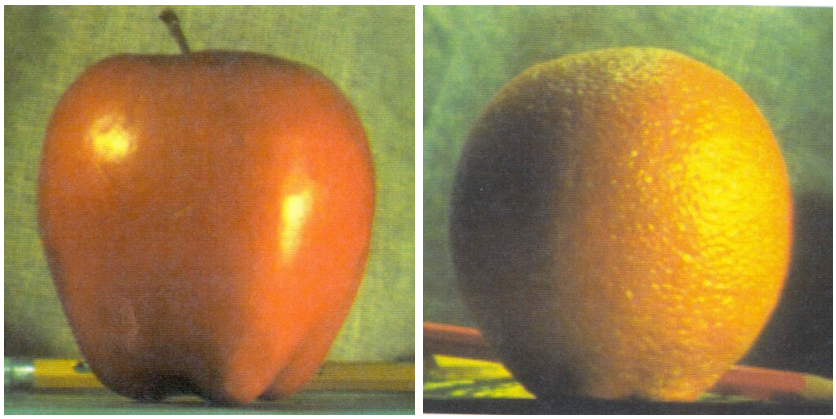
3.1.3 Output



The resulting result is as above, which will be the same as the pdf.

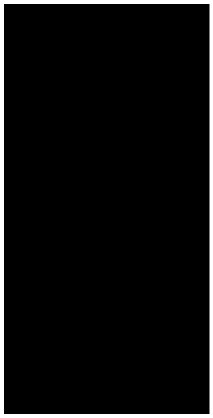
3.2 Example 2

3.2.1 Inputs

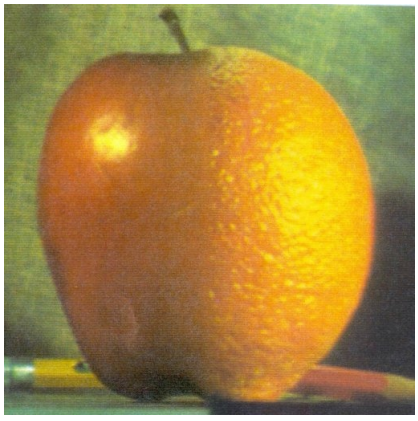


In this section, I have applied the most popular example made using Image Pyramid.

3.2.2 Mask



3.2.3 Output



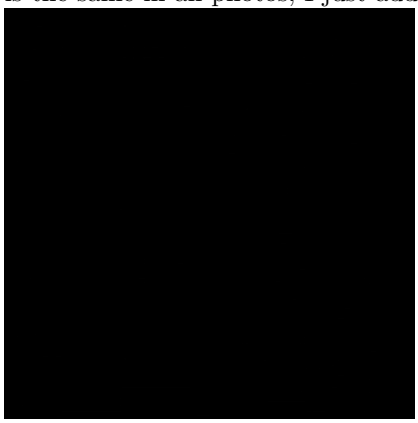
The output of this example is as above.

3.3 Example 3

3.3.1 Inputs

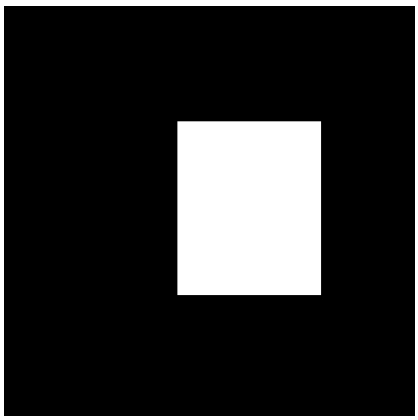

3.3.2 Laplacian Pyramid Part of Images

The images formed after the Laplacian pyramid operation, the second of the stages through which the two inputs I selected, are as below. Since this stage is the same in all photos, I just added this stage of these photos




When we look at the first photo, the edges did not appear because they did not have any details in their normal state and there was only a color transition. In the second photo, after the Laplacian stage, all the details are shown Decently in black and white

3.3.3 Mask



3.3.4 Output

The output used as pyramid level value 7 is as above.

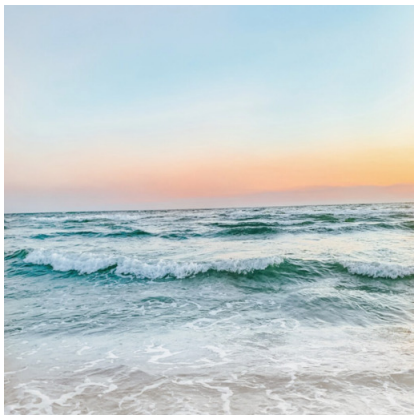
3.3.5 Pyramid Level Difference

I have used the value 7, which is the optimal value for pyramidLevel throughout my code and gives the best result. But in this example, I will show how much the output changes with different pyramidLevel values.


In the first photo that appears in this section, I have reduced the pyramid level that I use in each example from 7 to 5. In the second photo, I made the pyramid level value 3. As can be seen from the photos, the lower the pyramid level, the less smoothing occurs in the parts where the two images merge, and this transition becomes noticeable. This situation also creates a bad image.

3.4 Example 4

3.4.1 Inputs

3.4.2 Mask

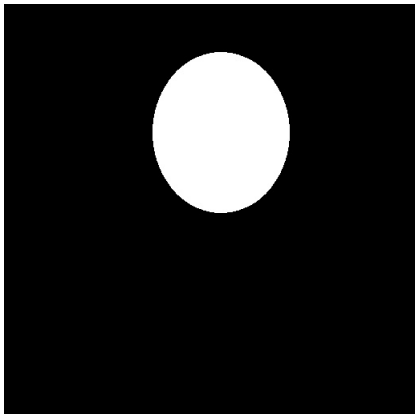
3.4.3 Output

3.5 Example 5

3.5.1 Inputs


In this example, I used two photos of Lana Del Rey. I especially struggled to find it at the same angle, because the difference in different poses is that when I tried two people, I could not provide a close-to-reality image. That's why I chose two photos I took from the same album shoot.

3.5.2 Mask



When creating a mask for these photos, I normally use cv2's rectangle function in the examples above, while this time I used cv2's ellipse function to create a cleaner image. The mask I created is as above. I also added the code I used below;

```
mask5 = np.zeros(img1.shape[:2], dtype="uint8")
#I drew an ellipse inside this array at the intervals
#I specified and made the inside of this ellipse white
cv2.ellipse(mask5, (270, 160), (100,85), 90, 0, 360, 255, -1)
#I saved the mask5 image I created as a jpg with the name Mask5
cv2.imwrite('Mask5.jpg',mask5)
mask5=cv2.imread('Mask5.jpg').astype(np.float32)
#I limited all elements of the array from 0 to 1.
mask5=np.clip(mask5, 0, 1)
```

3.5.3 Output

The resulting result is as above. There is no obvious problem, as it seems. To achieve this result, I specifically applied a black and white filter. This is the best result I have caught.

4 Conclusion

As a result of this assignment, I learned what the Gaussian and laplacian pyramid are. I've seen how I can write these using opencv. How to make Image blending using two different photos and a mask suitable for them I tested this. At the end of these processes, I completed this assignment. I have reached great information in terms of image processing.

5 References

[1] What is Gaussian and Laplacian Pyramid