Lab: Edge Detection

Table of Contents

Authors: Renn Jervis (3762) Jason Liu (4053)

Introduction

In this lab we will be exploring methods for detecting and displaying edges in an image. We will use the Gaussian and its derivative to create gradient coponents and display this gradient in various representations, including its magnitude as well as a weighted directional gradient image. We will also examine the effects of increasing scales of the Gaussian and generating multple images. We will aslo explore the effect of imposing a threshold on the effectiveness of edge detection.

Exploring Gradient Components

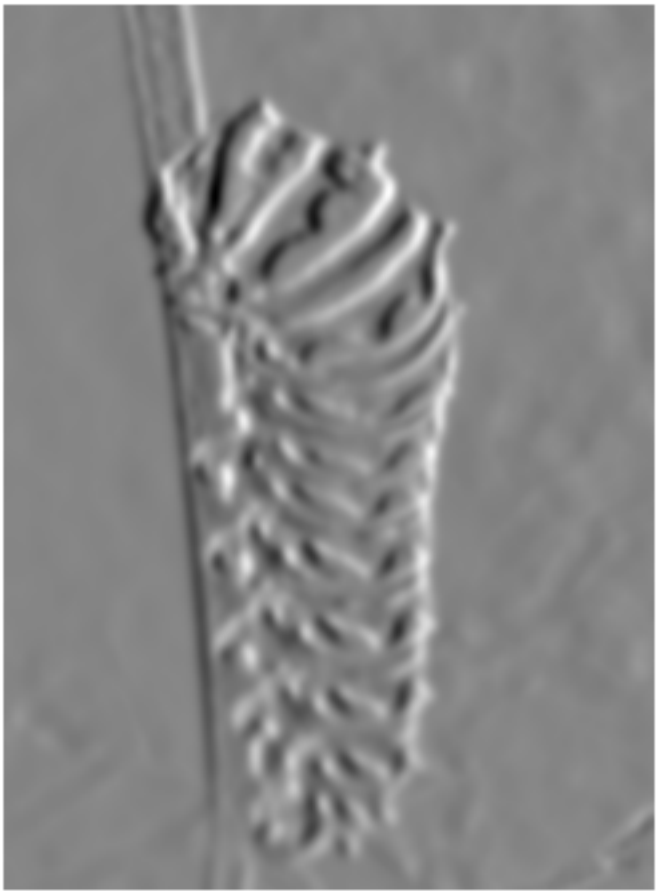
The gradient of a function gives us a vector containing the magnitude of the change in that function oriented in the direction of greatest change. We can create the gradient of an image by computing and combining its partial derivatives, and in this section we examine these partials.

Our quest begins with a caterpillar:

Original Image

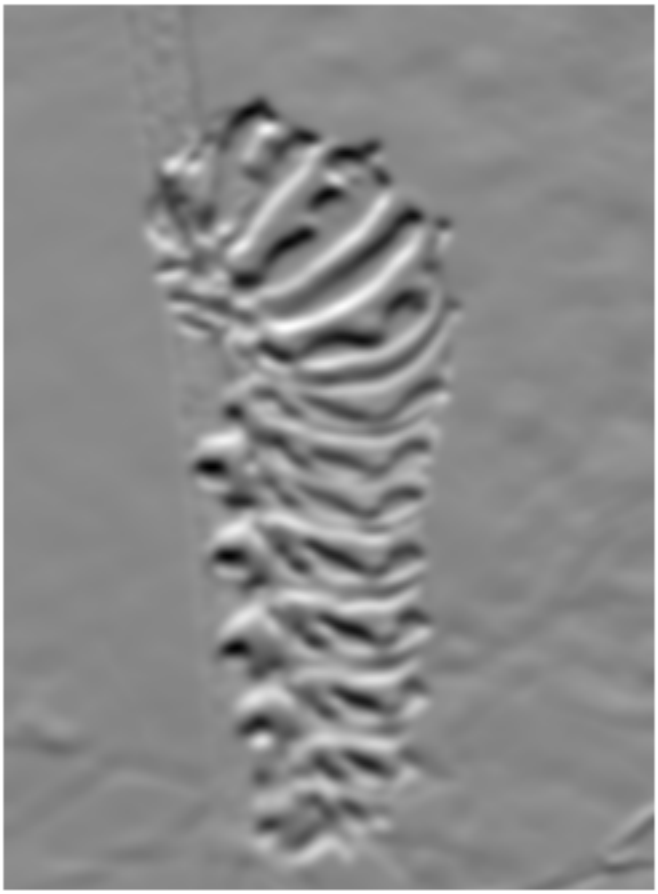
We begin by creating a one-dimentional Gaussian kernel with variance of 4 and the corresponding one-dimensional derivative of the Gaussian so that we may use seperable filtering to create two images, the first with the derivative along the rows, and the second with the derivative along the columns. We will accomplish this by using a separable filter with the one-dimentional Gaussian as the filter along the columns and the Gaussian derivative along the rows. The row derivative image will show us the areas where we transition sharply between pixel intensities along a row, thus indicating a vertical edge. For this reason we see gray portions of the image where no vertical edges have been detected, white portions where edges transition from dark to light, and dark portions where there is an edge moving from light to dark values.

Row Derivative Image



In the case of the column derivative below--which detects horozontal edges--we see a significantly smaller number of verical edges (notice the plant stem on which the bug stands). The stripes of the bug in this image are easier to see, which is reasonable as its stripes are mostly oriented horizontally. Again we see gray portions of the image where there are few or no horozontal edges, white portions where edges move from dark and light, and dark portions where edges transition light to dark.

Column Derivative Image



Calculate and Examine the Magnitude of the Gradient

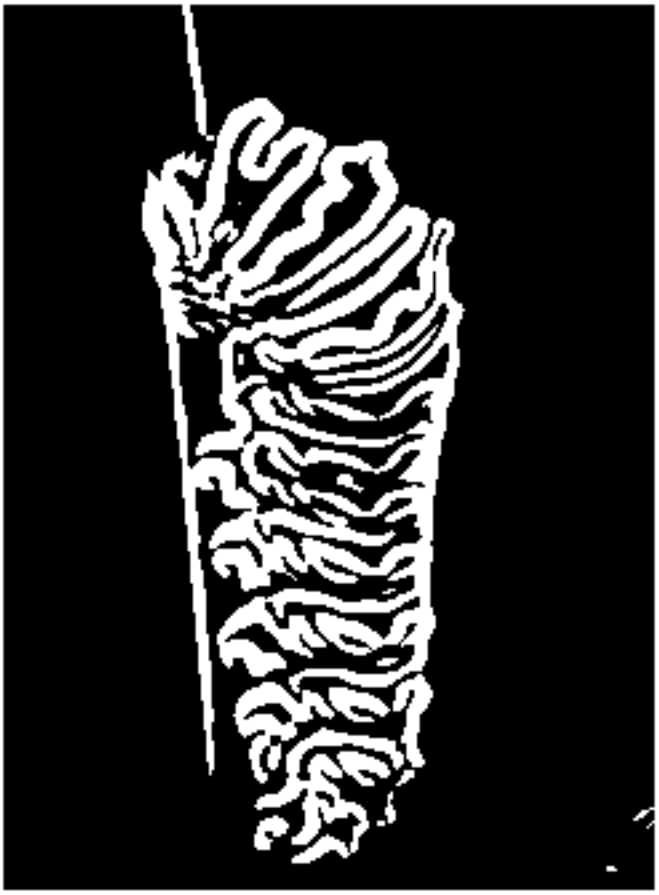
In calculating the derivative across the rows and then across the columns of the image, we have computed the two partial derivatives, or the components of the gradient of this image. Now we want to inspect the magnitude of this gradient, and see if it can reveal more about the edges present in our image.

Magnitude of Gradient

The strongest responses in this image appear where the original image has a strong edge oriented in any direction. This is to be expected as the magnitude of the gradient by definition shows us the strength of the change in the direction of greatest change. Thus, the magnitude of the gradient shows us a combination of edges detected by the two partial derivatives, and this produces a more complete view of the edges in the image. Edges appear white or light in the gradient magnitude image, and these edges correspond to edges that appear in either of the partial derivative images.

To reduce noise, it is common to place a threshold on the gradient magnitude image so that we can produce an edge detection result that is binary. We created a nice representation of the edges of the image by utilizing a threshold of .03.

Thresholded Gradient Magnitude Image

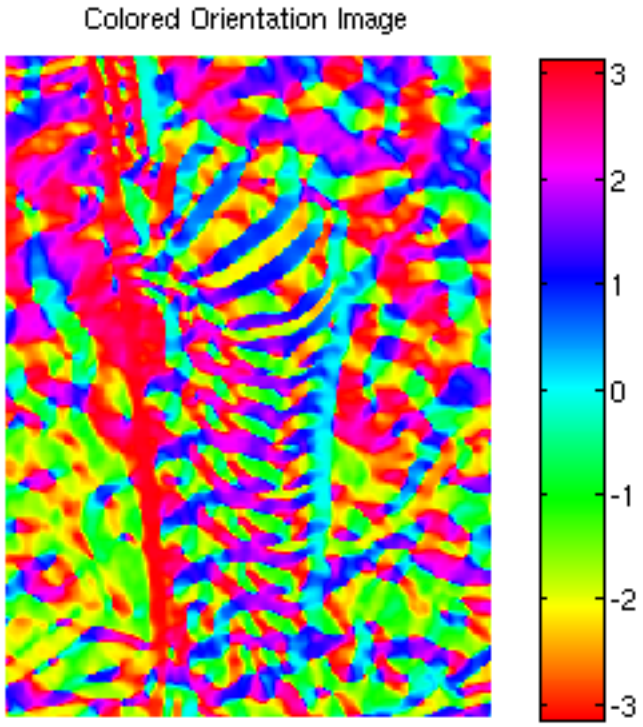


Ways to Present Gradient Orientation

If we seek to examine the image further, we can consider looking at the magnitude and the direction of the gradient at each pixel. We create this image using our row and column derivative images and the atan2 function which takes the inverse tangent at each pixel location. This orientation image is in the range of -pi to pi and so we treat -pi as black and pi as white to create a more comprehensible image.

Black and White Orientation Image

This black and white image is rather hard to decipher and so to refine further we apply a colormap to the image to make it easier to read.

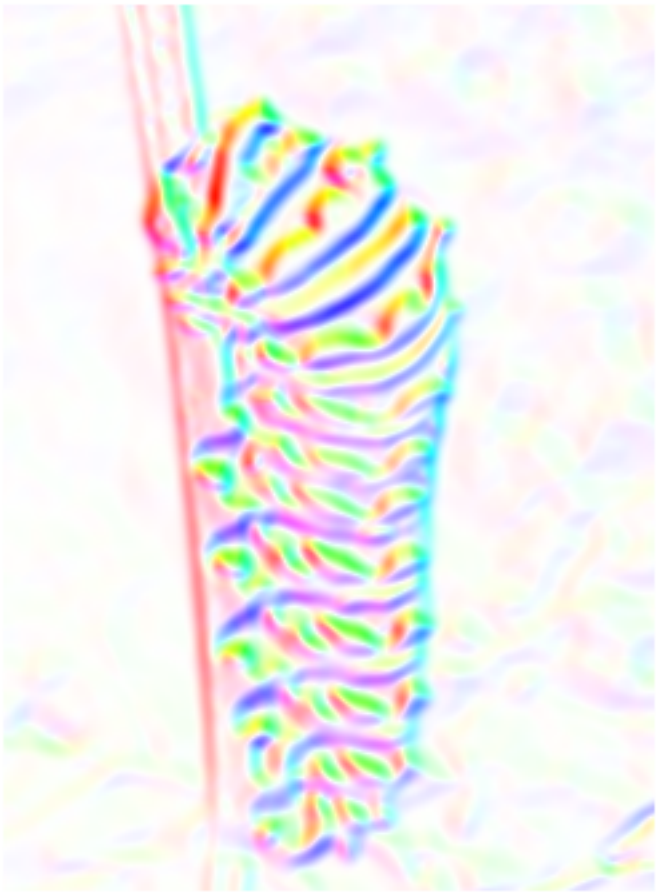


In the figure, the teal pixels represent a horizontal gradient direction left to right, and thus appear on vertical edges that move from dark to light, and the red pixels represent a horizontal gradient direction from right to left, and thus appear on vertical edges that move light to dark. The darker blue pixels represent a diagonal gradient direction to bottom right, and represent edges perpendicular to that diagonal, similarly the green pixels represent a diagonal gradient direction to top right, and so appear on edges oriented to top left. The yellow pixels represent a diagonal gradient direction to top left and show up on edges oriented to top right (like the first few stripes behind the caterpillar's head). Finally, the pink pixels represent a horozontal gradient direction to the bottom left. The pixels between pink and blue represent a vertical gradient direction to the bottom, and the pixels between green and yellow represent a vertical direction to the top. Purple and yellow-green pixels appear on horozontal edges (although there are few truly horozontal edges in this image). The colors reconcile with the image contents because they show the appropriate edges where expected, except that the background of the image, which appears on inspection of the original to have few obvious edges, is strewn with pixels of all colors. This is perhaps explained if we cosider the image of the magnitude of the gradient, in which the background noise has resolved into a pattern with many twist and turns, ie. with many disparate edge orientations.

Gradient Orientation Revisited

That fact that the background of our colored image is just as bright as all of the other portions of the image seems counterintuitive, as there appear to be no edges in that region of the original. In this section we take a different approach to inspecting the gradient orientation by separating our representation into components. We will still produce a color image, but one that uses saturation to determine the strenght of an edge, eliminating colored pixels where no edges are present.

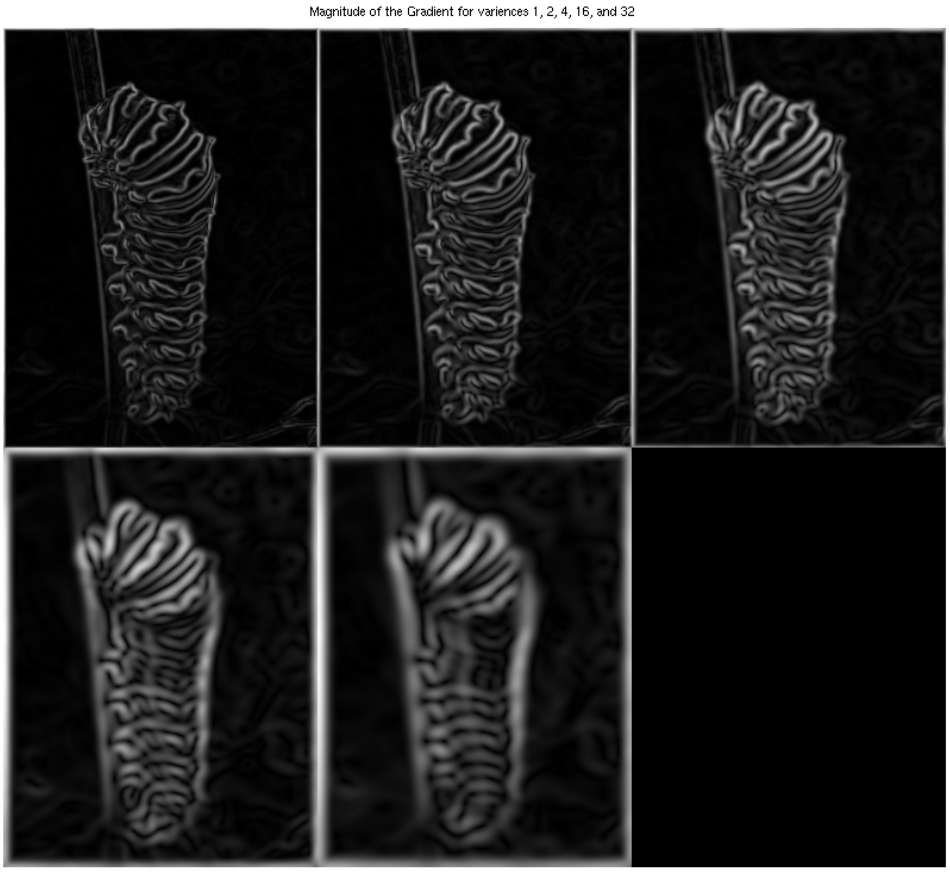
First we create an image to represent the hue, or the pure chroma, of the image by rescaling our orientation image in the range [0, 1]; then we create a representation of the saturation by rescaling the maximum value to 1. Finally, we create an value matrix of all ones, concatenate these three images into one, and convert to an RGB image.



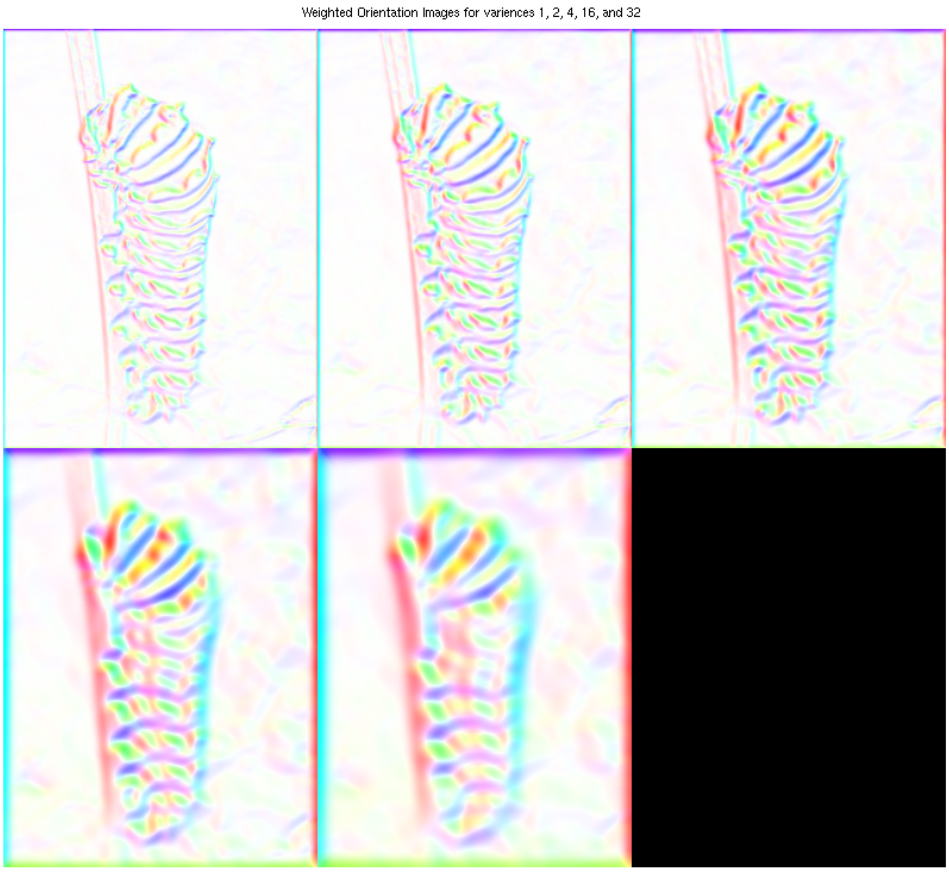
Detecting Edges and Analyzing the Effect of Scale on Edges

As we have now throughly examined the process of creating a gradient image, we will generate some more gradients and see how the scale of the Gaussian that we use affects the output.

We create an gradient magnitude image by the same process as before, and make spearate image for Gaussian variences of 1, 2, 4, 16, and 32 and will now diplay the various images of the magnitude of the gradient, with increasing scale.



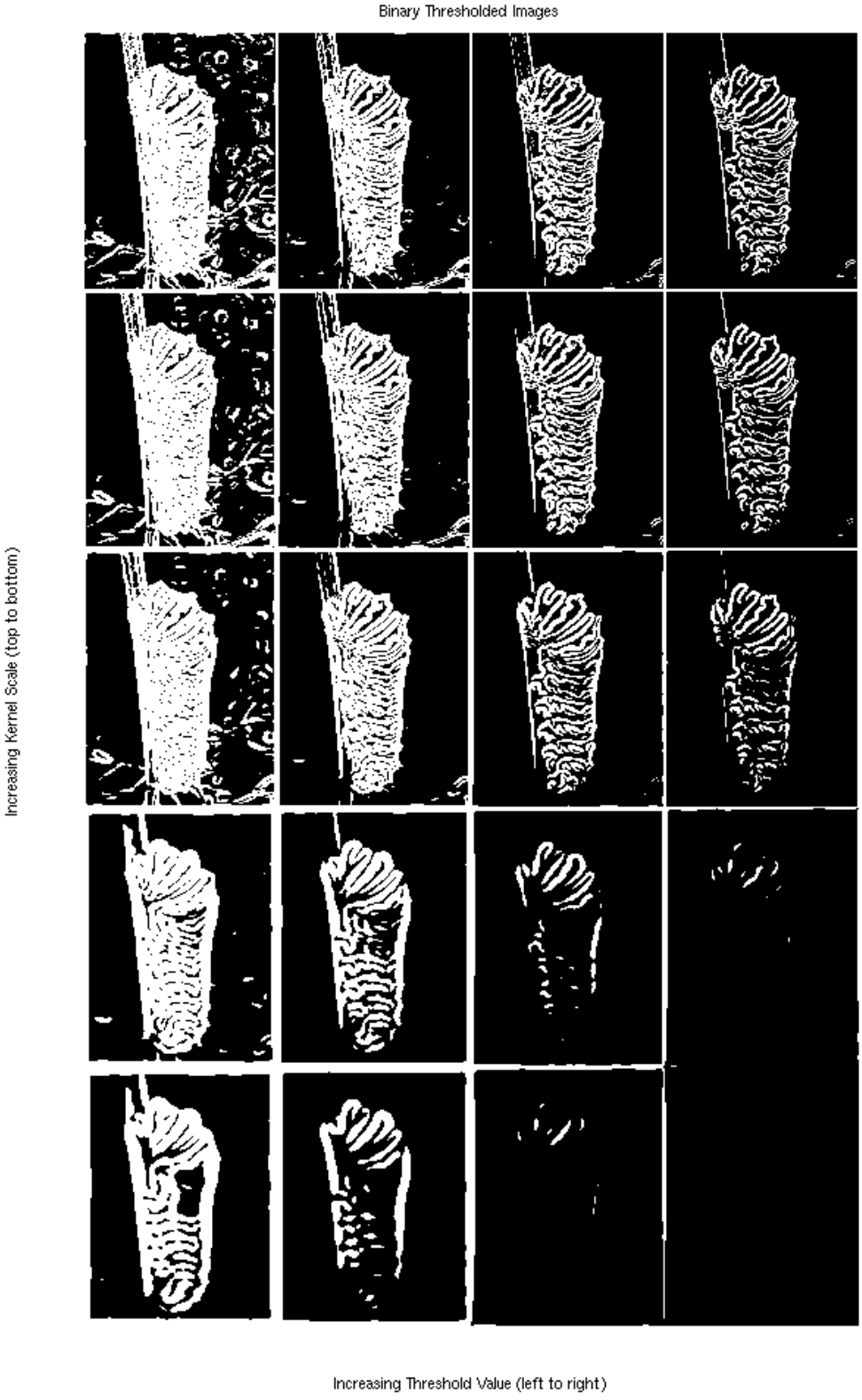
We notice the results of the gradient magnitude shows that the larger our scale gets, the larger our edges become. We also loose some edges and the background noise in the image appears to increase as the scale increases. The images with higher variances appear to be blurred, with the white pixels indicating an edge spread out over a large region.



The orientation images loose small details as out scale goes up. We notice especially that colors begin to appear around the edges of the picture itself, most likely because of the artifacts that appear around the edges of an image when we take a convolution. The colored regions become more smeared out and cover a larger area.

For each of the Gaussian scales indicated above, we test the effectiveness of using thresholds 2/256, 4/256, 8/256, and 12/256 to produce an array of binary threshold images.

```
Warning: Image is too big to fit on screen; displaying at 50%
```



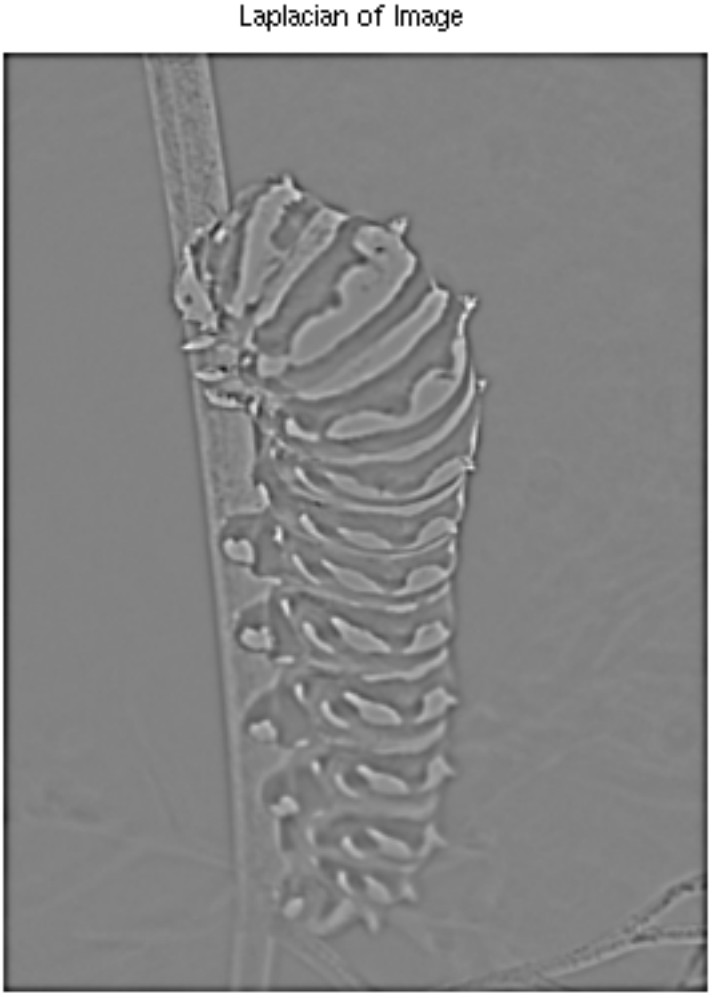
For the thresholded images, we observe that the smaller thresholds detect more edges, and so the left-hand image in each row has the most number of white pixels and the most number of false edges or noise. As we move to the right across a row, our edges become more detailed, thinner, and more easily distinguishable.

In the case of the top row the far right image appears to be the most closely approximating the edges in the original image, with a few noisy spots in the background and some edges running into eachother. As we increase our Gaussian scale (moving top to bottom in the image) we see that large-scale filters--especially with small threshold values--display mostly image noise and not a recognizable image. As we increase the size of the kernel the edges become wider, but if we put a larger threshold on high scales we can at least recover some semblance of form in the image.

Extra: Thinning Edges

The edges that we get in our thresholded images could be thinned by Szeliski's method of using the Laplacian to compute the true point at which there is an edge. The Laplacian is the sum of the second derivatives, and so a zero crossing indicates a point where the image transitions from darker to lighter, which in our case would indicate an edge!

We use the second derivative of the Gaussian with a variance of 4 to create second derivatives along row and column, and then add them together to get the Laplacian.



Conclusion

In this lab we considered different approaches to correctly identifying edges in an image. We began by experimenting with the gradient and its components and saw that we could create different representations of the gradient that could be interpreted differently. Our initial colored gradient image was rather hard to unpack, and tended to display colors where no edges were present. We reconciled this by creating a new representation of the orientation of the gradient that only showed color where an edge was present. We found this result to be the most readable gradient image that we produced, as it contains information on both edge orientation and strength. We then experimented with different sized Gaussian kernels and the effect that the scale had on edge detection, and concluded that as the size of our kernel increases, we generally loose detail in the image and are able to detect fewer reliable edges. We also found thresholding the images to create a binary edge image and refining this threshold is an good way to get more detailed edges, even at large scales.

Acknowlegements

The bug image was provided by Jerod Weinman, who is also the photagrapher. Copyrite 2007.

Published with MATLAB® R2013a