Image Denoising and Edge Detection

UTRGV CSCI6367

Orlando E. Garcia
Department of Computer Science
University of Texas Rio Grande Valley
orlando.garcia12@utrgv.edu

Damian J. Gomez
Department of Computer Science
University of Texas Rio Grande Valley
damian.gomez02@utrgv.edu

Chris J. Hinojosa Jr.
Department of Computer Science
University of Texas Rio Grande Valley
christopher.hinojosa06@utrgv.edu

Nayeli Gurrola
Department of Computer Science
University of Texas Rio Grande Valley
nayeli.gurrola01@utrgv.edu

1 Introduction

Digital image processing involves algorithms that convert an input image to another output image; they enhance, compress, denoise, and transform images for various applications. Many pictures can appear noisy or fuzzy because of the quality of the device on which they are taken, dim lighting, or movement. To make such images clearer in order to gain more information from them, removal of the noise in the image is necessary. Denoising a visual improves its quality and makes it usable for other purposes, such as image classification and medical imaging. Beyond denoising, another important issue in image processing is identifying points where there is a strong difference in color and brightness to identify boundaries, create structure, or outline the objects in a scene. A common image processing technique to do so is edge detection. To practice both these concepts, we utilize a noisy image and use a Median Filter to denoise it and Canny edge detection method to find the edges in the image. The median filter is effective at reducing salt-and-pepper noise, and the Canny method provides robust edge detection. The results are presented by comparing the original noisy input, the denoised visual, and the final edge-detected output.

2 Approach

2.1 Median Filter

Noise can appear in digital images in several different ways. One common type is salt and pepper noise, which appears as random white or black pixels scattered across the picture, resembling salt and pepper sprinkled on top of it. The median filter is particularly effective at removing this type of noise because reduces outliers and does not blur the image. It preserves edges while removing noise and is often used as a pre-processing step for other types of processing like edge detection. The filter operates with a fixed sliding window: for each pixel, the current pixel point is replaced with the median of its neighboring pixel values. This makes the median filter robust to outliers while preserving image details. It is defined by:

$$\mathrm{y}_i = median(x_1 + x_2 + ... + x_i + ... + x_{n-1} + x_n)$$

where n is the number of elements and x_i represents the value of the element that is in the center of the sliding window.

2.2 Canny Edge Detector

The Canny Edge Detector detects edges in images while maintaining accuracy and reducing noise. There are four steps involved:

1. Noise Reduction
2. Finding Intensity Gradient of the Image
3. Non-maximum Suppression
4. Hysteresis Thresholding

2.2.1 Step 1: Noise Reduction

Noise in the input image can lead to falsely detecting edges. To prevent this, a filter is first applied to the input. In this problem, the Median Filter was used.

2.2.2 Step 2: Gradient Calculation

To detect the intensity of edges, the Canny method computes the gradient of the input image. It does so by applying convolution with derivative filters in both the x and y directions. From this the magnitude (strength of the gradient) and direction (angle of the gradient) are computed as:

$$\mathrm{h}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad h_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\mathrm{M} = \sqrt{(h_x \cdot z)^2 + (h_y \cdot z)^2}$$

$$\mathrm{A} = \arctan\left(\frac{h_y \cdot z}{h_x \cdot z}\right)$$

where h_x and h_y are kernels that detect changes along the x-axis and y-axis, respectively.

2.2.3 Step 3: Non-Maximum Suppression

After identifying the intensity of the edges, the algorithm thins them out by removing the pixels that are not the strongest along the direction of an edge (non-maxima). The following algorithm is applied:

For each pixel (x, y) in M:

1. Look at the direction of the edge.
2. Compare the pixels magnitude to its neighbors along that same direction.
3. If the pixel is lower than its neighbors, the gradient is set to 0 and does not become part of the edge.

2.2.4 Step 4: Thresholding

The final step applies two layers of thresholding to classify edges in the image. Using the gradient magnitude, the edges that are above a high threshold are classified as strong edges. Those that fall between a low and high threshold are classified as weak edges, and those that are below the low threshold are classified as non-edges.

The second layer of thresholding involves performing edge tracking by hysteresis. Hysteresis refers to the process of identifying weak edges that are connected to strong edges so that they remain retained as part of the final edge detection. This process continues until no more weak edges remain, ensuring the identified edges are well-defined.

3 Experiments

3.1 Data

The denoising and edge detection methods were applied to Figure 1. The input was a JPG file, and the outputs are a denoised image and an edge-detected image.

Figure 1: Original noisy image used for the problem.

3.2 Experimental details

The original noisy image is loaded into a variable and read in grayscale using the IMREAD_GRAYSCALE flag of the $imread()$ function from OpenCV. An if statement is used to verify that the image was successfully loaded by checking whether the variable is empty. If the image fails to load, an error message is displayed.

Once loaded, the image is denoised using the $medianBlur()$ function with a kernel size of 5×5. The denoised output is then passed to the $Canny()$ function for edge detection, with the minVal and maxVal thresholds set to 100 and 200 for hysteresis, respectively.

Finally, the original, denoised, and edge-detected images are plotted for visual comparison of the processing steps.

3.3 Results

For the denoised image, the salt-and-pepper pattern of pixelation scattered across the image is no longer present. Although the noise has been removed, the median filter introduces a slight blurriness, which makes some edges harder to identify. Comparing the original image with the output of the Canny method, we can see the outline of the woman and her surroundings across most of the picture. Some edge information is missing along her chin, shoulder, hat, and where her hair ends, because the pixels in those regions have similar gradients, and the median filter blurred those boundaries even more.

Nevertheless, the Canny method clearly identifies the boundaries in the denoised image. The added blurriness suggests that a different filter might have resulted in sharper outlines and better detection of faint edges. Overall, the median filter was effective at reducing noise, as no noticeable salt-and-pepper traces remain in the denoised output, but improvements could be made in preserving weaker edges.

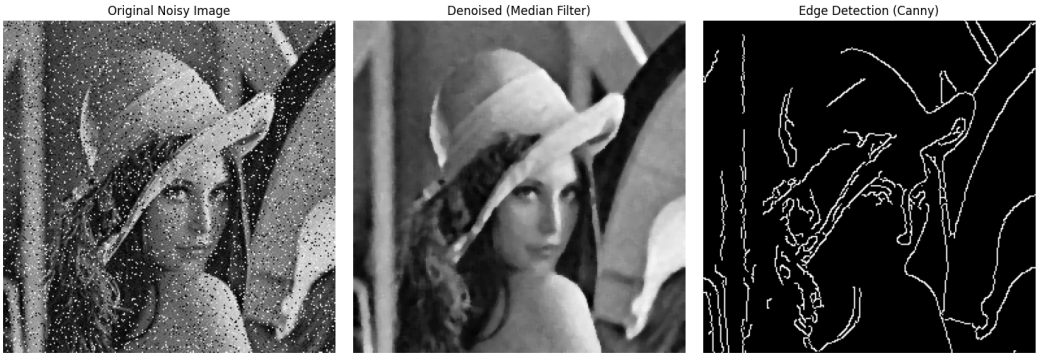

Figure 2: From left to right: original, denoised, and edge detected image.

4 Conclusion

The project showed that the median filter was effective at removing noise but blurred faint edges, while Canny detection emphasized strong boundaries.

References

[1] Yong Han, Min Xiang, Binglin Jiang, Junyi Huang, Xingwang Zhou, and Changjian Zhang. An image denoising algorithm based on improved median filter and threshold function. In *2024 36th Chinese Control and Decision Conference (CCDC)*, pages 1602–1607, 2024.

[2] Vincent Mazet. Basics of image processing - denoising, 2025.

[3] Kinda Technical. Lesson 10: Canny edge detector, 2025.

[1] [2] [3]