CMPE 465: Intoduction to Computer Vision

Homework 1 Report

Efe Berk ERGULEC

Instructor: Venera ADANOVA

October 25, 2019

Introduction

Edge detection is a crucial method for image processing. It is useful for Computer Vision applications such as Visual Odometry, Simultaneous Localization and Mapping (SLAM) and many others. This assignment covers edge detection methods using MATLAB Programming Language. There will be three sections in this report: edge detection, hybrid images and image blending. Each section includes definitions, outcomes and discussion of results. Outcomes of experiments will be provided under this report.

Question 1: Edge Detection

First task in this question is to define *SobelEdge* function that gets an image as an input and then performs edge detection using separable filters. Filters that has used provided below:

$$Kernel_{X_{vertical}} = \frac{1}{4} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad Kernel_{X_{horizontal}} = \frac{1}{2} * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$Kernel_{Y_{vertical}} = \frac{1}{2} * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \quad Kernel_{Y_{horizontal}} = \frac{1}{4} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

These kernels came from Sobel filters and they are helpful to find G_x and G_y which are partial derivatives of image in horizontal and vertical direction.

SobelEdge function has simple structure. It takes image as input. It assigns sobel kernels as variables. Normally, sobel filter is multiplying with $\frac{1}{8}$, therefore I need to split this value to $\frac{1}{4}$ and $\frac{1}{2}$ to my horizontal and vertical kernels in order to perform same same operation that sobel filter does. After implemetation of these kernels, function reads image and converts it to grayscale. After that, I put boundaries that includes 0 values to borders and expand each side by 1. This operation prevents me to but conditions that control sides. Next step of structure includes vertical and horizontal scan of image with sobel kernels that has shown above. First two kernels generate G_x and last two kernels generate G_y and function completes its mission after this operation. G_x and G_y outputs of *SobelEdge* function shown at Figure 1 below.

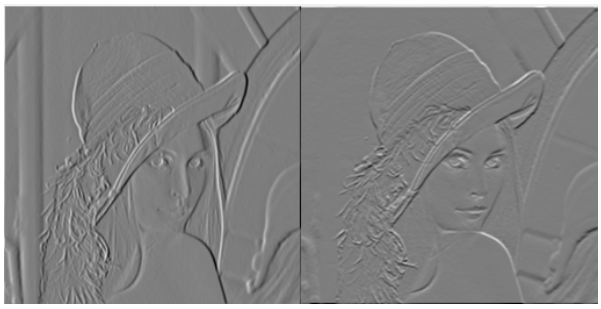

Figure 1: Gx(left) and Gy(right)

Due to the heavy operations they done, I implemented vertical and horizontal scan as dicrete functions. This approach saved me 0.5 seconds for this question.

Second task requests gradient magnitude of image using G_x and G_y that I found in previous task. It was a simple task and using eucledian distance formula, I solved it. Results provided at Figure 2 below.

Figure 2: Gradient Magnitude

Discussion of Results

This question includes two different tasks: partial derivative of images with respect to x and y, and finding gradient magnitude using results of first task. In my opinion, output of first task is very similar with the outcome that you gave us in assignment report. Thus, there isn't any concern in my mind about first task. On the other hand, second task did give more pale result than assignment and my expectations. However, it still gives edges clearly and I am satisfied with this result.

Question 2: Hybrid Images

This question requires to unite two different images with same size to generate hybrid image. Hybrid image includes low-pass filtered version of the one image and a high-pass filtered version of a second image. Reaching efficient hybrid matrix is based on arranging gaussian filters such that high-pass filtered image can fit onto low-pass filtered image shows high-pass filtered image at big size but low-pass filtered image at small size. With the help of MATLAB functions, I implemented all o them in a simple form. Results provided below:

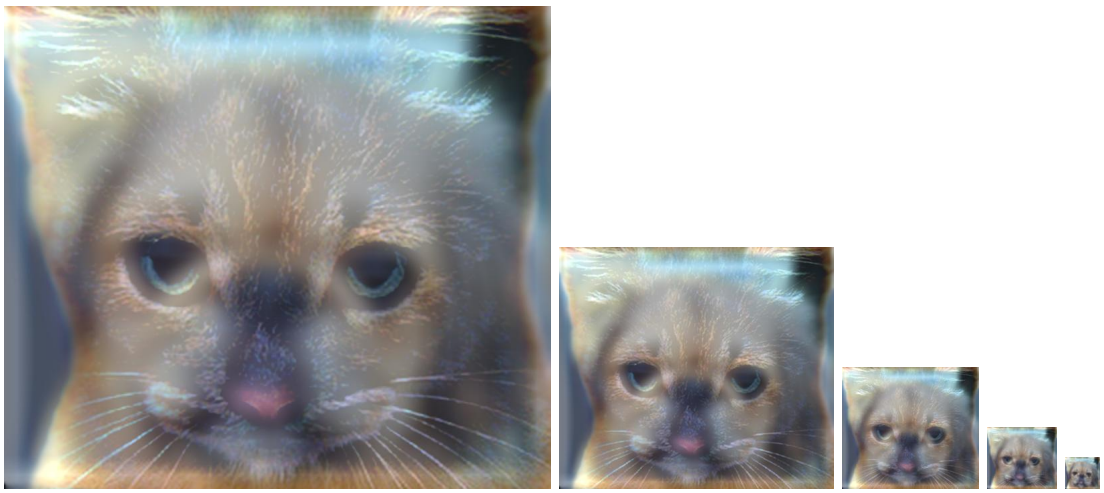

Figure 3: Transformation between image sizes.

From the picture above, hybrid image shows cat from close but when picture becomes smaller or distance increases, image turns into dog. Both cat and dog images are given by lecturer and has same pixel sizes. Pictures with different sizes doesn't work for this question because MATLAB functions used in this code works parallel with fundamental matrix calculations.

Code for this question does short operations on two different images. Code picks two images with same pixel sizes. After that, it generates high-pass and low-pass gaussian filters. High-pass filter has 100-by-100 filter size and σ=20. Low-pass filter has 20-by-20 filter size and σ=20. Filter implementation to images comes after filter generation. There is an important point in this step. In order to obtain high-pass filtered image, code needs to subtract high-pass filter from original image. Last step includes merging two filtered images.

Discussion of Results

To sum up, results of my hybrid image provided below. In my opinion, image shows both cat and dog as intended. This question was simpler than other questions because mostly MATLAB functions made important calculations. However, determining filter sizes and σ values are tougher than expected.

Question 3: Image Blending

Image blending does operations to connect two different images with each other using binary mask and Gaussian/Laplacian matrices. This question concentrates on using these tools to make a blending image. Input images and binary mask used in this question sprecified at Figure 4.

Figure 4: Two input images (left and middle) and binary filter (right).

First image is a passport photo of one of my best friends and my partner in multiple projects at Computer Engineering Department, Arda Andırın. Second photo is a hand picture that I found on internet. Third image is binary mask to blend right eye of Arda to the hand in image two in Figure 4.

Coding part of this question covers:

1. Implementation of images
2. Creating mask manually
3. Measuring gaussian pyramids by images
4. Finding laplacian pyramids with respect to gaussian pyramids
5. Merging images with laplacian pyramids of image 1 and image 2, and gaussian pyramid of image mask.

First step includes fundaental image processing functions that reads and converts image to grayscale. Two images occured after this step has shown in Figure 4.

Second step includes manual mask creation. This code doesn't provide users automatically created mask, which means user should enter mask with drag and drop tool. After user enters his filter, program generates a binary mask shown in Figure 4.

Third step is to generate gaussian pyramids of images. I implemented special *gaussPyramid* function that takes image as input parameter to create gaussian pyramid. In this function, I used MATLAB functions and ended up with gaussian pyramids shown in Figure 5, Figure 6 and Figure 7 below.

Figure 5: Gaussian Pyramid of Image 1.

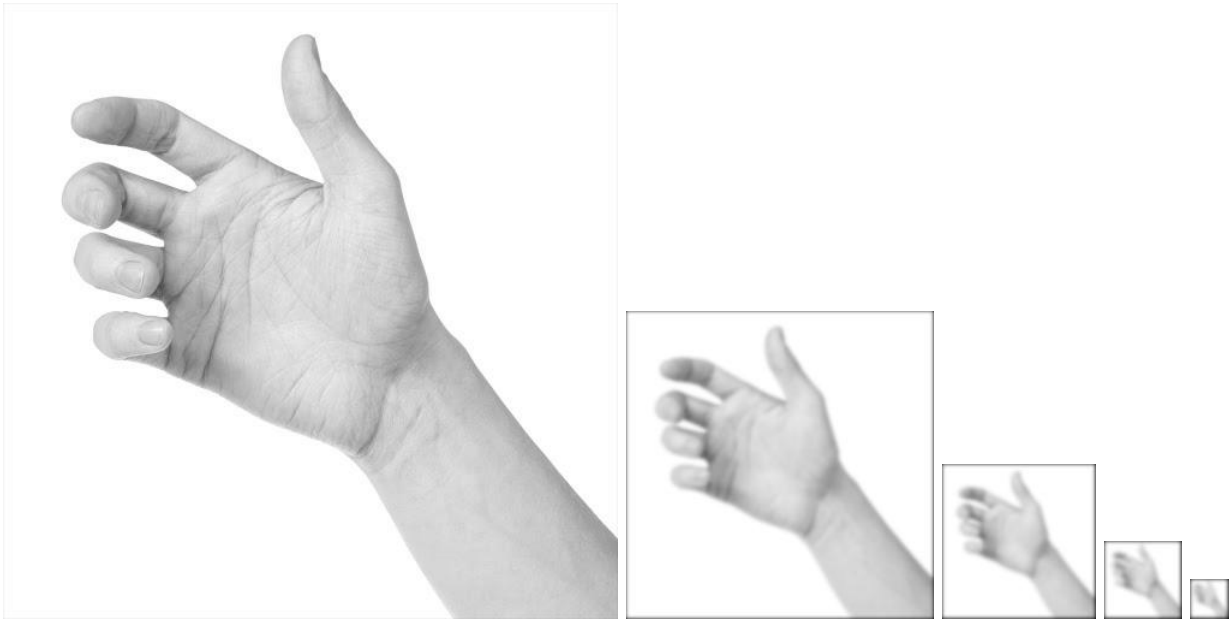

Figure 6: Gaussian Pyramid of Image 2.

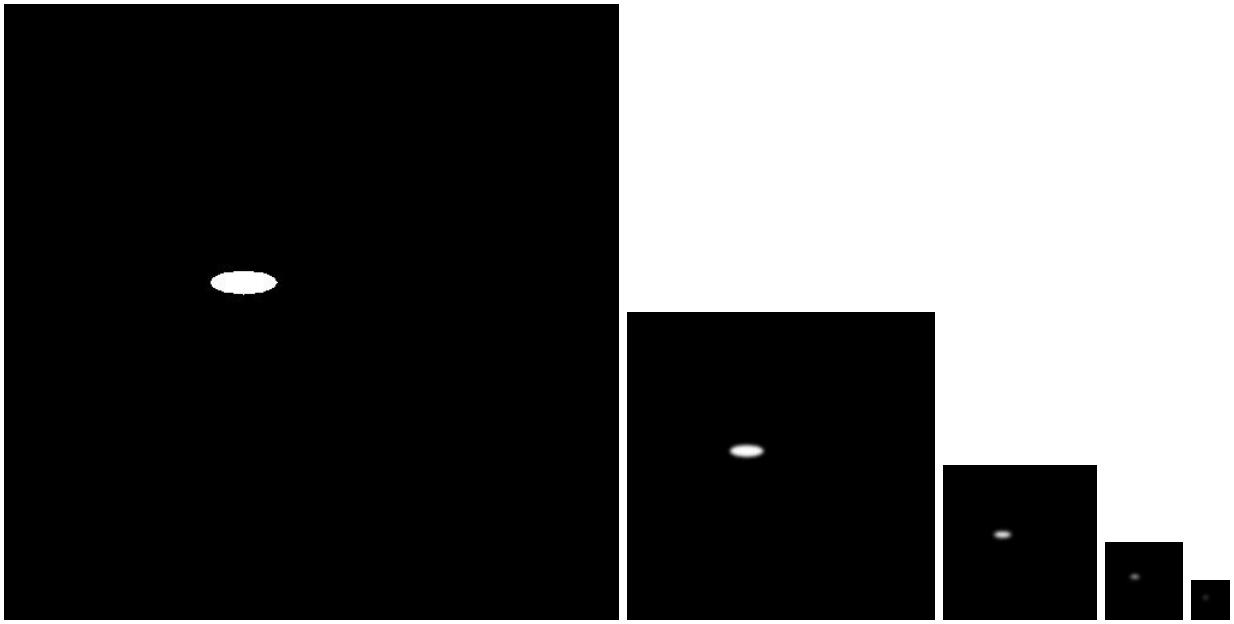

Figure 7: Gaussian Pyramid of Mask.

Fourth step is to create laplacian matrices of image one and two with respect to gaussian pyramids of these images. For this step, I implemented *laplPyramid* function in order to generate laplacian matrices. Results provided below with Figure 8 and Figure 9.

Figure 8: Laplacian Pyramid of Image 1.

Figure 9: Laplacian Pyramid of Image 2.

Merging images is the last step of this question. For this purpose, I wrote *merge_ images* function to merge two images using their laplacian pyramids. Also, gaussian pyramid of mask image is used in order to unite them. In *merge_ images* function, it took every elements of pyramid from smallest to biggest image and applied formula below to each pixel of these images. Function returns to new generated gaussian matrix.

$$R_i(x,y) = (G_{1_i}(x,y) * M_i(x,y)) + (G_{2_i}(x,y) * (1 - M_i(x,y)))$$

R_i: i^{th} element of result pyramid.
G_{1_i}: i^{th} element of gaussian pyramid of first image.
G_{2_i}: i^{th} element of gaussian pyramid of second image.
M_i: i^{th} element of laplacian pyramid of bianry mask.

After that, code uses *collapse* function in order to merge them to create an output image.

Results provided in Figure 10 below.

Figure 10: Output image.

Discussion of Results

This question mainly wants me to focus on blending two different images by using binary mask. From the results that provided above at Figure 10, I can clearly say that eye looks realistic on the hand. However, there are some problems that shows image paler than expected. I did some research about results of this and I ended up with one solid result: if there are sharp color transitions between objects in an image, both laplacian and gaussian pyramids could generate lossy images and both of my pictures have sharp color transitions from dark to bright colors. Hereby, I believe that my code does image blending with true structure and orders but due to the sharp transitions, my output has some pale points.

Conclusion

This assignment mainly focused on edge detection and intermediate image processing applications. There were three steps to work on: edge detection, hybrid images and image blending using masks. At the end of this assignment, I learned about edge detection with kernels and saw differences between filter and kernel approach. Another subject that I found chance to criticize was hybrid images. Arranging filtering matrices and σ value in order to catch more efficient hybrid images. Lastly, I learned about image blending, and implementation of gaussian and laplacian pyramids onto image. To sum up, assignment was very informative for me and I believe that I will use these subjects in my further works.