RBE-549: Homework0 - Probalistic Edge Detection

Saurabh Kashid
Wocester Polytechnic Institute
Robotics Engineering
Email: saurabhkashid157@gmail.com

***Abstract*—This report presents an implementation of a Probability of Boundary (Pb) edge detection pipeline based on the BSDS500 framework. Unlike classical edge detectors such as Sobel and Canny, this method incorporates high-level perceptual cues by combining texture (Tg), brightness (Bg), and color (Cg) gradients with baseline edge maps. We detail the implementation pipeline, demonstrate results, and compare against ground truth boundaries from the BSDS500 dataset..**

I. Introduction

Edge detection is a foundational task in computer vision. Traditional methods such as Sobel and Canny rely primarily on low-level intensity gradients, which often fail to detect edges defined by textural or chromatic changes.

The Pb edge detector [1] enhances boundary detection by incorporating higher-level features such as texture gradients (Tg), brightness gradients (Bg), and color gradients (Cg). In this project, we implemented a Pb edge detection pipeline and evaluated its performance on the BSDS500 dataset.

A. Filter Bank Generation

We used the following filter banks to extract texture and edge features:

- Derivative of Gaussian (DoG) filters
- Leung-Malik (LM) filters [2]
- Gabor filters

These filters respond to edges, textures, and patterns at multiple scales and orientations, mimicking aspects of human vision.

1) Derivative of Gaussian (DoG) Filters: DoG filters approximate the first derivative of a Gaussian function, and act as an edge detector. The 1D Gaussian is defined as:

$$G(x;\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

The derivative of Gaussian (DoG) is:

$$G'(x;\sigma) = -\frac{x}{\sigma^2} G(x;\sigma)$$

In 2D, applying this along the x-axis results in an oriented edge detector:

$$DoG(x,y) = G'(x;\sigma_x) \cdot G(y;\sigma_y)$$

DoG filters were used in multiple orientations to capture edge responses at various directions.

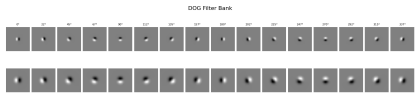

Fig. 1. Example Derivative of Gaussian (DoG) filter

2) Leung-Malik (LM) Filter Bank: The LM filter bank [2] is a multi-scale, multi-orientation set of filters designed to capture texture patterns. It consists of:

- First derivatives of Gaussian ($\partial G/\partial x$)
- Second derivatives of Gaussian ($\partial^2 G/\partial x^2$)
- Laplacian of Gaussian (LoG)
- Gaussian filters

The Laplacian of Gaussian is defined as:

$$LoG(x,y;\sigma) = \left(\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right) G(x,y;\sigma)$$

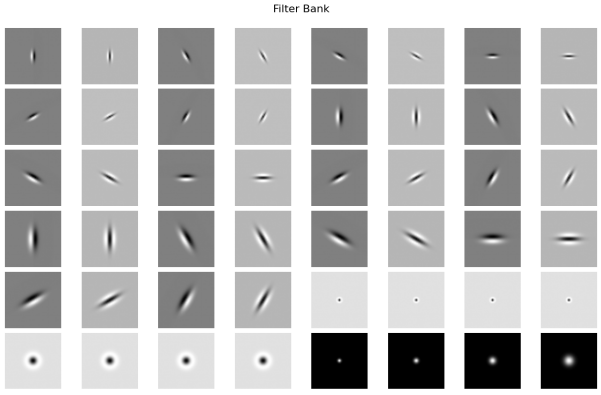

Fig. 2. Example Leung-Malik (LM) filter bank visualization

LM filters were applied at multiple scales and orientations to build the **texton map** used in Pb edge detection.

3) Gabor Filters: Gabor filters model **frequency- and orientation-selective** receptive fields in the visual cortex, and are defined as:

$$G(x,y) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\psi\right)$$

Where:

$$x' = x\cos\theta + y\sin\theta, \quad y' = -x\sin\theta + y\cos\theta$$

Parameters:

- λ: wavelength of the sinusoidal component
- σ: standard deviation of the Gaussian envelope
- θ: orientation
- γ: spatial aspect ratio
- ψ: phase offset

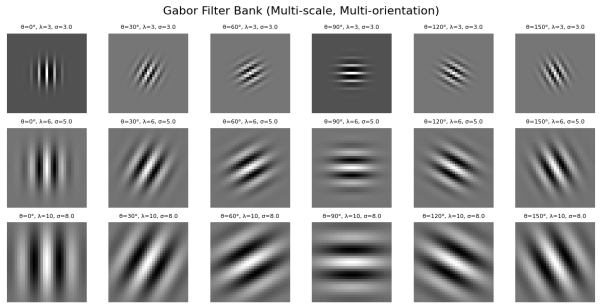

Fig. 3. Example Gabor filter bank visualization

Gabor filters are useful for modeling **texture and orientation-selective features**, and can complement DoG and LM responses.

B. Feature Maps

1) Texton Map (T): Filtering an input image with each element of your filter bank (you can have a lot of them from all the three filter banks you implemented) results in a vector of fillter responses centered on each pixel. For instance, if your filter bank has N filters, you'll have N filter responses at each pixel. A distribution of these N-dimensional filter responses could be thought of as encoding texture properties. We will simplify this representation by replacing each N-dimensional vector with a discrete texton ID. We will do this by clustering the filter responses at all pixels in the image in to K textons using kmeans. Each pixel is then represented by a one dimensional, discrete cluster ID instead of a vector of high-dimensional, real-valued filter responses (this process of dimensionality reduction from N to 1 is called "Vector Quantization"). This can be represented with a single channel image with values in the range of [1,2,3,. . . ,K]. K=64 is chosen for clustering.

2) Brightness Map (B): The brightness map is as simple as capturing the brightness changes in the image. Here, again we cluster the brightness values using kmeans clustering (grayscale equivalent of the color image) into a chosen number of clusters (k=16) clusters. We call the clustered output as the brightness map B .

3) Color Map (C): RGB images were converted to Lab space, and color vectors were clustered into $K = 16$ bins.

C. Texture, Brightness and Color Gradients Tg,Bg,Cg

To obtain Tg,Bg,Cg , we need to compute differences of values across different shapes and sizes. This is achieved very efficiently by the use of Half-disc masks.

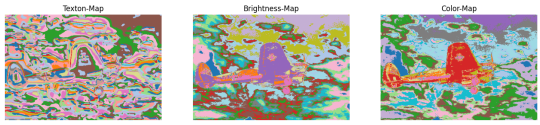

Fig. 4. Example Map of Image

D. Half-Disc Masks

We generated half-disc masks at multiple scales ($r = [4, 10, 20]$) and orientations ($N = 8$). These were used to compare histogram distributions between left and right halves around each pixel.

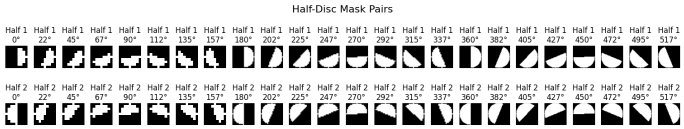

Fig. 5. Example of Half Disc

For each feature map, we computed the chi-square distance between half-disc histograms:

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

This produced gradient maps $Tg(x,y), Bg(x,y), Cg(x,y)$.

E. Final Pb Edge Computation

Feature gradients were combined with precomputed Canny and Sobel baselines to form the Pb edge map:

$$Pb(i,j) = \left(\frac{Tg + Bg + Cg}{3}\right) \times (0.5 \cdot Canny + 0.5 \cdot Sobel)$$

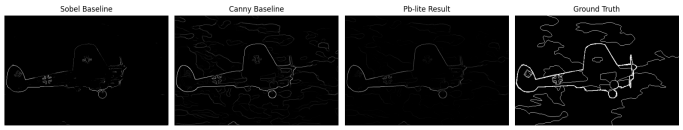

Fig. 6. Result of image 1

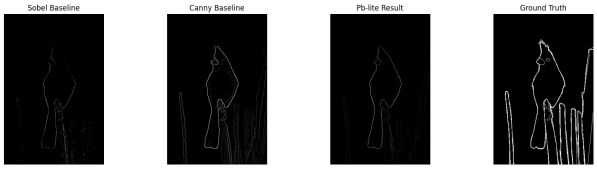

Fig. 7. Result of image 2

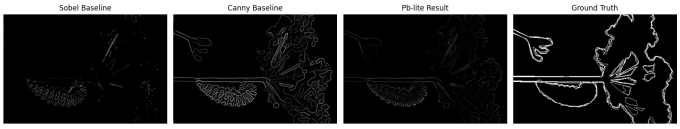

Fig. 8. Result of image 3

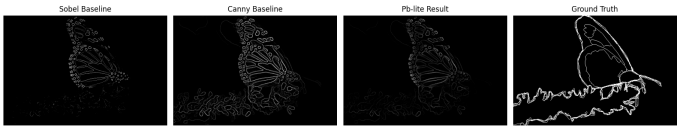

Fig. 9. Result of image 4

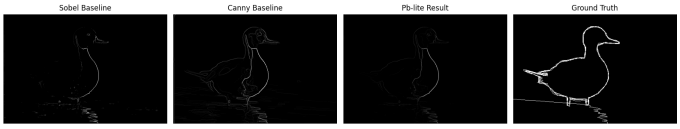

Fig. 10. Result of image 5

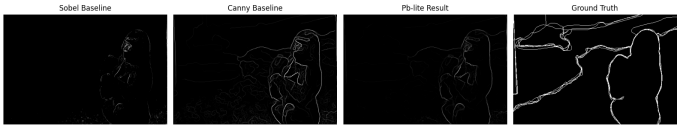

Fig. 11. Result of image 6

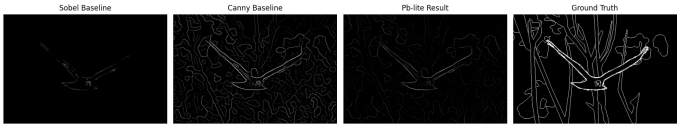

Fig. 12. Result of image 7

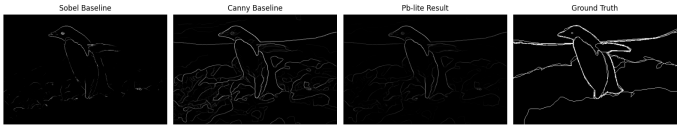

Fig. 13. Result of image 8

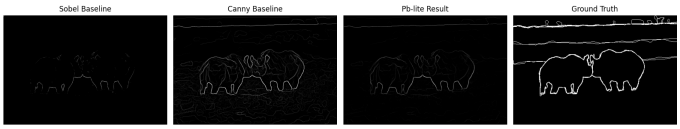

Fig. 14. Result of image 9

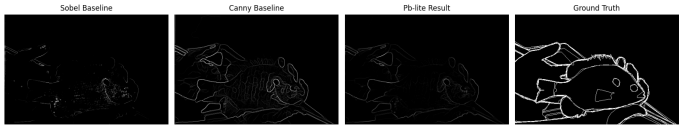

Fig. 15. Result of image 10

REFERENCES

[1] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[2] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 29–44, 2001.

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011.