HW0: Alohomora

Blake Bruell
Worcester Polytechnic Institute
Worcester, Massachusetts 01609
babruell@wpi.edu

***Abstract*—In this paper pb-lite edge detection algorithm is implemented during Phase 1. The math behind the derivative of Gaussian filters is discussed, and the method of creating a texton gradient is explained. In Phase 2 image classification networks are explored.**

I. Introduction

This paper is divided into two parts: Phase 1 and Phase 2. During Phase 1 the pb-lite edge detection algorithm is implemented. The implementation of pb-lite includes the derivation of different filter banks, and the creation of texton, brightness and color gradient maps. Finally, the method of combination is explained, as well as the key insights about the algorithm. During Phase 2 a simple model for image classification is created and trained, and some aspects of the model and training as discussed.

II. Phase 1: Pb-lite

Edge detection is a very common and widespread problem in computer vision, and of fundamental importance to the field. An edge detection algorithm's goal is to find discontinuities in an image, or edges. What exactly an edge is hard to define, with answers even varying between humans, and as such the problem is considered very challenging.

One of the simplest techniques for performing edge detection is the Sobel kernel. The horizontal and vertical Sobel kernels are the following:

$$\mathbf{S}_h = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \mathbf{S}_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

and are applied to the intensity map of an image, $\mathbf{A}$, to find the gradient magnitude at each pixel, $\mathbf{G}$:

$$\mathbf{G} = \sqrt{(\mathbf{S}_h * A)^2 + (\mathbf{S}_v * A)^2}$$

where $S*A$ is the convolution of A by the kernel S. $\mathbf{G}$ gives an idea of the edges in an image, but obviously only takes into account the changes in intensity in the image. Canny is a similar algorithm, but it takes into account the magnitude of the gradients of the nearby pixels. Canny obviously also suffers from the same shortcoming of only considering intensity.

The pb-lite algorithm is an attempt to address this shortcoming by also considering the *texture* of an image as well. This is accomplished by clustering pixels in an image (using an algorithm such as K-means) according some some feature set, and then considering the gradients of these clusters. Specially, pb-lite uses three feature sets: brightness (1 feature), color (3 features), and texture (number of features equal to the number of filters). These gradients are then combined with the Sobel and Canny edge detection algorithms according the pipeline shown in Fig. 1.

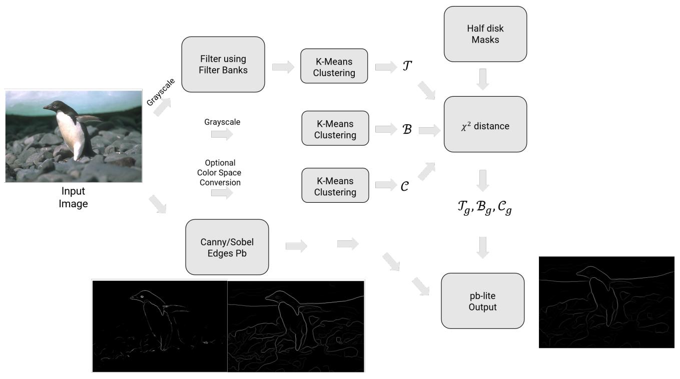


Fig. 1: Overview of pb-lite pipeline.

Much of this section will be discussing the last feature set, the *texture*. As previously hinted at, this feature set is defined by a set of filters, or a filters bank. There are many possibilities for the filter bank, and some of those possibilities are discussed in the following section.

A. Filter Banks

Each of the following filter banks is based on the 2-d Gaussian filter [1]:

$$G_{\sigma,\mu}(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Note that in this paper when filter banks are visualized, 0 is represented by gray, white is more positive, and black is more negative.

1) Oriented Derivative of Gaussian: This filter bank is defined by looking at the derivative of the Gaussian filter across a set of orientations. The derivative of the Gaussian in the x direction is equal to $-\frac{x}{\sigma^2}G_\sigma(x,y)$ [1]. This can the be rotated to an arbitrary angle, taking advantage of the Gaussian's circular symmetry to produce an arbitrary orientation, a trick that will be used repeatedly in the following sections. An example bank is as follows:

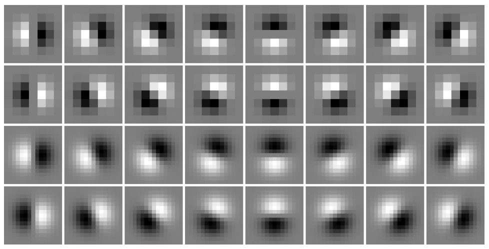


Fig. 2: Oriented DoG filter bank at 7 and 15 px sizes, and 16 orientations

2) Leung-Malik Filters: Leung-Malik filters are a combination of four different kinds of Gaussian filters: oriented first and second derivatives, Laplacian of Gaussian (LoG), and a normal Gaussian distribution. The function for the second order derivative of the Gaussian in the x direction is given by [1] (again other orientations can be found by simply rotating the result):

$$G''_\sigma(x,y) = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) G_\sigma(x,y)$$

and finally the Laplacian is given by [1]:

$$\nabla^2 G_\sigma(x,y) = \left(\frac{x^2+y^2}{\sigma^4} - \frac{2}{\sigma^2}\right) G_\sigma(x,y)$$

Leung-Malik contains the following filters: first and second order derivatives of Gaussians at 6 orientations and 3 scales, 8 scales of LoG, and 4 normal Gaussians.

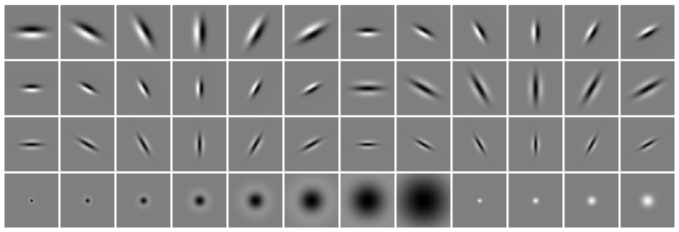


Fig. 3: Set of Leung-Malik filters

3) Gabor Filters: The last type of filter that is considered is the Gabor filter. Gabor filters are simply the Gaussian filter modulated by a sinusoidal plane wave (again, simply rotate result to get different orientations):

$$Gb_\sigma(x,y) = \sin(x) G_\sigma(x,y)$$

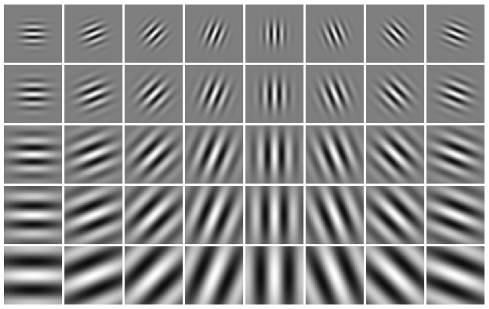


Fig. 4: Set of Gabor filters at 5 scales and 8 orientations

B. Texton Map and Gradient

Now that we have some candidate filter banks, lets consider the texton map of an image. For the following sections Fig. 5 will be used as the input image.

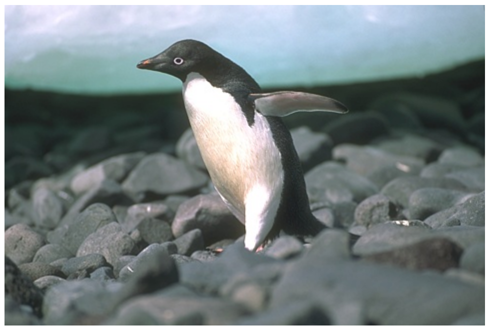


Fig. 5: Penguin image used as an input

The first step of calculating the texton map is converting the image ($N \times M \times 3$) to a brightness map ($\mathcal{B} : N \times M$), simply done by converting the image to grayscale. Then each filter in a filter bank (F filters) is convolved with $\mathcal{B}$ resulting in a texton map $\mathcal{T}$ of dimension $N \times M \times F$. At this point we have the texton of the image, which is visualized using the mean across the F dimension in Fig. 6

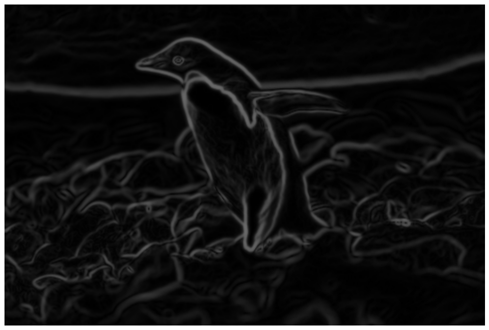


Fig. 6: Mean of texton map

This map is then clustered using KMeans into $K = 64$ clusters ($N \times M \times F \rightarrow N \times M$), reducing the high dimensional filter data to 1 dimension. This yields a single channel image in which each pixel has a value between $1, 2, \ldots, K$, shown in Fig. 7 using a jet color map, and this is the texton map $\mathcal{T}$

The final step of this process is to take a pair of half-disk masks to compute the gradient of each pixel using the chi-squared metric between the two halfs for each given cluster. This is performed for a whole bank of half-disk pairs (Fig. 8), and averaged. This yields $\mathcal{T}_g$ which has dimensions $N \times M \times K$, which must finally be condensed to $N \times M$. This is achieved by simply taking the mean across the K dimension. The output of this process for Fig. 7 is shown in Fig. 9.

C. Brightness and Color Gradients

The exact same process as discussed for the texton Graident is performed on two other feature sets, color ($N \times M \times 3$) and

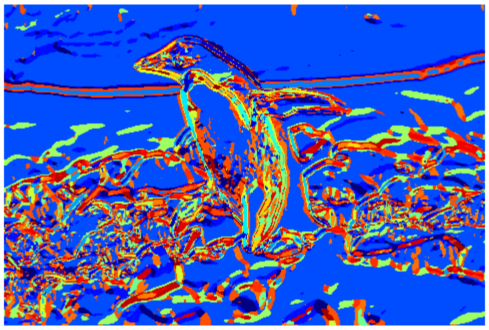


Fig. 7: Texton Map $\mathcal{T}$

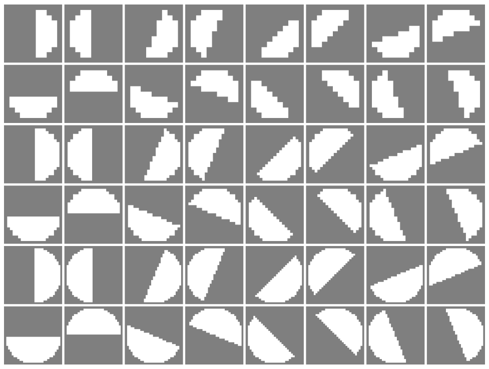


Fig. 8: Half-disk Masks

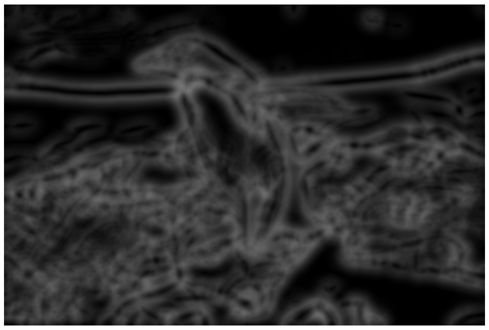


Fig. 9: Texton map gradient $\mathcal{T}_g$

brightness ($N \times M$), but with $K = 16$. The results are shown in Fig. 10.

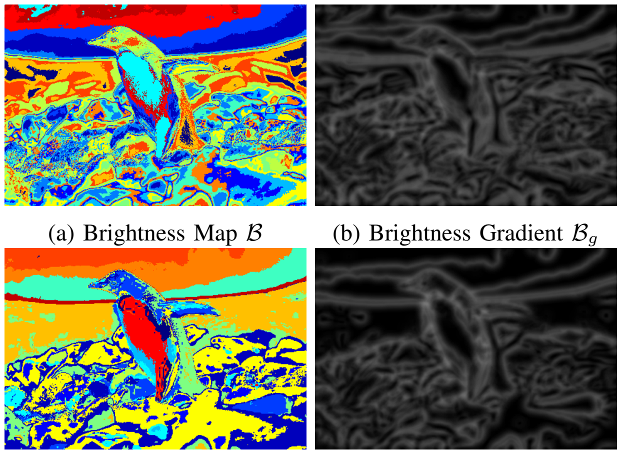
D. Putting It All Together

The final pb-lite output is defined by taking the mean of the gradients, a weighted average of the Canny and Sobel baselines, and performing piece-wise multiplication, as follows:

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

The mean of the gradients, the Canny and Sobel baselines, and the final pb-lite result are shown in Fig. 11.

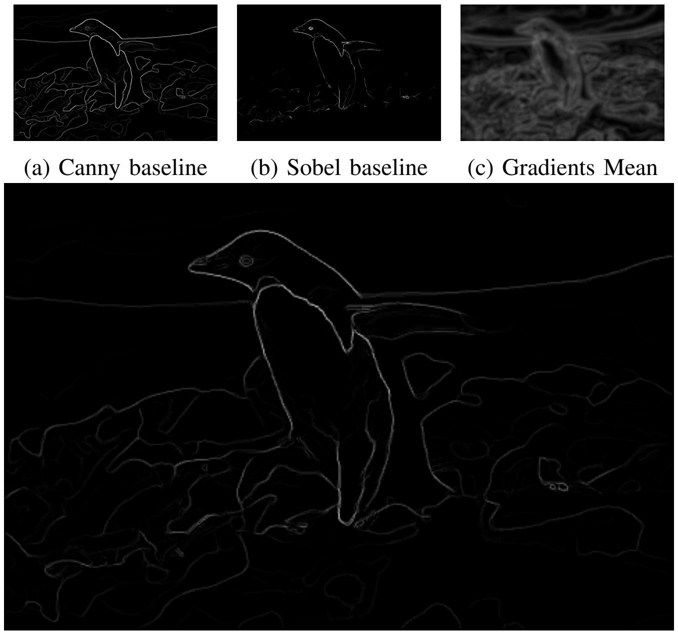
To understand the advantage of pblite, consider the example input image. The Canny baseline has too much detail, and the Sobel too little. This, as previously mentioned, is a result of the fact that these algorithms only consider brightness gradients, and not texture and texture gradients. Pblite directly addresses this by calculating the three gradients of clusters (texton, color, and brightness) and including these in the final result. By looking at the gradients of the *clusters* instead of the texton/brightness/color directly, pb-lite is considering regions of similar texture, and changes across these regions. As can be seen by looking at the equation for $PbEdges$ the textural aspects of the image modify the brightness by weighting the results of the Sobel and Canny baselines, not adding new edges.



(a) Brightness Map $\mathcal{B}$ (b) Brightness Gradient $\mathcal{B}_g$

(c) Color Map $\mathcal{C}$ (d) Color Gradient $\mathcal{C}_g$

Fig. 10: Gradients for color and brightness



(a) Canny baseline (b) Sobel baseline (c) Gradients Mean

(d) Final pb-lite result

Fig. 11: Final step of pb-lite pipeline

E. Results for 10 Example Images

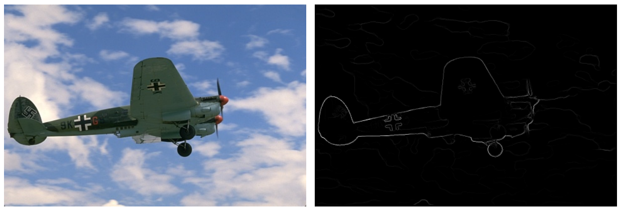


Fig. 12: Image 1 and pblite result

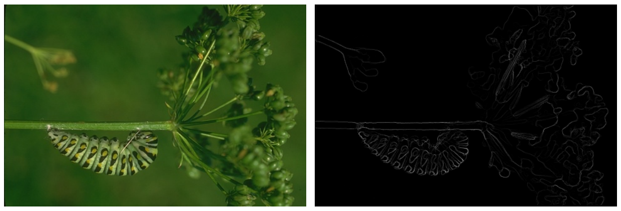


Fig. 13: Image 2 and pblite result

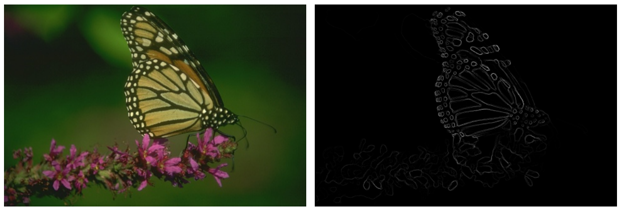


Fig. 14: Image 3 and pblite result

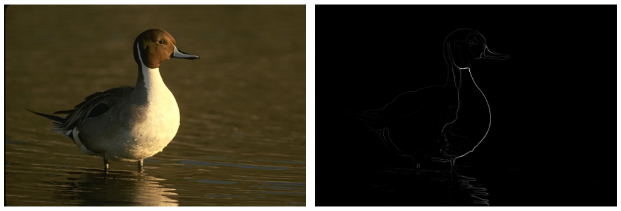


Fig. 15: Image 4 and pblite result

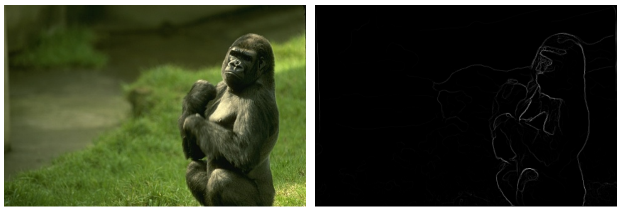


Fig. 16: Image 5 and pblite result

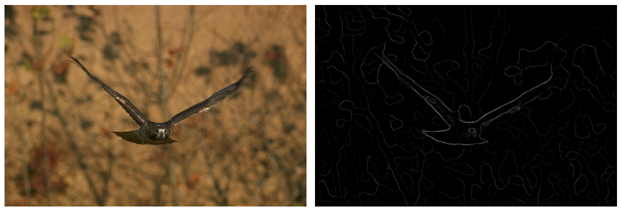


Fig. 17: Image 6 and pblite result

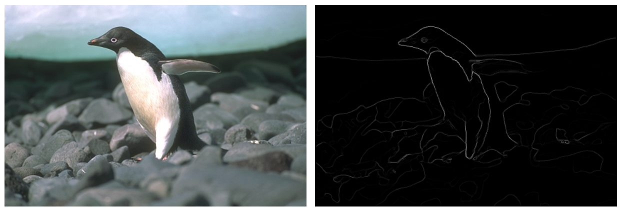


Fig. 18: Image 7 and pblite result

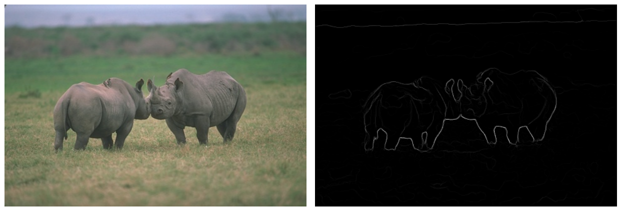


Fig. 19: Image 8 and pblite result

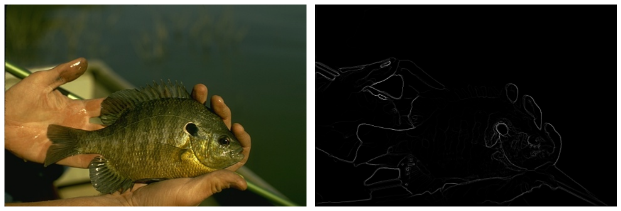


Fig. 20: Image 9 and pblite result

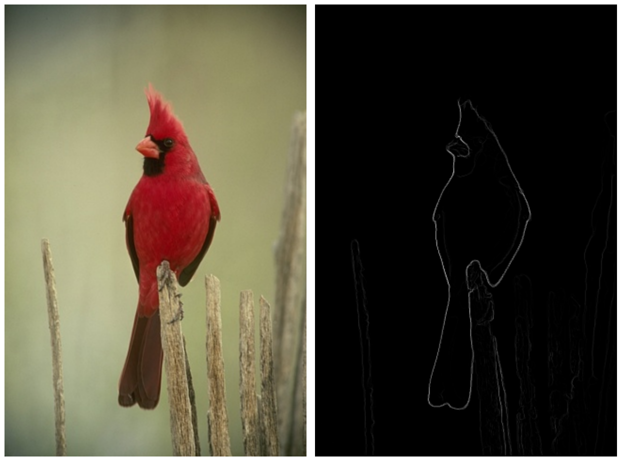


Fig. 21: Image 10 and pblite result

III. Phase 2: Image Classification

In this section of the paper a set of neural networks are implemented and tested on the CIFAR10 dataset.

A. Baseline Model

A baseline model of 10 parameters was tested, with a simple model of convolution, relu, pool, convolution, relu, pool, flatten, and two final relu layers (Fig. 22). The loss per epoch is shown in Fig. 23, and training accuracy per epoch is shown in Fig. 24.

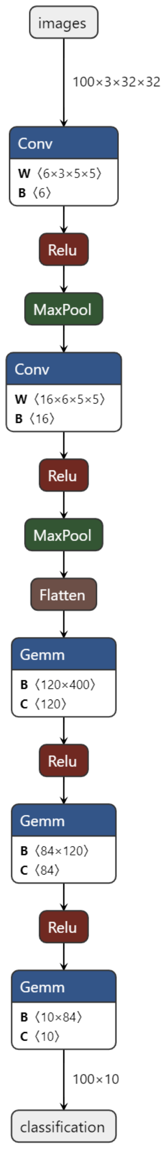


Fig. 22: Model Architecture

AdamW was chosen as the optimizer, with a learning rate of .001, weight decay of 0, epsilon of 1×10^{-8}, beta 1 of .9 and beta 2 of .999. Mini-batches of size 100 were chosen to strike a balance between speed of training and accuracy.

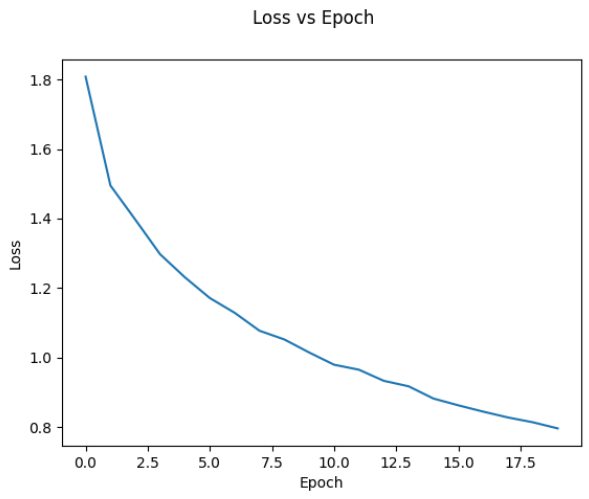


Fig. 23: Loss Per Epoch

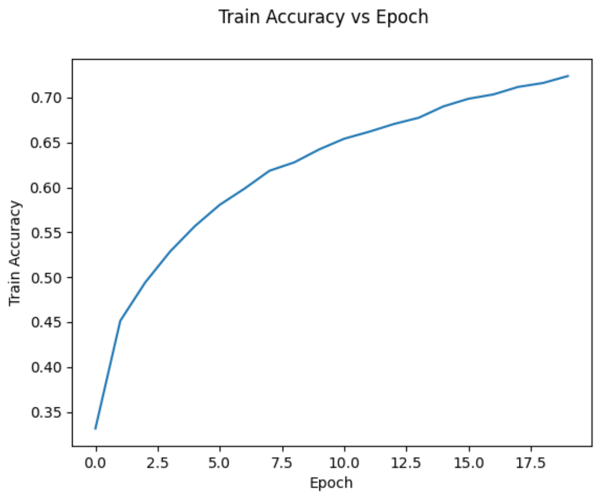


Fig. 24: Training Accuracy Per Epoch

IV. Conclusion

In Phase 1 pb-lite was successfully implemented, and its advantages over Canny and Sobel were explored and explained. The image classification networks in Phase 2 were not fully implemented or tested, but a baseline model was created and trained. The next steps for this paper would be to explore more sophisticated models, such as ResNet, or ResNetX.

References

[1] J. Maucher. “Gaussian filter and derivatives of gaussian.” (Jan. 2021), [Online]. Available: https://hannibunny.github.io/orbook/preprocessing/04gaussianDerivatives.html.