Homework 0 : Alohamora

Advait Patole
Masters of Engineering in Robotics
University of Maryland, College Park
Email: apatole@umd.edu
Phase 1

***Abstract*—The field of computer vision has been a great topic of research for many researchers.The problem of boundary detection is a well known problem that can be solved using various image processing algorithms.The paper talks about the technique of boundary detection in an image using Probability of boundary detection algorithm which is different from classical image processing algorithms like Sobel and Canny edge detection algorithms.**

I. PHASE 1 : SHAKE MY BOUNDARY

In this phase we develop a simplified version of Probability of Boundary (pb) detection algorithm to detect the edges and boundaries in image. We have utilized the brightness, color, and texture information at various scales to find the edges and boundaries in the image. The first step in this process is to develop a filter bank of various filters that can provide the information about texture information of the image. After filtering the image with the filters, we cluster the filtered image using K-Means algorithm to generate a texton map. The filters that are the part of the filter bank are oriented DoG(Difference of Gaussian) filters, Leung Malik filters and Gabor filters.

A. Creating Filters

1) Oriented DoG Filter: The Derivative of Gaussian filter is the first derivative of the Gaussian. It can be calculated by convolving the Sobel filter with the Gaussian. The derivative of Gaussian was found for two different dimensions (G_x and G_y) and the resultant of both the derivatives (G) gave the derivative of Gaussian filters. The DoG filters were calculated for two different scales (σ_1=1 ,σ_2 = 2) and 16 different orientations.

$$G = G_x cos\theta + G_y sin\theta \tag{1}$$

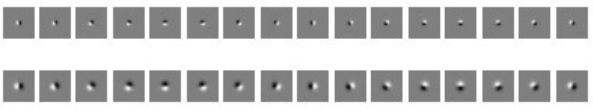

Fig. 1: Derivative of Gaussian filter with σ = 1, 2 and 16 different orientation

2) Leung Malik Filter: It is a set of different filters at different orientations and scales. It comprises of total of 48 filters in its filter bank. It consists of first and second derivatives of Gaussian at 6 orientations and 3 scales. There are 8 Laplacian of Gaussian filters and 4 Gaussian. There are two variations of Leung Malik filters depending on the scales namely LM Small (LMS) whose $\sigma = \{1, \sqrt{2}, 2, 2\sqrt{2}\}$ and LM Large (LML) whose $\sigma = \{\sqrt{2}, 2, 2\sqrt{2}, 4\}$.The first and second derivative of Gaussian was found out with separate standard deviation values in x and y direction. Also, the first and second derivative was calculated for the first 3 scales only while the Laplacian of Gaussian and Gaussian were calculated for all the given scales. To find out the Laplacian of Gaussian the Laplacian kernel was used, and it was convolved with the Gaussian.

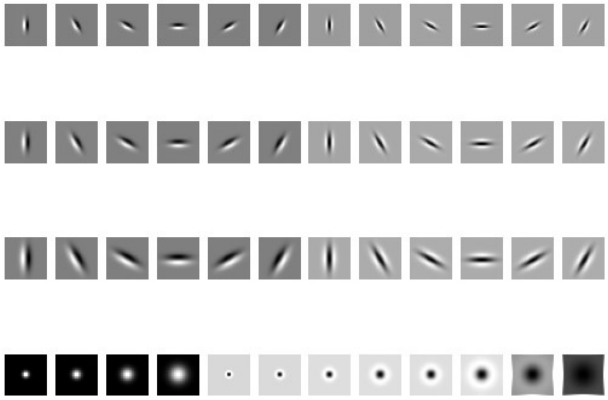

Fig. 2: Leung Malik filter with 3 rows the first and second derivative of Gaussian alternately.In last row The first 4 entries are of the Gaussian and the last 8 entries are Laplacian of Gaussian

3) Gabor filter: Gabor filters are special filters which are used for textural analysis. To extract the texture information, they particularly search for any change in frequency in a particular region. Gabor filter is a Gaussian kernel function modulated by sinusoidal plane wave. Image analysis done through Gabor filters on an image are somewhat like perception in human system. To create Gabor filters, we modulated Gaussian function sinusoidally for 8 scales and 6 orientations. There are different parameters like lambda, gamma that govern the sinusoidal modulation of the Gaussian are used. The lambda is the wavelength of sinusoidal component, gamma is the aspect ratio of the Gaussian, the values of lambda and gamma are set as 10 and 1 respectively. The lambda component is related to the frequency of the sinusoidal component. The Gabor function is then rotated for 6 different orientations spanning over a total range of 360 degree.

B. Texton, Color and Brightness Map

1) Texton Map T: The next step in the process is to generate the Texton Map. In the texton map the pixels of image are

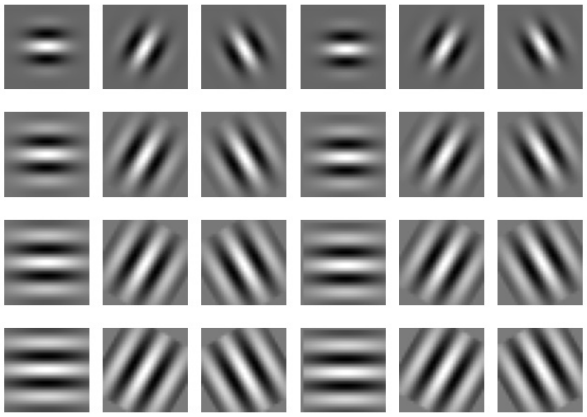

Fig. 3: Gabor filter

clustered based on different textures that are found in the image. The image is filtered with different filters present in the filter bank and then that image is clustered. K-Means clustering algorithm is used for clustering the image. After filtering the image with all the filters, we end with a stack of images with dimension h x w x N where h is the height of image, w is width of image and N is the number of filters. We cluster this image stack into k clusters and each pixel has a unique texton id. The output of texton map is an image which has information about the texture changes in the original image. The output of texton map can be seen in figure 4.

2) Color Map C: The color map is generated in similar manner as Texton map but here the criterion of clustering is the color of each pixel rather than textures. We have clustered the image into 16 clusters depending on the color distribution in the image. The purpose of color map is to get the color properties present in image and segment the image based on different colors into different clusters. The output of color map is shown in figure 4.

3) Brightness map B: Brightness map is used to find out the intensity changes present in the image. It is generated in the similar way as the color map but here the clustering is done based on the intensity values at each pixel. The image is clustered into 16 clusters to find the intensity information in the image and the output can be seen in figure 4.

C. Gradients of Texture, Brightness and Color Map

1) Gradients: To find the gradients of the texton, color and brightness map we need to find the difference of values between a pixel and its neighboring pixels. For finding the gradient we need to find out the chi-square (χ^2) distance. To find chi square ((χ^2)) distance and to save computation we make use of half disc's mask that we convolve on the image to find difference in the value between a particular pixel and all the pixels in its neighborhood rather than looping over each pixel in its neighborhood's half discs are just binary images of half discs that help to calculate chi square (χ^2) distance. We have created half discs in pair with 3 scales and 8 orientations. To create the half discs, we have set the radius of half discs as 8, 12, 16. Using these half discs along with

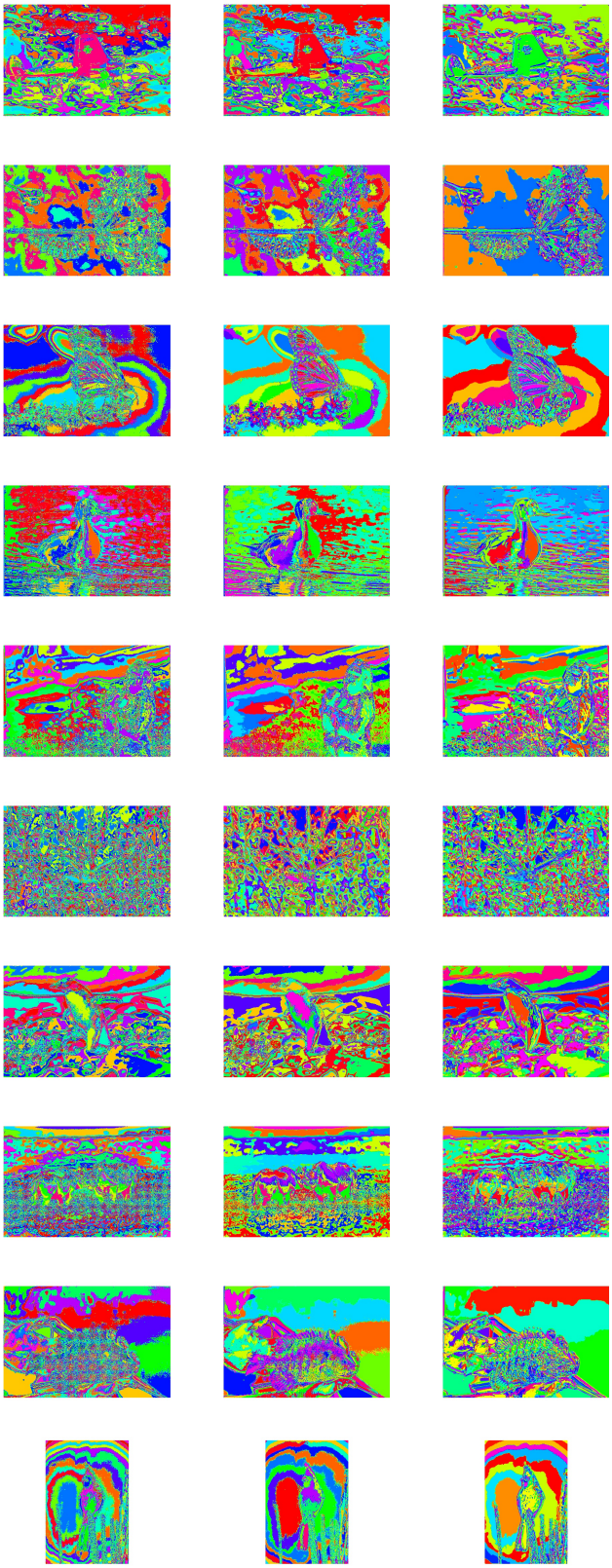

Fig. 4: $\mathcal{B}$, $\mathcal{C}$, $\mathcal{T}$ for Images 1 - 10

the chi square distance can help us to find out the gradient for color, brightness and texton map. The half discs are stored as a pair and these half discs are convolved over each cluster in the color, brightness and texton map. We find the chi square (χ^2) distance of the result of convolution between the image and right and left half discs. The resultant gradient is an image stack of the dimension h x w x p where h Is height of image, w is the width of image and p Is the number of the half discs pairs. We then find the mean of the image stack over its third dimension (number of half disc's pairs) and we get the gradient of dimension h x w. This method is repeated to find the gradient of color, brightness, and texture map. The output of texton gradient (T_g), brightness gradient (B_g) and color gradient (C_g) are shown in figure 6.

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

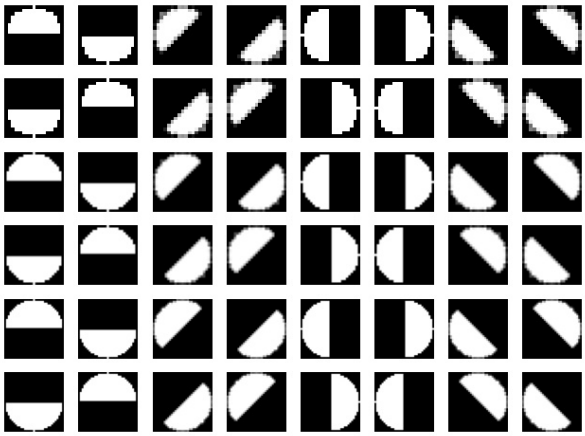

Fig. 5: Half Discs

D. Pb Lite Output

The output of pb lite can be found by combining the output of Canny and Sobel baseline output of the images and the gradient of color, brightness, and texture output. We find the average of the gradient output and do element wise multiplication with Sobel and Canny outputs. Another aspect is that we have used weights to give relative weight-age to Canny and Sobel output such that the total of weights adds up to 1.The resulting pb lite outputs are as shown below.

$$PbEdges = \frac{(\mathcal{T}_g + \mathcal{B}_g + \mathcal{C}_g)}{3} \circ (w_1 * CannyPb + w_2 * SobelPb) \tag{3}$$

II. Conclusion

From the output of pb lite we conclude that output varies depending on the weights that are assigned to the Sobel and Canny baselines. For example, if we increase the weights of Canny baseline, we see that there are more edges that are observed in the image this is since Canny edge detector is capable to capture weak edges as compared to Sobel edge

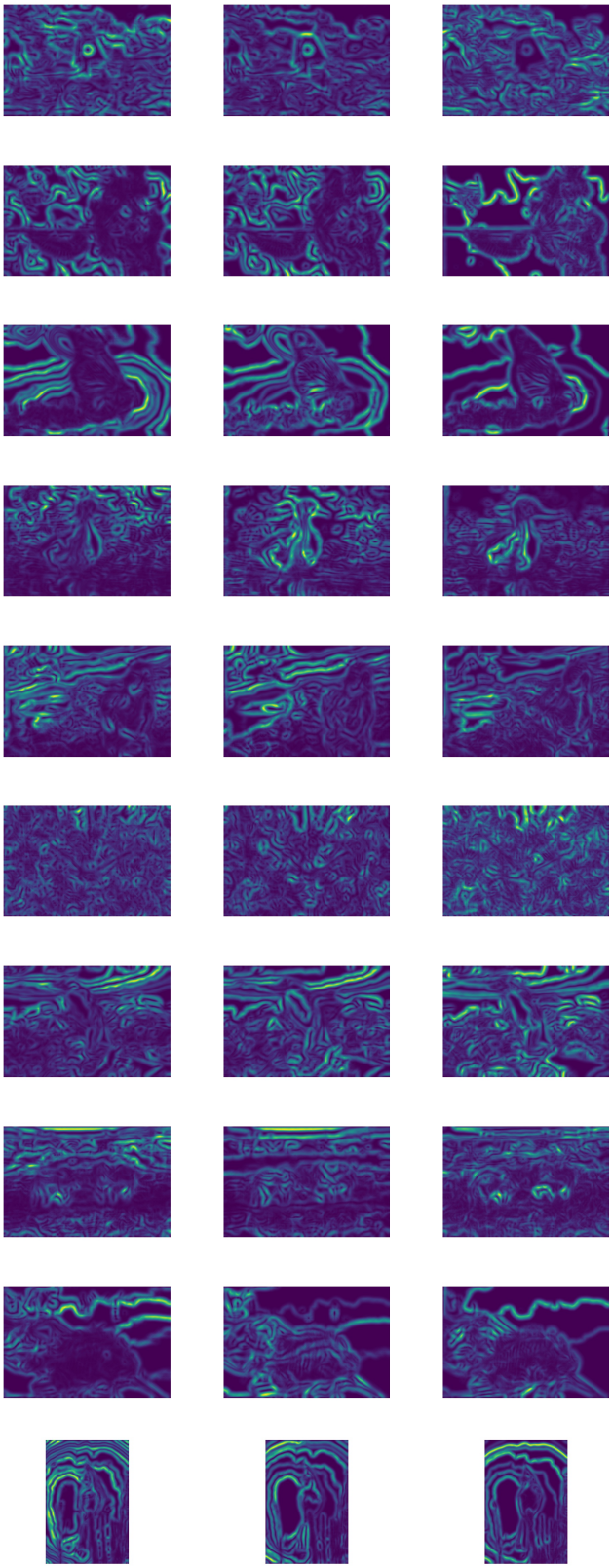

Fig. 6: $\mathcal{B}_g$, $\mathcal{C}_g$, $\mathcal{T}_g$ for Image 1 - 10

detector. This gives us more flexibility to change the output of pb lit depending on the contribution of Sobel, Canny or the gradients of the texture, color, and brightness map. The Pb lite can suppress these weak edges that are captured by the Canny baseline output. Another benefit of pb lite is that it considers color, brightness and texture information which is not possible in Canny and Sobel. Hence it can be concluded that Pb lite is better than Canny and Sobel. The gradients of the Color, Brightness and Texton map is affected by if we change the radius of half discs because it changes the chi square distance, but its effect is not much. In this implementation some of the features got suppressed that's why it is not like ground truth.

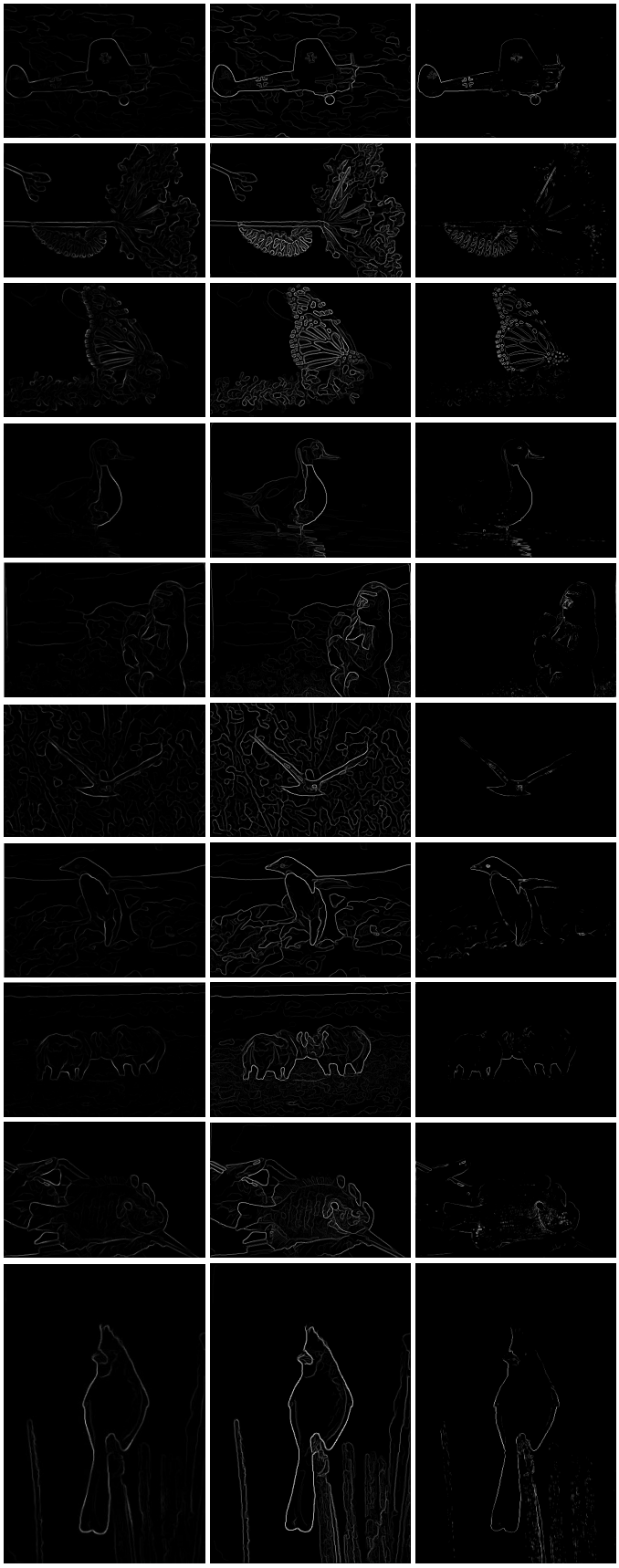

Fig. 7: PB-Lite, Canny, Sobel Output for Image 1- 10