Improved Performance of Canny Edge Detection in Low-Light Conditions

Rishikesh S (21BCE5304)
Github link: https://github.com/rishi292978/DIP.git
Gautham P Harish (21BCE5382)
Aditya Kushwaha(21BCE6104)

VIT university

***Abstract*—In the realm of image processing and computer vision, edge detection is a critical operation. Among various edge detection methods, the Canny Edge Detection algorithm is widely recognized for its superior performance. However, its effectiveness can be significantly compromised in low-light conditions. This paper proposes a real-time light enhancement algorithm that substantially enhances the overall performance of the Canny Edge Detection algorithm under low-light conditions. The proposed algorithm is designed to improve edge recognition in dimly lit environments, which can be particularly beneficial in applications such as autonomous navigation and surveillance systems. The algorithm's effectiveness is demonstrated through rigorous testing and comparison with existing methods. The results indicate a significant improvement in edge detection performance, paving the way for more reliable image processing in low-light scenarios.**

Keywords- Edge detection, Canny edge algorithm, Facial Expressions, Active Vision, Surveillance, Mobile Robots, Slam.

I. Introduction

Edge detection is a fundamental operation in image processing and computer vision. It is used to identify the boundaries of objects within an image. The Canny Edge Detection algorithm is one of the most commonly used methods for edge detection due to its superior performance. However, its performance can be significantly affected in low-light conditions. This literature review focuses on the improved performance of the Canny Edge Detection algorithm in low-light conditions.

Edge detection is a fundamental operation in image processing and computer vision. It is used to identify the boundaries of objects within an image. These boundaries, or edges, are significant attributes of an object or a certain location in relation to feature recognition, classification of an object, location, and mapping. An edge is essentially a pair of affixed pixels forming the boundary between two disjointed regions. Such discontinuities usually fit into one of four categories: level discontinuities, area path discontinuities, changes in material properties, and variations in scene illumination.

Edges can be further classified as horizontal, right, and diagonal edges. Therefore, edge detection is a method of segmenting an image into discontinuous parts. It's a method that is certainly preferred for digital images, such as design recognition, image morphology, and function reduction. Edge recognition allows people to look at photo features. For a start, this is certainly a considerable level. That is gray. This surface denotes the finish of morphology and features. Only the final features. Within an image while keeping its functions, which are often architectural. Detection providers. They are this surface denotes the finish of morphology and features.

The Canny Edge Detection algorithm is one of the most commonly used methods for edge detection due to its superior performance. However, its performance can be significantly affected in low-light conditions. Although this feature detection algorithm performs well in great lighting conditions, it doesn't identify edges or any feature in dim-light-to-dark dilemmas. To address this issue, an algorithm has been proposed which will be used to enhance the image or video before applying the Canny Edge, which in turn optimizes the image/video frame-by-frame and detects many more edges/features, therefore improving the overall performance by approximately 60%.

This enhancement is of great advantage to the OPENCV Python library and Canny Edge detection library, which are more beneficial than mathematical digital tools. This study deduces a MATLAB algorithm that produces far better outputs. Python and MATLAB are utilized in this research study. Python is one of the most widely used open-source programming languages

II. Literature review

A. **A Distributed Canny Edge Detector and Its Implementation On FPGA**: This research paper presents an in-depth exploration of a distributed Canny edge detection algorithm. The algorithm is designed to optimize memory usage, reduce latency, and enhance throughput, all while maintaining the edge detection performance when compared to the original Canny algorithm. The novel algorithm employs a low-complexity 8-bin non-uniform gradient magnitude histogram to compute block-based hysteresis thresholds. These thresholds are integral to the functioning of the Canny edge detector, as they determine which edges are considered significant. Furthermore, the paper presents an FPGA (Field-Programmable Gate Array)-based hardware architecture of the proposed algorithm. This provides valuable insights into the practical implementation of the theoretical concepts discussed, and demonstrates the feasibility of implementing complex edge detection algorithms on hardware platforms.

B. **An improved adaptive threshold canny edge detection algorithm**: This paper proposes an innovative adaptive threshold for the Canny Edge Detection algorithm, utilizing an Actor-Critic Algorithm. The research introduces an edge preserving filter as a replacement for the original Gaussian filter. This enhancement significantly improves the edge detection capabilities of the algorithm by preserving more edge information while reducing noise. The paper also presents a new design for templates that calculate the magnitude and direction of image gradient from x direction, y direction, and two oblique directions. This multi-directional approach allows for more accurate edge detection, as it can capture edges of various orientations. The Otsu algorithm is employed to calculate the thresholds, thereby eliminating the need for repeated threshold setting and enhancing the efficiency of the process.

C. **Object Tracking Algorithm Implementation for Security Applications**: This paper discusses the importance of object tracking, a key component for many machine vision applications. It presents a method using a semi-foreground map for stable training of the proposed Spatio-Temporal Fusion Network (STFN). The STFN has the capability to extract temporal and spatial information using a temporal network and a spatial network. This dual-network approach allows the STFN to provide a comprehensive understanding of the object's movement and position, making it highly effective for tracking objects in real-time video feeds.

D. **Canny edge detection enhancement by scale multiplication**: This paper examines the technique of scale multiplication within the framework of Canny edge detection. A scale multiplication function is defined as the product of the responses of the detection filter at two scales. This approach allows the algorithm to capture edges at different scales, making it more robust to variations in edge size. Edge maps, which are crucial for understanding the boundaries and features of objects within an image, are constructed as the local maxima by thresholding the scale multiplication results. This approach ensures that only the most significant edges are preserved, reducing the likelihood of false edge detection.

E. **Edge detection algorithm of plant leaf image based on improved Canny**: This paper proposes an improved Canny edge detection algorithm specifically designed for plant leaf images. The algorithm employs an enhanced filtering method to de-noise the image, improving the clarity and accuracy of the edge detection. The four-direction gradient template is used to calculate the gradient amplitude, providing a comprehensive understanding of the edges in the image. This approach is particularly effective for plant leaf images, as it can accurately capture the complex edge structures often found in these images.

F. **Research on image text recognition based on canny edge detection algorithm and k-means algorithm**: This paper proposes a solution to improve the accuracy of image text recognition effectively. It employs the canny algorithm to process edge detection of text, and the k-means algorithm for cluster pixel recognition. This combination of techniques allows for a more accurate and efficient text recognition process. The use of the canny algorithm ensures that the text edges are accurately detected, while the k-means algorithm groups similar pixels together, allowing for effective text segmentation.

III. Proposed Method

The proposed work has employed the medial side. This is certainly the method that Canny Edge is explaining, as the advanced detection technique is just a multistep algorithm to identify the edge of any input photo. The below mentioned to be followed while finding the edges of an image.

$$S = I * g(x,y) = g(x,y) * I \quad (1)$$

$$g(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{x^2+y^2}{2\sigma^2}} \quad (2)$$

Gausian Distribution = g(x,y).

Where I = input image

Calculate the derivative of the filter with respect to X and Y dimensions and convolve with the gradient magnitude along the dimensions [4]. Moreover, the direction of the image can be calculated using the tangent of the angle between the two dimensions.

$$\nabla S = \nabla(g * I) = (\nabla g) * I$$
(3)

$$\nabla S = \begin{bmatrix} gx \\ gy \end{bmatrix} * I = \begin{bmatrix} gx * 1 \\ gy * 1 \end{bmatrix} \quad (4)$$
∇

$$\nabla g = \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix} = \begin{bmatrix} gx \\ gy \end{bmatrix} \quad (5)$$

The above convolution results in a gradient vector that has magnitude and direction.

$(S_x, S_y)\, Gradient\ Vector$

$$magnitude = \sqrt{{S_x}^2 + {S_y}^2} \quad (6)$$

$$direction = \theta = tan^{-1}\frac{S_y}{S_x}$$
(7)

Along with a benefit, it's generally seen that few things make the presence regarding the advantage better. So we can neglect those side things which don't contribute more towards feature visibility [5]. To ultimately achieve the same, the non is used by us Maximum Suppression method. Here we mark the real things from the bend regarding the edge where in fact the magnitude is largest. This is got by buying a maximum along with a piece regular to the curve [6].

Consider the edge of the figure, this is certainly below, which has three advantage points. Believe the point (x, y) the point obtains the gradient, that is, the largest of sides. Search for the side points of the course perpendicular to the advantage and verify if their particular gradient is less than (x,y). Then those non-maximal things can be controlled .

The image/video frame has been optimized by employing a set of operations meant to modify the image before applying the Canny Edge advantage detection method. Transform the feedback image or each video frame to grayscale: an image that is grayscale one in which the worth of each pixel is a single sample indicating simply a quantity of light; this is certainly, conveys just intensity information in portrait digital photography, computer-generated imaging, and colorimetric [7]. Grayscale photographs, often known as black-and-white or monochrome, which are certainly grey, are entirely made up of grayscale hues.

The contrast goes from black at the level that is certainly the least expensive to white at the greatest. The grayscale images need less information for each pixel compared to fullcolor photos. The Gray color is one where the purple, green, and blue elements all have equal intensity in the RGB room, and thus it's just required to specify an intensity that is solitary for every pixel. Hence, further computations become a good deal easier [8]. Opencv is a massive computer system that is certainly an open-source, machine learning, and image processing library. It may analyze photos and films to discover products, people, and sometimes even the handwriting of a human. In this guide, we are going to view just how to convert a color movie to grayscale [9]. *A. Method:*

• Import the cv2 module.

• The video clip file is changed making use of the cv2.VideoCapture() function.
• Creating a cycle is certainly countless.
• With the read() purpose, retrieve the video clip frames within the cycle.
• To transform the frame to grayscale, use the cv2.cvtColor() function aided by the argument cv2. Color BGR2GRAY.

The cv2.imshow() purpose is employed to show the framework.

Calculate the histogram regarding the image, this is certainly grayscale: The histogram of an image offers details about the brightness comparison and power circulation in an image [10]. cv2.calcHist() has been applied to calculate. The histogram of the image in Fig. 4. The histogram depends on the regularity of the luminance in the image. Therefore, the colors which are purchased are graphed on the x-axis in addition to the number of pixels for every shade regarding the y-axis, giving an overview associated with the circulation of colors in the image. Whenever an image is converted to grayscale, the colors tend to be limited to 100 or 255 levels of gray [11]. A histogram is just a visual depiction associated with regularity with which a color that is different can be found in an image. It is a plot or graph that depicts an image's intensity of circulation. This is a story with pixel values (usually ranging from 0 to 255) within the X-axis; additionally, the true range of pixels within the photo is within the Y-axis [12].

The cv2.calcHist() strategy in Python opencv determines the histogram of 1 or even more arrays. This enables you to calculate the histogram of single and photos which are multichannel photos.

Maskmask: it's an image of a mask. The histogram associated with the image that is supplied as "None." tSize: This parameter reflects the actual range of bins provided as a record.
Range: the power is represented by an appreciative range.
Needed Segments:

- opencv (cv2)
- Numpy

The signal for processing the histogram of a total image is certainly the grayscale supplied under. Hist is really a (256,1) array in this instance. Each range product presents the number of pixels with the tone value that is connected. In this case, the histograms are plotted using the matplotlib package. A graphing hist() method.

Histogram equalization can be used automatically to change the intensity levels of picture pixels. Histogram equalization entails adjusting the intensity values so that the resulting image closely resembles a specified histogram. Histeq, the histogram equalization function, by default attempts to match a histogram, that is, flat 64 containers; however, you could provide a different histogram instead. Hexagonal histogram equalization is a technique for modifying strength values. Histeq yields an image with pixel values equally distributed within the range.

The histogram is a collective histogram in which the straight axis provides not only the counts for the single bin, but also counts for the container plus all bins for lower values of the reaction variable. After calculating the histogram's collective distribution, locate the items in the histogram for clipping and calculate the value of variables alpha and beta, which determines the factor by which the image power must be altered based on the histogram's collective distribution. The factor alpha will be beta and increased in this product to obtain an illuminated version of the image.

After optimizing the brightness and contrast of the image by modifying the power distribution, we make the following adjustments using the Canny Edge detection method for much better side detection:

The image is convolved through a Gaussian filter, which is really a low-pass filter that removes the outlier pixels or even high frequency elements through the image to cut back sound in Gaussian Blur operation. The edges have power, this is certainly low eliminated. The opencv Gaussian filtering provides a cv2.GaussianBlur() solution to blur a graphic by using Gaussian Kernel. Each pixel in an image is multiplied by a Gaussian Kernel, which is a square array. Conversion to grayscale before applying Canny Edge detection. The Canny Edge technique works closely with a simpler picture, and the grayscale photos are mostly the ones that are simplified. Grayscale is probably decreasing complexity: coming from a 3D pixel value (Roentgen, G, B) up to a price, that is, 1D. Numerous jobs like advantage detection try not to fare better with 3D pixels. Converting a graphic to grayscale does not break down the image by any means because after all, the Canny Edge technique only produces black colored photos which are white. So grayscale doesn't impact edge detection in any way.

Now, we use the Canny Edge Detection on the edge (this is certainly cv2. Canny strategy in OpenCV. Canny Edge advantage recognition is really an algorithm that multistage detects an array of edges in pictures. When we compare the output, the true difference can be seen in the total results. When a picture is preprocessed using the method described above, many more functions are detected than in the original image.

A Gaussian filter is used to remove noise from the input image. Processing the Gaussian filter byproduct to look for the gradient of image pixels with magnitude along the x and y dimensions. Considering a small grouping of neighbors for almost any curve within a direction perpendicular to the supplied edge, we suppress the benefit that this pixel is undoubtedly non-max. Finally, we use the Hysteresis

Thresholding strategy to protect pixels larger than the gradient magnitude and ignore pixels smaller than the paid-off limitation value. Before delving into the steps, here are the three conclusions reached by J.K Canny Edge, the algorithm's creator: Good Detection: The best sensor must remove the possibility of false positives and negatives. Good localization: The detected edges must be close to the genuine edge response. Single: The detector must reunite only one point from each side. Steps to take the Canny Edge Algorithm: Image Smoothing or Noise Removal: During the noise, the pixel is almost certainly not similar to its neighbouring pixels that are neighbouring. This may result in poor or recognition that is unacceptable. The noise, preventing the desired edges in outcome images. In the under instance, we are convolving the filter. This is certainly the Gaussian kernel g (x, y) having an image we want to avoid comparable; we apply a Gaussian filter that will evolve due to the photo and be removed. Here we wish to make sure that any provided pixel should really be comparable to its pixels which are neighboring outputs, so we use the matrix [1 1, 1] to keep the similarity between pixels and take away the sound.

DATASET DESCRPTION

The LOL (Low-Light) dataset is a meticulously curated benchmark dataset that has been specifically designed to address the real-world challenge of low-light image enhancement. It comprises a total of 500 pairs of images, with each pair consisting of a low-light image and a corresponding normal-light image. The dataset is neatly divided into 485 pairs for training and 15 pairs for testing. This division provides a balanced and robust platform for both training and validating low-light enhancement algorithms, ensuring that the models developed using this dataset are well-equipped to handle real-world scenarios.

The low-light images in the dataset are not artificially darkened versions of the normal-light images. Instead, they authentically represent the noise and quality degradation that occurs during the photo capture process in low-light conditions. This authenticity makes the LOL dataset particularly valuable for developing algorithms that can handle real-world lighting variations. It ensures that the models trained on this dataset are not just theoretically sound but also practically applicable.

A significant portion of the images in the LOL dataset are indoor scenes. This is a deliberate choice, as indoor environments often present challenging lighting conditions for image processing algorithms. Indoor scenes can have complex lighting due to multiple light sources, shadows, and reflections. By including a wide variety of indoor scenes, the LOL dataset provides a rich testing ground for low-light enhancement algorithms. It ensures that these algorithms are tested against a wide range of scenarios, thereby increasing their robustness and applicability.

All the images in the LOL dataset have a resolution of 400×600 pixels. This uniformity in resolution helps maintain consistency while training and testing algorithms for low-light image enhancement. It ensures that all images contribute equally to the learning process, regardless of their content. This uniformity also simplifies the preprocessing steps, as all images are of the same size and do not need to be resized or cropped.
The LOL dataset is structured into two main data fields: highlight_images and lowlight_images. the highlight_images field contains images taken under normal light conditions, while the lowlight_images field contains the corresponding images taken under low light conditions. this paired structure allows for direct comparison between the original and enhanced images, making it easier to evaluate the performance of low-light enhancement algorithms.

In conclusion, the LOL dataset is a well-structured, realistic, and challenging dataset that serves as a valuable resource for the development and benchmarking of low-light image enhancement algorithms. Its use of real-world low-light images, its focus on indoor scenes, and its structured format make it particularly relevant for applications such as surveillance, autonomous navigation, and any other application that involves image or video capture in low-light conditions. The LOL dataset, with its unique features and comprehensive structure, is poised to drive forward the field of low-light image enhancement, enabling the development of more effective and robust algorithm

References

[1] S. Sivanantham, N. Nitin Paul and R. Suraj Iyer, "Object Tracking Algorithm Implementation for Security Applications", Far East Journal of Electronics and Communications, vol. 16, no. 1, 2016.

[2] P. Bao, L. Zhang and X. Wu, "Canny edge detection enhancement by scale multiplication", IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 9, pp. 1485-1490, Sep. 2005.

[3] "A Distributed Canny Edge Detector and Its Implementation On FPGA", School of Electrical Computer and Energy Engineering Arizona State University IEEE, pp. 500-505, 2011.

[4] "An improved adaptive threshold canny edge detection algorithm", IEEE International Conference on Computer Science and Electronics Engineering, pp. 164-168, 2012.

[5] "Edge detection algorithm of plant leaf image based on improved Canny", Shen Xizhen; Zeng Wei; Guo Yiling; Yin Shengyang.

[6] "Research on image text recognition based on canny edge detection algorithm and k-means algorithm", Jinxiu Xu,Fangsheng Wu,Changan Zhu

[7] "Improved Performance of Canny Edge Detection in Low-Light Conditions",P. Bao, Lei Zhang, Xiaolin Wu