Lineariy II: Final Report

Anna Griffin and Enmo Ren

December 2018

1 Introduction

As we move further into the Information Age, we have a growing dependence on computers and thus, a growing amount of digital information. Digital processing has made way for more powerful and precise algorithms especially in image processing. The broader array of algorithms make many image processing techniques possible. Edge detection is at the heart of image processing. This technique was practically impossible with previously used analog techniques. This fundamental process analyzes images and identifies discontinuities to extract information about the image. Edge detection serves as a gateway to many more advanced processing technologies in image processing.

2 Edges

An edge is defined as the bounding limit of an object, area, or surface. In an image, edges are represented by significant changes in value intensity. There are two different kinds of discontinuities: step and line. A step change occurs when there is one value on one side of the discontinuity and a completely different one on the other. On the other hand, a line discontinuity describes a situation in which the value changes significantly and then returns to the original value almost immediately after. If you imagine a black line on a white background, as you cross over it, the values go from white to black for only a short bit and then return back to white. In real images, however, these discontinuities are less precise and therefore yield ramp edges and roof edges which correspond to the step and line discontinuities respectively. The changes in these cases occur over a short distance rather than instantly.

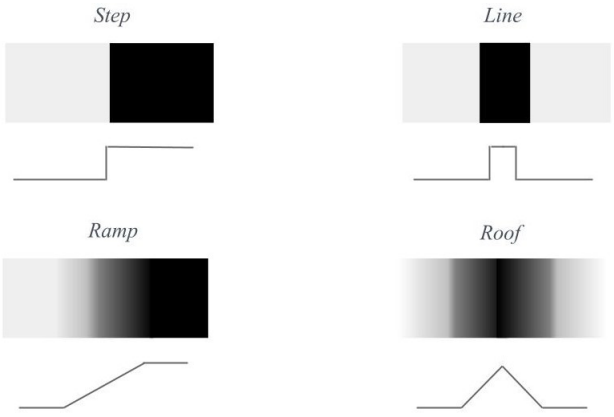

Figure 1: This figure describes different types of edge.

3 Approaches

In general, there are two main approaches, identifying the local extrema in the first derivatives or the zero-crossings in the second derivative. When an edge occurs, there will be a spike or drop in the intensity over a short period of time. When we apply the gradient to these values, which is essentially calculating the first derivative in either the x- or y-directions with respect to time, we see either a minimum or maximum. When using first derivative approaches, these changes are the points of interest. In the other circumstance, when we implement second derivative methods, we used the Laplace operator which is the divergence of the gradient; in other words, the 2D spatial derivative. Using this method, the zero-crossings are now observed because the rate of change at the minimums and maximums of the gradient is zero.

One main difference between these two approaches is the way in which they deal with classification of edges. For the gradient methods, a thresholding process is applied afterward. Only local extrema that fit this threshold are considered edges. As for the Laplacian methods, it is crucial that a Gaussian is applied before operator to reduce noise because this technique is especially sensitive to noise.

3.1 First Derivative

To detect these discontinuities, the gradient of the image is observed. The gradient describes local changes in the intensity by taking the discrete approximation of the first derivative. In methods of edge detection that rely on the gradient approach, the minimum and maximum points mark the edges because the rate of change in intensity at those points is the greatest. For a two dimensional object, the image in this case, the gradient is represented as a vector containing the change in the x-direction and the change in the y-direction.

$$G = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{1}$$

Since gradient is a vector, it has both a magnitude and direction. The magnitude is calculated by taking the square root of the squares of both gradients.

$$|G| = \sqrt{{G_x}^2 + {G_y}^2} \tag{2}$$

The direction of the gradient can be found using the inverse tangent function.

$$\theta = \arctan \frac{G_y}{G_x} \tag{3}$$

The magnitude is significant in this application because it is a measure of the greatest increase of intensity and the direction is the direction in which that path follows.

3.2 Second Derivative

A different approach to identifying edges is by using the Laplacian method. This operation could be done by calculating second derivatives, in which local maxima or minima will appear as zero-crossings. It is a scalar operator, which defined as

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{4}$$

Laplacian can be used as a 2D spatial second derivative for image processing. It takes the second derivatives of pixel intensity of images and finds the zero-crossing which describes the most significant changes in the image. The formula for the Laplacian of a function I(x,y) is:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{5}$$

Since images are discrete representations using pixel intensity value, Laplacian is transformed into a discrete convolution kernel that can approximate the second derivatives. Due to the fact that Laplacian utilizes second derivatives to emphasize any rapid change, a Gaussian Smoothing filter is always applied in order to smooth the image, preventing unwanted noise from being detected as edges. In order to reduce computational complexity, the LoG (Laplacian of Gaussian) kernel, which combines Gaussian smoothing filter and Laplacian filter, is always applied instead so that only one convolution mask is needed for finding second derivatives. The Laplacian of Gaussian-filtered images is the same as taking Laplacian of Gaussian and convolving the resulting kernel in the applied image.

4 Applying Masks

Since images are two dimensional, it is convenient to use a kernel which is also called a convolution mask. This allows us to observe the surrounding pixels and use them as a source of information to detect edges. The approximation of the partial derivatives is found by subtracting the pixel values that are near the target pixel.

4.1 Gradient Approaches

In gradient methods of edge detection, the partial derivatives of x and y are computed separately and then summed. Since the x derivative just looks at changes in the x-direction we can subtract the pixels on the left and right of the target pixel. For the y-direction, the pixels above and under are now relevant. We can write the subtraction of the neighboring pixels using the Taylor expansion and solve for the first derivative. Below we show how the first derivative can be approximated by subtraction the two neighboring pixels in the x-direction. The same method is used for the y-direction but it is calculated separately.

$$\begin{array}{rcl} \text{f(x+h)} & = & \text{f(x) + f'(x)h + ...} \\ \text{- f(x-h)} & = & \text{f(x) - f'(x)h + ...} \\ \hline \text{f(x+h) - f(x-h)} & = & \text{2f'(x)h} \end{array} \tag{6}$$

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) \tag{7}$$

$$\frac{\partial u}{\partial x} = \frac{u(x+h) - u(x+h)}{2h} \tag{8}$$

This is actually an approximation of the points $(x + \frac{1}{2}h, y)$ for Gx and $(x, y + \frac{1}{2}h)$ for Gy. Both points lie in the middle of the two pixels observed. Because this is not very intuitive when trying to visualize, it is common to center around a whole pixel instead of a location in between pixels. To achieve this, we can use a 1x3 mask for the x-direction and a 3x1 mask for the y-direction.

-1	0	+1

G_x

-1
0
+1

G_y

-1	0	+1
-1	0	+1
-1	0	+1

G_x

-1	-1	-1
0	0	0
+1	+1	+1

G_y

It is also advantageous to get information in both the x- and y-direction relative to the target pixel. A 3x3 convolution mask is often desired and can be calculated through expanding the 1x3 or 3x1 mask by looking at its two neighboring rows or columns.

A more sophisticated way of generating the 3x3 convolution masks for the partial derivatives is to incorporate blurring which reduces noise sensitivity. Gaussian blurring is the most popular method and it puts more weight on the pixels that are close in proximity to the center pixel. The two masks that are produced after the blurring is applied are known as the Sobel operators.

-1	0	+1
-2	0	+2
-1	0	+1

G_x

-1	-2	-1
0	0	0
+1	+2	+1

G_y

4.2 Laplacian Approaches

Laplacian is computed through summing the second partial derivatives of x and y. In order to approximate the second derivatives using discrete pixel intensity values, we expand the neighboring pixel using the Taylor series.

$$\begin{array}{rcl} \text{f(x+h)} & = & \text{f(x) + f'(x)h + } \frac{1}{2} \text{ f"(x)h}^2 + ... \\ \text{+ f(x-h)} & = & \text{f(x) - f'(x)h + } \frac{1}{2} \text{ f"(x)h}^2 + ... \\ \hline \text{f(x+h) + f(x-h) - 2f(x)} & = & \text{f"(x)h}^2 \end{array} \tag{9}$$

By isolating the second derivatives on the right side of the equation and summing up the first and second equations, we can express the second derivatives of point x in terms of its neighboring pixel.

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{10}$$

In order to get Laplacian, which is a scalar, the second derivatives from x and y direction are added together.

$$\nabla^2 I = Div \ \ Grad \ \ I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{11}$$

$$\frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = \frac{I(x+h,y) + I(x-h,y) + I(x,y+h) + I(x,y-h) - 4I(x,y)}{h^2} \tag{12}$$

After finding the relationship between the second derivatives of the target pixel and its neighboring pixels, a discrete convolution kernel that can approximate the Laplacian is derived.

0	1	0
1	-4	1
0	1	0

Discrete approximation to Laplacian filter

5 Canny Edge Detection Algorithm

The Sobel operator detects the magnitude and direction of edges by approximating the gradient of pixel intensity. However, it suffers from the noise within images. Therefore, the Canny Edge Detection algorithm is used to compensate for the disadvantages of Sobel operator.

The Canny edge detector is one of the well-known edge detectors in the computer vision field. It computes the gradients by convolving the Sobel kernel in x-direction and y-direction. The magnitude and direction, which demonstrates the local change of pixel, can be computed using Gx and Gy based on the method we mentioned earlier. Compared to other approaches such as Sobel operator alone, the Canny edge detection algorithm aims to eliminate problems of false edge detection, cases of missing true edges, and improving signal to noise through non-maximum suppression and double thresholding.

5.1 Non Maxima Suppression

Having found the magnitude and direction of the gradient, any intensity of pixel that is not considered as local maxima will be set to zero through Non-maxima Suppression. Each pixel will be checked if it is a local maximum against its neighboring pixels. Any local maximum will occur at a peak in the gradient direction. In other words, a pixel, whose gradient is smaller than those of other two pixels, will be suppressed (put to zero). This will result in thin-edges.

5.2 Thresholding

After locating the local extrema in the intensity, the image is passed through a threshold filtering process in an attempt to reduce the effect of noise. Instead of using a single threshold, a technique called hysteresis thresholding is often implemented. This approach relies on two thresholds, an upper and a lower bound. All of the pixels with absolute values that exceed the upper limit are labeled as edges and conversely, all of the values that lie below the lower threshold are disregarded. The values that are in the middle of the two thresholds are only considered to be edges if they are connected to a section that is above that upper limit. This method helps identify edges that might not be extremely clear; for example, if they are in a shadow or if the colors in the background that surround the image are similar.

6 Fuzzy logic

Fuzzy logic is an alternative approach to thresholding. Instead of using binary logic, recent methods have started moving in the direction of using many-valued logic, specifically fuzzy logic. Instead of classifying values as either 0 (black) or 1 (white), fuzzy logic allows variables to be anywhere in between. Every value is measured as a degree of membership to a certain variable. This takes into account the many shades of grey in an image instead of just categorizing data into "edge" and "no edge" groups.

Fuzzy logic has gained a lot of popularity since it was first introduced in 1965. This way of classifying values is more closely allied to how humans think and make decisions. Fuzzy logic reduces the amount of uncertainty and indefiniteness in a given situation by allowing for more options beyond just TRUE and FALSE. Instead of setting a single threshold and classifying a pixel as black or white depending on where it falls relative to the threshold, fuzzy logic would describe a grey pixel as a certain percentage of black. Black would be 100% , while white would be 0%, and anything in between would receive a fraction of membership. This allows the system to be more flexible and accommodate images with lower contrast areas while still achieving an acceptable amount of clarity in the result.

The fuzzy logic process is based on a classification system and IF-THEN rules. As the system traverses through the image data, each pixel is examined along with its 8 neighboring pixels. Before fuzzy logic is applied, a Gaussian mask is used to smooth the image. Then, similar to the Canny approach, the Sobel filters are used to find the gradient in the x and y direction. However, there are two additional filters that are applied in the fuzzy logic method. A low pass filter and a high pass filter are passed over the original image. After each edge strength value is calculated from applying the three masks individually (G_x or G_y, HP, and LP), they are passed through a membership function. This point of the process is called the fuzzification step. For the purpose of edge detection, "Low", "Medium", and "High" values are used to distinguish between varying levels of edge strength. The membership function graphs the classification of each variable (low, medium, high) along the intensity scale. Each point that serves as an input returns with three different values corresponding to amount of "lowness", "mediumness", and "highness". The membership functions are limited to values between 0 and 1 which is called the fuzzy domain. Now the conditional statements can be applied to come to a conclusion of whether or not the target pixel corresponds to an edge pixel (see appendix).

Then the values are defuzzified. The values in the fuzzy sets are combined in using the Mamdani method to calculate a final output of the classification of the original pixel.

7 Implementation of Canny Edge detection

Canny edge detection consists of five steps: Gaussian blur, Gradient, Non-maximum suppression, double thresholding, and edge tracking. In this section, we explored the implementation of Canny edge detection based on the mathematical theories we explained in the previous section. We generated result images after each operation being applied in order to provide a visual representation of image processing in Canny edge detection.

We used numpy library to convert images to matrixs for the future process. In our gradient_intensity function, we defined two kernels for the use of convolution in x-axis and y-axis. After computing gradient from the images using gradient_intensity, we applied non-maxima suppression through suppression function. To suppress non-maxima also means to select local maximum within the image. Recall the fact that images consist of discrete pixels, so in order to compare the pixel

with the neighboring pixels in its gradient direction, we round the gradient angle to four numbers of direction using round_angle function: 0, 45, 90, and 135. Based on the direction of the rounded gradient, we were able to compare the center pixel with the other two neighboring pixels in its gradient direction.

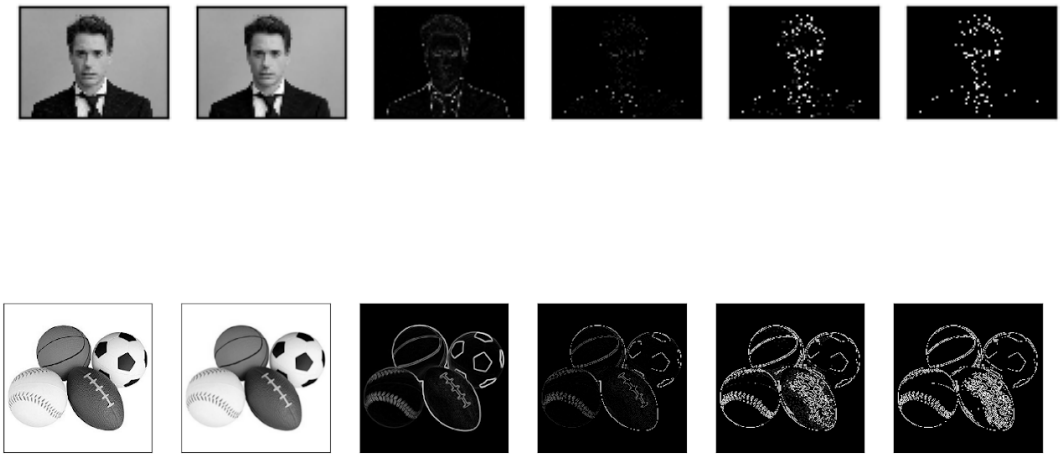

Figure 2: These graphs illustrate the output image after each steps in Canny edge detection we built.

Appendix

Conditional Statements

If $edginess_{Lp}$ is LO and $edginess_{So}$ is LO and $edginess_{Hp}$ is LO then p_{edge} is E_{L}

If $edginess_{Lp}$ is LO and $edginess_{So}$ is LO and $edginess_{Hp}$ is MD then p_{edge} is E_{L}

If $edginess_{Lp}$ is LO and $edginess_{So}$ is LO and $edginess_{Hp}$ is HI then p_{edge} is E_{L}

If $edginess_{Lp}$ is LO and $edginess_{So}$ is MD and $edginess_{Hp}$ is LO then p_{edge} is E_{L}

If $edginess_{Lp}$ is LO and $edginess_{So}$ is MD and $edginess_{Hp}$ is MD then p_{edge} is E_{L}

If $edginess_{Lp}$ is LO and $edginess_{So}$ is MD and $edginess_{Hp}$ is HI then p_{edge} is E_{M}

If $edginess_{Lp}$ is LO and $edginess_{So}$ is HI then $edginess_{Hp}$ is LO then p_{edge} is E_{L}

If $edginess_{Lp}$ is LO and $edginess_{So}$ is HI and $edginess_{Hp}$ is MD then p_{edge} is E_{H}

If $edginess_{Lp}$ is LO and $edginess_{So}$ is HI and $edginess_{Hp}$ is HI then p_{edge} is E_{H}

If $edginess_{Lp}$ is MD and $edginess_{So}$ is LO and $edginess_{Hp}$ is LO then p_{edge} is E_{L}

If $edginess_{Lp}$ is MD and $edginess_{So}$ is MD and $edginess_{Hp}$ is LO then p_{edge} is E_{L}

$\vdots$

If $edginess_{Lp}$ is HI and $edginess_{So}$ is LO and $edginess_{Hp}$ is HI then p_{edge} is E_{H}

If $edginess_{Lp}$ is HI and $edginess_{So}$ is MD and $edginess_{Hp}$ is HI then p_{edge} is E_{H}

If $edginess_{Lp}$ is HI and $edginess_{So}$ is HI and $edginess_{Hp}$ is HI then p_{edge} is E_{H}

Figure 3: Fuzzy Logic Rules [8]

References

[1] Samta Gupta, Susmita Ghosh Mazumdar *Sobel Edge Detection Algorithm*,

https://pdfs.semanticscholar.org/6bca/fdf33445585966ee6fb3371dd1ce15241a62.pdf

This source provided us a comprehensive understanding on implementing Sobel operator in edge detection. We were able to get a better grasp on basic principles on Sobel operator.

[2] Daniel Kim *Sobel Operator and Canny Edge Detector*,

https://www.egr.msu.edu/classes/ece480/capstone/fall13/group04/docs/danapp.pdf

This source provided us the comparison between Sobel operator and Canny edge detector.

[3] *Machine Vision: Chapter5 Edge Detection*,

http://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter5.pdf

This source gave us a good introduction into edge detection and image processing. It also provided an overview of all the existing edge detectors.

[4] Dharampal, Vikram Mutneja *Methods of Image Edge Detection: A Review* ,

https://www.omicsonline.org/open-access/methods-of-image-edge-detection-a-review-2332-0796-1000136.pdf

This paper pointed us towards the research in edge detection based on fuzzy logic. It also offered a clear explanation on how those edge detectors differ from each other mathematically.

[5] Raman Maini *Study and Comparison of Various Image Edge Detection Techniques*,

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.301.927&rep=rep1&type=pdf

This paper offered clear explanations of the different kernels and their strengths and weaknesses. There was also a pretty detailed procedure for canny edge detection that was very helpful and gave us a good sense of what was going on in that algorithm.

[6] Dmitrij Cestverikov, Dr. Himanshu Aggarwal *Basic Algorithms for Digital Image Analysis*,

http://www.inf.u-szeged.hu/~ssip/2003/lectures/chetverikov/course_on_basic_algorithms_for_digital_image_analysis/lecture07col.pdf

This source helped us understand hysteresis thresholding. The authors provided a easy to understand explanation of the steps and reasoning behind each.

[7] Sunita Sarangi, N. P. Rath *Performance Analysis of Fuzzy-based Canny Edge Detector*,

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4426380

This source helped us put together the different steps of the fuzzy logic method. The authors provided short descriptions but it was still useful when trying understand the bigger picture of the algorithm.

[8] David O. Aborisade *Novel Fuzzy Logic Based Edge Detection Technique*,

`http://www.matlabi.ir/wp-content/uploads/bank_papers/ipaper/i37_www.Matlabi.ir_Novel%20Fuzzy%20logic%20Based%20Edge%20Detection%20Technique.pdf`

This paper helped us understand the fuzzification step of the process and provided a lot of background information about the application and development of fuzzy logic.

[9] Yuan-Hang Zhang, Xie Li, Jing-Yun Xiao *A Digital Fuzzy Edge Detector for Color Images*,

`https://arxiv.org/pdf/1701.03364.pdf`

The source provided a good introduction and overview of fuzzy logic. It also put this process in context and walked through other steps of the process that are related.

[10] Yasar Becerikli, Tayfun M. Karan *A New Fuzzy Approach for Edge Detection*,

`https://pdfs.semanticscholar.org/869c/48c918e134a261dd2c5fe588dc69fbf94cb4.pdf`

This source communicated the significance of fuzzy logic in edge detection and provided a clear and concise outline of the process. Also, there was a visual explanation of how the kernel is formed.

[11] Abdullah H. Muhammad, Hussain S. Akbar *Algorithms for Edge Detection by Using Fuzzy Logic Technique*,

`https://www.researchgate.net/publication/294925526_ALGORITHMS_FOR_EDGE_DETECTION_BY_USING_FUZZY_LOGIC_TECHNIQUE`

This journal article provided a visual explanation of non-maximal suppression which aided our understanding.