MO446 – Introduction to Computer Vision
Project 1

Breno Leite
Guilherme Leite

28/08/2017

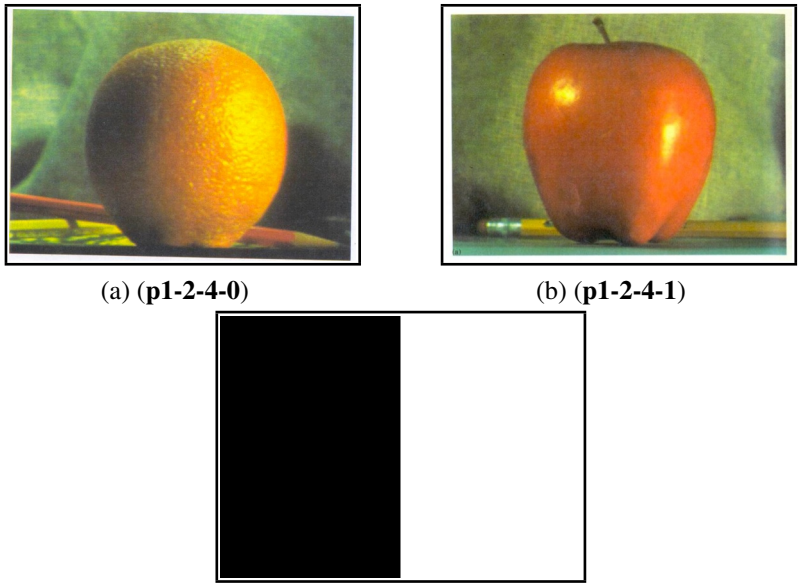
Input Images

Throughout this project some images were used as input to test the algorithms. Figure 1 was used as input for the pyramids exercises **2.1, 2.2, 2.3** and **3.1**, its dimensions are 400x300 and it is a colored image.

Figure 1: Input image for pyramids and fourier transform exercices. (**p1-1-0**)

Important note: *The borders seen in the figures are not part of the image, they are figurative information about the starting and ending points of the image. Moreover, all the image scales in this report were changed in order to make the text more readable.*

Images in Figure 2 are used for the blending exercises (**2.4** and **3.2**), their dimensions is 540x392. Note that the images are slightly rotated, this will affect some results which comes from the image and not by any error on the process itself.



(a) (**p1-2-4-0**) (b) (**p1-2-4-1**)

(c) (**p1-2-4-2**)

Figure 2: Images (a), (b) and (c) used for the blending operations.

Question 2 - Spatial blending

2.1) In the spectrum of image processing, convolution can be understood as the action of applying a mask to an image, in this case every pixel in the image is affected by the weighted sum of the filter applied to the pixel neighborhood. This approach requires some special attention when dealing with the edges of the image, since the application of the mask will overshoot the image borders, our solution to this problem was to extend the original image filling its new borders with zeros, as seen in Figure **??**, by doing so the convolution simply ignore those values and don't account them into the result, a minor downside to this adpproach is the lessen effect of the mask around the edges, this solution was chosen for its simplicity to implement. Additionally the convolution was tested with three smoothing masks, 3x3, 7x7 and 15x15, in comparison to the embedded solution by OpenCV our convolution was noticeably slower as seen in Table 1. As expected the masks smooth the edges around the image and suffers with a darkening close to the borders as seeing in Figure **??**, with bigger masks the image suffered more distortion to the point of loosing all of its fine details, see Figure **??**, the loss of such details is accounted by the size of the neighborhood that affect each pixel, thus a larger mask smooths more than a smaller one.

	Time (ms)		
Mask Size / **Convolution**	**3x3**	**7x7**	**15x15**
Implemented	5.123	4.8681	5.238
OpenCV	0.006	0.004	0.019

Table 1: Comparission of time between our implementation and OpenCV convolution.

(a) (**p1-2-1-0**)

(b) (**p1-2-1-1**)

(c) (**p1-2-1-2**)

Figure 3: Images (a), (b) and (c) are the result of the convolution with gaussian masks, respectively 3x3, 7x7 and 15x15

2.2) To store the gaussian and laplacian pyramids of this project we chose a list, in which each node of it holds a level of the pyramid. Since the data structure is a list the access method is as in a array, $pyramid[i]$ returns the image in the ith pyramid level, disregarding then, the necessity to implement an access function. The strategy chosen to implement the interpolation was the bilinear interpolation. These decisions were taken regarding the time to implement and simplicity to understand. It is also worth noting that due to some confusion about the meaning of Up and Down, these functions were renamed as follow: Expand referring to the action of expand the image width and height, and Contract analogously. Figure 4 shows the pyramid formed with the implemented function.

2.3) The laplacian pyramid demanded more attention to its details, such as the fact that the last level of the pyramid is a copy of the same level at the gaussian pyramid. This particular level is used to perform the reconstruction of the original image, first the image is extended and interpolated, to match the previous level size, then it is summed with the previous level of the pyramid, the repetition of these steps result in the original image that was compressed in the pyramid, Figure 5 shows the laplacian pyramid.

2.4) The process of blending two images implement here is referred as "Splining Regions of Arbitrary Shape" in Burt and Adelson's paper, it consists of three gaussian pyramids *GA*, *GB* and *GM*, and three laplacian pyramids *LA*, *LB* and *LS*, in which *LS* is the resulting pyramid and the image at its base is the blended image. The algorithm to blend the images uses both *LA*, *LB*

(a) (**p1-2-2-0**)

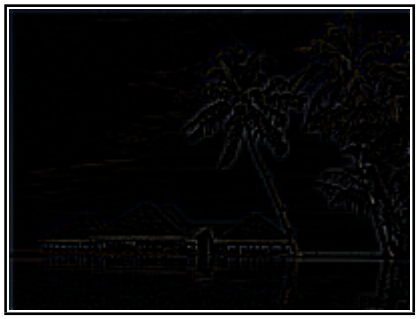
(b) (**p1-2-2-1**)

(c) (**p1-2-2-2**)

Figure 4: Images (a), (b) and (c) composing the Gaussian Pyramid formed from image **p1-1-0**

and *GM* pyramids simultaneously, from their top it multiply each lower resolution image with the equivalent mask (Figure 6 (a) and (b)), sum the resulting images with each other (Figure 6 (c)) and expand the resulting image, repeat these steps and the resulting image at the base of *LS* is the blended image. The core point of this process is the gaussian filter applied to the mask that smooth its borders and the expand operation performed in the pyramid. Two tests were performed reproducing the original results, in Figure 2 (a) and (b) were merged using Figure 2 (c) as mask and Figure 7 (a) and (b) were merged using Figure 7 (c) as mask, resulting in Figure 8 (b).

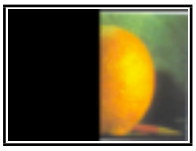
(a) (**p1-2-3-0**)



(b) (**p1-2-3-1**)

(c) (**p1-2-3-2**)

Figure 5: Images (a), (b) and (c) composing the LaPlacian Pyramid formed from image **p1-1-0**

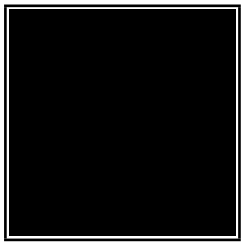


(a) (**report-p1-2-4-0**)

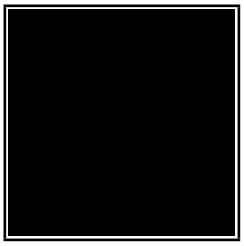
(b) (**report-p1-2-4-1**)

(c) (**report-p1-2-4-2**)

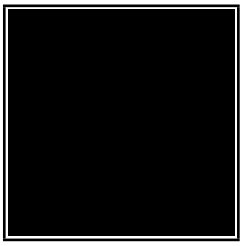
Figure 6: Images (a) and (b) are the result of the blending operation.



(a) (**p1-2-4-3**)



(b) (**p1-2-4-4**)

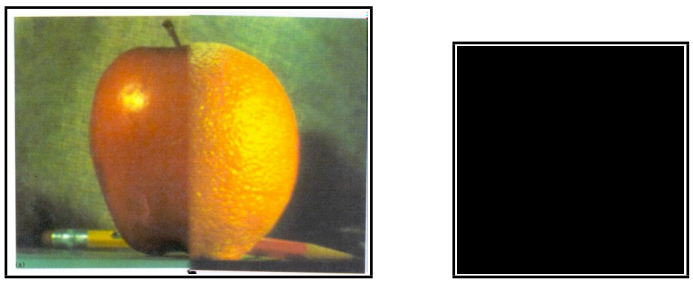


(c) (**p1-2-4-5**)

Figure 7: Images (a), (b) and (c) used for further testing the blending operations.

Question 3 - Frequency Blending

In this question, we will present some experiments developed using *numpy* and *OpenCV* func-



(a) (**p1-2-4-0**) (b) (**p1-2-4-1**)

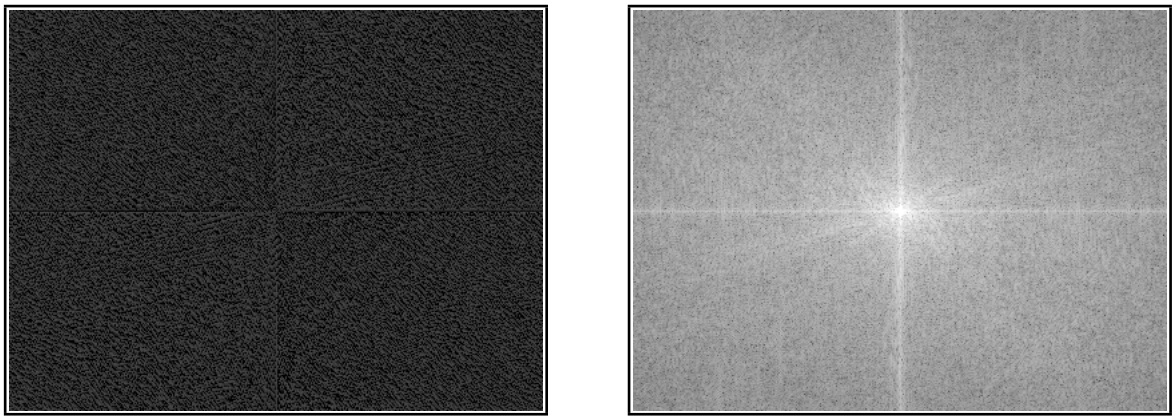
Figure 8: Images (a) and (b) are the result of the blending operation.

tion in order to transform the image from the spatial domain to the frequency domain. The process will be divided into two different experiments.

On the first one (**3.1**), we will use the transformation and make some modifications to the phase and magnitude in order to see the impact on the image recontructed. On the second experiment (**3.2**), we will be blending the two images used into the question **4.2** in order to check how it is compared to the spatial blending.

3.1) Exploring Fourier Space

In order to explore the fourier space, we implemented two functions. One to transform an image into magnitude and phase, and the other to reconstruct the image from magnitude and phase. The Figure 9 shows the magnitude and phase obtained transforming the Figure **??** to the frequency domain.



(a) (**p1-3-1-0**) (b) (**p1-3-1-1**)

Figure 9: (a) Phase and (b) Magnitude images created on the fourier transform using image **p0-1-0**.

In order to create these images, the results of the fourier transform were shifted to the center and then reduced by the functions, $20 * log_e(magnitude)$ and $40 * log_e(phase)$. These values were used to mantain the values between 0 and 255, which could be showed as an image. These values are translated back when doing the reverse of the fourier transform.

The Figure 9a shows the phase of the image, it is hard to obtain some information from this image. However, it is respective to the directions of the image. Moreover, the Figure **??** shows the values to the magnitude, the intensity shows the values of the magnitude. In both Figures, light

pixels indicates high values.

We can have a little more information with the magnitude image, it is noticeably that the image is bright, which indicates that the image on Figure **??** has more high frequency than low frequency.

3.2) Blending