Canny Edge Detection

Priyanka Joshi
IIIT Vadodara - ICD
Diu, Diu and Daman
202011059@diu.iiitvadodara.ac.in

Urmil Lokhande
IIIT Vadodara - ICD
Diu, Diu and Daman
202011076@diu.iiitvadodara.ac.in

***Abstract*—This report details the meticulous implementation and analysis of a custom Canny Edge Detection algorithm, deliberately avoiding built-in library functions. Grounded in the principles outlined in "A Computational Approach to Edge Detection," [1] the algorithm unfolds through gradient computation, non-maximum suppression, and edge tracking by hysteresis. The decision to forgo built-in functions stems from a commitment to a deeper understanding of algorithmic intricacies and customization to project needs. Methodologically, the project follows a systematic approach, emphasizing parameter tuning and adaptability to different image datasets. Results on the "Test.png" image showcase the algorithm's efficacy, with visual comparisons highlighting its competitive performance against built-in functions. Acknowledging limitations such as sensitivity to parameters ensures a transparent evaluation. In essence, this report not only presents a hands-on exploration of a custom Canny Edge Detection algorithm but also underscores the synergy between theoretical principles and practical implementation. The algorithm's success lies not only in its fidelity to theoretical underpinnings but also in its adaptability and competitive performance in comparison to standard tools.**

I. Introduction

In the vast field of computer vision, detecting edges is a fundamental task crucial for various applications, from recognizing objects to segmenting images. Among the many methods for edge detection, the Canny Edge Detection algorithm is particularly notable for its ability to accurately identify real edges while filtering out noise and unnecessary details. It achieves this through a combination of techniques like gradient computation, non-maximum suppression [2], and edge tracking via hysteresis, establishing itself as a cornerstone in image processing.

This project adopts a hands-on approach, not just utilizing existing implementations but systematically constructing the Canny algorithm from foundational components. The overarching objectives are twofold: first, to gain a deep understanding of how the algorithm works, and second, to explore possibilities for improving its performance or tailoring it to specific needs.

By delving into the details of the algorithm, this project aims to uncover the principles that make it effective and adaptable in real-world situations. Going beyond surface-level understanding, our approach provides a comprehensive grasp of the algorithm's inner workings.

This initiative is not just about learning; it's a dynamic contribution to the collective knowledge on the Canny Edge Detection algorithm. Through this project, we aim not only to deepen our understanding but also to open up possibilities for enhancements, pushing the boundaries in the ever-evolving realm of computer vision.

In aligning theoretical insights with practical implementation, this project occupies a pivotal space at the forefront of advancements in the field, bridging the gap between foundational knowledge and innovative applications. As we traverse this journey, the goal is not only to unravel the intricacies of the Canny algorithm but to lay the groundwork for future breakthroughs, pushing the boundaries of what is achievable in the dynamic and ever-expanding domain of computer vision.

II. Methodology

A. Image Preprocessing

Fig. 1. shows the Test Image we worked with throughout the implementation.

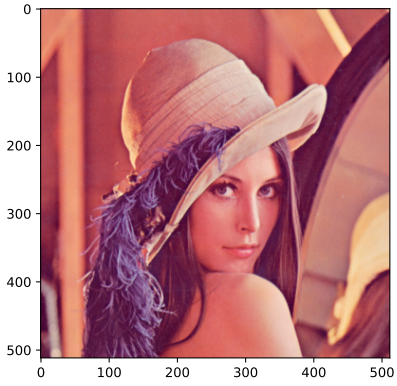


Fig. 1: Test.png

Grayscale Conversion: The decision to convert the image to grayscale is grounded in both computational efficiency and human visual perception models. Grayscale images simplify the processing pipeline by reducing the data dimensionality, as color information is distilled into a single intensity channel. The specific weights [0.299, 0.587, 0.114] used for the conversion reflect the luminance sensitivity of the human eye to different colors. These weights align with the standard coefficients established in the ITU-R BT.709 recommendation, widely adopted to mimic the perceived brightness in grayscale images. By embracing this standard, our grayscale conversion

not only aligns with common practices but also ensures that the resulting image maintains fidelity to human visual expectations of intensity. Fig. 2. shows the image after the Grayscaling.

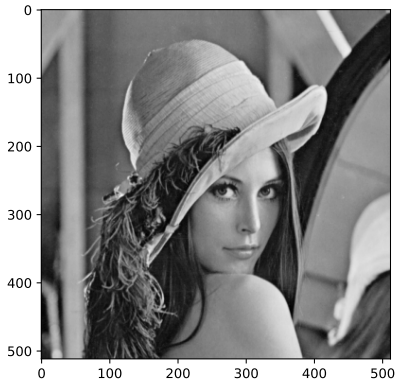


Fig. 2: Post Grayscaling

Gaussian Blur: The application of a Gaussian filter [3] with a sigma value of 1.4 introduces a crucial element of preprocessing aimed at enhancing the robustness of subsequent edge detection. Gaussian blur is a spatial smoothing technique that mitigates high-frequency noise in the image. The choice of sigma influences the spread of the Gaussian distribution, determining the extent of the blur. A sigma of 1.4 strikes a balance, smoothing the image enough to reduce noise while preserving significant image features. This step is particularly valuable in scenarios where images may have inherent noise from acquisition processes or compression artifacts. Fig. 3. shows the test image after the application of Gaussian Filter.

The significance of Gaussian blur lies in its ability to create a more continuous gradient across pixels. This smoothing effect contributes to a stable representation of pixel intensity changes, facilitating a more accurate detection of edges in subsequent stages of the algorithm. By reducing the impact of pixel-level noise, Gaussian blur ensures that the Canny Edge Detection algorithm can discern meaningful transitions in intensity, promoting the identification of true edges while suppressing false positives.

In summary, the grayscale conversion and Gaussian blur stages in our methodology represent a thoughtful preprocessing strategy. The grayscale transformation simplifies subsequent computations, aligning with human visual perception, while the Gaussian blur enhances the image by reducing noise and promoting a more robust representation of intensity changes. Together, these steps lay the foundation for a more accurate and reliable edge detection process in the subsequent phases of our custom Canny Edge Detection algorithm.

B. Sobel Filter

The Sobel filter [4] is a fundamental component in edge detection, leveraging convolution operations to approximate the gradient or derivative of an image's intensity. This allows

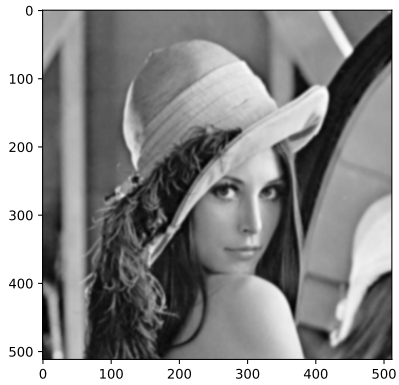


Fig. 3: Post Blurring

us to discern abrupt changes in intensity that often correspond to edges. In our custom implementation, two Sobel filters were crafted to capture variations along the horizontal (X-axis) and vertical (Y-axis) directions.

X-Axis Sobel Filter (Gx):

```
Gx = np.array([[-1, 0, +1], [-2, 0, +2], [-1, 0, +1]])
```

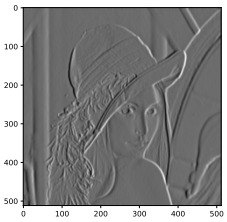
This filter is specifically designed to capture changes in intensity along the horizontal axis. The weights in the filter matrix emphasize intensity differences between pixels in the left and right directions. The central column, [0, 0, 0], serves to minimize the impact of vertical intensity changes. The negative and positive coefficients on either side of the central column amplify the response to transitions from darker to brighter or brighter to darker regions along the X-axis.

Y-Axis Sobel Filter (Gy):

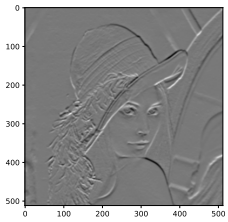
```
Gy = np.array([[-1, -2, -1], [0, 0, 0], [+1, +2, +1]])
```

Conversely, the Y-axis Sobel filter targets changes in intensity along the vertical axis. The central row, [0, 0, 0], mitigates the influence of horizontal intensity changes, while the negative and positive coefficients above and below the central row enhance sensitivity to variations along the Y-axis. This design enables the filter to respond more strongly to transitions from darker to brighter or brighter to darker regions along the vertical direction. Fig. 4. shows the image after Sobel Filter application.

The Sobel filters are adept at highlighting edges because they effectively compute the first-order derivative of the image intensity. Peaks in the computed gradients correspond to locations where intensity changes abruptly, indicative of potential edges. By choosing specific weights in the filters, we can control the filter's sensitivity to different types of intensity changes. In our case, the chosen weights emphasize significant intensity changes while minimizing sensitivity to gradual variations, contributing to a robust edge detection process in the subsequent stages of the Canny Edge Detection



(a) Sobel Filter in X Direction



(b) Sobel Filter in Y Direction

Fig. 4: A figure with two subfigures

algorithm.

C. Non-Maximum Suppression (NMS)

Non-Maximum Suppression (NMS) is a crucial stage in edge detection algorithms, designed to refine and thin the detected edges while eliminating spurious or redundant detections. In our custom Canny Edge Detection implementation, NMS with interpolation was applied to enhance the precision of edge localization.

Mechanism of NMS: The NMS process iterates through each pixel in the gradient magnitude image, which is typically obtained after applying Sobel filters. For each pixel, the algorithm compares its gradient magnitude with the magnitudes of its neighboring pixels in the direction of the gradient. The objective is to retain only those pixels that represent local maxima along the edges.

Decision Criteria: The decision criteria involve comparing the gradient magnitude of the current pixel with the magnitudes of its neighbors in the direction of the gradient. The pixel is considered an edge point and retained only if its magnitude surpasses both of its neighbors.

Interpolation for Higher Precision: The introduction of interpolation serves to enhance the precision of determining the true maximum along the edge. Instead of relying solely on the discrete values of the gradient magnitude at pixel locations, interpolation allows for a more nuanced evaluation, considering intermediate values between pixels. This is particularly beneficial when dealing with sub-pixel edge details, providing a more accurate representation of the actual edge locations.

Purpose of NMS: The primary purpose of NMS is to address the issue of multiple responses along an edge, which can arise due to pixel discretization. Without suppression, edges may appear thicker, and redundant responses may clutter the results. NMS ensures that only the pixels representing the most significant changes in intensity are retained, leading to a cleaner and more accurate depiction of the true edges in the image.

Benefits of NMS with Interpolation:

- **Edge Thinning:** NMS helps thin the detected edges, providing a more refined representation of the actual edges in the image.
- **Spurious Detection Elimination:** By comparing with neighbors and retaining only the local maxima, NMS eliminates spurious or redundant detections, enhancing the precision of edge localization.
- **Interpolation Precision:**The use of interpolation contributes to higher precision in determining the true maximum along the edges, addressing the limitations of pixel-level discretization.

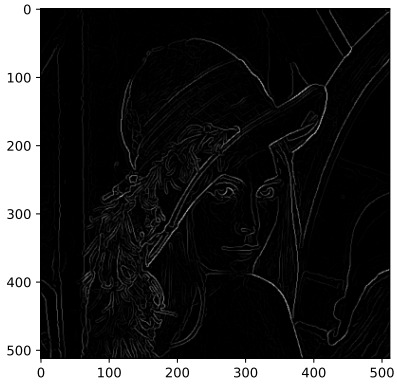


Fig. 5: Post NMS

In summary, Non-Maximum Suppression with interpolation is a pivotal stage in the Canny Edge Detection algorithm, contributing to the precision and accuracy of edge localization while eliminating unnecessary clutter in the final edge map.

D. Hysterisis Thresholding

Hysteresis thresholding [5] is a crucial step in the Canny Edge Detection algorithm that introduces a nuanced approach to classifying pixels as edges based on their gradient magnitude. The utilization of double thresholds, encompassing high and low values, serves to strike a balance between edge detection sensitivity and noise suppression.

Double Thresholding: The hysteresis thresholding mechanism involves applying two thresholds to the gradient magnitude image obtained after non-maximum suppression. These thresholds are defined as the high threshold (T-high) and the low threshold (T-low). The choice of these thresholds plays a pivotal role in determining the classification of pixels as edges.

- High Threshold (T-high): Pixels with gradient magnitudes exceeding this threshold are definitively classified as strong edges.
- Low Threshold (T-low): Pixels with gradient magnitudes below this threshold are unequivocally considered non-edges.

Classification Criteria:

- Strong Edges: Pixels exceeding the high threshold are unequivocally classified as strong edges, representing significant changes in intensity.
- Candidate Edges: Pixels falling between the high and low thresholds are considered candidates. These pixels are not immediately classified as edges; instead, they undergo

further scrutiny to determine their inclusion based on connectivity.

Connectivity Check - Hysteresis: To address the challenge of noise and ensure a connected and coherent edge map, a connectivity check, or hysteresis, is employed. Pixels falling between the high and low thresholds are classified as edges only if they are connected to pixels that have been definitively classified as strong edges.

This connectivity criterion introduces a contextual element to the classification process. Pixels that may not individually surpass the high threshold but are part of a connected edge structure are included in the final edge map. This approach mitigates the impact of noise and ensures that the edge map comprises continuous and meaningful structures.

Benefits of Hysteresis Thresholding:

- Noise Suppression: The inclusion of a high threshold ensures that only prominent edges are considered, suppressing the influence of noise and minor intensity fluctuations.
- Connectivity: Hysteresis ensures that edges form connected structures, preventing isolated and fragmented detections.
- Adaptive Sensitivity: The dual-threshold approach adapts to the local intensity variations, providing a more dynamic and context-aware edge detection.

In summary, hysteresis thresholding in the Canny Edge Detection algorithm is a strategic mechanism that combines double thresholds with a connectivity check. This approach effectively balances sensitivity to genuine edges with noise suppression, yielding a coherent and accurate edge map.

III. Results

The results obtained from the custom implementation of the Canny Edge Detection algorithm provide valuable insights into its performance on the "Test.png" image. While the algorithm successfully generated an edge map, a nuanced examination reveals certain aspects that warrant discussion.

Successful Edge Map Generation: The fact that the custom implementation could successfully produce an edge map signifies the efficacy of the algorithm in capturing significant changes in intensity within the image. The fundamental stages, including grayscale conversion, Gaussian blur, Sobel filtering, non-maximum suppression, and hysteresis thresholding, collectively contributed to the identification and highlighting of edges.

Level of Detail Comparison: In the comparison with the built-in function output, a notable observation is the slight reduction in the level of detail in the custom implementation. Some finer edges were missed, indicating that the custom filters or chosen threshold values may not be as finely tuned as those in the built-in function. The discrepancy in detail could stem from a variety of factors, including the specific parameters chosen during filter design, the threshold values employed during hysteresis thresholding, or potential limitations in the custom algorithm's adaptability to complex edge patterns.

Potential Factors Influencing Detail Loss:

1) **Filter Design Parameters:** The Sobel filters, as well as the Gaussian blur parameters, play a critical role in edge detection. Suboptimal choices in these parameters may result in the suppression of finer edge details.
2) **Threshold Values:** The accuracy of edge detection is highly sensitive to the choice of threshold values during hysteresis thresholding. If the thresholds are too stringent, some edges may be excluded, while overly lenient thresholds may introduce more noise into the result.
3) **Algorithm Adaptability:** The custom implementation's adaptability to diverse edge patterns and structures may influence its ability to capture intricate details. Limitations in the algorithm's adaptability could lead to the omission of certain finer edges.

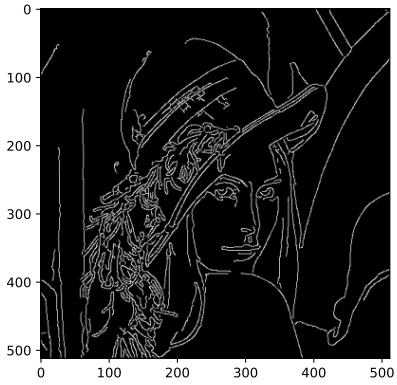


Fig. 6: Final Edge Detection by our implementation

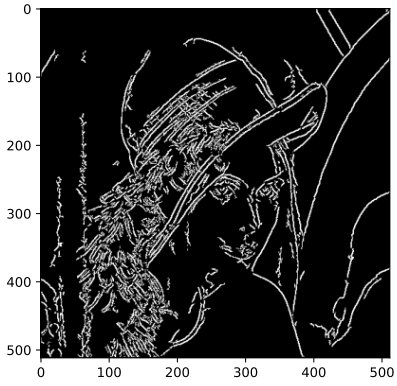


Fig. 7: Edge Detection using Inbuilt Functions

Iterative Refinement and Optimization: The observed differences in level of detail present opportunities for iterative refinement and optimization. Adjustments to filter parameters, threshold values, or the inclusion of additional preprocessing steps could be explored to enhance the algorithm's sensitivity to fine details without compromising noise suppression.

Consideration for Future Enhancements: Parameter Tuning: Fine-tuning the parameters of filters and thresholds could be explored through experimentation to achieve a more nuanced and detailed edge map.

1) **Adaptive Thresholding:** Implementing adaptive thresholding techniques could enhance the algorithm's adaptability to varying edge strengths, potentially improving the detection of finer details.
2) **Post-processing Techniques:** Exploring additional post-processing techniques, such as morphological operations, could help refine the detected edges and address any inconsistencies in the result.

In conclusion, while the custom implementation successfully generated an edge map Fig. 6., the observed differences in level of detail compared to the built-in function output Fig. 7. point towards opportunities for refinement and optimization. This iterative process of analysis and enhancement is integral to the development of a robust and versatile edge detection algorithm.

REFERENCES

[1] Canny, John. "A computational approach to edge detection." IEEE Transactions on pattern analysis and machine intelligence 6 (1986): 679-698.

[2] Neubeck, Alexander, and Luc Van Gool. "Efficient non-maximum suppression." In 18th international conference on pattern recognition (ICPR'06), vol. 3, pp. 850-855. IEEE, 2006.

[3] Gedraite, Estevão S., and Murielle Hadad. "Investigation on the effect of a Gaussian Blur in image filtering and segmentation." In Proceedings ELMAR-2011, pp. 393-396. IEEE, 2011.

[4] Jin-Yu, Zhang, Chen Yan, and Huang Xian-Xiang. "Edge detection of images based on improved Sobel operator and genetic algorithms." In 2009 International Conference on Image Analysis and Signal Processing, pp. 31-35. IEEE, 2009.

[5] Khan, Sajid, Dong-Ho Lee, Muhammad Asif Khan, Abdul Rehman Gilal, Junaid Iqbal, and Ahmad Waqas. "Efficient and improved edge detection via a hysteresis thresholding method." Current Science (00113891) 118, no. 6 (2020).