Homework 0 - Alohomora
CMSC 733

Saurabh Palande (UID: 118133959)
Masters in Robotics Engineering
University of Maryland College Park
Email: spalande@umd.edu
Choice of phase: Phase 1 Shake my boundary

***Abstract*—This homework focuses on developing a simplified version of pb-lite boundary detection algorithm, which finds boundaries by examining brightness, color, and texture information across multiple scales (different sizes of objects/image) by using filter banks. The output of the algorithm will be a per-pixel probability of boundary. The simplified boundary detector significantly outperforms the well regarded Canny and Sobel edge detectors.**

I. Introduction

The pb-lite algorithm gives the per-pixel probability of the boundary in an image. The entire algorithm is divided into 4 steps which are as follows:-
1. Filter bank generation.
2. Texton, Brightness and Color Map generation
3. Texture, Brightness and Color gradient generation
4. Combining information from features with sobel and canny baselines.
The above mentioned steps are explained in detail in the following subsections.

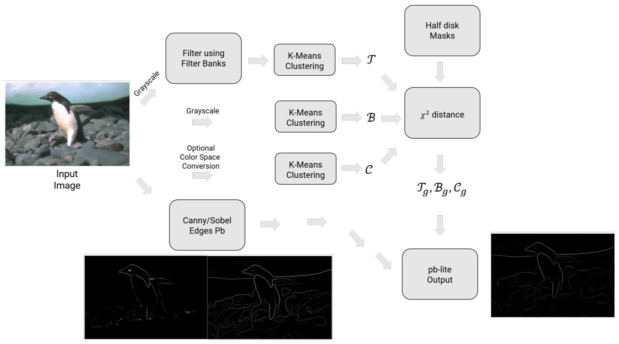


Fig. 1: Overview of pb-lite algorithm

A. Filter bank generation

The first step of the pb-lite boundary detection algorithm is to filter the image with different filter banks. The 3 different types of filter banks used in this homework are Oriented DoG filters, Leung-Malik Filters, and Gabor Filters.

1) Oriented DoG filters: Oriented DoG filters can be created by convolving a simple Sobel filter and a Gaussian kernel and then rotating the result. The oriented DoG filters for 2 scales() and 16 orientations(from 0 to 360 degrees) are shown in Figure 2.

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

$$Sobel filter = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2}$$

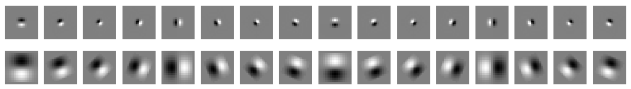


Fig. 2: Oriented DoG Filters

2) Leung-Malik filters: The Leung-Malik filters or LM filters are a set of multi scale, multi orientation filter bank with 48 filters. It consists of first and second order derivatives of Gaussians at 6 orientations and 3 scales making a total of 36; 8 Laplacian of Gaussian (LOG) filters; and 4 Gaussians. In this filter bank we implement two versions of LM filters. In LM Small (LMS), the filters occur at basic scales. The first and second derivative filters occur at the first three scales with an elongation factor of 3. The Gaussians occur at the four basic scales while the 8 LOG filters occur at sigma and 3-sigma. For LM Large (LML), the filters occur at the basic scales. Both LMS and LML are shown in Figure 3 and Figure 4 respectively.

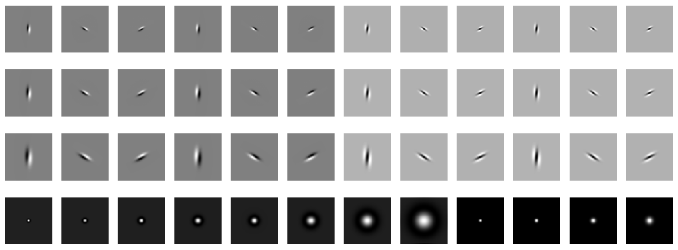


Fig. 3: LM Large filters

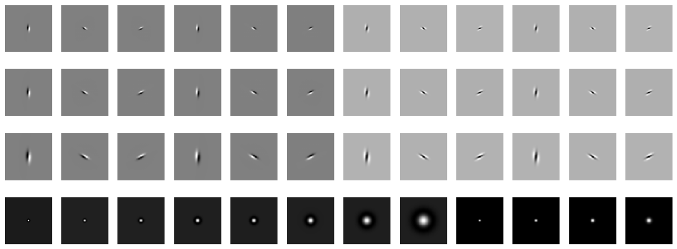


Fig. 4: LM Small filters

3) Gabor filters: A gabor filter is a gaussian kernel function modulated by a sinusoidal plane wave of a particular frequency and orientation.When a Gabor filter is applied to an image, it gives the highest response at edges and at points where texture changes. It has the following parameters:
λ - Wavelength of the sinusoidal component
θ-The orientation of the normal to the parallel stripes of Gabor function
ψ-The phase offset of the sinusoidal function
σ-The sigma/standard deviation of the Gaussian envelope.
γ - The spatial aspect ratio and specifies the ellipticity of the support of Gabor function
The effect of these parameters on the filter is explained in [2]. The gabor filters for 5 scales and 8 orientations are shown in the Figure 5.

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = exp(-\frac{\grave{x}^2 + \gamma^2 \grave{y}^2}{2\sigma^2})cos(2\pi\frac{\grave{x}}{\lambda} + \psi) \quad (3)$$

where,

$$\grave{x} = xcos\theta + ysin\theta \quad (4)$$

$$\grave{y} = -xsin\theta + ycos\theta \quad (5)$$

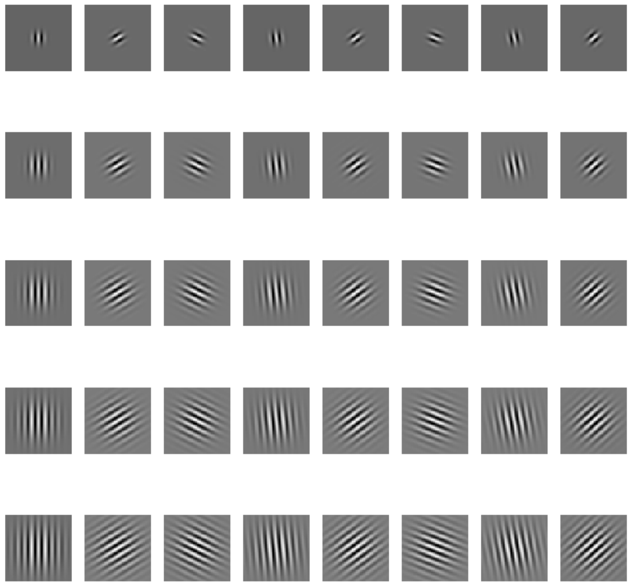


Fig. 5: Gabor Filters

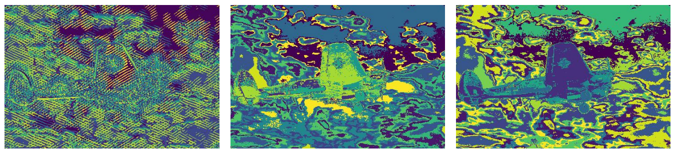
B. Texton, Brightness and Color map generation

After creating the filter banks, the next step is to generate the texton, brightness and color map.

1) Texton Map: The image is filtered using the filters from the filter bank which gives a vector of filtered responses on each pixel where the length of the vector is the number of filters in the filter bank. To simplify this representation we cluster the filter responses using K means clustering. After clustering, each pixel is represented by a discrete cluster ID and this process of dimensinality reduction is called vector quantization. The number of clusters chosen in this homework are 64.

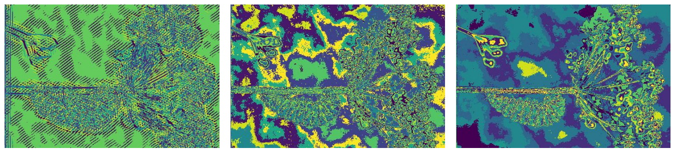
2) Brightness Map: The brightness map captures the brightness change in the image. The brightness values are clustered using K means clustering on the grayscale equivalent of the original image and the number of clusters chosen are 16.

3) Color Map: The color map captures the color changes or chrominance content in the image. Here, again we cluster the color values using kmeans clustering where the number of clusters chosen are 16.



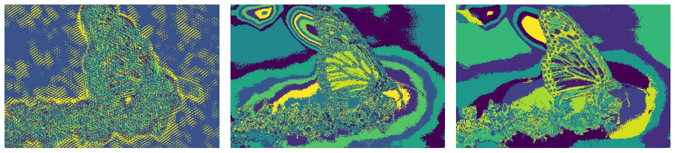
(a) Texton Map (b) Brightness map (c) Color map

Fig. 6: Texton, Brightness and Color map for image 1



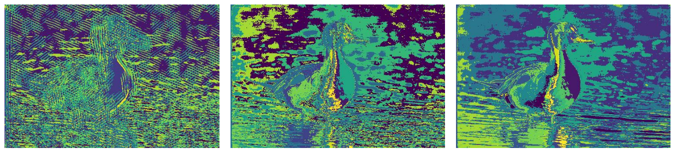
(a) Texton Map (b) Brightness map (c) Color map

Fig. 7: Texton, Brightness and Color map for image 2



(a) Texton Map (b) Brightness map (c) Color map

Fig. 8: Texton, Brightness and Color map for image 3



(a) Texton Map (b) Brightness map (c) Color map

Fig. 9: Texton, Brightness and Color map for image 4

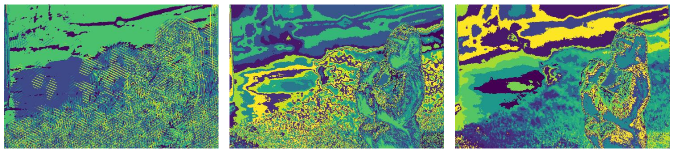

(a) Texton Map (b) Brightness map (c) Color map

Fig. 10: Texton, Brightness and Color map for image 5

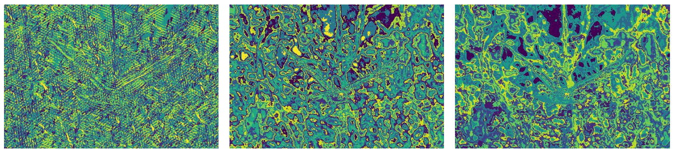

(a) Texton Map (b) Brightness map (c) Color map

Fig. 11: Texton, Brightness and Color map for image 6

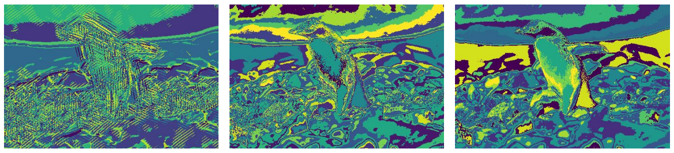

(a) Texton Map (b) Brightness map (c) Color map

Fig. 12: Texton, Brightness and Color map for image 7

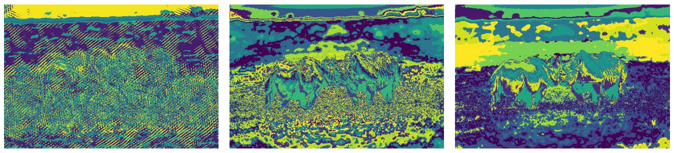

(a) Texton Map (b) Brightness map (c) Color map

Fig. 13: Texton, Brightness and Color map for image 8

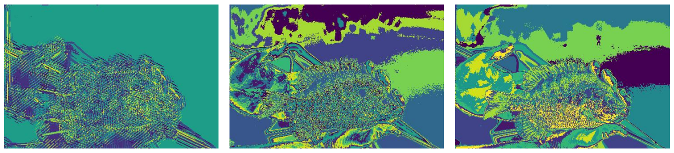

(a) Texton Map (b) Brightness map (c) Color map

Fig. 14: Texton, Brightness and Color map for image 9

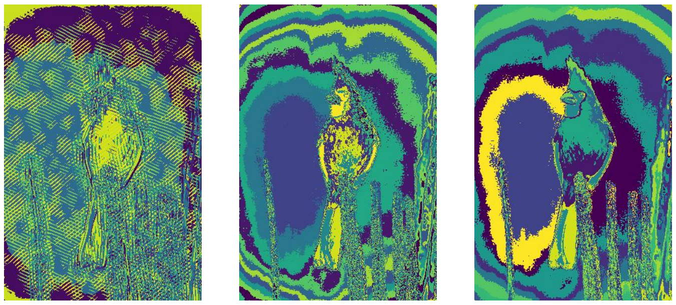

(a) Texton Map (b) Brightness map (c) Color map

Fig. 15: Texton, Brightness and Color map for image 10

C. Texton, Brightness and Color gradient generation

The texton, brightness and color gradient is generated by computing the differences of values across different shapes and sizes. To compute this difference half disk masks are used.The half-disc masks are pairs of binary images of half-discs. It allows us to compute the chi-square distances using a filtering operation, which is much faster than looping over each pixel neighborhood and aggregating counts for histograms. The half disc masks are shown in Figure 15. Texture,Brightness and Color gradient encode how much the texture, brightness and color distributions are changing at a pixel. They are computed by comparing the distributions in left/right half-disc pairs (opposing directions of filters at same scale, in Fig. 16)centered at a pixel. If the distributions in the image are the similar, the gradient is small and if the distributions are dissimilar, the gradient is large. The the Tg,Bg,Cg computed by comparing the distributions in left/right half-disc pairs along with the chi-square distances are shown for each image.

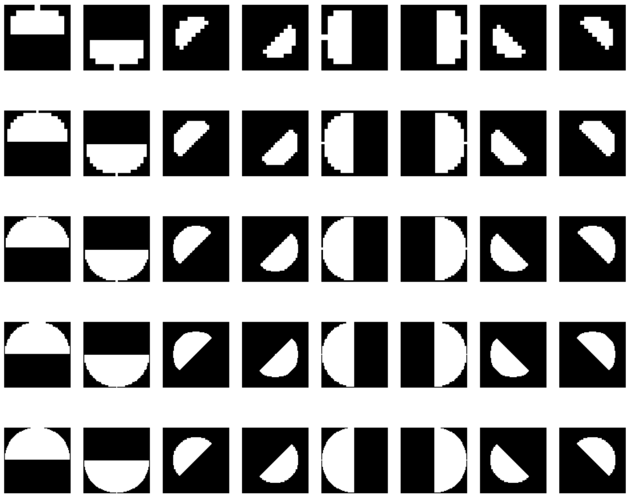


Fig. 16: Half Disc masks

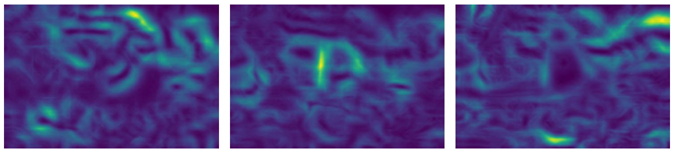

(a) Texture gradient (b) Brightness gradient (c) Color gradient

Fig. 17: Texture, Brightness and Color gradient for image 1

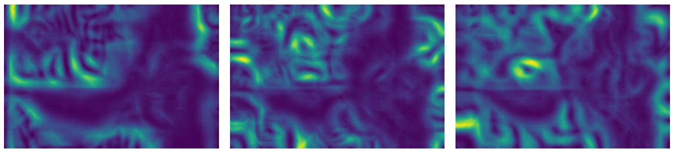

(a) Texture gradient (b) Brightness gradient (c) Color gradient

Fig. 18: Texture, Brightness and Color gradient for image 2

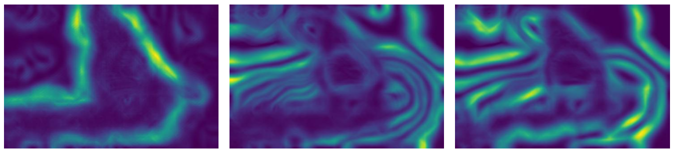

(a) Texture gradient (b) Brightness gradient (c) Color gradient

Fig. 19: Texture, Brightness and Color gradient for image 3

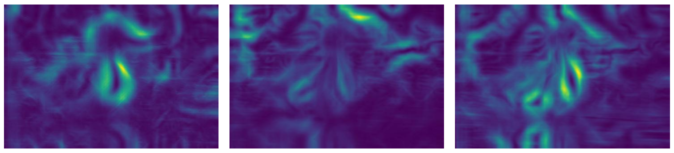

(a) Texture gradient (b) Brightness gradient (c) Color gradient

Fig. 20: Texture, Brightness and Color gradient for image 4

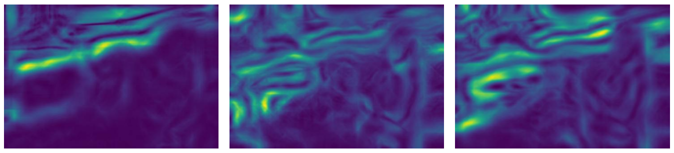

(a) Texture gradient (b) Brightness gradient (c) Color gradient

Fig. 21: Texture, Brightness and Color gradient for image 5

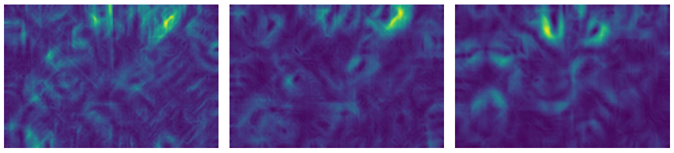

(a) Texture gradient (b) Brightness gradient (c) Color gradient

Fig. 22: Texture, Brightness and Color gradient for image 6

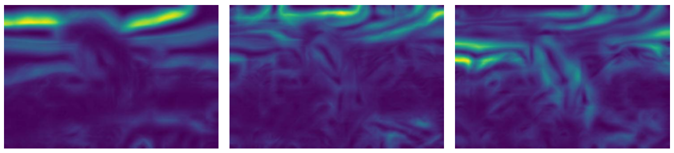

(a) Texture gradient (b) Brightness gradient (c) Color gradient

Fig. 23: Texture, Brightness and Color gradient for image 7

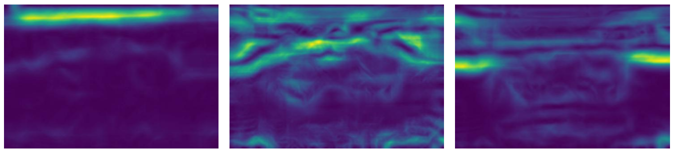

(a) Texture gradient (b) Brightness gradient (c) Color gradient

Fig. 24: Texture, Brightness and Color gradient for image 8

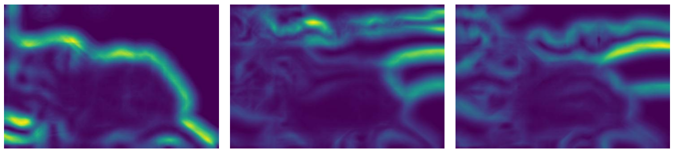

(a) Texture gradient (b) Brightness gradient (c) Color gradient

Fig. 25: Texture, Brightness and Color gradient for image 9

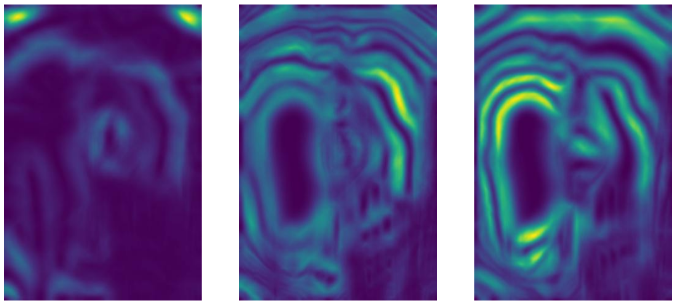

(a) Texture gradient (b) Brightness gradient (c) Color gradient

Fig. 26: Texture, Brightness and Color gradient for image 10

D. *Combining information from features with sobel and canny baselines*

In the final step of the pb-lite algorithm, the information from the texture, brightness and color gradient is combined with the Canny and Sobel baselines using the Hadamard product operator and choosing suitable weights. The results are shown for each image.

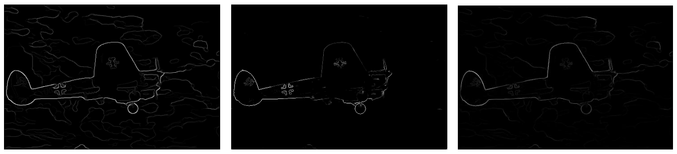

(a) Canny Baseline (b) Sobel Baseline (c) Pb-lite

Fig. 27: Canny, Sobel and Pb-lite result of image 1

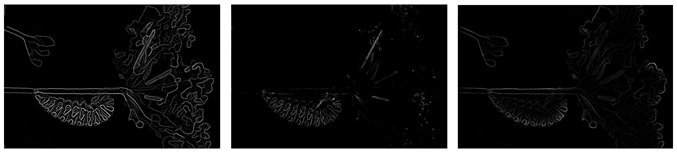

(a) Canny Baseline (b) Sobel Baseline (c) Pb-lite

Fig. 28: Canny, Sobel and Pb-lite result of image 2

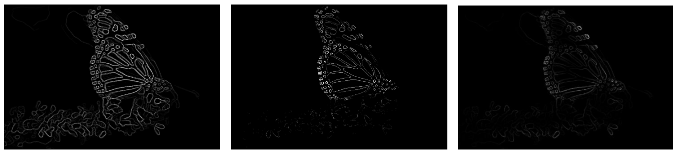

(a) Canny Baseline (b) Sobel Baseline (c) Pb-lite

Fig. 29: Canny, Sobel and Pb-lite result of image 3

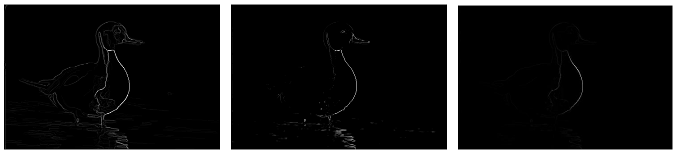

(a) Canny Baseline (b) Sobel Baseline (c) Pb-lite

Fig. 30: Canny, Sobel and Pb-lite result of image 4

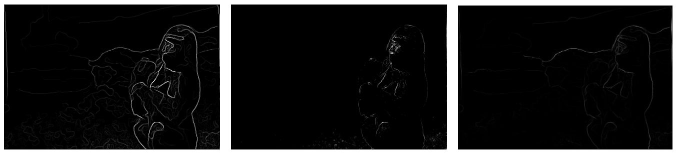

(a) Canny Baseline (b) Sobel Baseline (c) Pb-lite

Fig. 31: Canny, Sobel and Pb-lite result of image 5

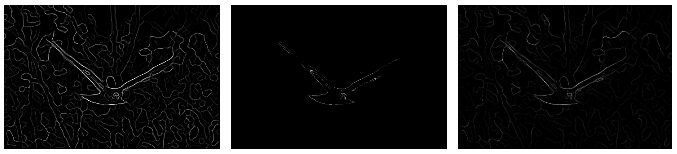

(a) Canny Baseline (b) Sobel Baseline (c) Pb-lite

Fig. 32: Canny, Sobel and Pb-lite result of image 6

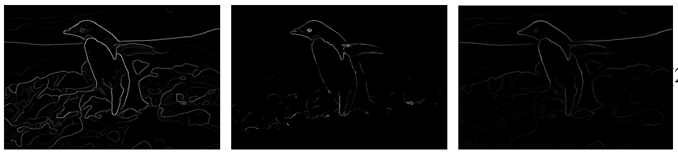

(a) Canny Baseline (b) Sobel Baseline (c) Pb-lite

Fig. 33: Canny, Sobel and Pb-lite result of image 7

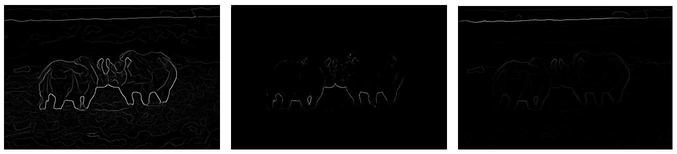

(a) Canny Baseline (b) Sobel Baseline (c) Pb-lite

Fig. 34: Canny, Sobel and Pb-lite result of image 8

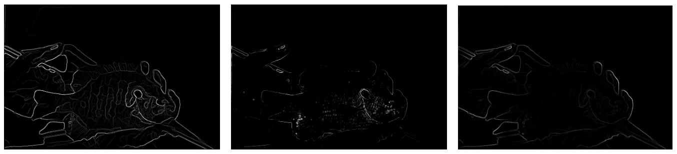

(a) Canny Baseline (b) Sobel Baseline (c) Pb-lite

Fig. 35: Canny, Sobel and Pb-lite result of image 9

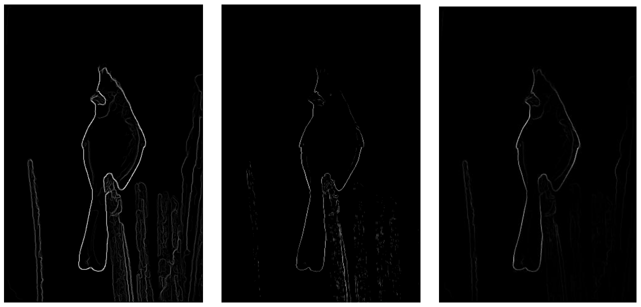

(a) Canny Baseline (b) Sobel Baseline (c) Pb-lite

Fig. 36: Canny, Sobel and Pb-lite result of image 10

Why Pb-lite is better than Canny and Sobel filters

The pb-lite algorithm performs very well to reduce the background noise and detect the edges of the main object in an image.The main step in pb-lite algorithm is to generate the texture, brightness and color gradients using the half disc masks. The more accurately the gradient change information is captured, the more accurate and clear the results are. It is observed that the edges detected by pb-lite are not as clearly visible as Canny and Sobel baselines. Possible solutions to improve the results could include:
1. Using more half- disc masks to capture the gradient change accurately.
2. Using a more effecint way of combining the features
3. Adjusting the weights of Canny and Sobel baselines

References

[1] https://www.cs.cornell.edu/courses/cs6670/2011sp/lectures/lec02$_{filter.pdf}$
2] https://medium.com/@anuj$_{s}hah/through - the - eyes - of - gabor - filter - 17d1fdb3ac97$