

# Comparative Study of Edge Detection Methods

Christodoulos Zerdalis AEM:03531

## 1 Summary of the methods used

### 1.1 Sobel Edge Detector (First-Order)

The Sobel operator performs a 2D spatial gradient measurement to highlight regions of high spatial frequency.

- **Logic:** Approximates the first derivative to find the intensity "peak" in horizontal ( $G_x$ ) and vertical ( $G_y$ ) directions.
- **Kernels:** Uses a weight of 2 for pixels directly adjacent to the center for weighted smoothing.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

- **Magnitude:**  $G = \sqrt{G_x^2 + G_y^2}$

### 1.2 Prewitt Edge Detector (First-Order)

A simpler version of the Sobel operator that treats all neighboring pixels with equal importance.

- **Logic:** Acts as a "pure" gradient operator without added smoothing.
- **Kernels:** All non-zero coefficients are 1 or -1.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix}$$

### 1.3 Laplacian Edge Detector (Second-Order)

An isotropic operator that detects edges by looking for the rate of change of the gradient.

- **Logic:** Identifies the **zero-crossing** (the exact center of an edge).
- **Derivatives:** First derivative results in a *maximum*, second derivative results in a *zero-crossing*.
- **Standard Kernel:**

$$\nabla^2 f = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

## 1.4 Canny Edge Detector (Multi-Stage)

Designed for optimal detection, localization, and minimal response.

1. **Noise Reduction:** Gaussian blur application.
2. **Gradient Calculation:** Finding intensity changes (usually via Sobel).
3. **Non-Maximum Suppression:** Thinning edges to 1-pixel width.
4. **Double Thresholding:** Categorizing pixels as Strong, Weak, or Non-edges.
5. **Hysteresis Tracking:** Connecting weak pixels to strong ones.

## 1.5 Edge Detection in the Frequency Domain

Analyzes the image globally by transforming spatial data into frequency waves.

- **Core Concept:** Low Frequencies = Smooth regions; High Frequencies = Edges.
- **Process:** Fourier Transform ( $FFT$ ) → Shifting → High-Pass Filtering ( $HPF$ ) → Inverse Fourier Transform ( $IFFT$ ).

## 2 Comparison of Edge Detection Methods

These methods differ fundamentally in their mathematical approach, with some operating on the first derivative, some on the second, and others utilizing the frequency spectrum.

**Sobel and Prewitt Detectors:** Both the Sobel and Prewitt operators work on the first derivative. The primary difference between them is that the Sobel kernel uses a weight of 2 for its central coefficients. This provides a weighted smoothing effect that makes it more robust against image noise than the Prewitt operator.

**Laplacian Detector:** The Laplacian operator works on the second derivative. Unlike first-order operators that require separate horizontal and vertical passes, the Laplacian is isotropic, meaning it requires only one convolution per pixel to detect edges in all directions simultaneously.

**Canny Edge Detector:** The Canny edge detector also relies on the first derivative (typically using Sobel gradients) however, it is a multi-stage algorithm. The built-in functions used in modern libraries integrate additional stages such as Gaussian blurring, non-maximum suppression, and hysteresis thresholding to significantly improve the accuracy and thinness of the detected edges.

**Frequency Domain Edge Detection:** Lastly, edge detection in the frequency domain does not calculate spatial pixel differences directly. Instead, it uses the pixel intensity values to perform a Fast Fourier Transform ( $FFT$ ), representing the image in the frequency domain. This allows the algorithm to interpret edges as high frequencies, which can then be isolated using a high-pass filter before transforming the image back into the spatial domain.

## 3 Methodology Adjustments

Before we start comparing the different methods, it is important to point out that the method of clipping the histogram requested for the exercise (manually selecting a valley in the gradient histogram) did not work at all for four of the five images. Most of the time the results were unusable thus, we clipped the histogram at the 90th percentile in order to be able to make the comparisons with quality images. This happened because, in some images, the valleys were false (very small valleys) or there weren't any valleys at all. Usually, the histogram after edge detection does not actually present many valleys, but just a tail from the positive y-axis to zero.

Also after running the notebook and waiting for about 10 minutes to finish (for my pc) all the images mentioned in the report and more will be saved in the directory in the Outputs subfolder.

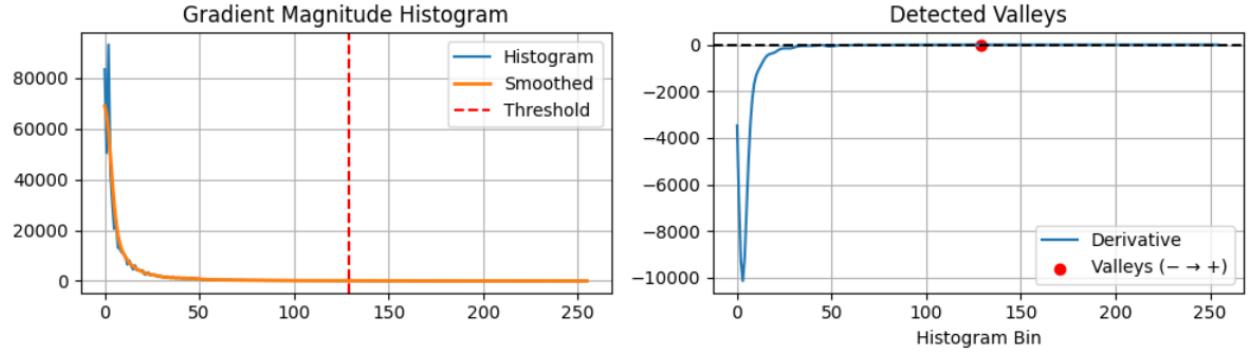


Figure 1: Here we can see how the gradient does not have any significant valleys. The same applies for the derivative as it does not cross the  $xx'$  axis.

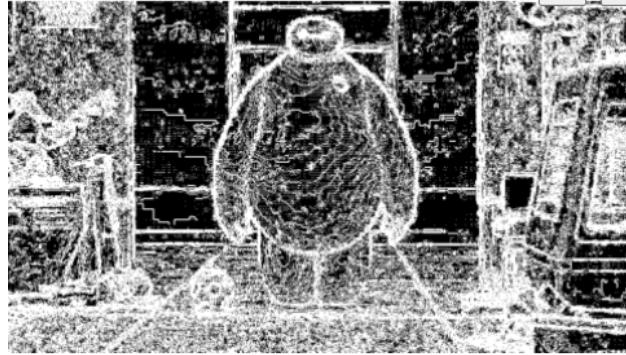
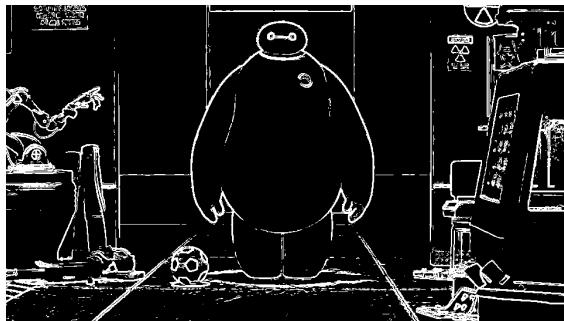


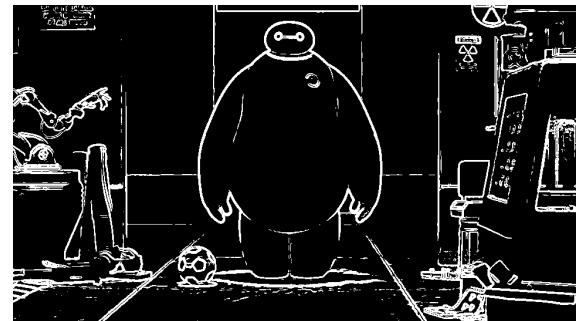
Figure 2: Example of an unusable image.

## 4 Effect of smoothing

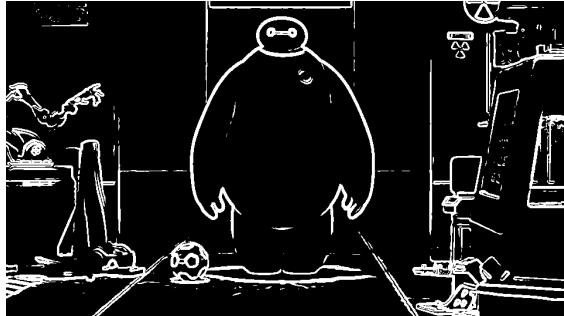
Smoothing was used only for the first three edge detectors because Canny has one already built in, and I thought that I should smooth in the spatial domain and then detect edges in the spatial domain. For all three methods, smoothing worked well in eliminating noise that would otherwise be detected as small point-like edges. An exception was the image depicting the New York skyline where, after smoothing, the final result in my opinion was a bit worse, as small details in the skyscrapers windows were not visible in reality, the best version was the one with the  $3 \times 3$  Gaussian smoothing. I picked the Sobel images for the comparisons below as I liked them the most, but the results are similar for the other two kernels.



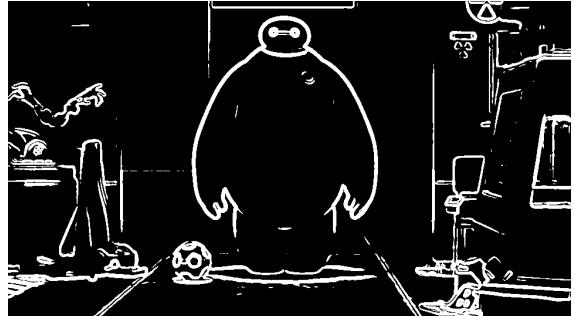
(a) No Smoothing



(b)  $3 \times 3$  Gaussian



(a)  $5 \times 5$  Gaussian



(b)  $7 \times 7$  Gaussian

Figure 3: Sobel edge detection: (File name: `edges_threshold_sobel_YxY_1.png`).



(a) No Smoothing



(b)  $3 \times 3$  Gaussian



(c)  $5 \times 5$  Gaussian

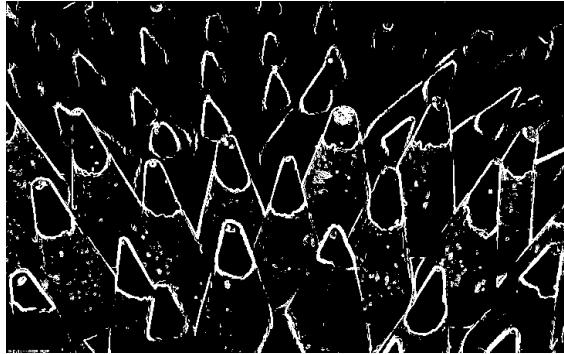


(d)  $7 \times 7$  Gaussian

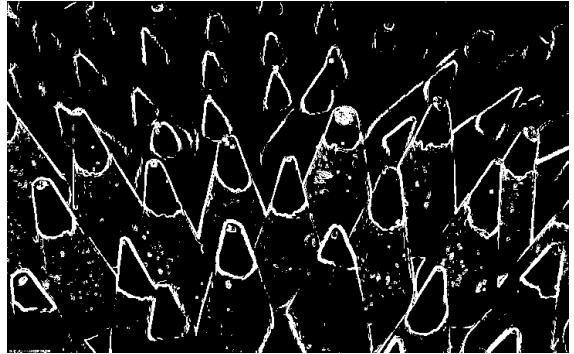
Figure 4: Example of the New York skyline (File name: `edges_threshold_sobel_YxY_3.png`).

## 5 Noise sensitivity

For noise sensitivity, I compare the images with no smoothing because smoothing eliminates most of the noise. The best detector was the Canny detector by far, but the Canny detector, as previously mentioned, has a built-in smoothing function, so this result might not make sense. As far as the detectors in the spatial domain are concerned, the Sobel and Prewitt perform very similarly as they are almost the same perhaps the Sobel detector is the best, as its kernel is fundamentally better for noise. The Laplacian detector is very susceptible to noise and overall does not work very well, as it creates blurred edges and multiplies noise. In the frequency domain, results are very different, and I cannot really discern if the difference is caused by noise or the inherent way the edge detection works however, in my opinion, noise does not have a big effect as the high-pass filter should eliminate it all.



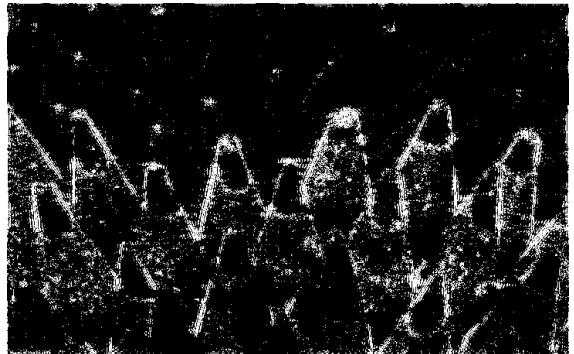
(a) Sobel (edges\_threshold\_sobel\_no\_4.png)



(b) Prewitt (edges\_threshold\_prewitt\_no\_4.png)



(c) Laplacian  
(edges\_threshold\_laplacian\_no\_4.png)

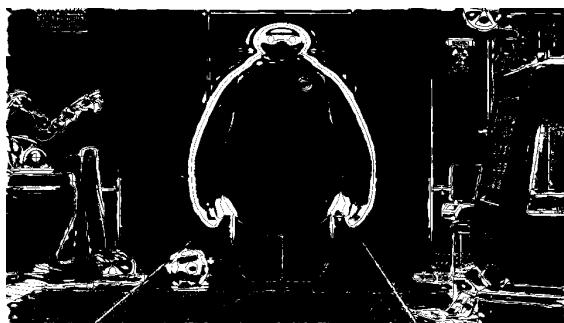


(d) High Pass  
(high\_pass\_fft\_image\_4\_radius\_100.png)

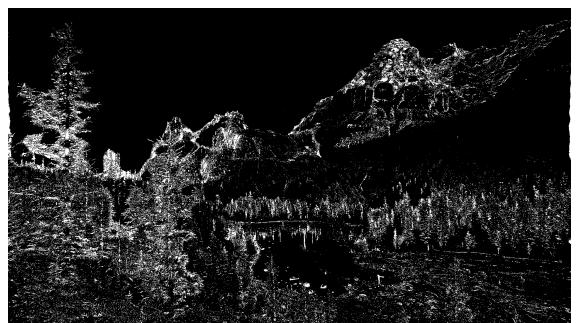
Figure 5: Comparison of noise sensitivity across different detectors.

## 6 Comparison of spatial vs frequency domain

In the frequency domain, I did not use smoothing as mentioned before. The results in the frequency domain were a little bit different from those in the spatial domain and not necessarily bad. The edges stop being streamlined and having a certain width and start being more "cartoon-like." Also, there is almost always interference near every edge this looks like waves. This phenomenon is more prevalent in the first image (Baymax). The second and third images work better, but they still look more like a sketch than a strict edge detection. The last two images, the pencils and the text, do not work well at all. Lastly, the best cutoff radius looks to be 30.



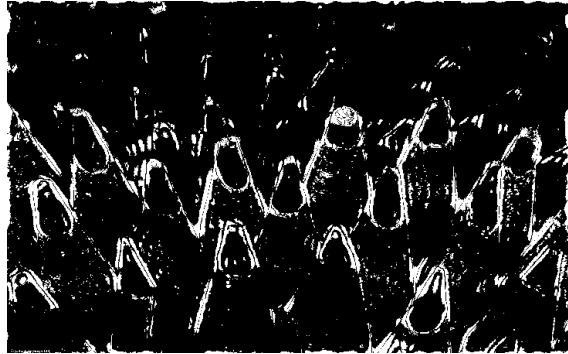
(a) First image  
(high\_pass\_fft\_image\_1\_radius\_30.png)



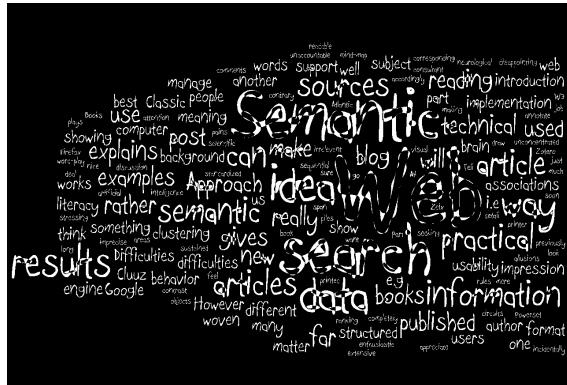
(b) Second image  
(high\_pass\_fft\_image\_2\_radius\_30.png)



(a) Third image  
 (high\_pass\_fft\_image\_3\_radius\_30.png)

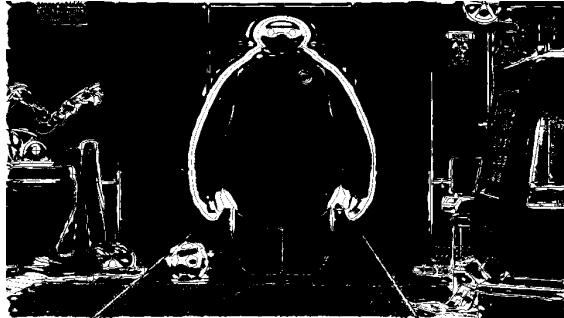


(b) Fourth image  
(high\_pass\_fft\_image\_4\_radius\_30.png)

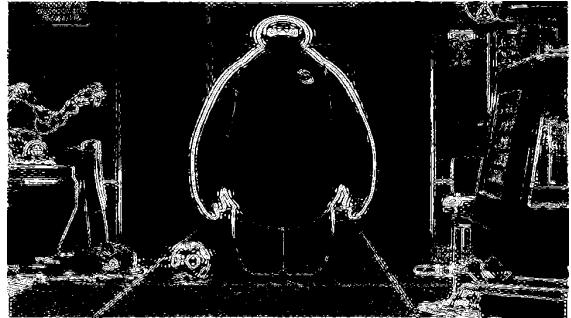


(c) Fifth image  
 (high\_pass\_fft\_image\_5\_radius\_30.png)

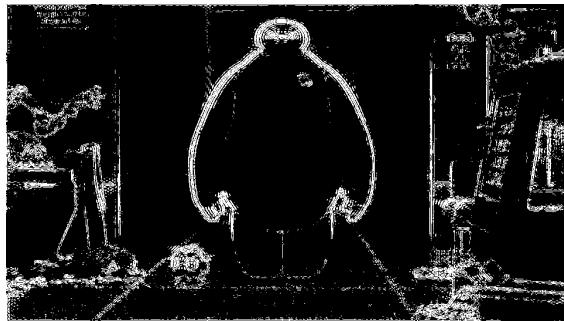
Figure 6: Frequency domain results for the remaining images. We can see the interference in image 4 and how the 5th image has not worked very well.



(a) Radius 30  
(high\_pass\_fft\_image\_1\_radius\_30.png)



(b) Radius 75  
(high\_pass\_fft\_image\_1\_radius\_75.png)



(c) Radius 100  
(high\_pass\_fft\_image\_1\_radius\_100.png)



(d) Radius 150  
(high\_pass\_fft\_image\_1\_radius\_150.png)

Figure 7: Comparison of different cutoff radii for frequency domain detection.



(a) Skyline with frequency  
(high\_pass\_fft\_image\_3\_radius\_30.png)



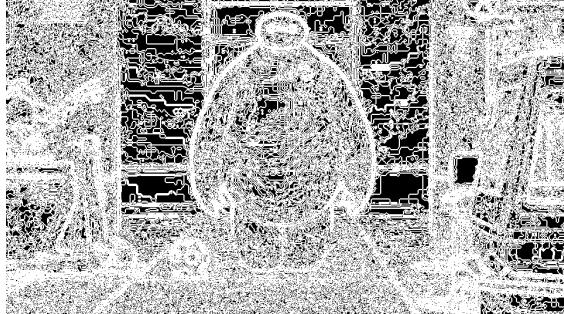
(b) Best skyline result  
(edges\_threshold\_sobel\_3x3\_3.png)

Figure 8: Comparison of the "cartoonish" frequency result versus the most accurate in my opinion spatial result.

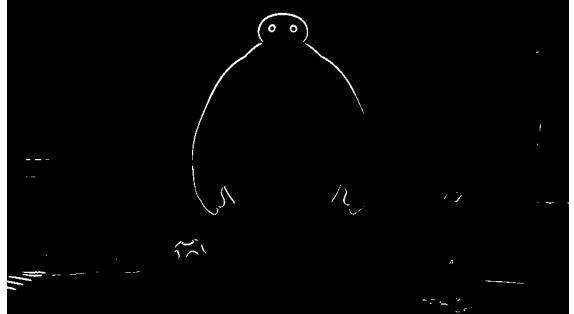
## 7 Extra observations

### 7.1 Histogram-based thresholding

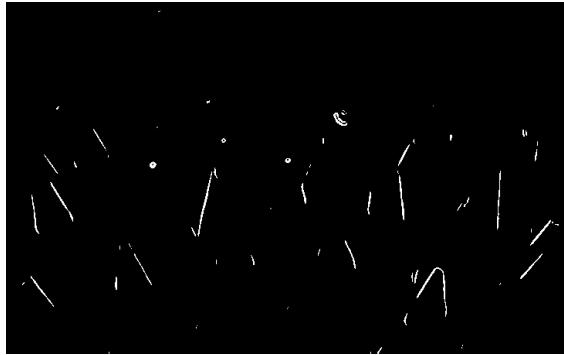
As mentioned above, valley-based thresholding did not work well for complex images, and I had to use 90th percentile thresholding to have usable results.



(a) edges\_valey\_laplacian\_5x5\_1.png



(b) edges\_valey\_prewitt\_5x5\_1.png



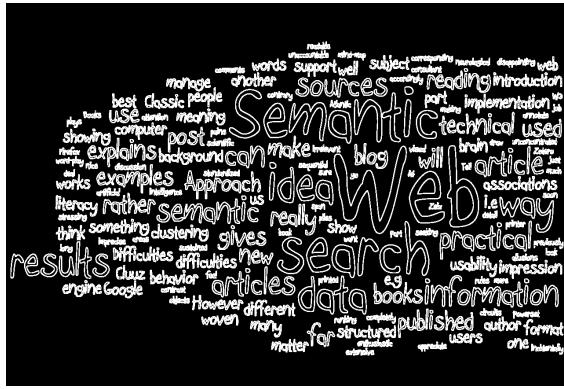
(c) edges\_valey\_sobel\_5x5\_4.png



(d) edges\_valey\_sobel\_no\_3.png

Figure 9: Some bad results of valley detection.

Only for the fifth image which just depicted text the valley detection worked well.



(a) edges\_valey\_sobel\_no\_5.png

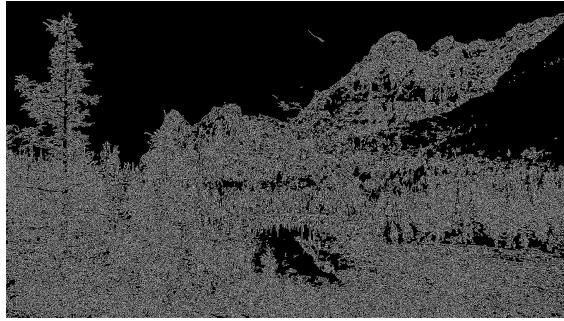


(b) edges\_valey\_prewitt\_no\_5.png

Figure 10: Comparison showing that results stay consistent regardless of smoothing.

## 7.2 Canny edge detector

Even though the Canny edge detector is considered to be the best edge detector out of the ones being tested, it still performs poorly for every threshold in some images where the scene is very complex. In my images, the second one depicting a forest has very bad results.



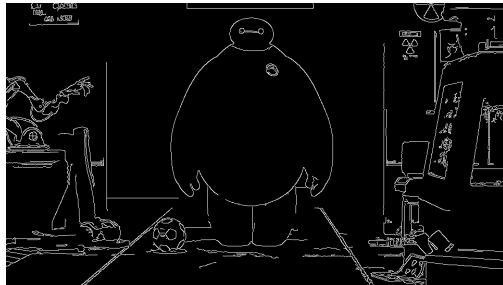
(a) Canny (`canny_image_2_t1_50_t2_150.png`)



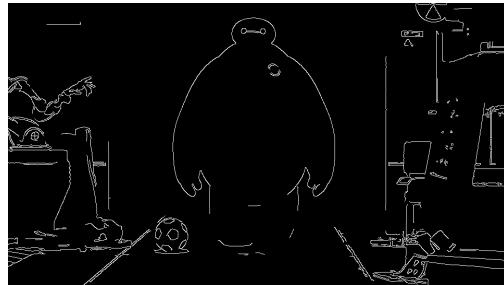
(b) Sobel (`edges_threshold_sobel_3x3_2.png`)

Figure 11: Comparison of Canny and Sobel results for a complex forest image.

Also, I tested this detector with double threshold values of 50–150, 100–200, and 20–70. While the best results came from the first two thresholds, the 20–70 one always had the worst result as it detected many edges that weren't actually edges.



(a) `canny_image_1_t1_50_t2_150.png`



(b) `canny_image_1_t1_100_t2_200.png`



(c) `canny_image_3_t1_50_t2_150.png`



(d) `canny_image_3_t1_100_t2_200.png`



(e) `canny_image_4_t1_50_t2_150.png`



(f) `canny_image_4_t1_100_t2_200.png`

Figure 12: Canny edge detection results comparing different thresholds for three test images.

### 7.3 Edge density results

In the first graph, the edge density is plotted for all the tests but with 90th percentile thresholding. We can see that the results are similar, with the Laplace detector being slightly reduced. This happens because all the spatial detectors use the same thresholding (except the Canny one) as far as the frequency one is concerned, the results happen to be the same. In conclusion, the Laplacian detector has lower values, the most aggressive is the Canny 50 to 150 threshold and the least is the Canny 100 to 200.

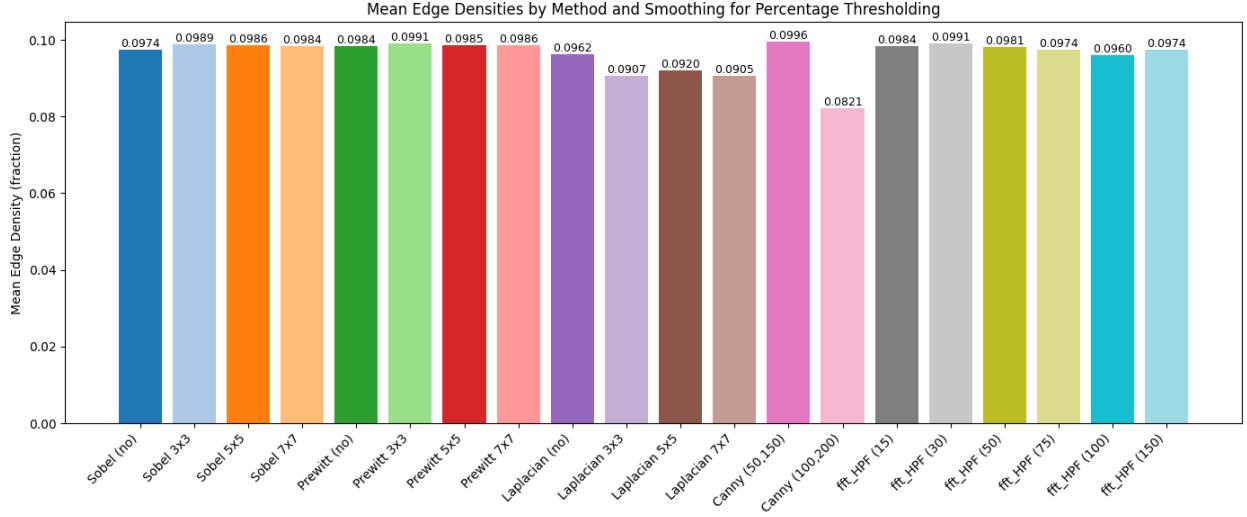


Figure 13: Mean edge densities per detection method using 90th percentile (File name: `edge_density_1.png`).

And here is the plot with valley thresholds. We can see that the results differ wildly as this method is not consistent and yields bad results.

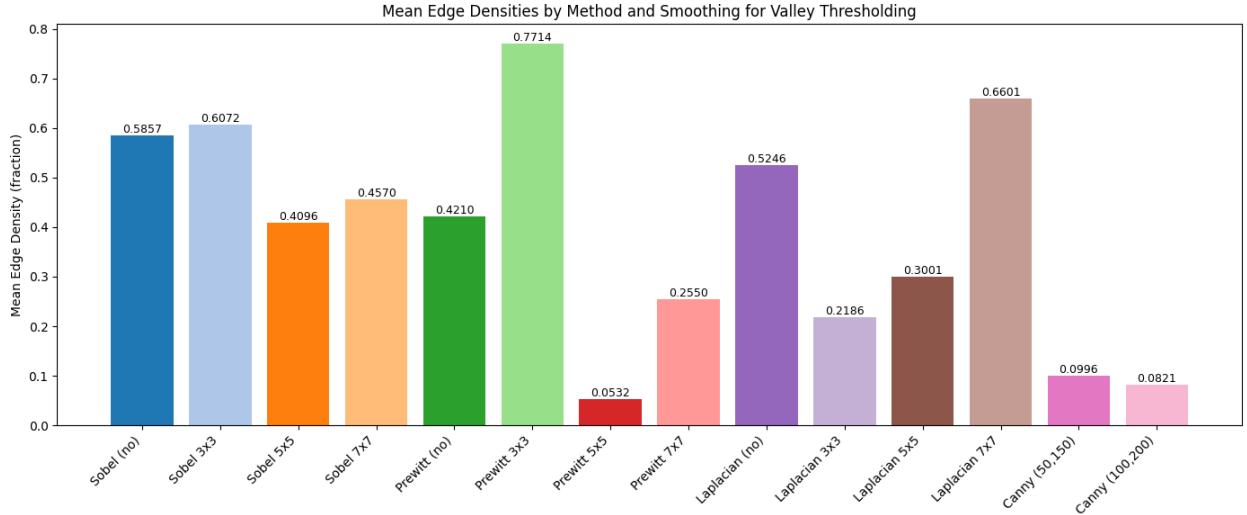


Figure 14: Mean edge densities per detection method using valley thresholding (File name: `edge_density_valley.png`).