Implementation of Real-Time Edge Detection Using Canny and Sobel Algorithms

N D Lynn¹, A I Sourav², and A J Santoso³

¹Magister Informatika, Universitas Atma Jaya Yogyakarta, Yogyakarta, Indonesia
²e-mail: 195303057@students.uajy.ac.id

Abstract. The process of performing tasks related to image processing requires the use of potential techniques such as edge detection. The technique can be used to accomplish tasks related to video and image manipulation. The process of edge detection is very virtual as it acts as the first phase of image analysis and understanding. In this paper, the edges of real-time images were detected for easy extraction of meaningful information from images. A prototype was implemented on OpenCV which is an in-built function cv2. () using python. Canny and Sobel edge detection algorithms are chosen to be used in this paper for comparison purposes to find out which method is better at edge detection. After using both algorithms to detect edges of real-time images, the result showed that the Canny algorithm produced thick edges compared to the Sobel algorithm. The canny algorithm follows the criteria of; Good detection, Good localization, and Minimal response, to output good edge detection results More so, canny uses a double threshold for edge revelation and applies the Gaussian filter which removes of any noise from an image, unlike Sobel which is not resistant to noise. The developed prototype for this study can detect edges of real-time objects without the intervention of any sensor input.

Keywords: Real-time, Edge detection, Canny and Sobel Algorithms, OpenCV, Image processing

1. Introduction

The advancement of new technologies for image processing and computer vision today gives long term solutions in different fields of life such as medical [1], [2] security [3], agriculture [4], and photo/video production [5]. The process aimed to capture very useful properties of objects and images is called Edge detection. This process is helpful in a way that eases image processing and allows the computer vision industry to capture image processes.

In edge detection, the variation in the grey level image must be increased, and the main variations used in edge detection are step edges, local extrema, and 2D features. During this process, the input is a grayscale image and the output is a binary image that displays both non-edged and edged pixels [6]. Edge detection aims at orienting these variations by identifying the real and physical phenomena that produce them. However, the edge detection process requires efficiency and reliability to ensure the possibilities plus the validity of accomplishing and guiding the edge detection process [7].

Due to advancements in computer vision Technology, several methods for edge detection have been developed. However, these two methods are categorized into two main groups ie; the Search-based (First order based) and the Zero-crossing (second-order based) categories. This paper uses both Canny and Sobel algorithms for real objects and image edge detection. The Canny algorithm, which bases itself on
the Gaussian filter to formulate the optimal edge detector is grouped in the Second-order category of edge detection techniques, while Sobel algorithm is found in the First order-based category of edge detection techniques [8]. After formulating these algorithms, many researchers have contributed a lot to algorithms’ innovation.

Real-time is an important subject that is employed in a wide range of applications of real-life activities today. Therefore, this is the main reason that motivated the authors of this paper to capture the edges to a real-time. For example; It can be employed in carrying out activities of authentication systems, video surveillance systems, and traffic surveillance systems. In addition to that, Real-time image processing can be employed to perform tasks of face recognition and iris recognition while carrying out activities such as in the medical field [9], [10]. To produce the edges of real-time objects and images we used OpenCV which is built-in python function.

2. Understanding Theory

Real-time edge detection usually aims at producing the edges of images and objects without pre-given information. To carry out successful edge detection in this study using the two chosen algorithms, i.e, canny and Sobel, the authors successfully went through the steps that each algorithm uses for real-time edge detection activity. In the next section, the categories of edge detection, edge detection techniques, and the procedure of edge detection, are discussed.

2.1. Categories of Edge detection

Image processing consists of several edge detection methods. All these are classified into two major groups namely; search-based and zero-crossing based methods, and these are also known as the “First-order based”, and “second-order” methods respectively. The Search-based methods (first-order based) detect edges through computing the degree edge strength i.e, (gradient magnitude). Next, this method then uses a calculated approximation of the local orientation of the edge (gradient direction) to find the maximum local direction of the gradient magnitude.

However, the zero-crossing based (second-order) edge detection methods search for zero crossings through a second-order derivative expression, computed straight from an image to find edges. This action is always known as the “zero-crossings of a non-linear differential expression” and sometimes called the zero-crossings of the Laplacian. Image processing involves the achievement of the use of the magnitude of the gradient first-order derivatives. The main reason why several filter types calculate gradient estimates in the x- and y- directions is that several edge detection techniques depend on the calculation of image gradients.

Eg: For a function \( f(x, y) \), as the dimensional column vector, the differential of ‘f’ at coordinates \((x, y)\) is denoted by:

\[
\nabla f = G[f(x, y)] = \begin{bmatrix} Gx \\ Gy \end{bmatrix} = \begin{bmatrix} \frac{df}{dx} \\ \frac{df}{dy} \end{bmatrix}
\]

(1)

Where \( \nabla f \) indicates the gradient of vectors, then \((x, y)\) points follow the highest change rate, which is normally the gradient direction. Vector’s magnitude of; \( |\nabla f| \) is defined as \( M(x, y) \):

\[
M(x, y) = \text{magnitude}(\nabla f) = |G| = \sqrt{Gx^2 + Gy^2}
\]

(2)

\[
\text{LoG}(x,y) = -\frac{1}{\pi \sigma^4} \left[ 1 - \frac{x^2 + y^2}{2 \sigma^2} \right] e^{-\frac{x^2 + y^2}{2 \sigma^2}}
\]

(3)

However, to approximate this quantity, the square root operation is used. Basing on the concept of gradient operator of edge detection, there is always a rapid change in the image gray value. Therefore, the method which bases on the derivation of the gradient operator is always used.
The expression below is used to calculate derivatives of Gx and Gy of an image in the x and y directions:

\[ G = \sqrt{Gx^2 + Gy^2} \]  

(3)

2.2. The common Techniques of edge detection

This section of the study discusses the various edge detection operators (algorithms) employed in the computer vision today to fulfill the purpose of edge detection.

2.2.1. The Canny operator.

The Canny edge detection technique is the optimum edge detector, commonly used for the edge detection activity, and was developed in 1986. The canny algorithm has been used to improve the edge detection activity compared to the techniques that were employed before its introduction [11]. This operator uses a 3x3 kernel pair, as shown in figure 1.

![Figure 1. The kernels of Canny operator](image)

2.2.2. The Prewitt operator.

Prewitt operator is an edge detection algorithm that was introduced by Prewitt in the year 1970. The Prewitt operator is almost alike Sobel operator its properties. When compared to other operators in performance, the Prewitt operator performs better than Roberts algorithm. This operator uses two kernels of size 3x3 as shown in figure 2.

![Figure 2. The kernels of Prewitt](image)

2.2.3. The Sobel operator.

Sobel operator is an edge detection technique founded by Sobel in the year 1970 and gives more emphasis to the image pixels that are nearer to the mask centre [9]. The Sobel algorithm detects edges at the points whose gradient is maximum and follows a criterion for determining the pixel image edge pixels [12]. This edge detection method applies 2 kernels of 3 x 3 size to detect edges of images. To implement this algorithm, one has to carry out the filtering process of an image using the left kernel [13]. This operator employs maximum points for its edge detection, thus making it a gradient-based operator. Figure 3 displays the kernels of the Sobel edge detection Algorithm.

![Figure 3. The kernels of Sobel edge detection Algorithm](image)
2.2.4. Roberts operator.
This operator was introduced in 1963, by Roberts. Roberts edge detection algorithm detects edges with a simple and fast 2-D gradient. This edge detector also requires high-frequency positions that usually matches with the edges. Roberts operator has an easy and fast structure that uses 2 kernels of size 2x2 to perform its function of edge detection. Figure 4 displays the kernels used by the Roberts edge detection technique.

![Figure 4. The kernels of Roberts](image)

Apart from the above three methods, other algorithms can be used for edge detection. Figure 5 illustrates various algorithms that are used for edge detection, categorized into two main groups. The first order-based methods and second-order methods.

![Figure 5. Categories of edge detection operators](image)

2.3. The Edge Detection procedure
One of the most commonly used procedures in digital image processing is Edge detection [14]. The process edge detection requires one to carry out three main steps for successful edge detection. The steps involved in this process are Filtering, Enhancement, and finally, Detection [15].

(1). Filtering: The first step of edge detection is filtering. The differences in the intensity values usually make images to become corrupted by unwanted noise, filtering is therefore meant to reduce noise in edge detection. The common types of noises include salt and pepper, as well as impulse Gaussian. Usually, the variability of white and black intensity values is contained in the Salt and pepper kind of noise. However, an increase in filtering leads to the loss of strength in produced edges.

(2). Enhancement: The second step toward edge detection is enhancement. Enhancement deals with understanding variations in the strength of the closest points, as this facilitates the successful detection of edges. Computing the gradient magnitude in enhancement helps to emphasize pixels in occurrences of significant changes in the local intensity values.

(3). Detection: It is the final step of the edge detection process. All points in images usually have non-zero gradient values. However, not all the points that exist in an image are detected as edges. A method must be applied to understand real points that must be considered, in other words, to differentiate
which points are to be used and which points are not supposed to be used. It is therefore in this last part that thresholding is applied to provide the edge detection criteria. The criteria for edge detection are illustrated in figure 6.

![Image of edge detection criteria]

Figure 6. The edge detection criteria

3. Methodology
This study uses both Canny and Sobel algorithms, one at a time, to output experimental results; to detect the edge of real-time objects/images. The algorithms were chosen for comparison. This study aims to compare the two algorithms’ nature of edge detection, majorly by analyzing the results obtained from employing both algorithms to understand which algorithm performs better than the other in the edge detection activity.

To capture the input, the study used the OpenCV library. OpenCV is an Open source Computer Vision library that supports several algorithms associated with Computer Vision and Machine Learning. The library works well with operating systems like Android, Windows, OS X, and Linux. The library supports a variety of programming languages such as Python, C++, and Java. This study uses OpenCV-Python to capture input data of real-time images and objects.

3.1. Edge detection using the Canny operator
The Canny method is the most frequent algorithm in the computer vision community to detect edges because of its good working ability. Previous studies show that the canny algorithm detects edges appropriately without moving features within the edges [16], as it employs a non-maxima suppression protocol to make the width of the edge equivalent to 1 pixel [17].

In canny operation processes, obtaining a smooth image edge \( k^\Lambda (s, t) \), is attained through convolving the image \( k(s) \) using the Gaussian function below:

\[
k^{s,t} = k(s, t) * (s, t, 6)
\]  

(4)

Three basic criteria of Canny. The Canny algorithm is ruled by three main criteria to produce the best edge detection results [18]. They are; (1) Good detection; where, the detector must choose several edges from the initial image. (2) Good localization; where, the marked edges must be so close to the edges in the real image, and lastly, (3) Minimal response; For any given edge of an image, it must be marked once. Also, noise must not lead to the production of unreal edges in the output.

Steps followed to implement the Canny edge detector. The Canny edge detector uses a few steps for its implementation. These steps are; (1) Using suitable Gaussian filter to Smoothen the image and remove unwanted image details, (2) Determining the and gradient direction as well as the gradient magnitude of pixels, (3) Marking each pixel as an edge or as the only if the pixel’s gradient magnitude is greater than the gradient of its two neighbors in the gradient direction, (4) Using hysteresis to remove weak edges, and lastly (5) Carrying out thresholding to output detected edges.

3.2. Edge detection using Sobel operator
While employing the Sobel algorithm for real-time edge detection, this study used the input of this method a real-time image and the method produced output as edges of the input image. To detect edges using Sobel edge detector (Sobel, 1970), the following steps must be followed; (1) Allow the input image, (2) Apply mask \( G_x, G_y \) to the input image, Next, (3) Sobel edge detector is applied to the initial image, (4) manipulation of \( G_x, G_y \) masks is then done differently on the input image, (5) Combining of
results to find the absolute magnitude of the gradient is done, and finally, (6) The magnitude is then detected as edges from the real image.

4. Results and Analysis
The proposed algorithms were implemented in Python version 3.4, with OpenCV library version 3.4. The operators were tested using a human face real image on an Intel R Core™ i3-3770K CPU at 4GHz, 8GB RAM, and Windows 10 Professional OS with NVIDIA GEFORCE. Next, the results of using both Canny and Sobel algorithms for edge detection using a real image are displayed below respectively.

![Original Image](image1)

![Result of Canny operator](image2)

**Figure 7.** Results with the Canny operator

![Original image](image3)

![Sobel edge magnitude](image4)

![Thresholded magnitude](image5)

**Figure 8.** Results with the Sobel operator

This section analyzes the results from employing both Canny and Sobel algorithms for edge detection using real images. As shown above, in the results of both algorithms, the Canny algorithm in figure 7 (b) displayed an output with better visible edges compared to the Sobel algorithm. This is because the algorithm uses double thresholding to detect edges. Furthermore, the Canny procedure of edge detection is based on the characteristic of intensity-values of considered pixels and this makes the algorithm a good signal-to-noise ratio so that all edges can be found even if the potential field data quality is poor. On the other hand, the Sobel algorithm produced unpleasant edges in the results of this study, seen in figure 8 (b), (c) compared to the Canny algorithm. This is because Sobel operator is less sensitive to noise, in a way that when the noise level in an image increases, the magnitude of the edges falls and makes the algorithm to lose accuracy. When edge detection was performed on the original real time image of 276KB using Canny edge detection algorithm, edge size detected was 22.7 KB. On the other hand, when edge detection was performed on a real image of 201 KB using Sobel method, the edge detected was 308 KB. Following the comparison, the Canny egde detection technics proved to be better than Sobel and are good at saving storage.

5. Conclusion
This paper briefly discussed the task of edge detection and various approaches used for edge detection. Next, edge detection was implemented in real-time using two algorithms; ie, The Canny and Sobel
algorithms, and results displayed that the Canny edge detection algorithm has a good visualization of edges of images compared to the Sobel algorithm. The authors hope that this will help to improve the activities in which edge detection is commonly used such as in hospitals for iris detection, surveillance systems, and traffic systems by using the best edge detection algorithm, i.e., Canny for edge detection activities. This can be more comfortable since real-time edge detection requires no intervention of sensors for edge detection.

In conclusion, using python-OpenCV made this study simpler for the authors compared to other programming languages. However, future studies should be focused on speeding up the process of real-time edge detection by using parallel programming/CUDA.

Acknowledgments
We would like to thank Universitas Atma Jaya Yogyakarta, department of Magister Informatika, International Students office, Kemitraan Negara Berkembang (KNB) scholarship program, and the government of Indonesia for supporting this work. We also thank our anonymous reviewers, whose fair and thorough feedback significantly improved this paper.

References
[1] M. Yıldırım and F. Kacar, “Retina-inspired neuromorphic edge enhancing and edge detection,” AEU - Int. J. Electron. Commun., vol. 115, p. 153038, 2020, doi: 10.1016/j.aeue.2019.153038.
[2] F. Orujov, R. Maskeliūnas, R. Damaševičius, and W. Wei, “Fuzzy based image edge detection algorithm for blood vessel detection in retinal images,” Appl. Soft Comput. J., vol. 94, 2020, doi: 10.1016/j.asoc.2020.106452.
[3] K. Sha, T. A. Yang, W. Wei, and S. Davari, “A survey of edge computing based designs for IoT security,” Digit. Commun. Networks, 2020, doi: 10.1016/j.dcan.2019.08.006.
[4] Y. Qin, Y. Wu, Q. Wang, and S. Yu, “Method for pests detecting in stored grain based on spectral residual saliency edge detection,” Grain Oil Sci. Technol., vol. 2, no. 2, pp. 33–38, 2019, doi: 10.1016/j.gaost.2019.06.001.
[5] L. Wang, Y. Shen, H. Liu, and Z. Guo, “An accurate and efficient multi-category edge detection method,” Cogn. Syst. Res., vol. 58, pp. 160–172, 2019, doi: 10.1016/j.cogsys.2019.06.002.
[6] A. Kumar and S. Raheja, “Edge Detection using Guided Image Filtering and Enhanced Ant Colony Optimization,” Procedia Comput. Sci., vol. 173, no. C, pp. 8–17, 2020, doi: 10.1016/j.procs.2020.06.003.
[7] Y. Shi, Z. Huo, J. Qin, and Y. Li, “Automatic prior shape selection for image edge detection with modified Mumford–Shah model,” Comput. Math. with Appl., vol. 79, no. 6, pp. 1644–1660, 2020, doi: 10.1016/j.camwa.2019.09.021.
[8] M. Hagara, R. Stojanović, T. Bagala, P. Kubinec, and O. Ondráček, “Grayscale image formats for edge detection and for its FPGA implementation,” Microprocess. Microsyst., vol. 75, 2020, doi: 10.1016/j.micpro.2020.103056.
[9] N. Mathur, S. Mathur, and D. Mathur, “A Novel Approach to Improve Sobel Edge Detector,” Procedia Comput. Sci., vol. 93, no. September, pp. 431–438, 2016, doi: 10.1016/j.procs.2016.07.230.
[10] A. K. Bharodiya and A. M. Gonsai, “An improved edge detection algorithm for X-Ray images based on the statistical range,” Heliyon, vol. 5, no. 10, p. e02743, 2019, doi: 10.1016/j.heliyon.2019.e02743.
[11] Ş. Öztürk and B. Akdemir, “Comparison of Edge Detection Algorithms for Texture Analysis on Glass Production,” Procedia - Soc. Behav. Sci., vol. 195, pp. 2675–2682, 2015, doi: 10.1016/j.sbspro.2015.06.477.
[12] K. Zhang, Y. Zhang, P. Wang, Y. Tian, and J. Yang, “An improved sobel edge algorithm and FPGA implementation,” Procedia Comput. Sci., vol. 131, pp. 243–248, 2018, doi: 10.1016/j.procs.2018.04.209.
[13] D. Vikram Mutneja, “Methods of Image Edge Detection: A Review,” J. Electr. Electron. Syst.,
[14] M. Hagara, R. Stojanović, T. Bagala, P. Kubinec, and O. Ondráček, “Grayscale image formats for edge detection and for its FPGA implementation,” Microprocess. Microsyst., vol. 75, p. 103056, 2020, doi: 10.1016/j.micpro.2020.103056.

[15] R. S. R. Prasad and S. Suresh, “A Review on Edge Detection Algorithms,” vol. 4, no. 1, pp. 7–11, 2016.

[16] F. M. Alzahrani and T. Chen, “A Real-Time Edge Detector,” vol. 378, no. 1997, pp. 363–378, 2014.

[17] L. Xuan and Z. Hong, “An improved canny edge detection algorithm for Detecting Brain Tumors in MRI Images,” Proc. IEEE Int. Conf. Softw. Eng. Serv. Sci. ICSESS, vol. 2017-Novem, pp. 275–278, 2018, doi: 10.1109/ICSESS.2017.8342913.

[18] P. K. Chakravathi, K. V. Kumar, P. Devi Pradeep, and D. Suresh, “FPGA based architecture for realtime edge detection,” Glob. Conf. Commun. Technol. GCCT 2015, no. Gcct, pp. 12–17, 2015, doi: 10.1109/GCCT.2015.7342615.