EE550 - Image and Video Processing - Lab 3 Report

Baris Sevilmis
Lab Date: 06/11/2019
Due Date: 21/11/2019

1 Overview

Third lab assignment of EE550 focuses on various Edge Detection methods & algorithms. Template method with Sobel, Prewitt and Roberts filters, Compass Operator, Laplace Operator and Frei-Chen method are the main methods to be discussed in further sections. Thresholding, complexity & operating with various level of noise are to be discussed as well as the methods themselves. Following images are used for experimentation:

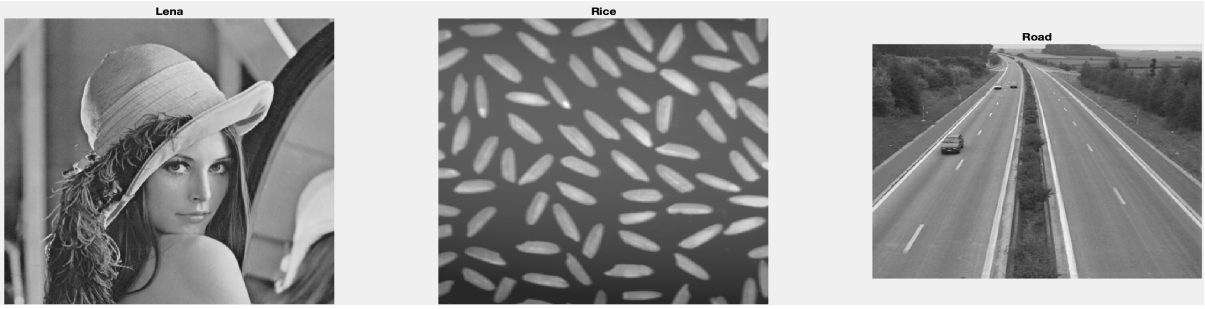

Figure 1: Lena, Rice & Road : Original

2 Template Method

Template method is one of the most useful and easiest techniques used for edge detection. Main focus of gradient methods is to model shape of the edges by providing an approximate gradient. Various filters are used for vertical and horizontal gradient approximation that are indicated in further subsections. Following 1-D gradient calculation may be helpful to visualize filters:

$$\nabla f(x) = \lim_{h->0} \frac{f(x+h) - f(x-h)}{2h}$$

L_1 & L_2 norms are used for different types of magnitude approximation.

$$L_1 : y(k,l) = |x(k,l) * *filt_1(k,l)| + |x(k,l) * *filt_2(k,l)| \quad (1)$$

$$L_2 : y(k,l) = \sqrt{(x(k,l) * *filt_1(k,l))^2 + (x(k,l) * *filt_2(k,l))^2} \quad (2)$$

2.1 Sobel

Sobel is an gradient approximation filter, which is smoothed. It emphasises more on the central points. Following S_1 & S_2 are mentioned Sobel filters:

$$S_1 = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, S_2 = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

These filters are convolved with given images in Figure 1. After convolution, L_1 & L_2 norms are taken by the Formulas (1) & (2). Following figures are the results of Sobel operator edge detection with different Gaussian noise levels such that noise$\in [0, 5, 11, 25]$. In addition, vertical and horizontal gradient approximations are provided. Figure 2, 3 & 4 depict gradient profiles of images in horizontal and vertical directions. Lastly, threshold is also given as a parameter, in which intensity values passing threshold, after convolution and norm operations, are chosen as edge points. For the sake of simplicity and experimentation, threshold interval $\in [45, 60]$ is chosen, in which most accurate edge detection is ensured. Parameter choice is demonstrated below in green boxes:

Gaussian

[0, 5, 11, 25]

Threshold

[45, 60]

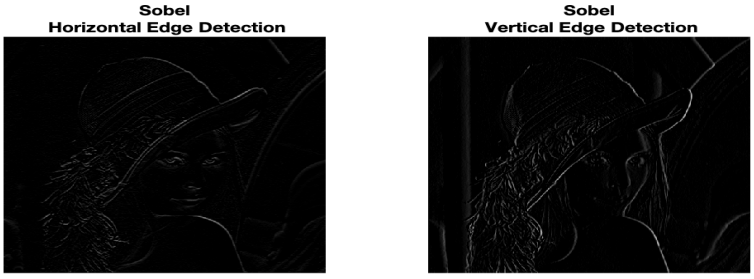

Figure 2: Lena Gradient Profiles

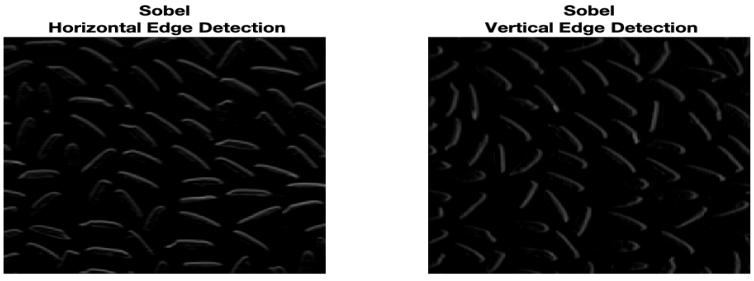

Figure 3: Rice Gradient Profiles

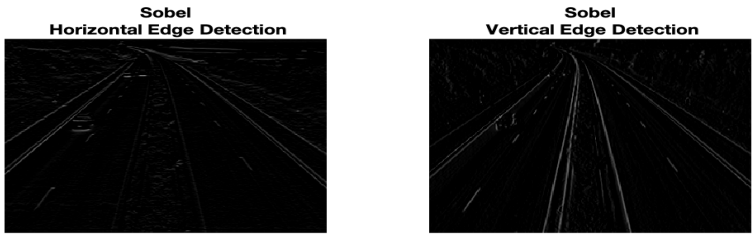

Figure 4: Road Gradient Profiles

Figure 5, 6, 7 & 8 demonstrate *lena* edge detection by Sobel operator with Gaussian noise levels $\in [0, 5, 11, 25]$. It is obvious that more the noise in original image less accurate are edge detection. Some of the unrelated noisy coordinates are detected as edges as well, which can be easily seen for Gaussian noise $\in [11, 25]$. Another point is L_2 norm provides better edge detection results than L_1 as expected. Even with noise=25, edge detection with L_2 proves more accurate edge detection as the magnitude computation is more robust to noise.

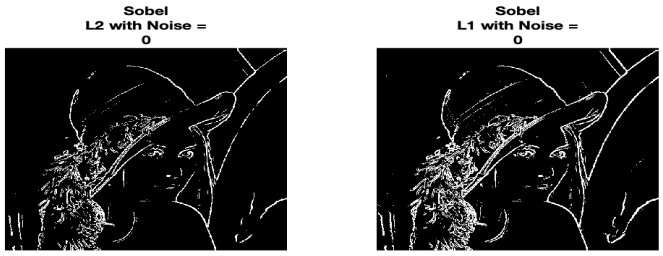

Figure 5: *lena* - Sobel - Noise = 0

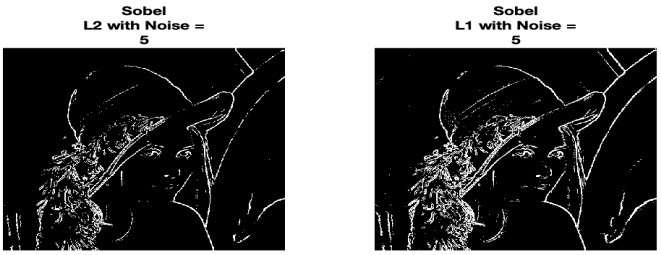

Figure 6: *lena* - Sobel - Noise = 5

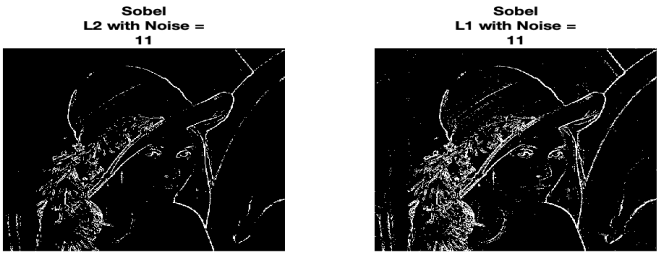

Figure 7: *lena* - Sobel - Noise = 11

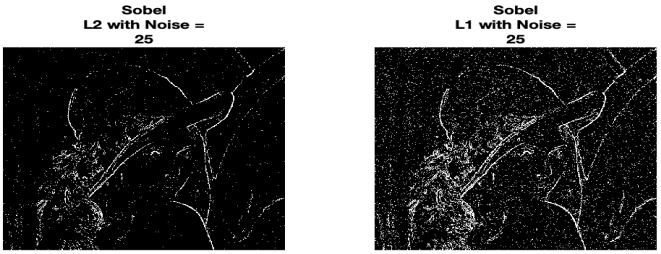

Figure 8: *lena* - Sobel - Noise = 25

On the other hand, Figure 9, 10, 11 & 12 demonstrate Sobel edge detection given same set of Gaussian noise and threshold parameters on the *rice* image as in *lena*. Same set of conclusions are observed, in which increase in noise reduces precision of edge detection and L_2 norm provides more robust edge approximation, as where as L_1 norm detects many of the non related noise based coordinates as edges.

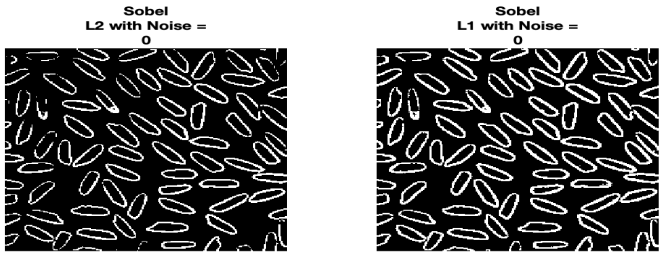

Figure 9: *rice* - Sobel - Noise = 0

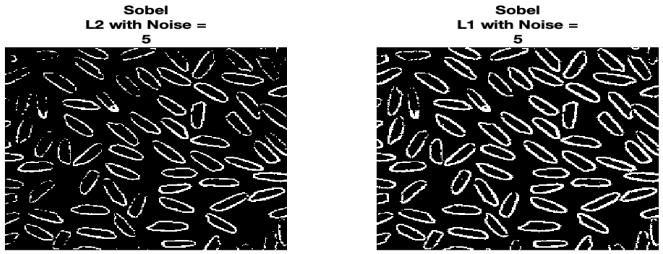

Figure 10: *rice* - Sobel - Noise = 5

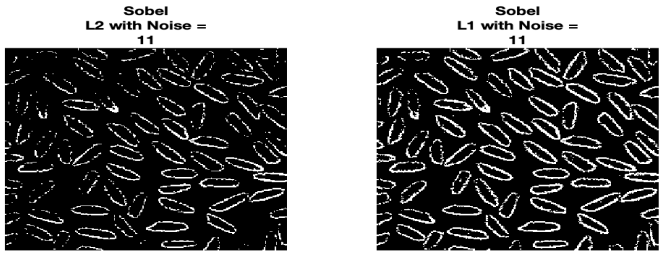

Figure 11: *rice* - Sobel - Noise = 11

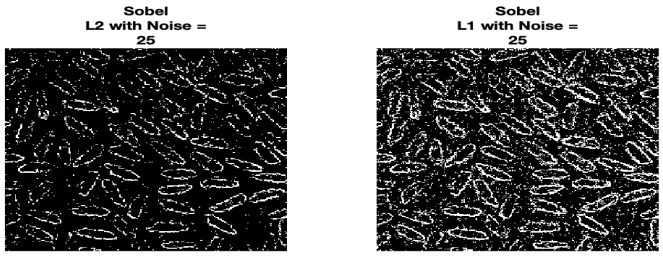

Figure 12: *rice* - Sobel - Noise = 25

Lastly, *road* image is used for experimentation in addition to **lena** & *rice* with the same set of threshold and Gaussian noise parameters. Figure 13, 14, 15 &16 depicts edge detection results for *road* image. As expected, resulting edge detection is same as in previous experimentation, L_2 ensures more precise results for images with higher noise, even though the edge detection performance decreases overall.

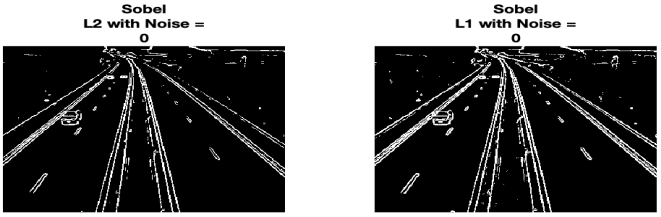

Figure 13: *road* - Sobel - Noise = 0

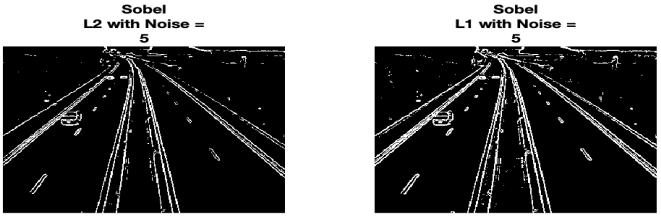

Figure 14: *road* - Sobel - Noise = 5

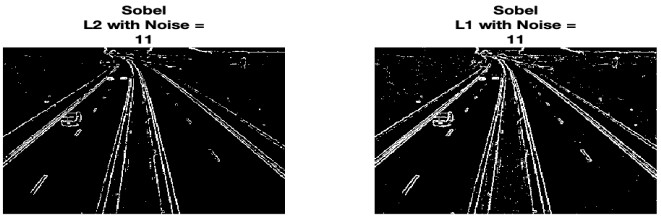

Figure 15: *road* - Sobel - Noise = 11

Figure 16: *road* - Sobel - Noise = 25

2.2 Prewitt

Prewitt operator is a really similar operator in comparison with Sobel operator. It is again a smoothed operator. Only difference is that Prewitt operator puts emphasis equally on all coordinates, whereas Sobel had higher weights for central points. Prewitt filters for vertical and horizontal gradient approximation are as follows:

$$P_1 = \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, P_2 = \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Rest of the algorithm is same as the Sobel, where both L_1 & L_2 norms are used for magnitude computation. Threshold values are same with Sobel's as in green boxes as well as the Gaussian noise levels. Figure 17, 18, 19 & 20 depict edge detection results for *lena* for noise $\in [0, 5, 11, 25]$ & threshold $\in [45, 60]$.

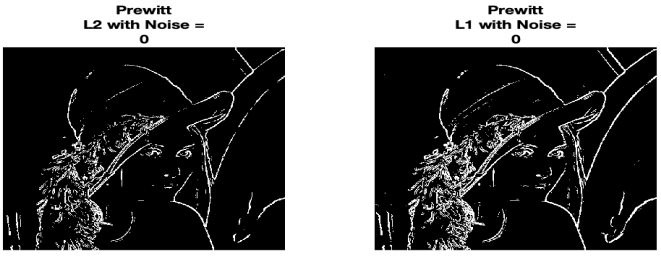

Figure 17: *lena* - Prewitt - Noise = 0

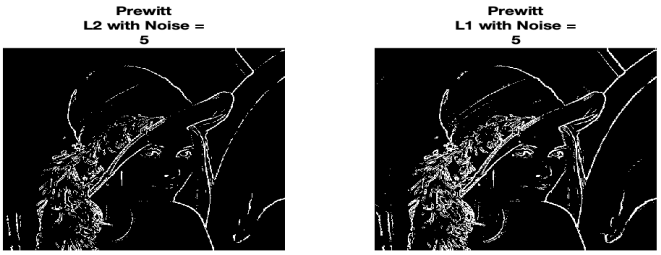

Figure 18: *lena* - Prewitt - Noise = 5

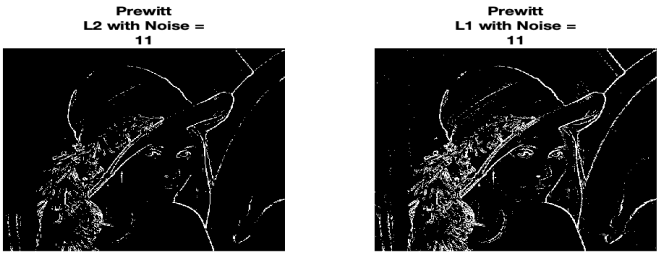

Figure 19: *lena* - Prewitt - Noise = 11

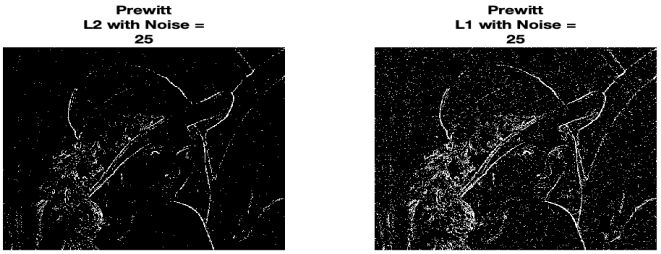

Figure 20: *lena* - Prewitt - Noise = 25

Obviously, results are very similar to Sobel's end results since the only difference is the operator and operators are really similar. Set of conclusions for L_1 & L_2 norms and noise effects are same as in Sobel. Figure 21, 22, 23 & 24 demonstrate resulting images for *rice* and finally, Figure 25, 26, 27 &28 depict edge detection results for image *road*. Conclusions about norms and noise are same as before.

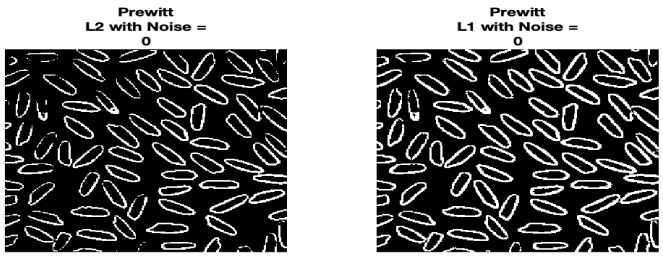

Figure 21: *rice* - Prewitt - Noise = 0

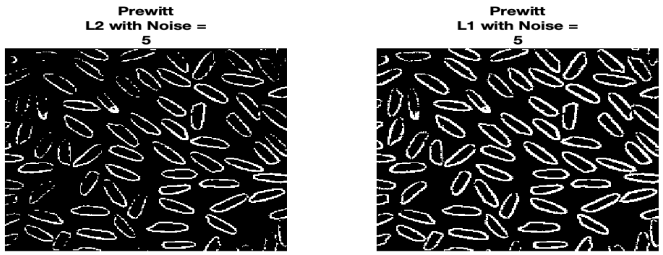

Figure 22: *rice* - Prewitt - Noise = 5

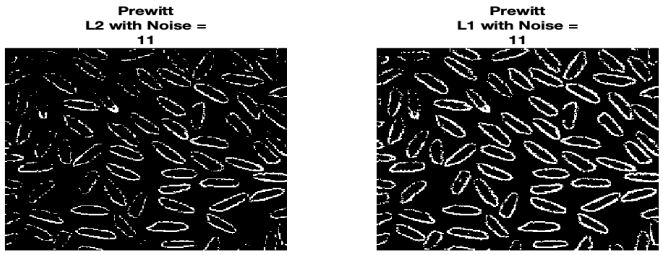

Figure 23: *rice* - Prewitt - Noise = 11

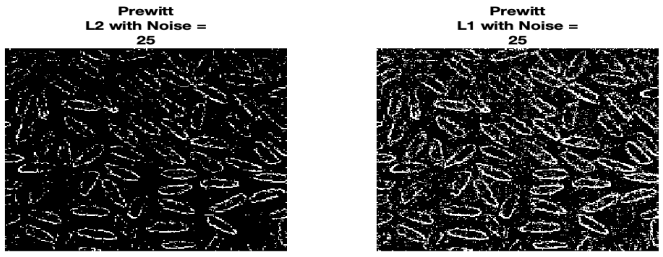

Figure 24: *rice* - Prewitt - Noise = 25

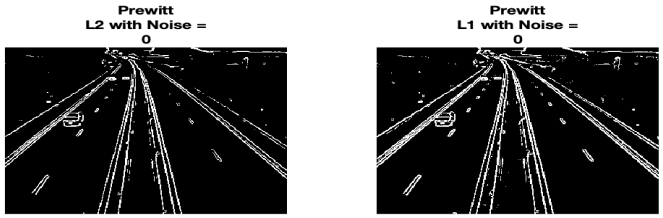

Figure 25: *road* - Prewitt - Noise = 0

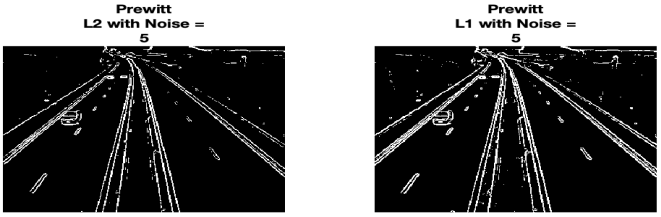

Figure 26: *road* - Prewitt - Noise = 5

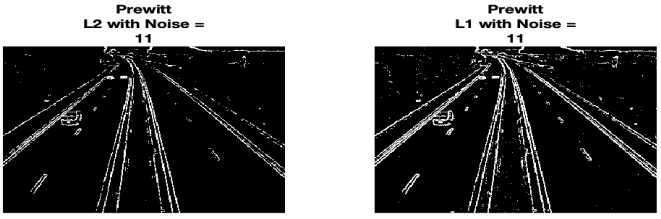

Figure 27: *road* - Prewitt - Noise = 11

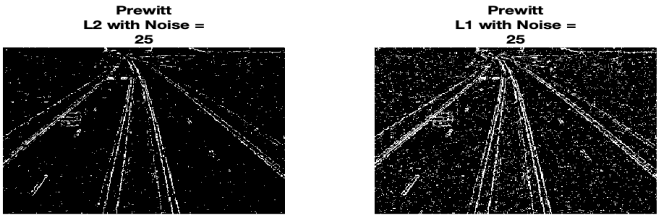

Figure 28: *road* - Prewitt - Noise = 25

2.3 Roberts

Roberts is another known operator, which is considered as an operator for template method. Main difference of this operator is that, it is actually a $2x2$ filter, which highlights changes in intensity values diagonally. One main disadvantage is Robert's operator is very sensitive to noise. For sake of experimentation same set of parameters are used as in Sobel and Prewitt, namely Gaussian noise($\in [0, 5, 11, 25]$) and threshold($\in [45, 60]$) & L_1 & L_2 norms are used for magnitude calculation. Roberts filters for vertical and horizontal gradient approximation are the following filters:

$$R_1 = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, R_2 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Figure 29, 30, 31 & 32 demonstrate Roberts operator on *lena* image with Gaussian noise and threshold parameters applied. Figure 33, 34, 35 & 36 depict same set of operations on *rice* image and lastly, Figure 37, 38, 39 & 40 provide edge detection results of Roberts operator on *road* image. Although, results of Roberts has similar accuracy and precision in terms of edge detection with Sobel and Prewitt, as the noise level increases Roberts performance decreases much more rapidly. In all of our experimentations with Roberts operator, precision is really low both in L_1 & L_2 norms. In case of Sobel & Prewitt, although L_2 norm resulted in fewer detection of edge coordinates, it was still accurate. However, Roberts is very noise sensitive as mentioned previously, therefore produces more and more inaccurate detection results in both L_1 & L_2 magnitude approximations as the noise level increases.

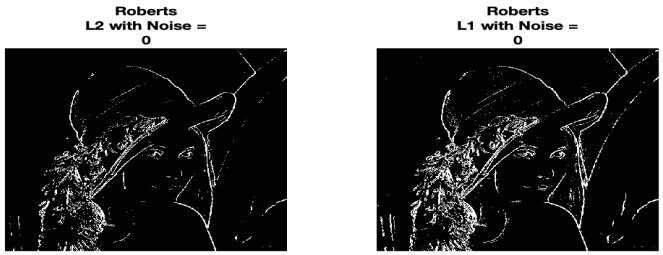

Figure 29: *lena* - Roberts - Noise = 0

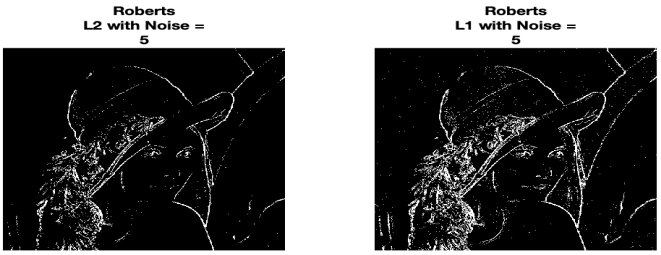

Figure 30: *lena* - Roberts - Noise = 5

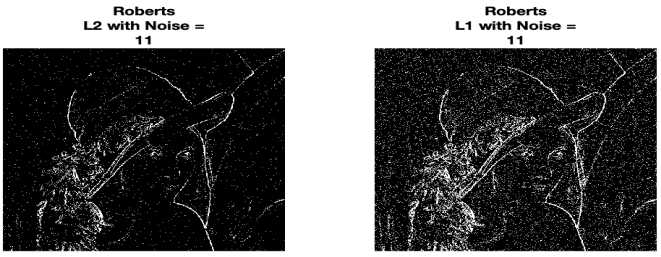

Figure 31: *lena* - Roberts - Noise = 11

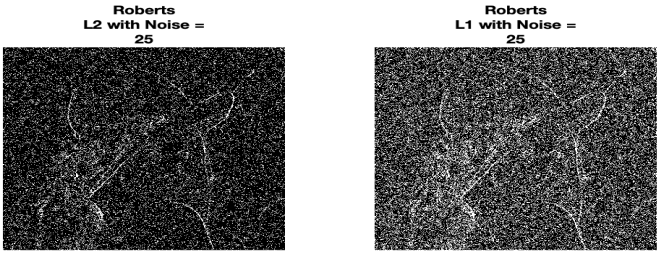

Figure 32: *lena* - Roberts - Noise = 25

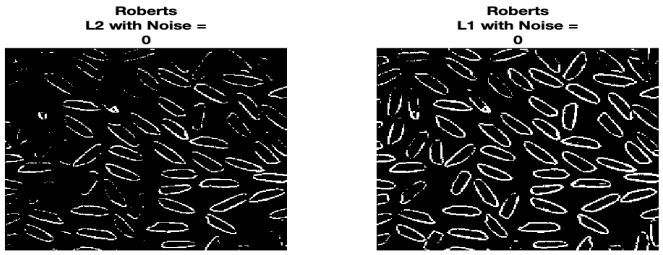

Figure 33: *rice* - Roberts - Noise = 0

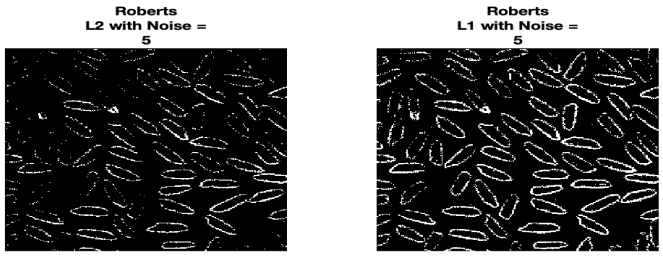

Figure 34: *rice* - Roberts - Noise = 5

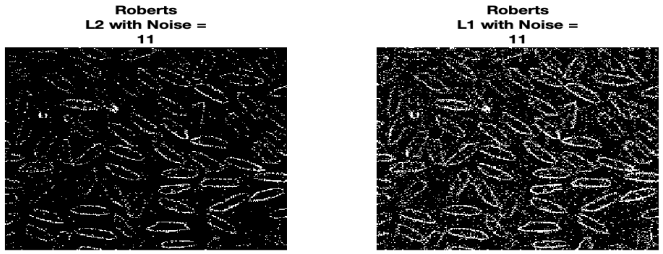

Figure 35: *rice* - Roberts - Noise = 11

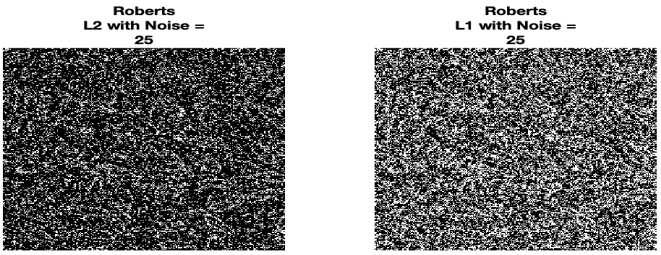

Figure 36: *rice* - Roberts - Noise = 25

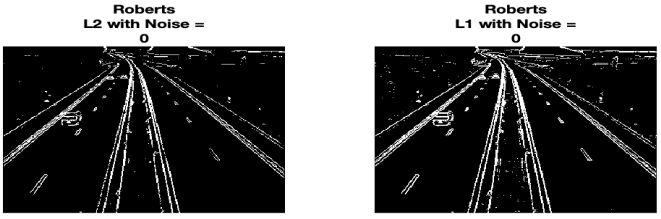

Figure 37: *road* - Roberts - Noise = 0

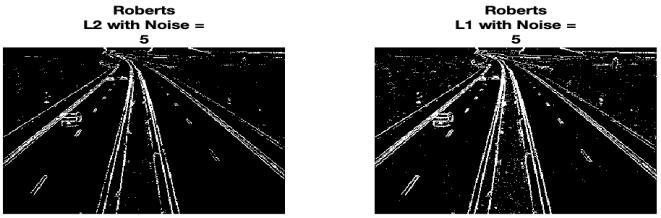

Figure 38: *road* - Roberts - Noise = 5

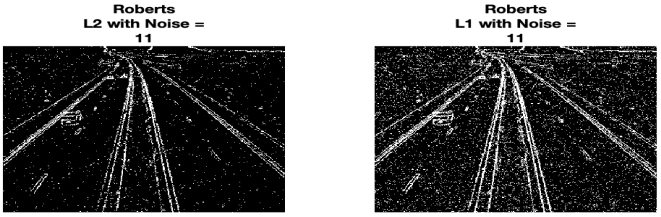

Figure 39: *road* - Roberts - Noise = 11

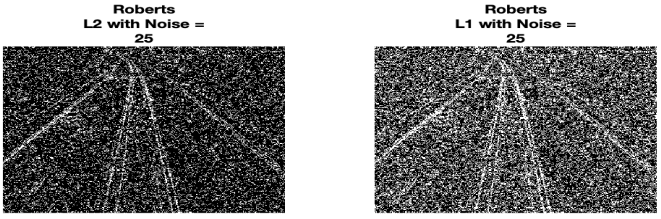

Figure 40: *road* - Roberts - Noise = 25

3 Compass Operator

Compass operators are utilizing a different differential gradient approximation technique, in which they are approximating gradients in given directions. In our case, Kirsch operator, namely a specific compass operator, uses 8 different directions. Typically, compass operators use 8 directions similar as the 8 different directions in the regular compass. These 8 different operators, namely $\{K_i | \forall i \in [0, 8]\}$, are actually 45^o counter-clockwise versions of each other for sequential i values. Mentioned Kirsch operators K_i are the following:

$$K_0(k,l) = \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix}, K_1(k,l) = \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix}, K_2(k,l) = \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

$$K_3(k,l) = \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}, K_4(k,l) = \begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix}, K_5(k,l) = \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix}$$

$$K_6(k,l) = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix}, K_7(k,l) = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}$$

After all the Kirsch operators are applied to the given image, gradient magnitude is chosen as the maximum absolute value among all 8 directions, and thresholded with given value. As in previous section of template method, Gaussian noise is applied to the original images as well before edge detection operation starts. Threshold and Gaussian noise parameters are as follows:

Gaussian	*Threshold*
[0, 5, 11, 25]	[165, 180]

Figure 41, 42 ,43 & 44 depict edge detection using Kirsch operators on image *lena*. On the other hand, Figure 45, 46 ,47 & 48 demonstrate Kirsch operators on *rice* and lastly, Figure 49, 50, 51 & 52 depict edge detection results of Kirsch operator on *road* image. Although, edge detection ensures sufficient results, even if threshold is high, there seems to be some noise that are included as edge points. Overall, method proves success in terms of existing edges and their detection. Therefore, it should be mentioned that compass operators are sensitive to noise. Moreover, when the Gaussian noise level increases, edge detection results are starting to be inadequate. When noise = 25, results are truly defective meaning that almost every coordinate are detected as edge points.

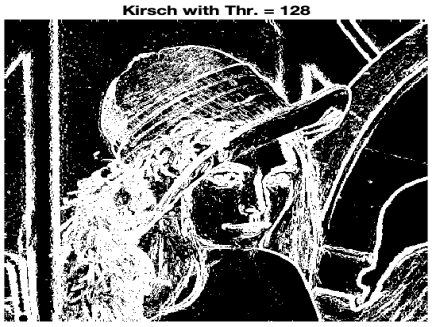

Figure 41: *lena* - Kirsch - Noise = 0

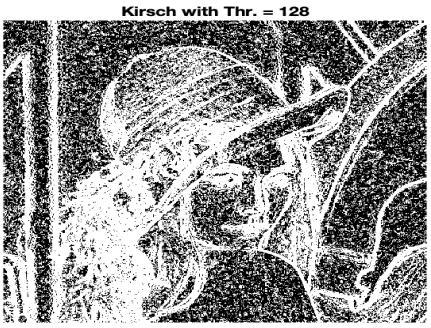

Figure 42: *lena* - Kirsch - Noise = 5

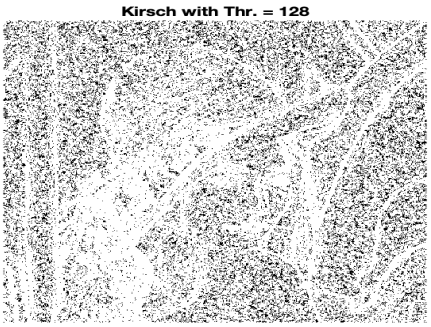

Figure 43: *lena* - Kirsch - Noise = 11

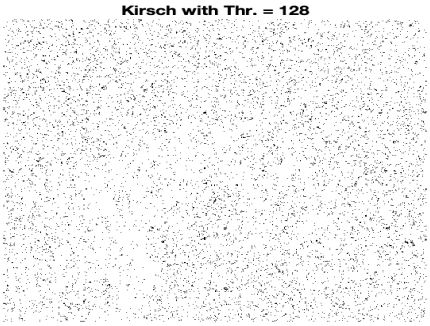

Figure 44: *lena* - Kirsch - Noise = 25

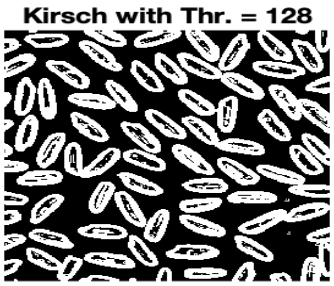

Figure 45: *rice* - Kirsch - Noise = 0

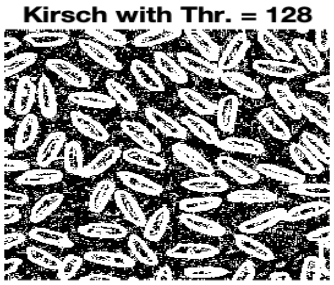

Figure 46: *rice* - Kirsch - Noise = 5

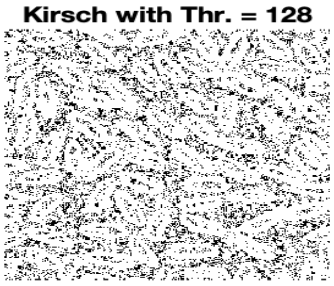

Figure 47: *rice* - Kirsch - Noise = 11

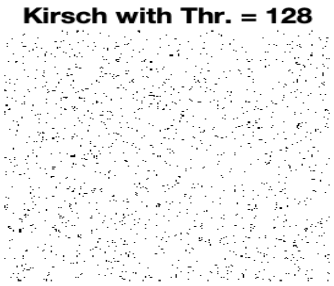

Figure 48: *rice* - Kirsch - Noise = 25

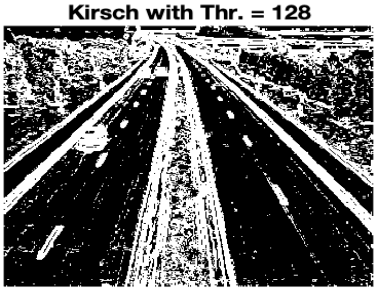

Figure 49: *road* - Kirsch - Noise = 0

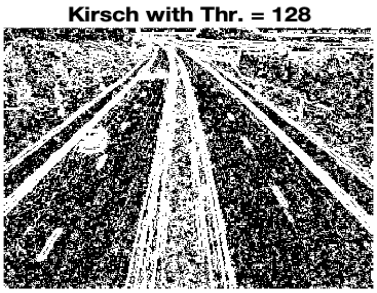

Figure 50: *road* - Kirsch - Noise = 5

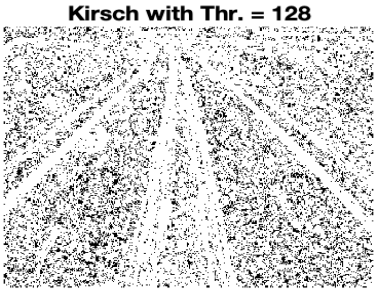

Figure 51: *road* - Kirsch - Noise = 11

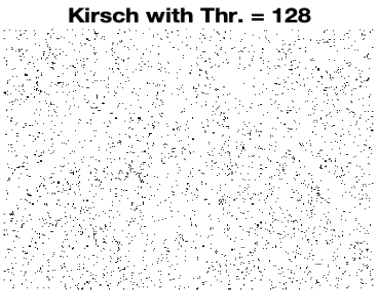

Figure 52: *road* - Kirsch - Noise = 25

4 Laplace Operator

Laplace operator is an advanced edge detection algorithm, in which noise sensibility is reduced to a certain level. This method uses LOG filter, whcih corresponds to laplacian of gaussian. Before edge detection, Gaussian filter is applied to the image to reduce Gaussian based noise in image. Main reason for this is, Laplace operator uses second order derivatives to approximate edges and second order derivatives are highly sensitive to noise. Therefore, Gaussian filter bridges the gap of noise sensitiveness. Gaussian filter is created from the gaussian formula, which is the following:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Gaussian filter can not be represented easily as σ values of Gaussian depend of the given sigma. A builtin Matlab function, **fspecial**, is used to form Gaussian filter with given filter size($2\lceil 3\sigma \rceil + 1$) and σ value. For sake of experimentation, *sigma* value is chosen as 2.5, $\in [2,3]$. Laplacian part of LOG filter uses second order derivatives as mentioned, which are the following:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

$$\frac{\partial^2 f(x,y)}{\partial x^2} \cong s(k+1,l) + s(k-1,l) - 2s(k,l) \quad (3)$$

$$\frac{\partial^2 f(x,y)}{\partial y^2} \cong s(k,l+1) + s(k,l-1) - 2s(k,l) \quad (4)$$

From 3 & 4 , equation 5 for the laplacian operator is derived as the following:

$$\nabla^2 = s(k+1,l)+s(k,l+1)+s(k-1,l)+s(k,l-1)-4s(k,l) = s(k,l)**laplacian(k,l) \quad (5)$$

Lastly, following parameters are used in our experimentation:

Gaussian	*Threshold*	σ
[0, 5, 11, 25]	[0.5, 0.9]	[2, 3]

Resulting Figures 53, 54, 55 & 56 demonstrate laplacian operator on *lena*, whereas Figure 57, 58, 59 & 60 depicts results of Laplacain operator on *rice*. Lastly, Figure 61, 62, 63 & 64 demonstrate Laplacian operator on *road* images. Obviously, Gaussian operation reduces noise of given images resulting in precise and accurate edge detection. Therefore, increasing noise does not affect performance of the method as in the previous methods. On the other hand, using second order derivatives to find edge coordinates turn out to be a highly effective method as thinner edges are detected in terms of necessity. Therefore, it would be reasonable to indicate that Laplace operator using LOG filter is one of the most successful methods in our lab exercise.

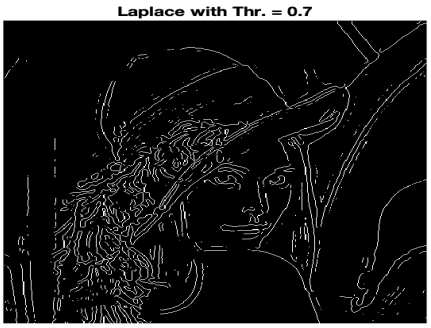

Figure 53: *lena* - Laplacian - Noise = 0

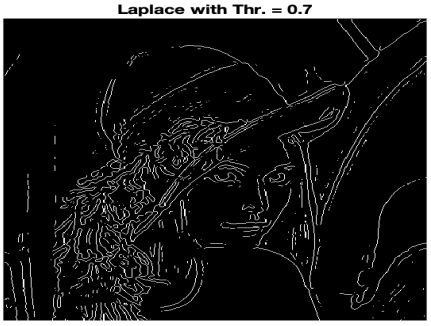

Figure 54: *lena* - Laplacian - Noise = 5

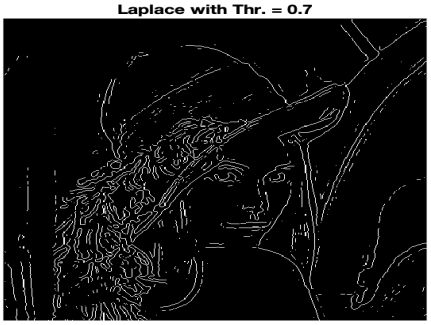

Figure 55: *lena* - Laplacian - Noise = 11

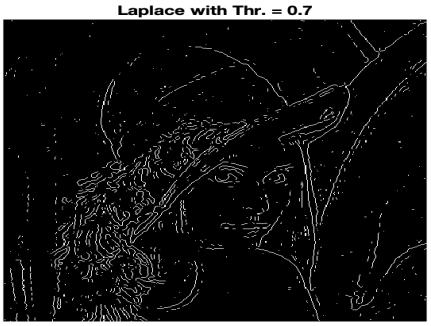

Figure 56: *lena* - Laplacian - Noise = 25

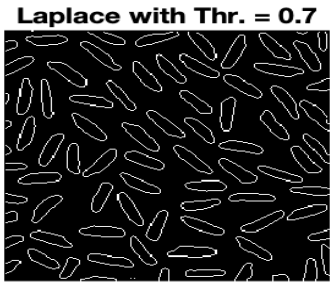

Figure 57: *rice* - Laplacian - Noise = 0

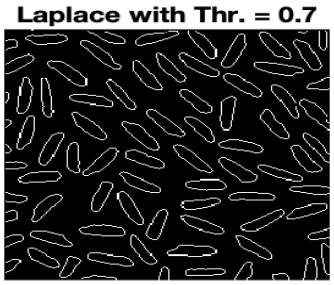

Figure 58: *rice* - Laplacian - Noise = 5

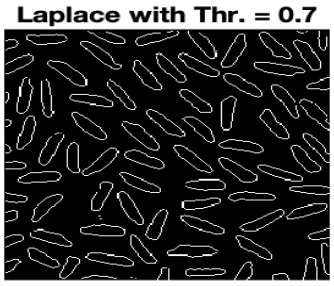

Figure 59: *rice* - Laplacian - Noise = 11

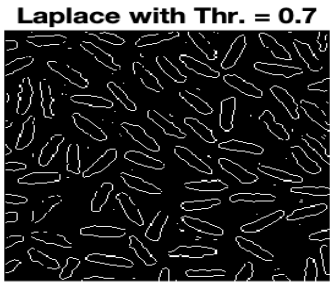

Figure 60: *rice* - Laplacian - Noise = 25

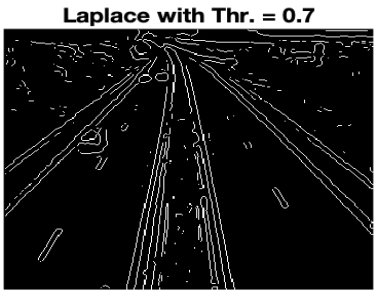

Figure 61: *road* - Laplacian - Noise = 0

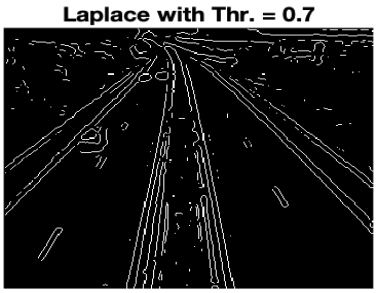

Figure 62: *road* - Laplacian - Noise = 5

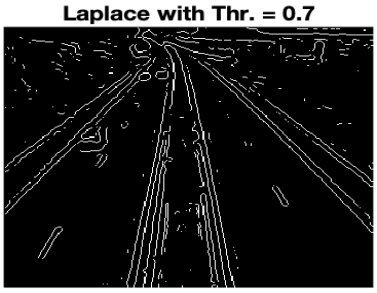

Figure 63: *road* - Laplacian - Noise = 11

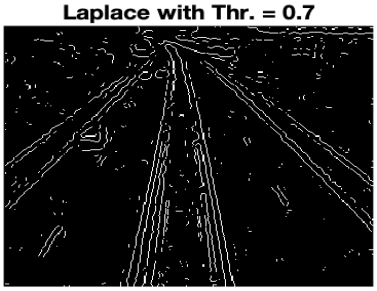

Figure 64: *road* - Laplacian - Noise = 25

5 Frei-Chen Method

Frei-Chen method is a different kind of approach to detect edges compared to the methods discussed. Frei-Chen method uses 9 different filter in order to detect local extremum in first order derivative and/or zero crossing in second order derivative. Image is projected onto 9 dimensional space by these filters, where 2 of the spaces capture magnitude information of the edges, rest 7 dimension capture information similar to line & uniform value distribution. Angles of corresponding pixels in 9 dimensions are calculated for edge's intensity information by the following equations:

$$m(k,l) = \sum_{k=0}^{1} (y_n(k,l))^2 \tag{6}$$

$$s(k,l) = \sum_{k=0}^{8} (y_n(k,l))^2 \tag{7}$$

From 6 & 7, equation 8 is ensured:

$$\alpha(k,l) = arccos(\sqrt{m(k,l)/s(k,l)}) \tag{8}$$

In our case, our only interest is the magnitude, therefore $cos(\alpha)$ is thresholded directly with given threshold parameters. Additionally, filters that are used for Frei-Chen are the following parameters:

$$FC_0 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix}, FC_1 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}, FC_2 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 0 & -1 & \sqrt{2} \\ 1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{bmatrix}$$

$$FC_3 = \frac{1}{2\sqrt{2}} \begin{bmatrix} \sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -\sqrt{2} \end{bmatrix}, FC_4 = \frac{1}{2} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, FC_5 = \frac{1}{2} \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}$$

$$FC_6 = \frac{1}{6} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}, FC_7 = \frac{1}{6} \begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix}, FC_8 = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

In blue boxes below, Gaussian noise parameters and threshold parameters are provided as in the previous methods.

Gaussian

[0, 5, 11, 25]

Threshold

[0.10, 0.15]

Figure 65, 66, 67 & 68 depict Frei-Chen results on *lena*, whereas Figure 69, 70, 71 & 72 depict Frei-Chen results on *rice* image. Lastly, Figure 73, 74, 75 & 76 Frei-Chen results on *road* image. Obviously, Frei-Chen is sensitive to noise as well, since edge detection accuracy decreases as the noise level increases. On the other hand, given small amount of noise, edge detection results are precise. Comparing to previous methods, Frei-Chen method can be considered as the most successful method after Laplace operator considering all of our implementations. Main problem of this method is the noise sensitivity, not as much as Compass operator, & thick edges. In terms of thinner edges, only Laplace operator proved prospering results. Nevertheless, Frei-Chen method should be considered as an unique operation in terms of methodology and accuracy.

Figure 65: *lena* - Frei-Chen - Noise = 0

Figure 66: *lena* - Frei-Chen - Noise = 5

Figure 67: *lena* - Frei-Chen - Noise = 11

Figure 68: *lena* - Frei-Chen - Noise = 25

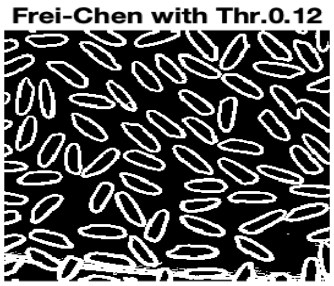

Figure 69: *rice* - Frei-Chen - Noise = 0

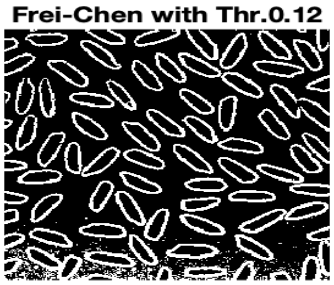

Figure 70: *rice* - Frei-Chen - Noise = 5

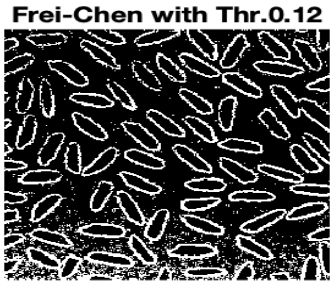

Figure 71: *rice* - Frei-Chen - Noise = 11

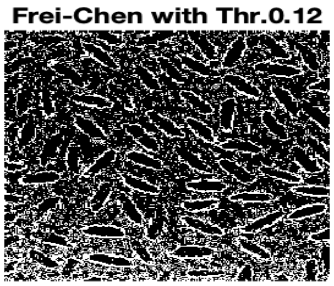

Figure 72: *rice* - Frei-Chen - Noise = 25

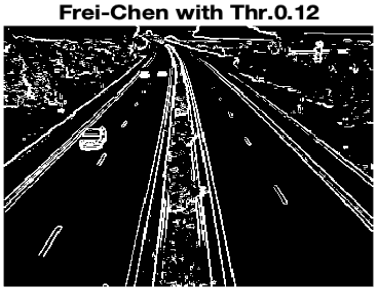

Figure 73: *road* - Frei-Chen - Noise = 0

Figure 74: *road* - Frei-Chen - Noise = 5

Figure 75: *road* - Frei-Chen - Noise = 11

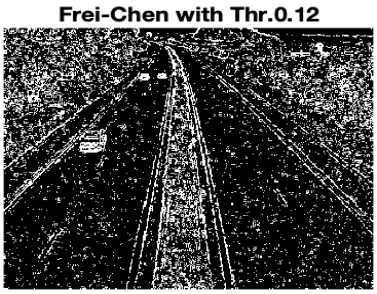

Figure 76: *road* - Frei-Chen - Noise = 25

6 Complexity Analysis & Performance Comparison

To conclude, we will be discussing algorithms complexity, efficiency and overall performances. For this reasons, we will denote image sizes as NxM and filters by FxF. In other words, doing a pixelwise operation on an image would cost $\Theta(NM)$ and a pixelwise convolution would cost $\Theta(F^2NM)$.

For template methods, $3x3$ filters are convolved with given images in both horizontal and vertical directions. Therefore, for each of these methods overall complexity is $\Theta(9NM)$, namely $\Theta(F^2NM)$. In terms of efficiency, Sobel and Prewitt filters provide more accurate results then Roberts and more robust to noise. Their, Sobel and Prewitt, overall performance are certainly worth to visualize as they have low complexities as well.

Compass operators, more specifically, Kirsch operator uses 8 different filters which are of size $3x3$ and they are rotations of each other. Creation of these filters takes almost no time, convolution of filters with image takes $\Theta(F^2NM)$ for each filter. Taking absolute value among all the projected dimensions take $\Theta(KNM)$, where K stands for filter amount/dimension size. In other words, total required time is $\Theta(KF^2NM) + \Theta(KNM)$, which comes to $\Theta(KF^2NM)$. Performance of compass operators shows promise, although noise sensitivity appears as a big problem.

Laplace operator, LOG filter, uses $\Theta(F^2NM)$ for Gaussian noise filtering. As laplacian operation, every pixel is compared with its neighbor. We will denote neighbourhood as specific window of size W, therefore Zero Crossing takes $\Theta(WNM)$. Total time required is $\Theta(F^2NM) + \Theta(WNM)$. LOG operator is one of the best methods in terms of its accuracy and precision as detected edges are very thin. Noise sensitivity is avoided with respect to Gaussian filtering.

Lastly, Frei-Chen method will be discussed. Filter creation takes almost no time, and 9 different filters of same sizes are created that are of size $3x3$. Total time required to convolve these filters is $\Theta(9 * F^2NM)$, more generally $\Theta(F^2NM)$. Noise sensitivity is still an issue in this method, however methodoloy is unique as mentioned in previous subsection. Edge detection performance of Frei-Chen is accurate as well. In terms of thinner edges, LOG is the best method. However, Frei-Chen method provides one of the best results in non-noisy images given the right set of parameters.