Detecting Edges using Linear Spatial Filtering

Michael Liu and Elika Ketabchi

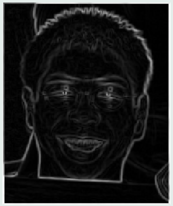
December 17, 2005

Abstract

The purpose of this project is to show and explain how edge detection works by the use of linear spatial filtering and Gradient Operators.

(a) Michael now

(b) Elika now



(c) Michael after edge detection

(d) Elika after edge detection

Home Page
Title Page
Page 1 of 26
Go Back
Full Screen
Close
Quit

1. Introduction

Edges are one of the gray-level discontinuities in a digital image, an element of image segmentation. Segmentation breaks an image into its basic parts or elements and the level to which segmenting is carried depends on the problem being solved. That is, segmentation should stop when the objects that we are analyzing have been isolated. For example, in automobile inspection of assemblies, the purpose of image segmentation is to identify defects and imperfections, such as missing parts. There is no point segmenting the image of the assembly even further once we identify those defects. Edge detection is by far the most common approach for detecting discontinuities in gray level. For our purposes, we will stay away from color images due to the complexities involved. In the following discussion we will explain what a gray-scale digital image is, how the first and second order derivative (the Gradient operator)is used in edge detection , and how filtering is used to detect edges.

2. Intensity Images in MATLAB

Before we can apply edge detection to images, we must first understand how intensity(gray-scale) digital images are stored and interpreted. For example, let's say that we want to store a 256×256 gray-scale pixel image. The image can be treated as a 256×256 matrix, with each entry in the matrix representing a number between 0 and 255, where 0 represents the color black, and 255 represents the color white. Every other number in between represents different shades of gray as shown in Figure 1.

3. How Edges are Detected

Edge detection is one of the most common methods for detecting discontinuities in gray level.

Intuitively, a digital edge is made up of pixels which are connected and lie on the boundary between two regions. Regions, in this case, refers to two different shades of gray. In order to better define an edge, it is required that we measure gray level transitions. Figure 2a shows the properties of a basic edge where the image consists of a set of connected pixels, half of which lie in the specific gray region and half of which lie in the black region of the gray-scale color bar. Figure 2b shows an image with gradual transition from black to the same gray color.

By the use of MatLab's *improfile* command we are able to present the gray-level profile beneath each image. The command *improfile* breaks each image into 20 pixels,(in this situation), and then computes the intensity values along a line path on an image. In the case of Figure 2a, the horizontal profile illustrates the sharp shift in color as there is a clear discontinuity detected by the first-and second-order derivatives. In Figure 2b, the slope of the ramp is inversely proportional to the degree

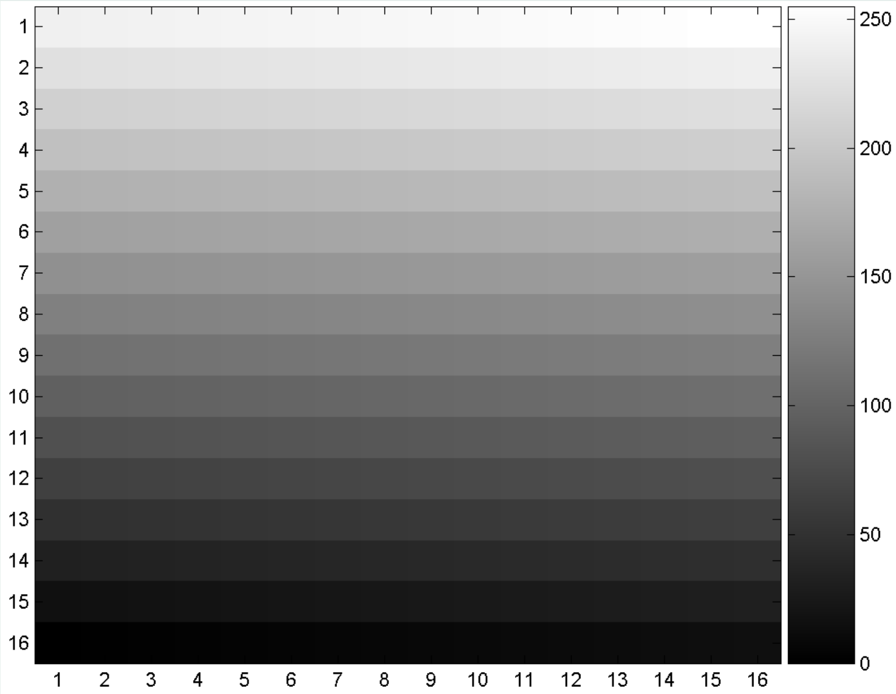
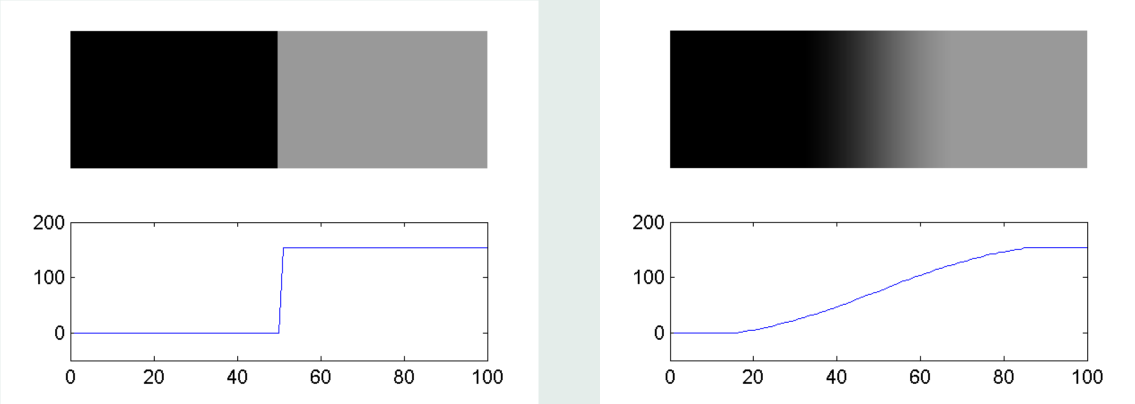


Figure 1: Shows the different gray-level intensities between numbers 0 to 255.



(a) sharp transition(step)

(b) gradual transition(ramp)

Figure 2: Two basic edges

of fading in the edge. An edge in this figure is now a set of connected points that is contained in the ramp. Thus, the length of the ramp, then determines the thickness of the edge–blurred edges have the tendency to be thick and sharp edges have the tendency to be thin.

4. First and Second Derivative

Now, we must look at some fundamental properties of these derivatives so that we can explain how gray-level discontinuities are detected. Looking at Figure 3 (from left to right), we can see that there is a fading region, a white dot, a line, and a corner of a rectangle. We are interested in the behavior of the first-and second-order derivatives as they detect these different shifts in gray-scale within in Figure 3.

As you can see in Figure 4, the **first derivative** must be zero where there are flat segments (areas of constant gray-level values), nonzero at the beginning of a gray level step or ramp, and nonzero along the ramps. As with the **second derivative**, it must be zero at every flat segment, nonzero at the beginning and end of a gray-level ramp, and zero along the ramp of a constant slope.

You can see that the first derivative is nonzero (-1) along the first ramp, or explicitly along the edge of the faded region; and, the second derivative is nonzero (roughly -1 at the onset and

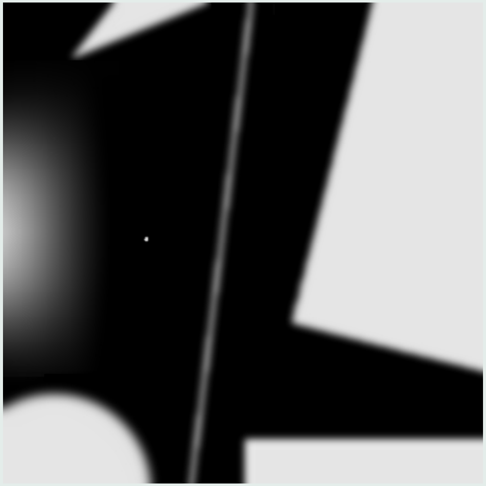


Figure 3: A simple grayscale image in which we will examine the gray level intensity with MATLAB's *improfile*. The dotted line shows the path we will examine (including the white point).

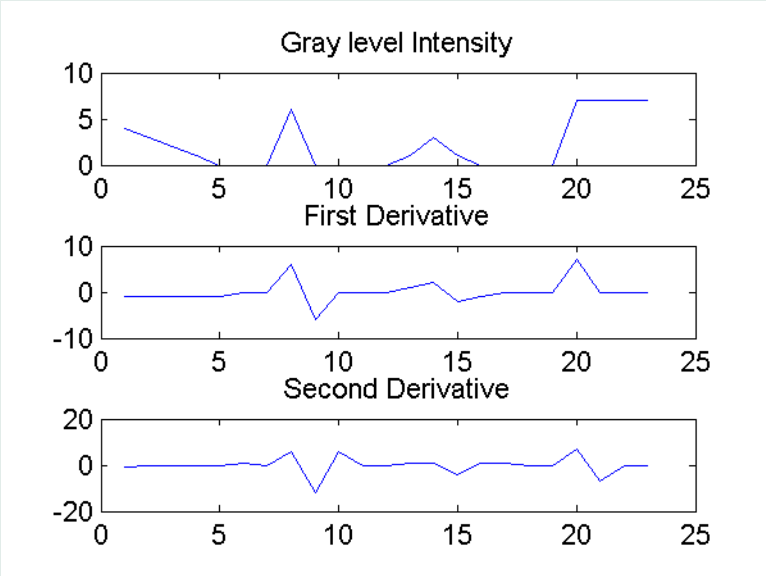


Figure 4: Plot of the gray-scale intensities, along with plots of the first-and second-order derivatives.

roughly 1 at the end of the ramp). Also, notice that the first derivative produces a big response as it detects the white line. By comparing the first and second derivatives, we can summarize that: the first-order derivatives generally detect thicker edges, edges that we can see with our eyes. However, the second-order derivatives have a stronger response to finer details, such as thin lines and small points.

In conclusion, the second-order derivative is better suited for edge detection due to it's ability to detect fine details. For our purposes, we will concentrate on the first-order derivative, because of it's simplicity and ability to detect obvious edges.

5. The Gradient Operator

Since images are two-dimensional, and based on a function $f(x, y)$, we must express edge detection in terms of the gradient; the gradient vectors determine the partial derivative with respect to x and the partial derivative with respect to y. The gradient of f at spatial coordinates (x, y) is defined as the vector

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

and the magnitude of this vector is

$$\|\nabla f\| = [G_x^2 + G_y^2]^{1/2}.$$

As you might have guessed, the magnitude of the gradient vector will produce our edges, but the question is, how?

6. How Filtering Works

Before we go into explaining the actual process of detecting edges, we must first explain the process of linear spatial filtering. Filtering works by multiplying each pixel in a neighborhood of an image by a corresponding coefficient and summing all the results to get the response at each point $f(x, y)$. If the neighborhood is the size of $m \times n$, mn coefficients are necessary. The coefficients are arranged in a $m \times n$ matrix called a filter mask.

For example, let us start with the spatial coordinate system shown in Figure 5 containing a section of an image and a filter mask. The red numbers are the coefficients of the mask and the blue numbers are the entries of the image. We will multiply each pixel in the neighborhood(every pixel

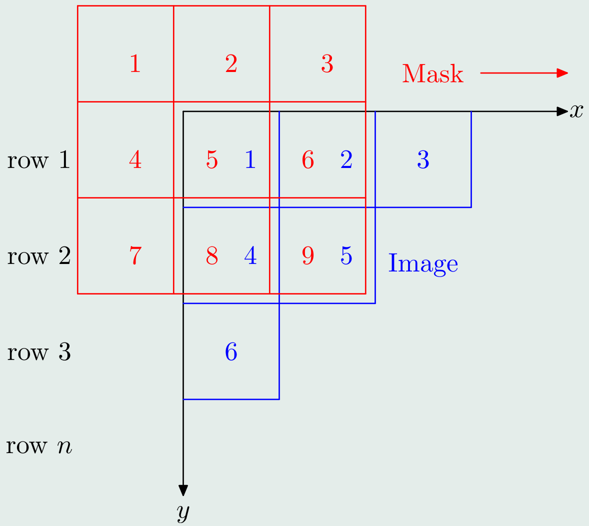


Figure 5: The 3×3 matrix (red) is the filter Mask and it is placed on top of the Image (blue).

under the 3×3 mask) by its corresponding coefficient and sum up the results. To get the response at the first (corner) entry of the image, we would take

$$(5 \times 1) + (6 \times 2) + (8 \times 4) + (9 \times 5) = 94.$$

Finally, the placement of response 94, is shown in Figure 6.

This filtering process repeats by shifting the mask over to the right by one pixel and repeating the necessary multiplications to get another response at entry $f(x, y)$. The entry $f(x, y)$ on the image, that will be subjected to change by the filter that lies upon it, is shown in Figure 7. So, if we did repeat the process, the next response will be located to the right of the response 94, which we obtained earlier. The filter mask will keep going through all the pixels in row 1 and, like a typewriter, it will move on to row 2 and repeat the same process until all the pixels have been filtered.

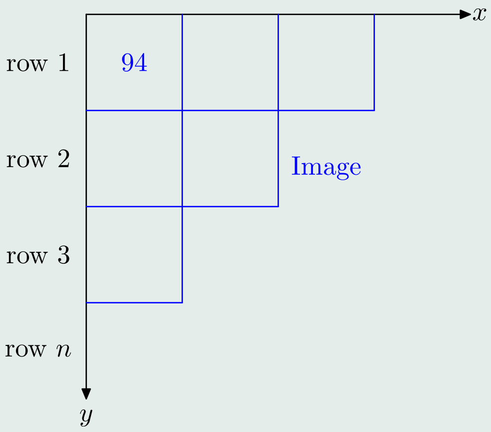


Figure 6: The resulting filtered pixel.

7. Prewitt's Mask

I believe we are now ready to relate linear spatial filtering to the gradient operator. Recall that the basic definition of the derivative is

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

as shown in Figure 8.

We will ignore the the scalar value 2, because it has no significance in the results. We'll begin by taking a function based on two variables, x and y. Then, when we take the partial derivative with respect to x, we get

$$\frac{\partial f}{\partial x} = 1 f(x+1, y) - 1 f(x-1, y)$$

and when we take the partial derivative with respect to y, we get

$$\frac{\partial f}{\partial y} = 1 f(x, y+1) - 1 f(x, y-1).$$

The scalar values are factored out of the partial derivatives for a reason. Remember that the pixels of the image under a filter mask is represented by Figure 7.

$f(x-1, y-1)$	$f(x, y-1)$	$f(x+1, y-1)$
$f(x-1, y)$	$f(x, y)$	$f(x+1, y)$
$f(x-1, y+1)$	$f(x, y+1)$	$f(x+1, y+1)$

Figure 7: The 3×3 matrix shows the pixels of the image that are under the mask. Each pixel, other than the one located at $f(x, y)$, is shifted with respect to the spatial coordinate system.

Introduction
Intensity Images in . . .
How Edges are Detected
First and Second . . .
The Gradient Operator
How Filtering Works
Prewitt's Mask
Recap
Filtering with Prewitt's . . .
Resulting Images
Conclusion
Home Page
Title Page
Page 10 of 26
Go Back
Full Screen
Close
Quit

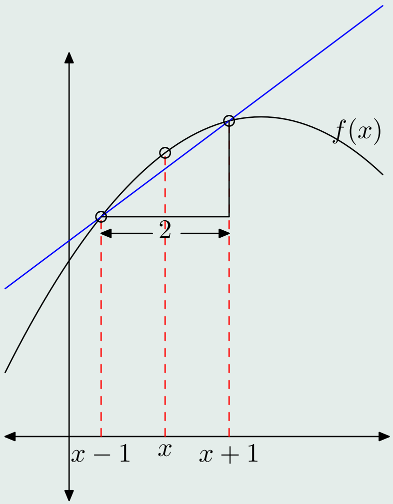


Figure 8: 1/2 of the difference between $f(x+1)$ and $f(x-1)$ is the approximation for the slope at x, or simply the first derivative.

$f(x-1,y-1)$	$f(x,y-1)$	$f(x+1,y-1)$
-1 $f(x-1,y)$	0 $f(x,y)$	1 $f(x+1,y)$
$f(x-1,y+1)$	$f(x,y+1)$	$f(x+1,y+1)$

Figure 9: The pixels of the mask represent the elements of the $\partial f/\partial x$ and the elements of the $\partial f/\partial x$ are replaced with the matching scalar values.

Home Page
Title Page
Page 12 of 26
Go Back
Full Screen
Close
Quit

-1	0	1
-1	0	1
-1	0	1

Figure 10: First and third columns are replaced with -1 and 1. Everything else is 0 because those entries don't exist in our partial derivatives.

The pixels represent where the partial derivative with respect to x will be taken. A better way to look at this is to look at that the second row which represents the $\partial f/\partial x$. This is how the filter mask is created for edge detection. When we filter the image, the mask is essentially taking the partial derivative with respect to x. The mask is created by placing the scalar values from the $\partial f/\partial x$ in the correct locations as shown in Figure 9. Remember that we are able to do this, because the pixels of the image are the points $f(x, y)$.

Also, the rest of entries in the first and third column of the filter mask will be replaced with -1 and 1 respectively (see Figure 10), because those entries are also locations of where the partial derivative with respect to x will be taken. Also, y does not only stay constant in the second row; y stays constant in the first and third row of the filter mask. This is why filtering will take the $\partial f/\partial x$.

Finally, we will create another filter mask to determine the partial derivative with respect to y as shown in Figure 11.

We will run each mask on separate, but identical images. We will then square every entry in

-1	-1	-1
0	0	0
1	1	1

Figure 11: This time, the first and third row are filled with -1 and 1 respectively.

x-mask

-1	0	1
-1	0	1
-1	0	1

y-mask

-1	-1	-1
0	0	0
1	1	1

Figure 12: The red mask takes the $\partial f/\partial x$ and the blue mask takes the $\partial f/\partial y$.

each of the two resulting filtered images, add them together, take the square root of the resulting matrix, and get

$$\|\nabla f\| = [G_x^2 + G_y^2]^{1/2}.$$

Now each pixel in our resulting image represents the magnitude of the gradient vector.

8. Recap

Again, remember the plots of the gray-level intensities in Figure 2? Recall that the slope(edge) of the step in Figure 2a and the slope(edge) of the ramp in Figure 2b represents the first derivative. Since an image is two dimensional, it requires a function of two variables to determine the first derivative at any point (x, y). This function is the magnitude of the gradient.

9. Filtering with Prewitt's Mask

Prewitt's mask is actually composed of two masks: One takes the $\partial f/\partial x$, and the other takes the $\partial f/\partial y$. For the sake of repetitiveness, we will call the mask that takes the $\partial f/\partial x$ the "x-mask", and the mask that takes the $\partial f/\partial y$ the "y-mask" as shown in Figure 12.

We will filter the 5×9 matrix(intensity image) in Figure 13. As you can see, there are numbers ranging from 0 to 255. Remember that 0 represents black, 255 represents white, and every other number in between represents different shades of gray. If you take a closer look at our image,

200	180	190	200	180	185	180	100	180
190	30 200	10 220	5 180	185	200	-290 -5 200	-15 15 90	200
200	-250 200	-260 200	180	195	175	205	80	190
140	0 120	100	120	110	105	100	90	80
200	220	180	190	205	210	210	70	180
200	210	205	210	220	215	200	80	195

Figure 13: A 5×9 pixel gray-scale image. The black numbers represent the pixels of the image, the blue numbers represent the responses produced by the y-mask, and the red numbers represent the responses produced by the x-mask.

there are big differences in our numbers between some rows and some columns–those big differences represent edges.

We will start by running the y-mask over entry (2,2). For our purposes, we will not start at the origin, as we are trying to show how our mask can detect edges. We will begin the process of filtering by multiplying every mask coefficient by its corresponding pixel, and sum up all of the results to get a response at $f(x, y)$. The response at entry (2,2) is 30. We will repeat this process at entries (2,3), (2,4), (3,2), (3,3), (4,2), (2,7), and (2,8) as shown in Figure 13.

Notice that in Figure 13, the y-mask is able to detect the horizontal dark line (assume that numbers around 100 is a dark shade of gray) at row 4. The y-mask produced big negative numbers

Introduction
Intensity Images in . . .
How Edges are Detected
First and Second . . .
The Gradient Operator
How Filtering Works
Prewitt's Mask
Recap
Filtering with Prewitt's . . .
Resulting Images
Conclusion
Home Page
Title Page
Page 16 of 26
Go Back
Full Screen
Close
Quit

at row 3 and at row 5, detecting what would be both sides of a line. Remember in Figure 4 that the first derivative produced a big response on both sides of the white line in Figure 3. Again, this is why the first derivative is best suited for detecting ”thick” edges(basically ramps, steps and lines that the human eye can see).

However, if we run the y-mask over the entries (2,7), and (2,8), the mask is not able to detect the vertical edge that is present in Figure 13 at column 8. If we try again and use the x-mask instead of the y-mask, the mask is able to detect the vertical edge as shown in Figure 13.

We must understand that the x-mask is able to detect changes in gray-level intensities in the x-direction(vertical edges), and that the y-mask is able to detect changes in gray-level intensities in the y-direction(horizontal edges) as shown and proven in Figure 13.

10. Resulting Images

We can finally apply what we have learned about edge detection. We will filter an image using Prewitt's mask in MATLAB.

We will start by opening up the command prompt in MATLAB and loading the image using the following command

```
J=imread('sample.jpg');
```

The result is the matrix of the image, with each entry representing a different color. We compressed our image due to its large size. Also, since the image is a RGB(true color) image, we have to convert it to an intensity image. Understand that true color images are too complex and difficult to work with. The following commands will convert the image into an intensity image and display it.

```
I=rgb2gray(J);
imshow(I);
```

Our image should look like Figure 14. Now let's load the x-mask and filter our image with the mask using the following commands.

```
xmask=[-1,0,1;-1,0,1;-1,0,1];
fx=imfilter(double(I),xmask);
figure,imshow(fx,[])
```

The *imfilter* command filters our image with the x-mask, and the double command converts our 8 bit image into double precision, so MATLAB can perform certain operations on it. So by taking the $\partial f/\partial x$, our result should look like Figure 15.

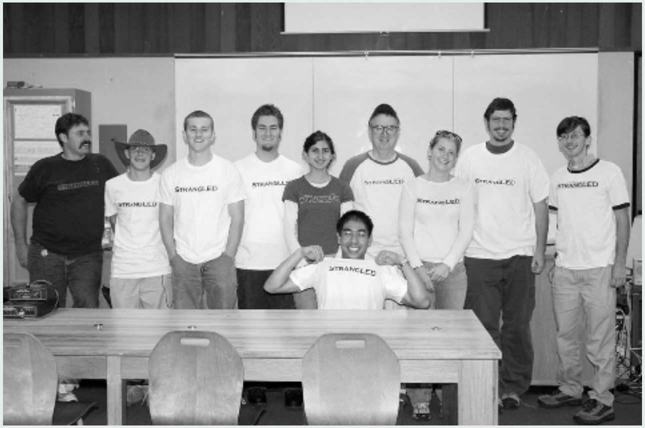


Figure 14: An intensity image of our class.

Now we need to ask ourselves why Figure 15 is mostly gray and why the edges are black and white. First, we need to use the following commands to find the maximum and minimum values in Figure 15.

```
max(fx(:))
min(fx(:))
```

Our maximum and minimum values are 633 and -608 respectively. The square brackets([ ]) in the *imfilter* command scales all the values in Figure 15 from 0 to 255(gray-scale intensity levels), which means almost all of the pixels in Figure 15 are shades of gray. Figure 16 shows the distribution of the entries, with each entry or pixel representing a number between 633 and -608.

The edges in Figure 15 appear to be black or white, because the pixels next to those edges can be a very big negative number (as shown in Figure 1, or can be a very big positive number. Those numbers are scaled near to either 0 or 255, which represent black and white respectively.

Now we will filter another image of our class with the y-mask using the previous commands, but loading the y-mask instead of the x-mask into MATLAB as shown in Figure 17; this image is mostly gray for the same reasons as Figure 15.

By comparing Figure 15 and Figure 17, we can see the vertical edges of our intensity image are detected in Figure 15 and the horizontal edges of our intensity image are detected in Figure 17. As

Introduction
Intensity Images in . . .
How Edges are Detected
First and Second . . .
The Gradient Operator
How Filtering Works
Prewitt's Mask
Recap
Filtering with Prewitt's . . .
Resulting Images
Conclusion
Home Page
Title Page
◀◀ ▶▶
◀ ▶
Page 18 of 26
Go Back
Full Screen
Close
Quit

Figure 15: Result of filtering with the x-mask.

Introduction
Intensity Images in . . .
How Edges are Detected
First and Second . . .
The Gradient Operator
How Filtering Works
Prewitt's Mask
Recap
Filtering with Prewitt's . . .
Resulting Images
Conclusion

Home Page

Title Page

Go Back

Full Screen

Close

Quit

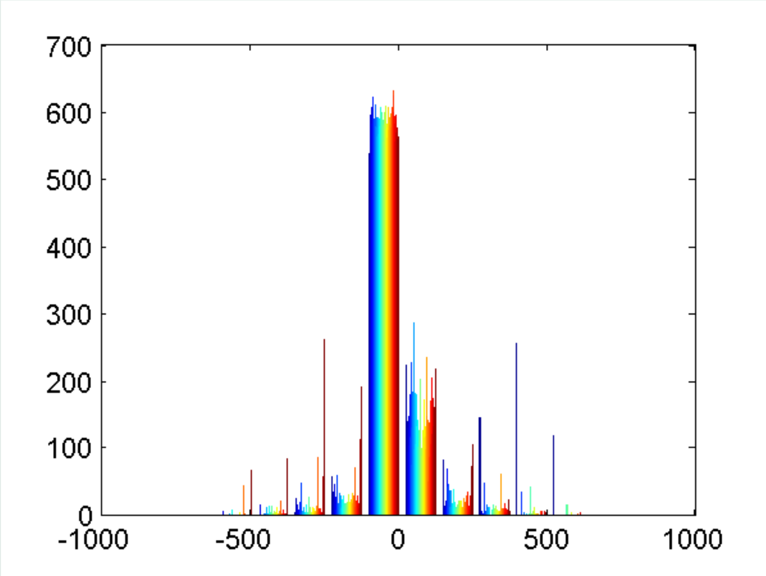


Figure 16: The histogram shows the gray-scale intensities versus the number of entries of the $\partial f/\partial x$. Understand that gray-scale intensities are not scaled from 0 to 255, but since most of the entries are in the middle, they represent different shades of gray.

Figure 17: Result of filtering with the y-mask or taking the $\partial f/\partial y$.

Introduction
Intensity Images in . . .
How Edges are Detected
First and Second . . .
The Gradient Operator
How Filtering Works
Prewitt's Mask
Recap
Filtering with Prewitt's . . .
Resulting Images
Conclusion

Home Page
Title Page
Page 21 of 26
Go Back
Full Screen
Close
Quit

Figure 18: Edges are detected by the magnitude of the gradient.

we proved earlier, the $\partial f/\partial x$ only detects edges in the x-direction (vertical edges), and the $\partial f/\partial y$ only detects edges in the y-direction (horizontal edges).

By squaring the resulting matrices, adding them together and taking the square-root of the result, we will get a matrix that represents the magnitude of the gradient.

$$\|\nabla f\| = [G_x^2 + G_y^2]^{1/2}.$$

We must remember that the first derivative will detect the "thick" and obvious edges. Since our image is a function of (x, y), the magnitude of the gradient will detect the edges with respect to the x-direction and the y-direction, as shown in Figure 18.

Although we have detected the edges in the image, we need to understand why Figure 18 appears to be black and white.

We will plot a histogram similar to the one in Figure 16, except we will now square every entry as shown in Figure 19.

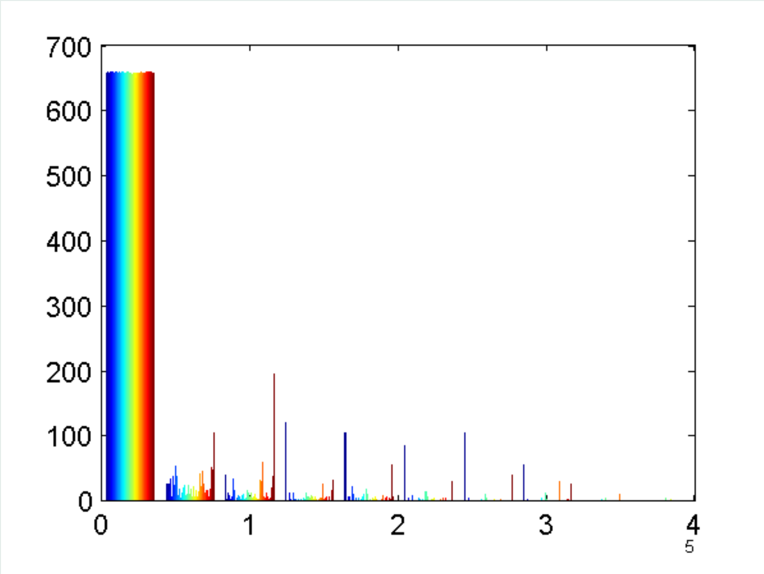


Figure 19: The histogram of $\partial^2 f/\partial x^2$.

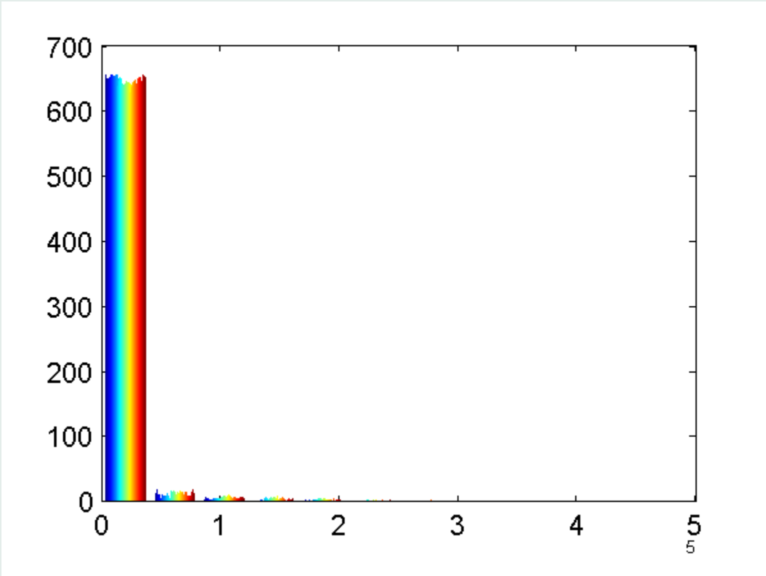


Figure 20: The histogram of $\partial^2 f/\partial y^2$.

As you can see in Figure 19, most of the pixels lie on the far left, which indicates that most of the pixels are near the color black. We should expect the same results if we plot a histogram of $\partial^2 f/\partial y^2$, and we do as shown in Figure 20.

Looking back at Figure 16, we can see most of the pixels lie around 0, and few pixels are lying on the far left and on the far right. Which means if we square all those pixels, we will get a histogram with most of the pixels lying around 0, and have a few pixels around 3×10^3. Remember that the maximum value for the resulting image from taking the $\partial f/\partial x$ is 633. Well, $633^2 = 4.0 \times 10^5$, which agrees to what our results are in Figure 19. Remember that when we square a small number (lets say for example, 2), we get a number that is small in magnitude (the result is 4). However, when we square a number that is much larger (lets say for example, 10), we get a number that is much bigger in magnitude (the result is 100). The same idea applies to what the histogram represents in Figure 19.

Finally, if we plot a histogram of the magnitude of the gradient (remember, that Figure 18 represents $\|\nabla f\| = [G_x^2 + G_y^2]^{1/2}$), we should expect that most of the pixels lie near 0 as shown in

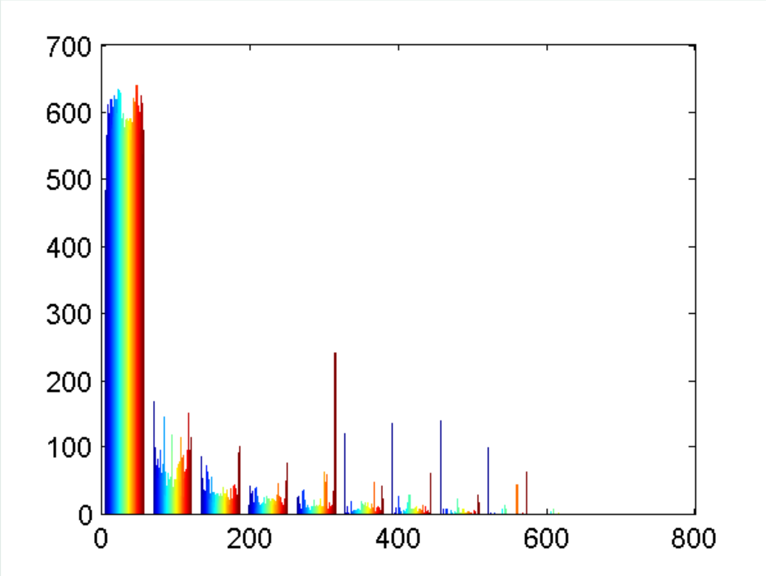


Figure 21: The histogram of the $\|\nabla f\|$.

Figure 21. If we scale the values in Figure 21 on the x-axis from 0 to 255, we would notice that most of the pixels are indeed black, and that the edges are mostly white as expected in Figure 18.

11. Conclusion

We have explained what an edge is, how edge detection is applied in the real world, how edge detection is based on the magnitude of the gradient, how filtering works, and how Prewitt's mask is able to detect edges. At this point, you might wonder if linear spatial filtering is limited to edge detection and how we can use the second derivative in filtering. We must understand that our project just explains the basics of edge detection and spatial filtering. There are many other applications with linear spatial filtering and many other masks out there. Please use this project as a foundation for future explorations in digital image processing.

References

[1] Gilbert Strang. *Introduction to Linear Algebra. 2nd ed. Wellesley-Cambridge Press, 2003*

[2] Gonzalez, and Woods. *Digital Image Processing. 2nd ed. Prentice Hall, 2002.*

[3] Gonzalez, Woods, and Edwins. *Digital Image Processing using MATLAB. Prentice Hall, 2004.*

[4] Arnold, Dave. Interview. Fall, 2005.