CMSC733: Homework 0 - Alohomora Phase 1

Gaurav Raut
University of Maryland
Email: gauraut@umd.edu

I. Introduction

This report provides a detailed explanation on the implementation of the pb-lite boundary detection algorithm. Pb-lite is based on extracting the intensity changes as well as the texture changes from the image. K-Means clustering is used for the generation of the three maps. The algorithm does so by following the given steps: 1) Generate filterbanks, 2) Evaluating the texton, brightness and color maps, 3) Finding the texture, brightness and color gradients and 4) Calculating the pb-lite output.

II. Implementation

This section will provide an insight on the components which make up the pb-lite boundary detection algorithm. We first generate the filter bank. The filters can detect the texture changes in the image. Using these filter bank, we will construct a texton map. Two other maps representing brightness and color clusters are generated. We further find the gradients of these maps and finally find the pb-lite output.

A. Filter bank Generation

The algorithm uses three different types of filters which are: 1) Derivative of Gaussian(DoG), 2) Leung-Malik filter, and 3) Gabor filter. Each filter can characterize textures in the input image.

1) Derivative of Gaussian(DoG) filter: As the name suggests, the DoG filters are made by getting the gradient of the Gaussian kernel. To get the gradient, we simply convolve our Gaussian kernel with a simple Sobel kernel(we can apply any one direction of the kernel but applying both works as well). Later, the generated filter will be rotated in 16 different orientations. The above procedure is carried for two distinct values of sigma which are, σ = 1, $\sqrt{2}$.

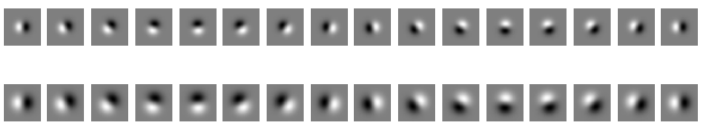

Fig. 1: DoG filters with σ = 1, $\sqrt{2}$

2) Leung-Malik(LM) filter: The Leung-Malik filter consists of four different filters namely the elongated derivative of gaussian, elongated double derivative of gaussian, the laplacian and normal gaussian filter. The filter bank uses three σ values and 6 orientations for the elongated filters. For the laplacian, 8 distinct σ values are used and 4 distinct σ values are used for the gaussian filter.

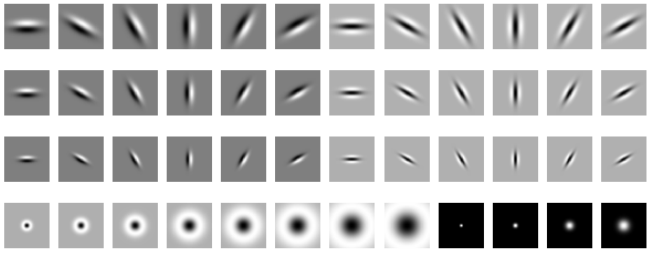

Fig. 2: Leung-Malik Small(LMS): The elongated DoG and DDoG filters uses σ = $\sqrt{2}$, 2, $2\sqrt{2}$. The laplacian uses σ = $0.5\sqrt{2}$, 1, $\sqrt{2}$, 2, $2\sqrt{2}$, 4.The gaussian uses σ = $0.5\sqrt{2}$, $\sqrt{2}$, $2\sqrt{2}$, 4.

3) Gabor filter: The Gabor filter are a set of filters at 8 different orientations and 4 distinct sigma. They are designed as per the filters in the human visual system. Gabor filters are basically gaussian kernels which are modulated by a sine wave. The below is the equation for designing a Gabor filter:

$$G_c = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi f x) \tag{1}$$

where,
G_c *is Gabor filter*
f is the frequency of modulation
I found out that by adjusting the frequency as we change σ yields better filter results which are shown in the Fig. 3 below.

B. Map Generation

Three different types of map clusters are used to generate the pb-lite output: The texton map($\mathcal{T}$), the brightness map($\mathcal{B}$) and the Color map($\mathcal{C}$). These maps are generated by assigning an ID to the pixels that provide similar information in the input image. $\mathcal{B}$, $\mathcal{C}$, $\mathcal{T}$ for all images are provided in the Fig. 4.

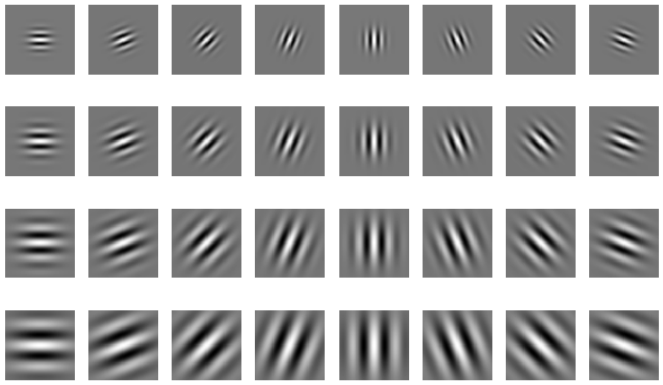

Fig. 3: Gabor filter generated using σ = 1, $\sqrt{2}$, 2, $2\sqrt{2}$

1) Texton Map: Texton map represent the texture clusters in the input image. Pixels with similar texture are binned in to the same cluster. K-Means clustering is used to create clusters. The number of clusters chosen are K = 64 and they provide satisfactory classification. The process of texton map generation follows three main steps:
1) Read input image in grayscale
2) Convolve the image with the filters in the filter bank
3) Apply K-Means clustering with K = 64

2) Brightness Map: Brightness map captures the brightness changes in the image. The process of developing the brightness map follows generating 16 clusters using K-Means on the grayscale channel of the input image.

3) Color Map: Color map captures the color changes in the image. The process of developing the color map follows generating 16 clusters using K-Means on the RGB channel of the input image.

C. Map Gradients Generation

Now, we need to calculate the gradients ($\mathcal{C}_g$, $\mathcal{B}_g$, $\mathcal{T}_g$) of the above generated maps. To do so, we make use of half disc masks. The masks are provided in Fig. 5. They are simply a pair of binary semi-circles with opposite orientation. 3 different radius and 8 pairs of orientations are used to generate these masks.
Later, we convolve these half disc masks in pairs on our $\mathcal{C}$, $\mathcal{B}$, $\mathcal{T}$ maps. These will generate two distribution across the same scale. If the difference between the distribution is not similar, the gradient will be large. We then calculate the χ^2 distance using the below equation,

$$\chi^2(g,h) = \frac{1}{2}\sum_{i=1}^{K}\frac{(g_i - h_i)^2}{g_i + h_i} \tag{2}$$

We generate the χ^2 distance for all bins. The number of bins chosen in this implementation are equal to the number of clusters generated using K-Means in the previous step. The

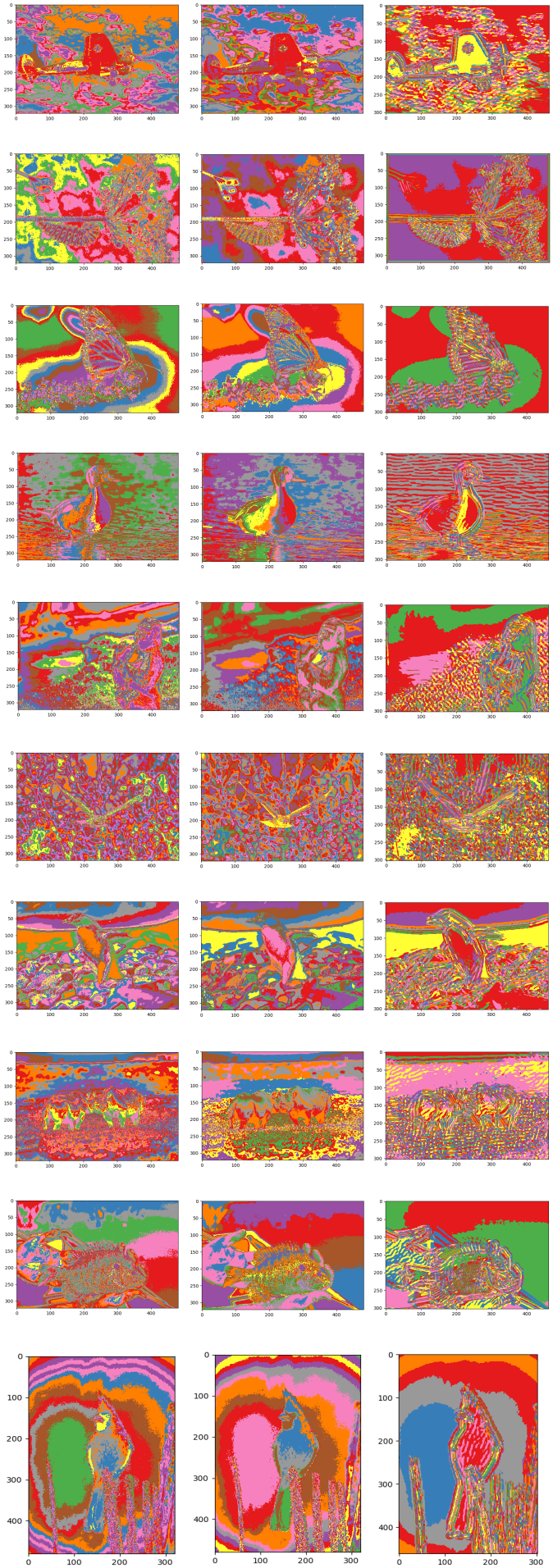

Fig. 4: $\mathcal{B}$, $\mathcal{C}$, $\mathcal{T}$ respectively

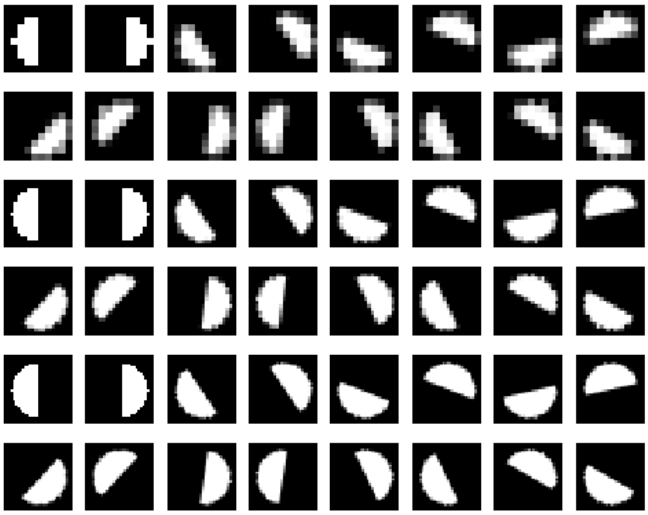

Fig. 5: Half disc maks generated with radius = 10, 20, 30 pixels

above procedure is repeated for all pairs and scales of half disc masks and the output is then averaged over all pairs. The output of this process is the gradient of the respective map. Fig. 6 provides the $\mathcal{C}_g$, $\mathcal{B}_g$, $\mathcal{T}_g$ for all images.

D. Pb-lite output

Now that we have the gradients of our maps, we proceed to calculate the final pb-lite output. To do so, we use the below equation,

$$PbEdges = \frac{\mathcal{T}_g + \mathcal{B}_g + \mathcal{C}_g}{3} \odot (w_1 * cannyPb + w_2 * sobelPb) \tag{3}$$

As can be seen in the equation, we perform element-wise multiplication with the canny and sobel baselines. The output gives us the probability of boundary.
Fig.7 provides the output results of pb-lite as compared to sobel and canny outputs for all images.

III. Conclusion

I have determined that the pb-lite output is better at edge detection than the canny and sobel baselines. This is because pb-lite is better adapted to detect the texture changes as well as the intensity changes. As can be seen in Fig. 7, pb-lite shows contours which are not easily detected with either sobel or canny. Alternatively, the outputs of pb-lite closely resemble the ground truth.
Adding to the conclusion, I have also determined that the pb-lite output is vastly subjective and depends on the user's need of the features or the way gradients are processed after calculations. For example, I have found nuances even while saving the images using the cv2.imwrite() and plt.savefig() method. The outputs from the latter are a lighter as compared to the former. Fig. 8 provides an example of this phenomenon.

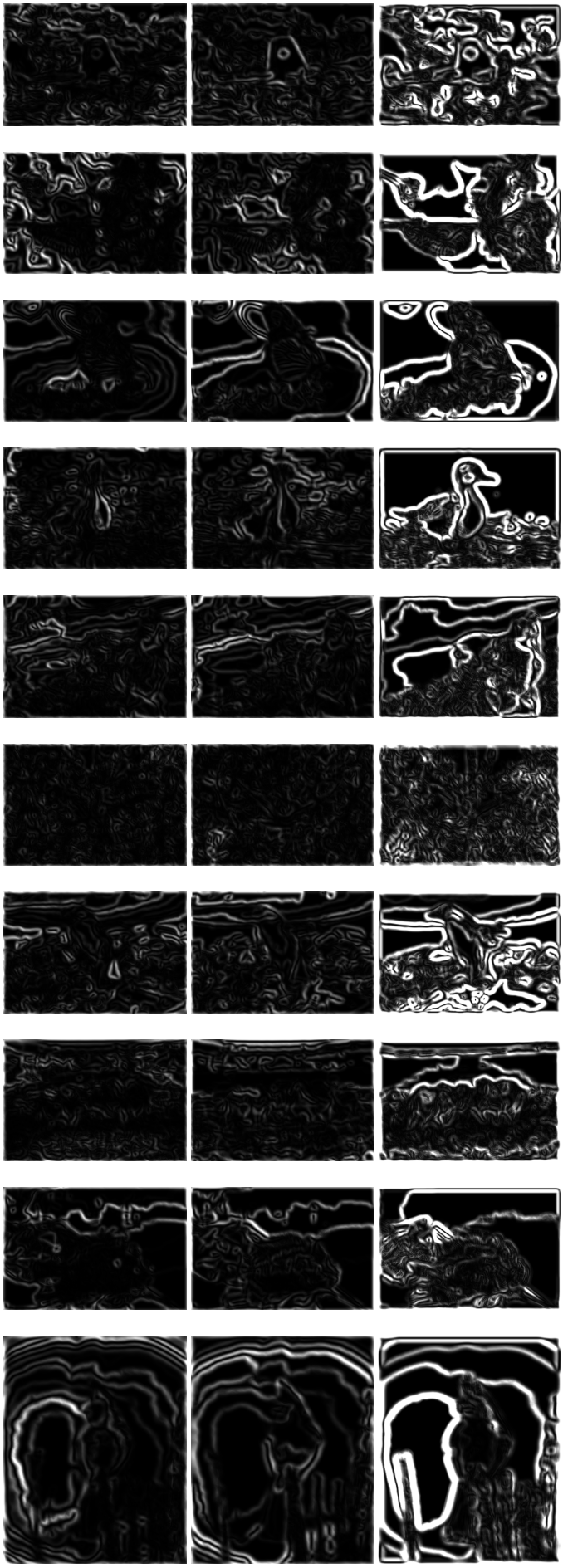

Fig. 6: $\mathcal{B}_g$, $\mathcal{C}_g$, $\mathcal{T}_g$ respectively

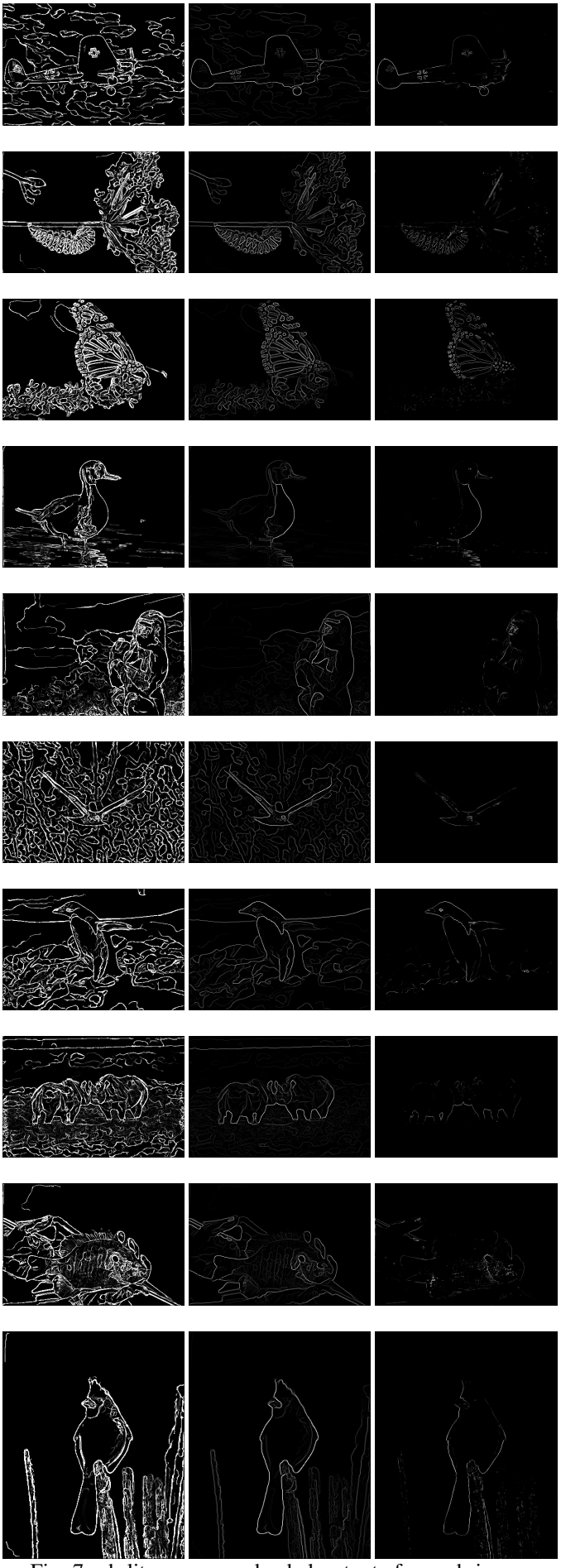

Fig. 7: pb-lite, canny and sobel outputs for each image respectively

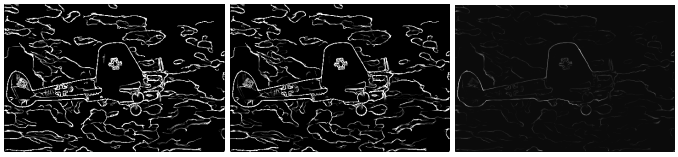

Fig. 8: Left: cv2.imwrite(), Middle: Scaled output, Right: plt.savefig()

Also, if we scale the gradients, sobel and canny in the range of 0 to 255, we get a different output. Even adjusting the weights for sobel and canny gives a different output. To conclude, I have determined that such subjective post-processing depends on the user and their expectations of the output. That's where I think pb-lite outperforms the traditional edge detection algorithm by providing different levels of detailing.