Image and Video Processing
Lab 3: Edge and contour detection

Yuliang Zheng, Valentine Santarelli

November 2016

Declaration

For each method here, we first did the experiment with the original, clean image (*i.e.* without adding any noise), and the result for this is titled as **clean**. As we are asked to repeat all the operations while adding Gaussian noise with different standard deviations (namely 5, 11, 25 with respect to dynamic range [=0, 255]), the corresponding results are titled as $\sigma = a$, where a is the standard deviation.

Template method

Using the gradient to find the intensity variation is one of the key ideas for edge and contour detection. Here, the *Template method* approximates the gradient. The templates we utilize here basically evaluate two directions. As we only consider the magnitude of gradient, two strategies are reasonable for selecting the largest magnitude, namely the *L1-norm* and *L2-norm*.

A function named *template_method* is created, with 5 parameters as the input (the template we choose, the input image, the norm and the threshold). Figure 1 to Figure 9 are the result of using *L1-norm*; while Figure 10 to Figure 18 are of *L2-norm*.

Obviously, the **clean** image offers the best result, and adding a Gaussian noise to the input image will also add proportional noise on the *edge and contour detection* result. As a result, the *edge and contour detection* result is noisy as well, and it's harder to find the edges. However, the implementation of *L1-norm* or *L2-norm* do make a difference to the *edge and contour detection* result. As *L1-norm* is the square root of the sum of two component square, while *L2-norm* is the sum of magnitude of the same two component. Clearly, the *L2-norm* will result in a larger number, which will have higher possibility to be set as 255(white) because of the threshold.

What's more, when changing the value of threshold, a larger threshold gives less edge and contour information. As we set the background as black (value of 0), with a higher threshold, more pixels are set of value 0 (black). Consequently, higher threshold will provide less edge and contour information. And among the three threshold(i.e. $T = 30, 70, 100$), $T = 30$ offers the best result with a more clear contour plot.

With the L1 norm

With T=30

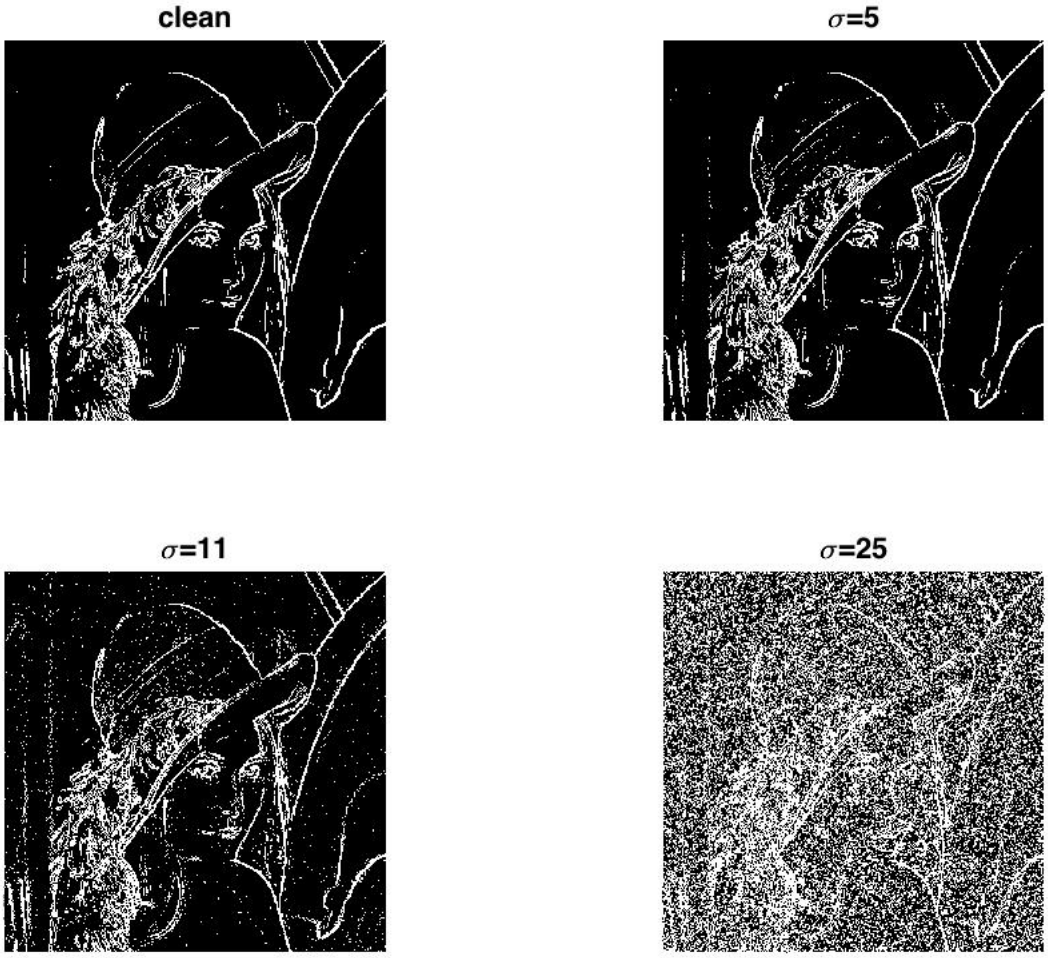

Figure 1: lena.png applying the template method with the L1 norm and a threshold T=30

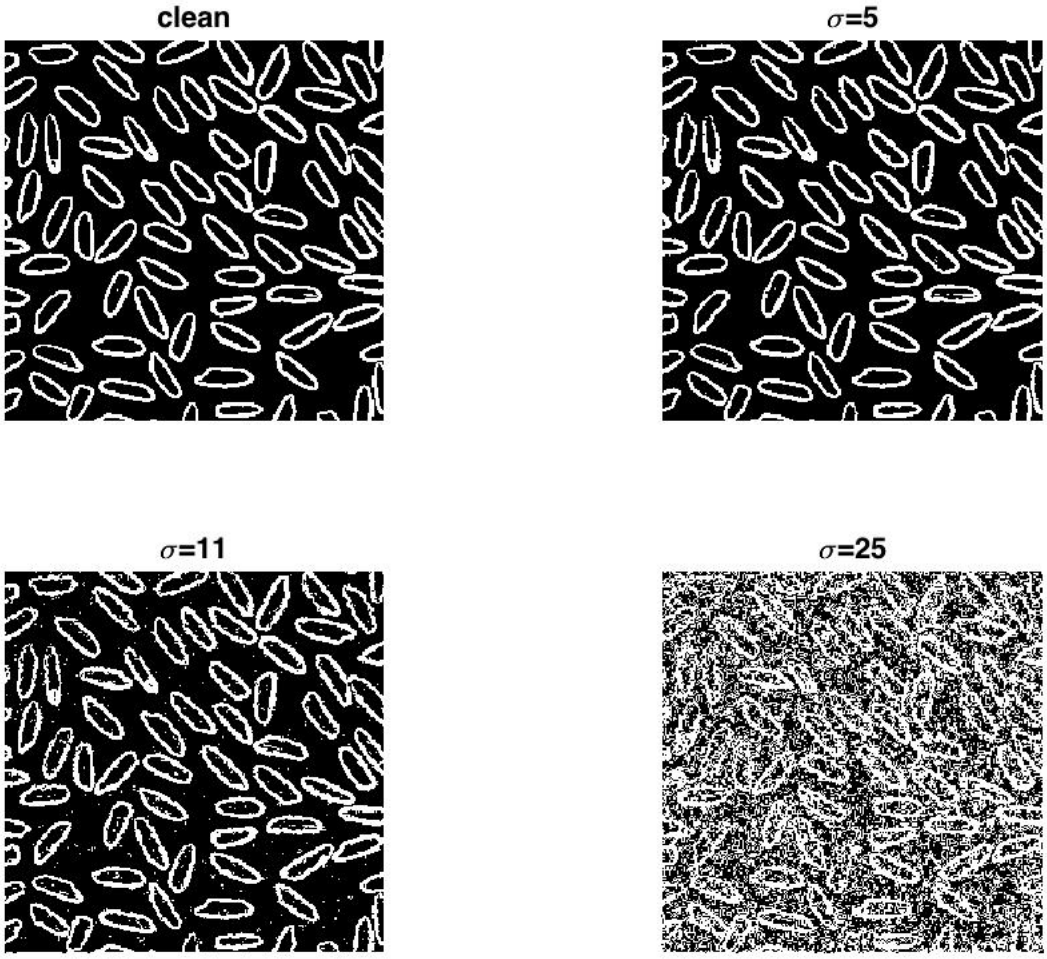

Figure 2: rice.png applying the template method with the L1 norm and a threshold T=30

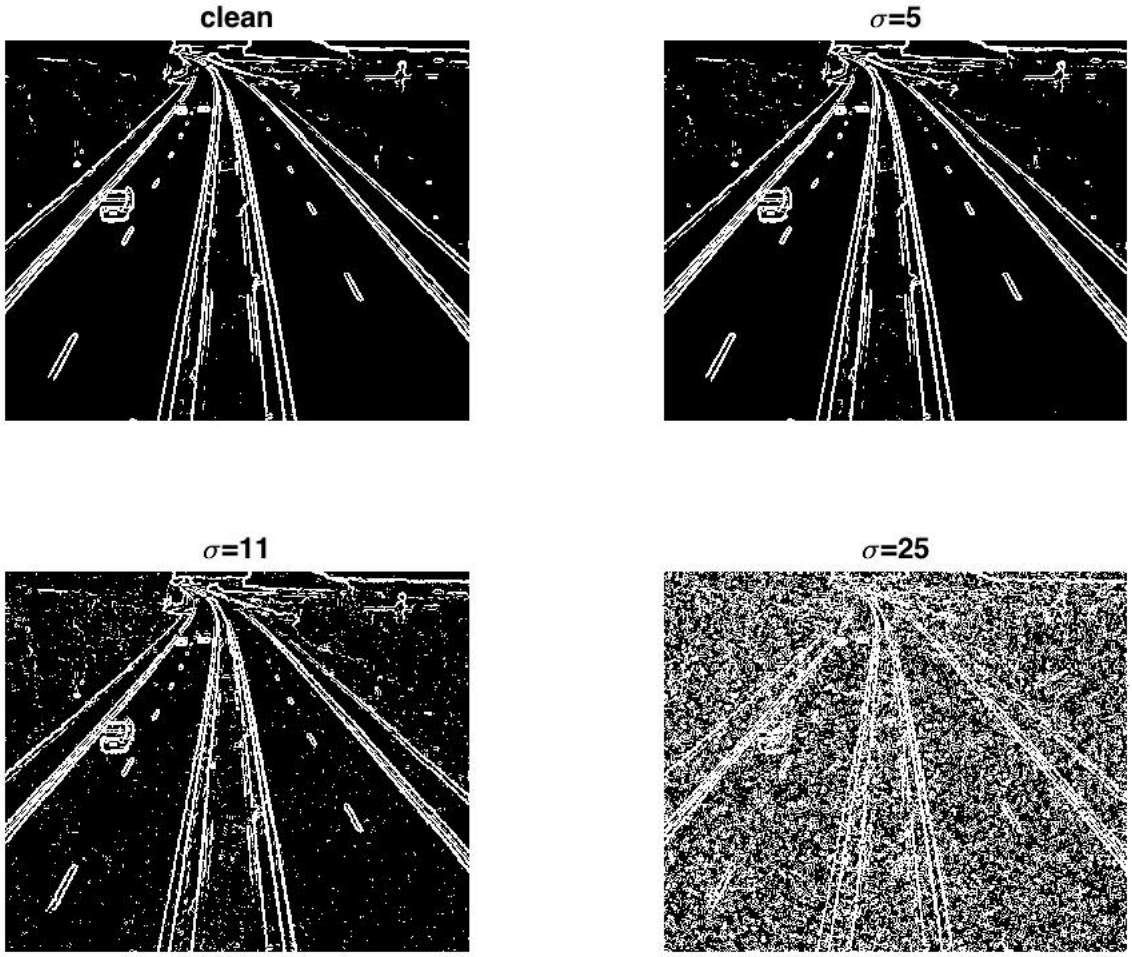

Figure 3: road.png applying the template method with the L1 norm and a threshold T=30

With T=70

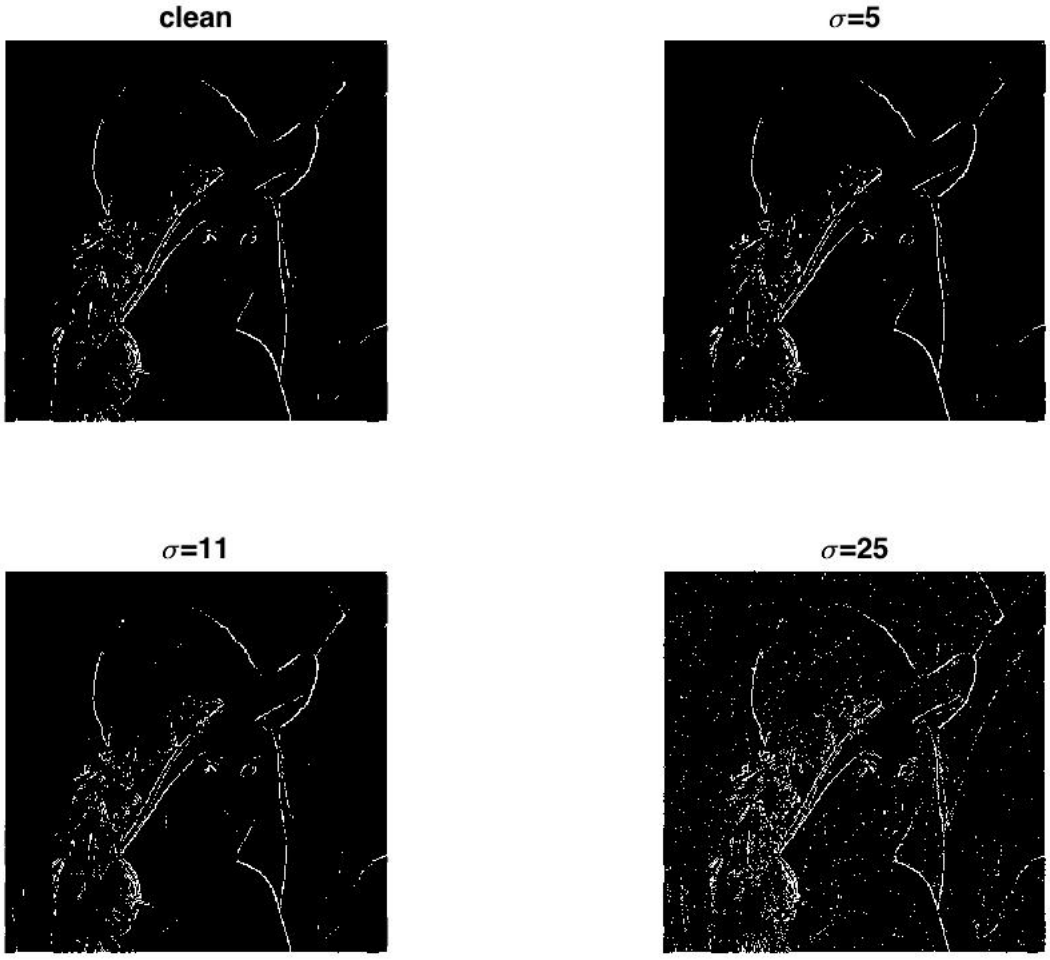

Figure 4: lena.png applying the template method with the L1 norm and a threshold T=70

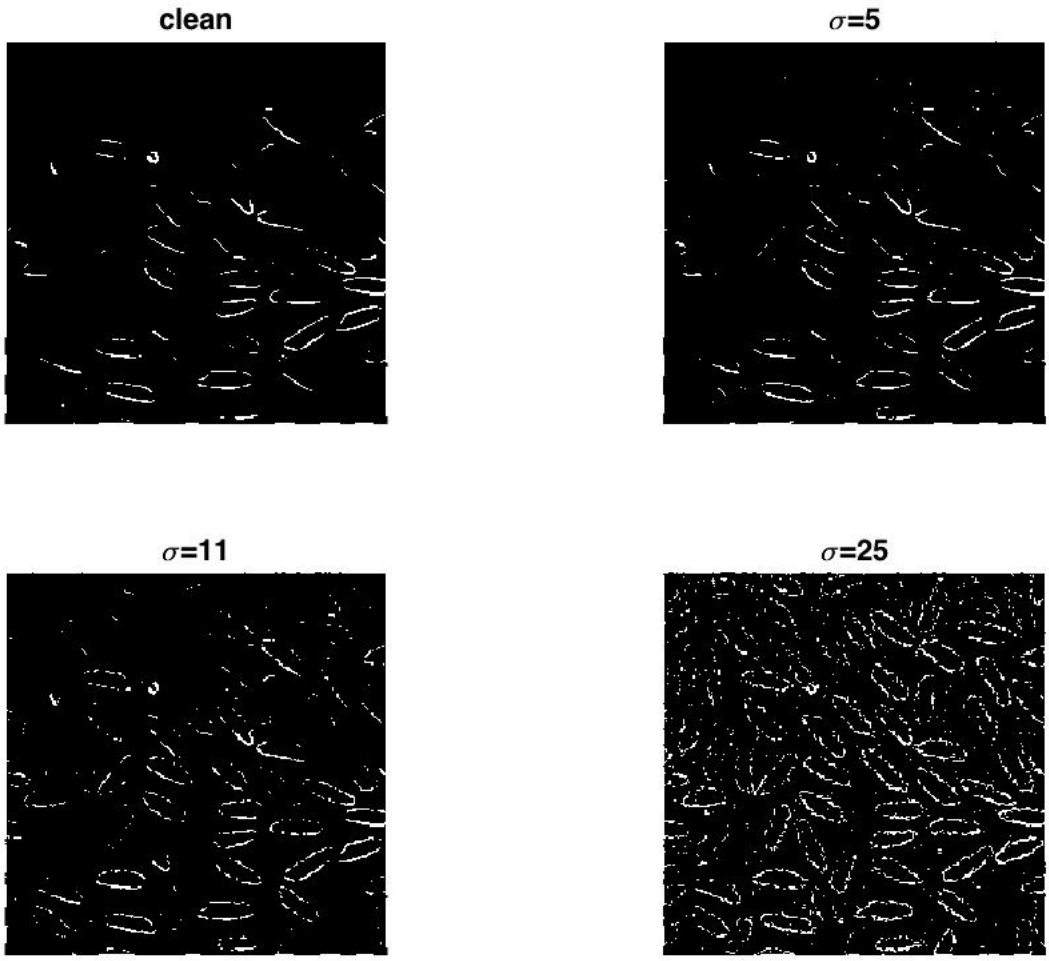

Figure 5: rice.png applying the template method with the L1 norm and a threshold T=70

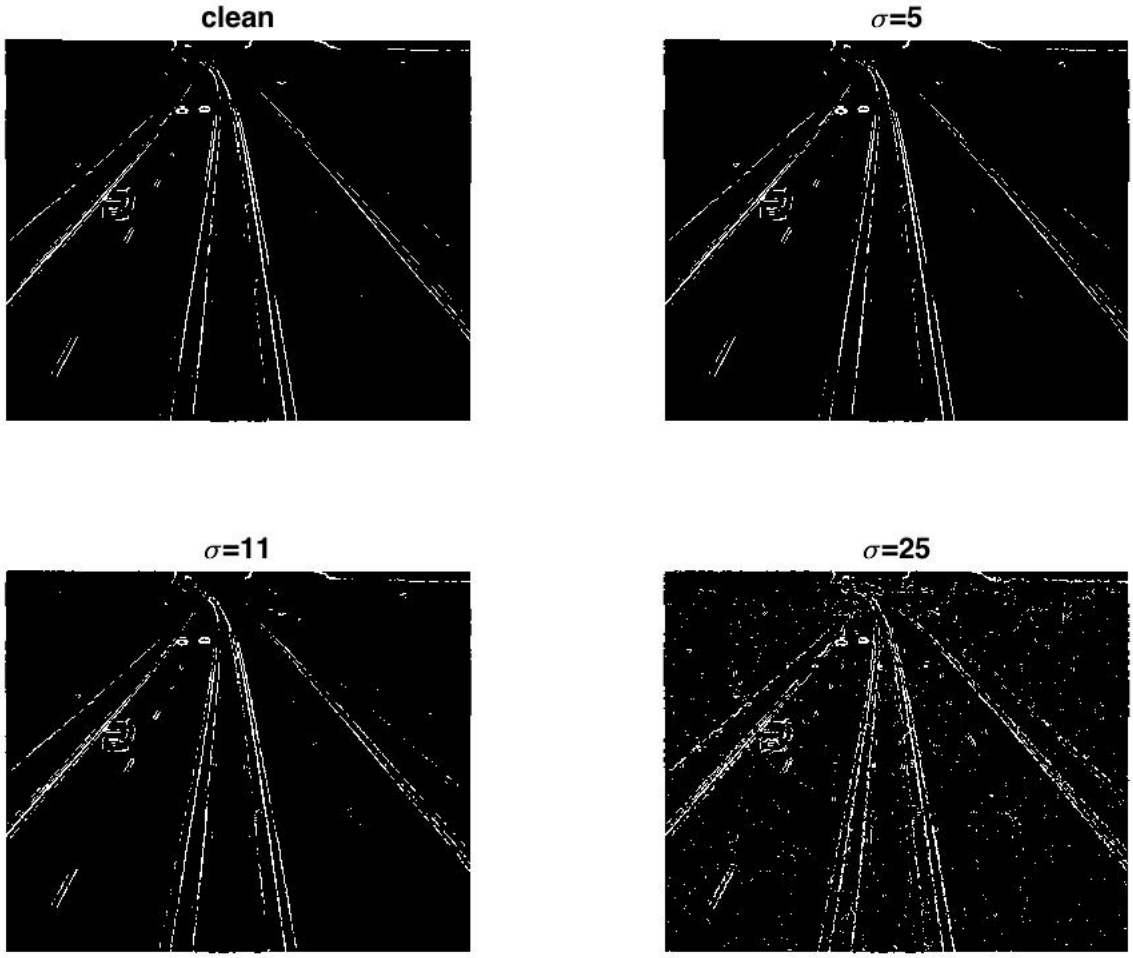

Figure 6: road.png applying the template method with the L1 norm and a threshold T=70

With T=100

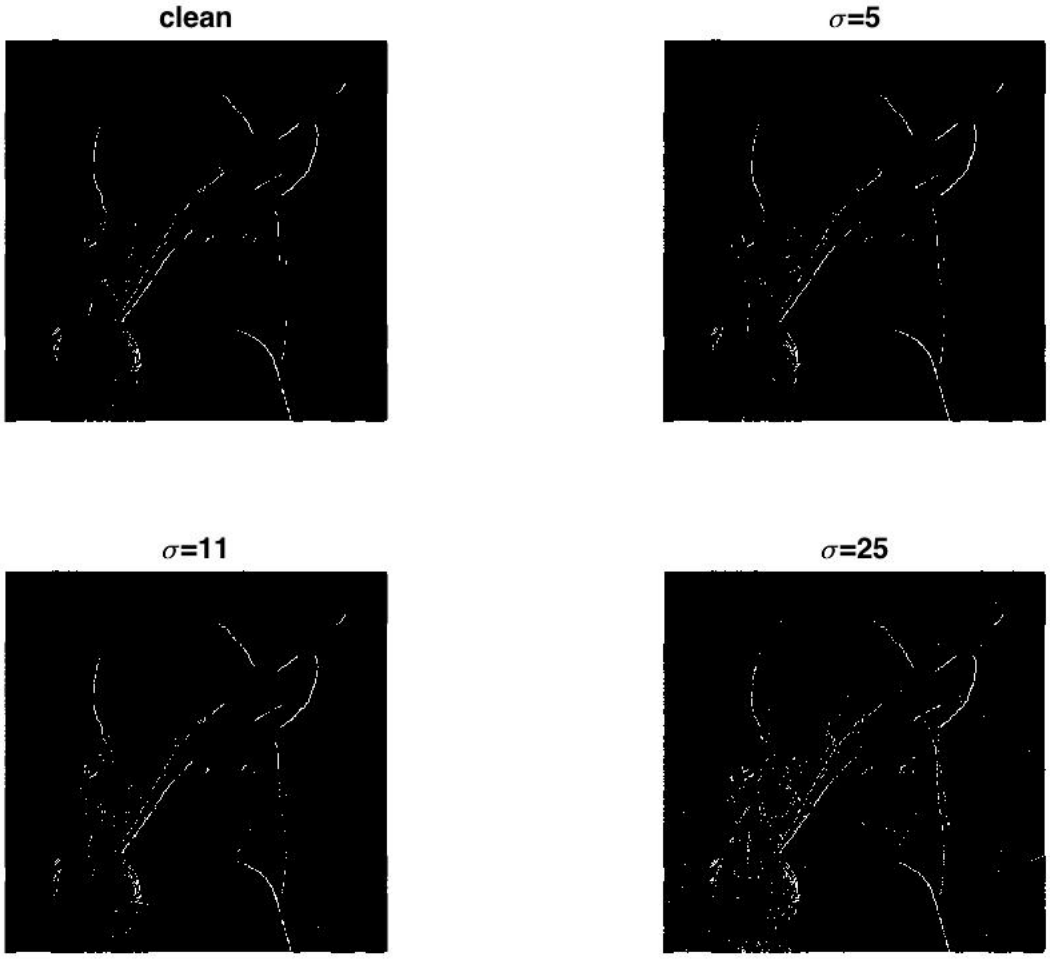

Figure 7: lena.png applying the template method with the L1 norm and a threshold T=100

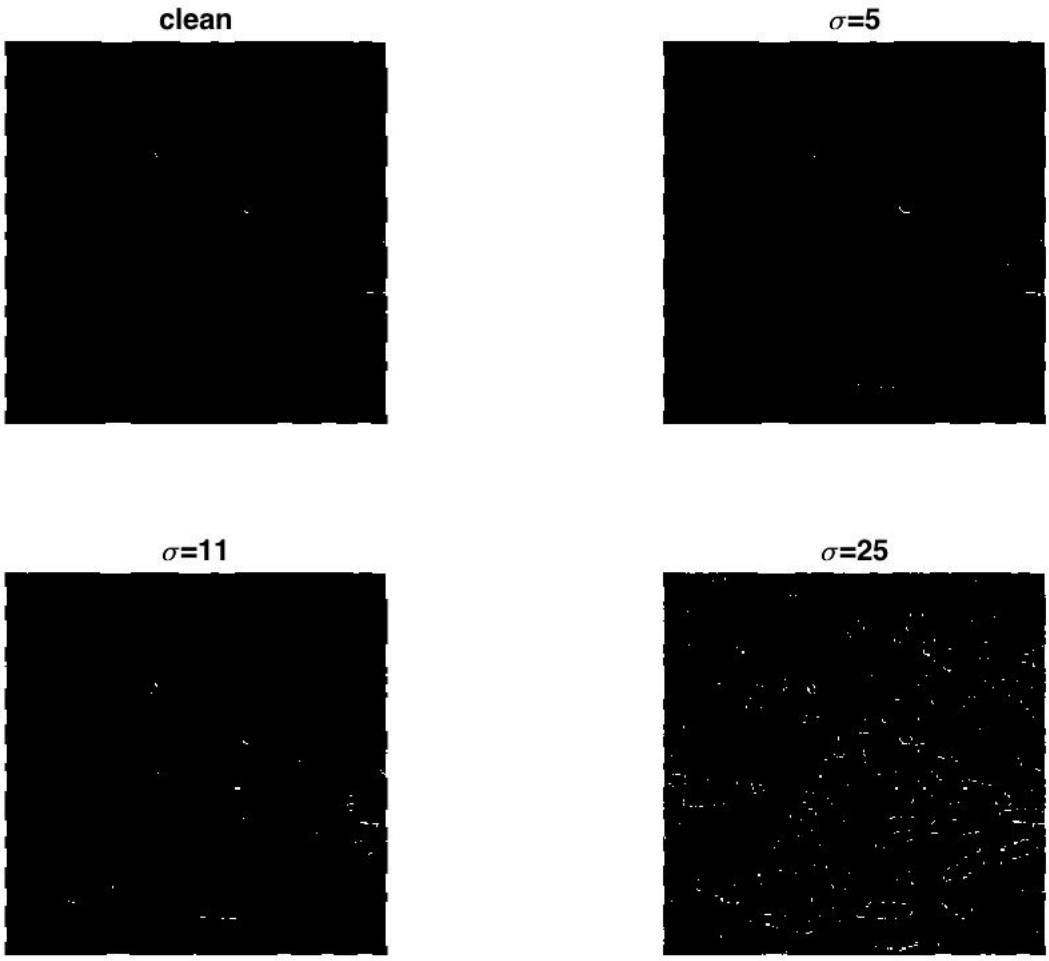

Figure 8: rice.png applying the template method with the L1 norm and a threshold T=100

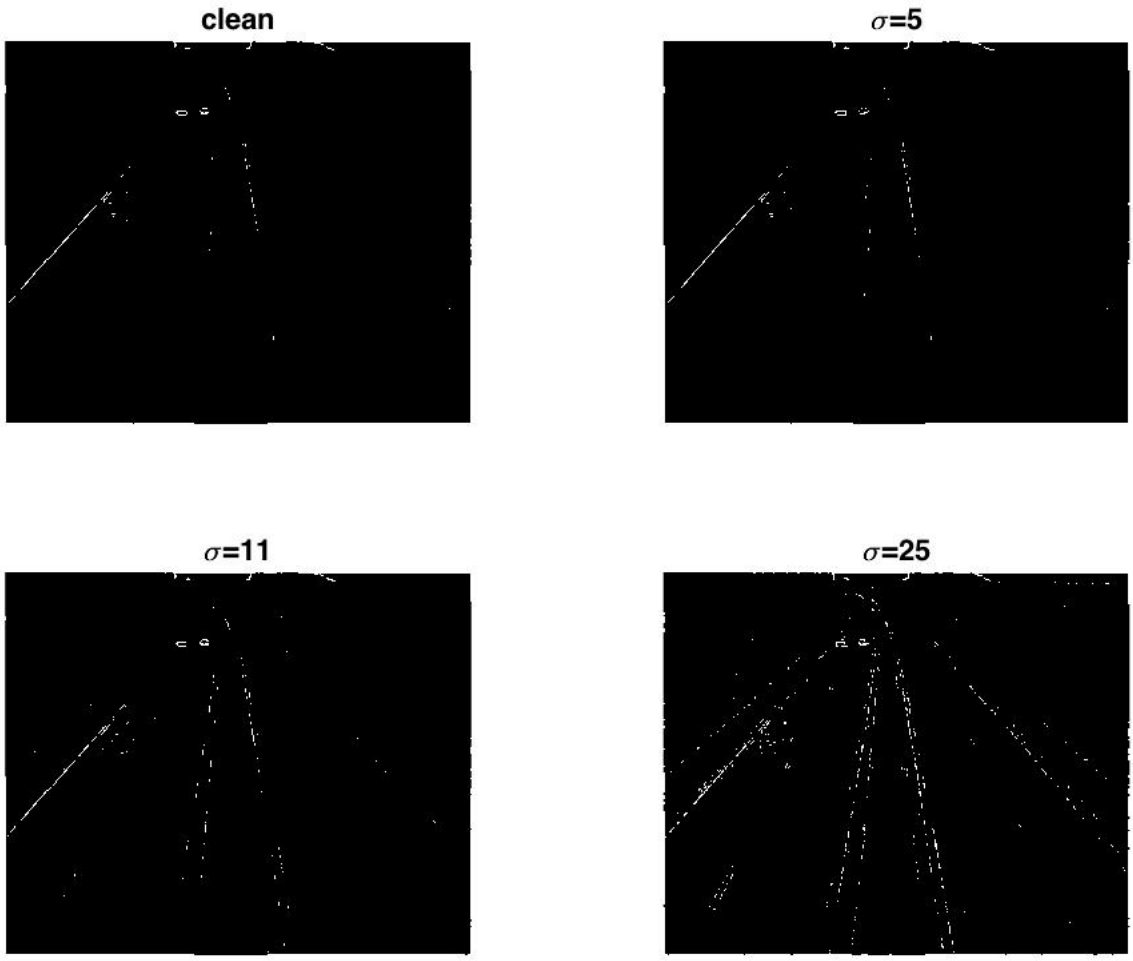

Figure 9: road.png applying the template method with the L1 norm and a threshold T=100

With the L2 norm

With T=30

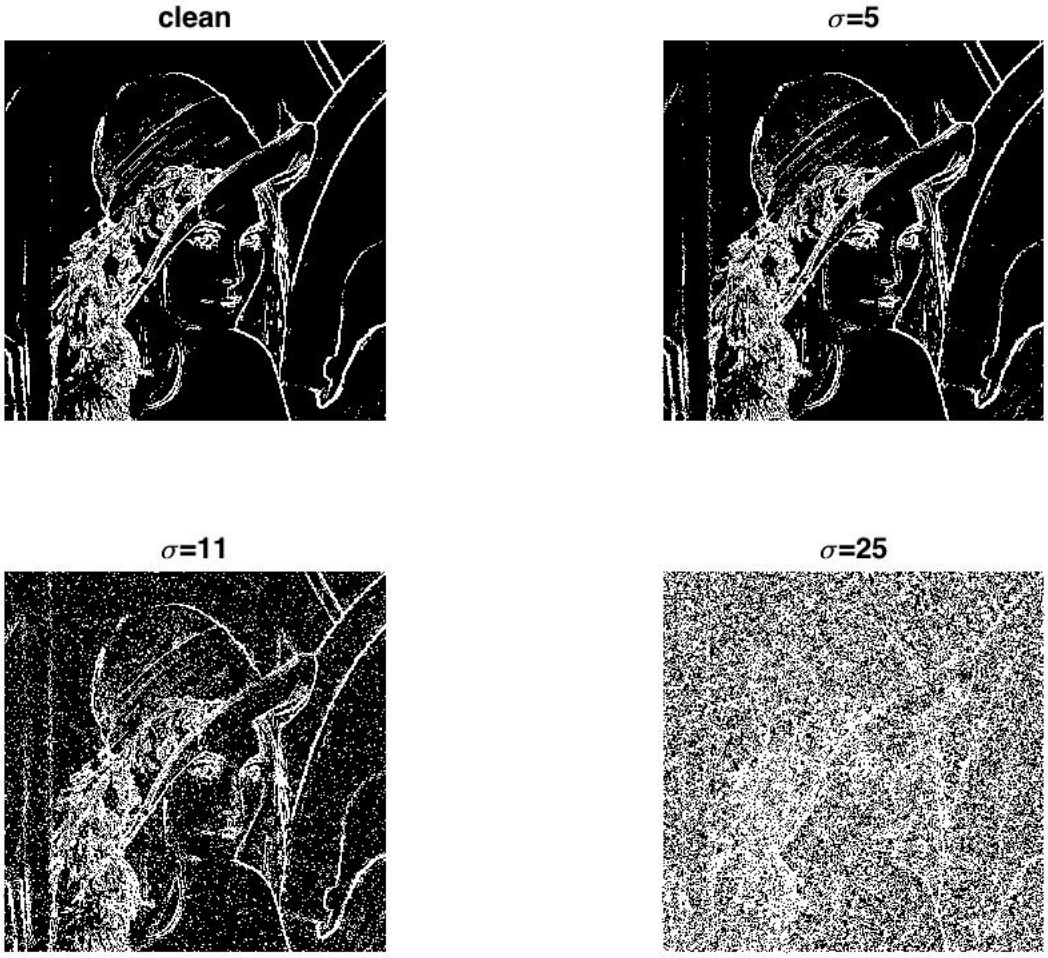

Figure 10: lena.png applying the template method with the L2 norm and a threshold T=30

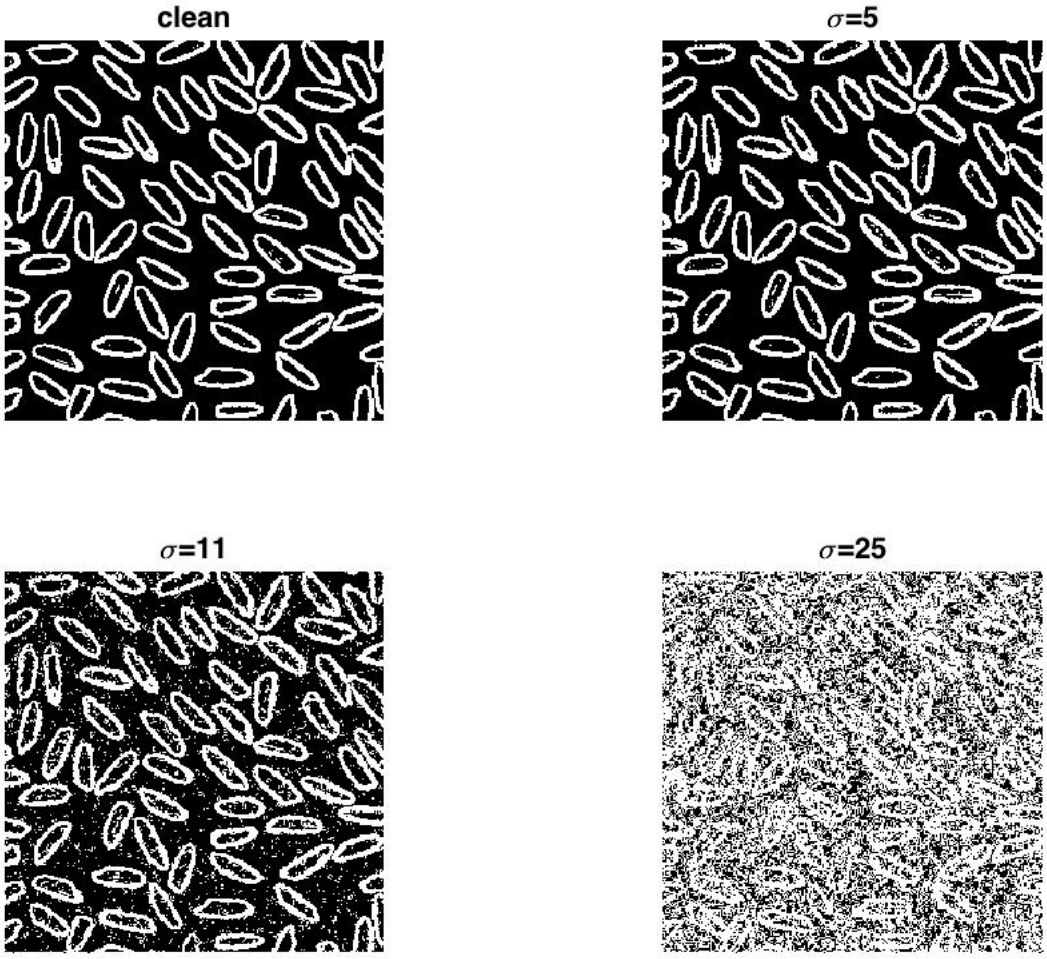

Figure 11: rice.png applying the template method with the L2 norm and a threshold T=30

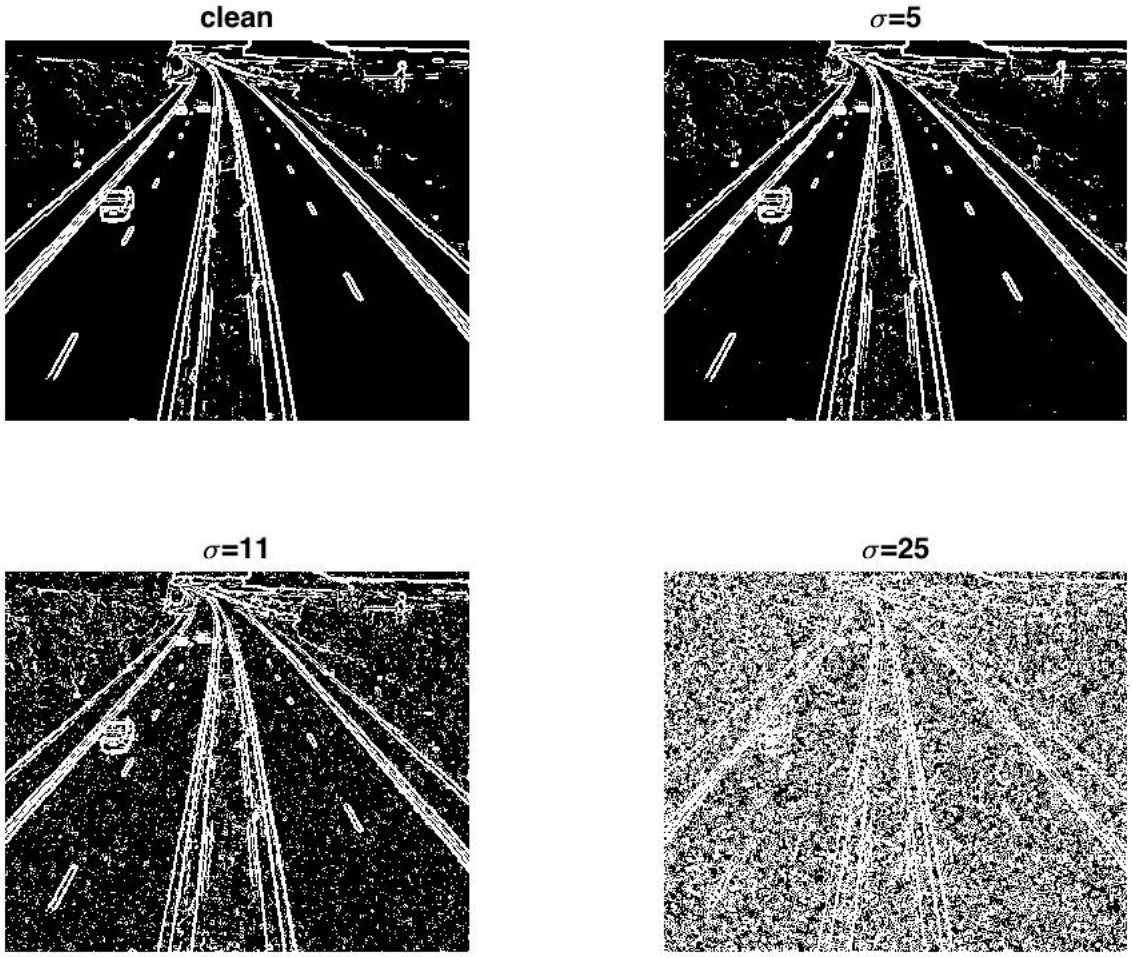

Figure 12: road.png applying the template method with the L2 norm and a threshold T=30

With T=70

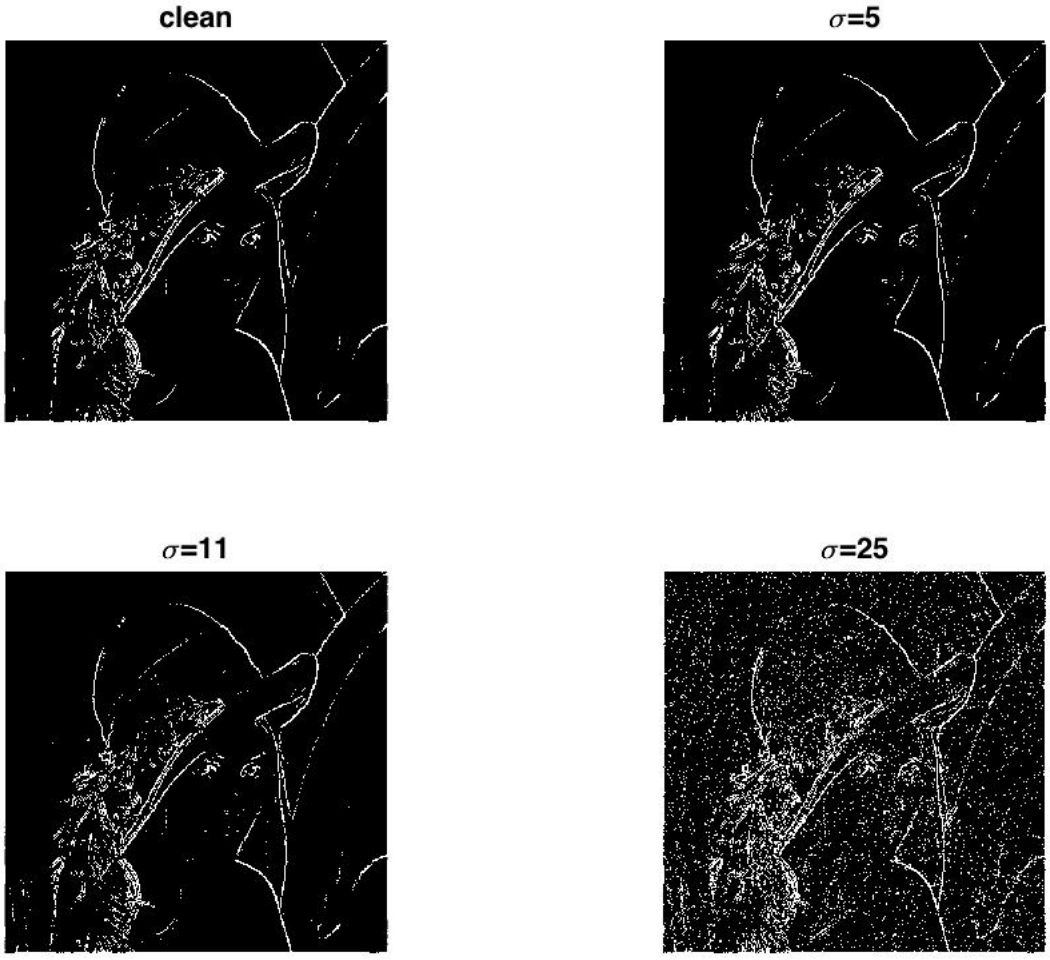

Figure 13: lena.png applying the template method with the L2 norm and a threshold T=70

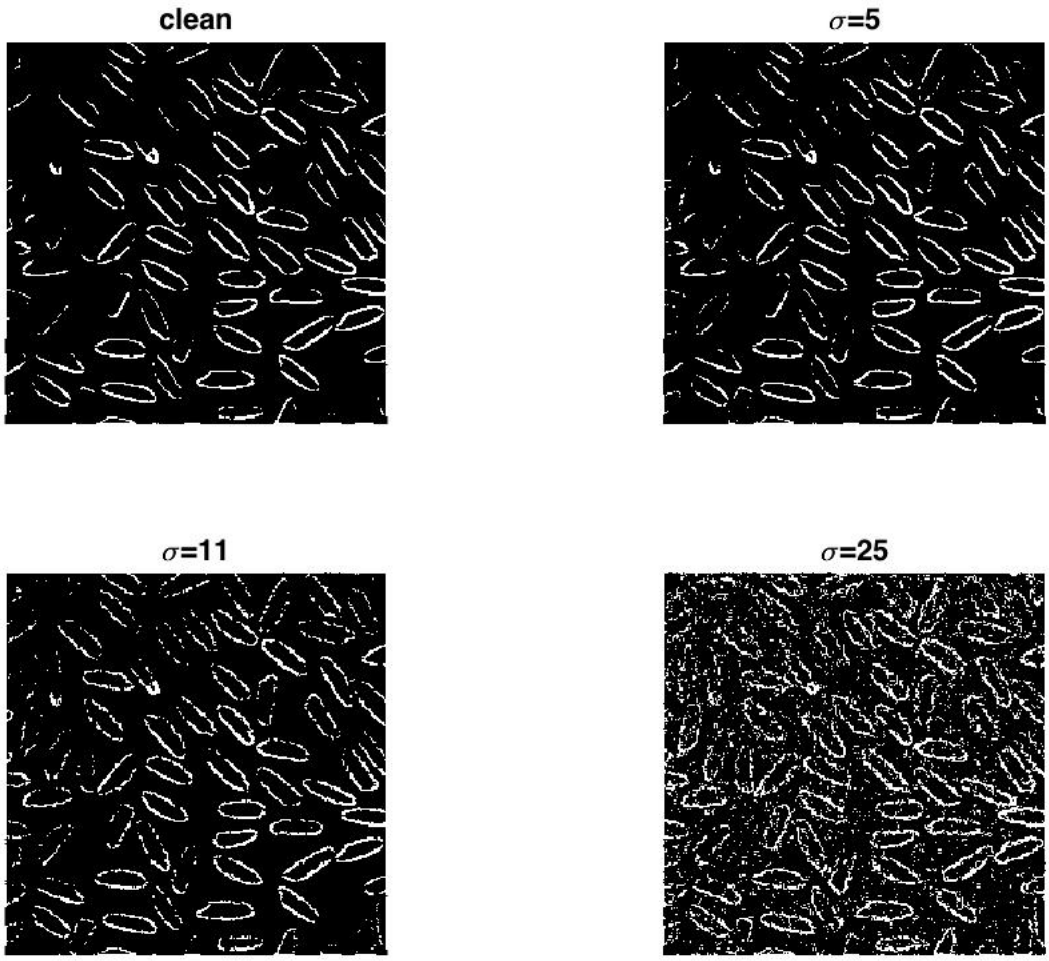

Figure 14: rice.png applying the template method with the L2 norm and a threshold T=70

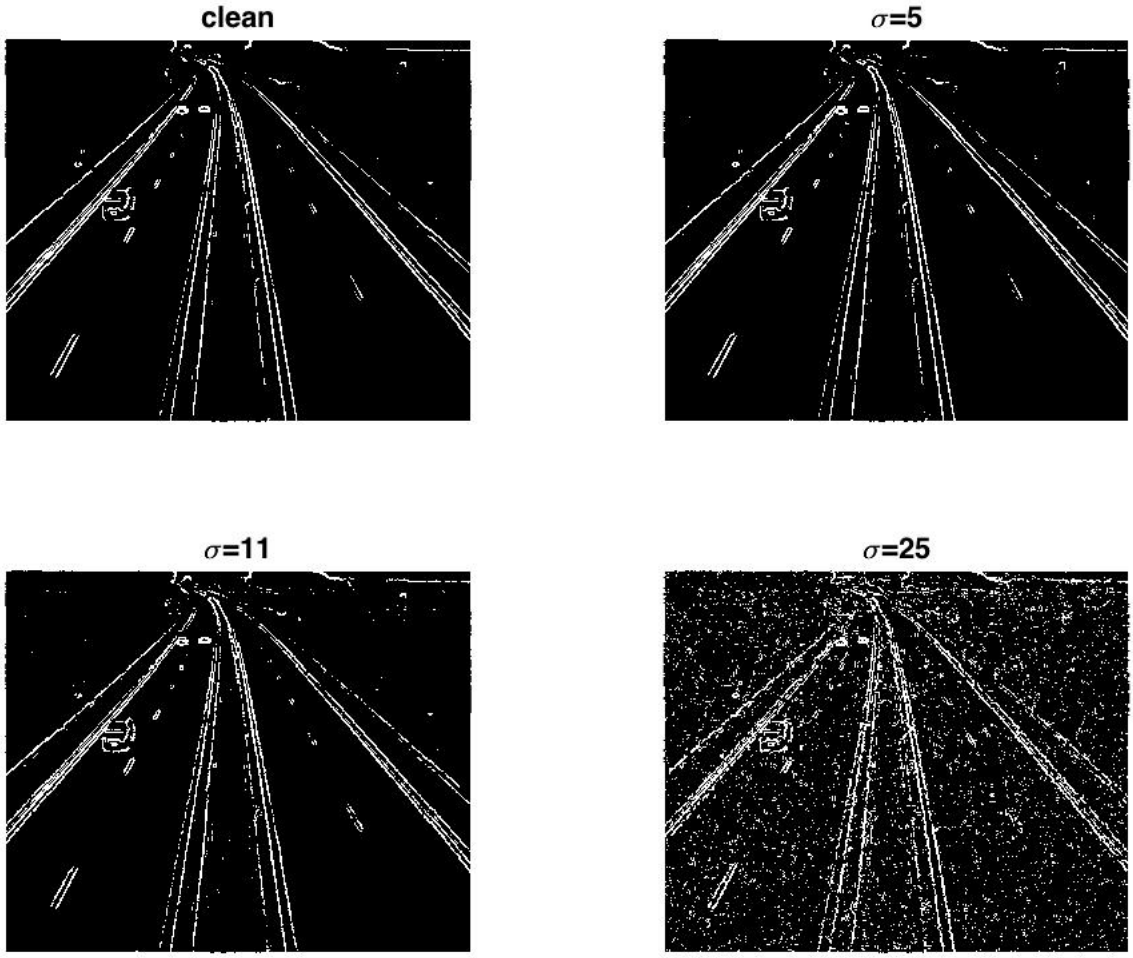

Figure 15: road.png applying the template method with the L2 norm and a threshold T=70

With T=100

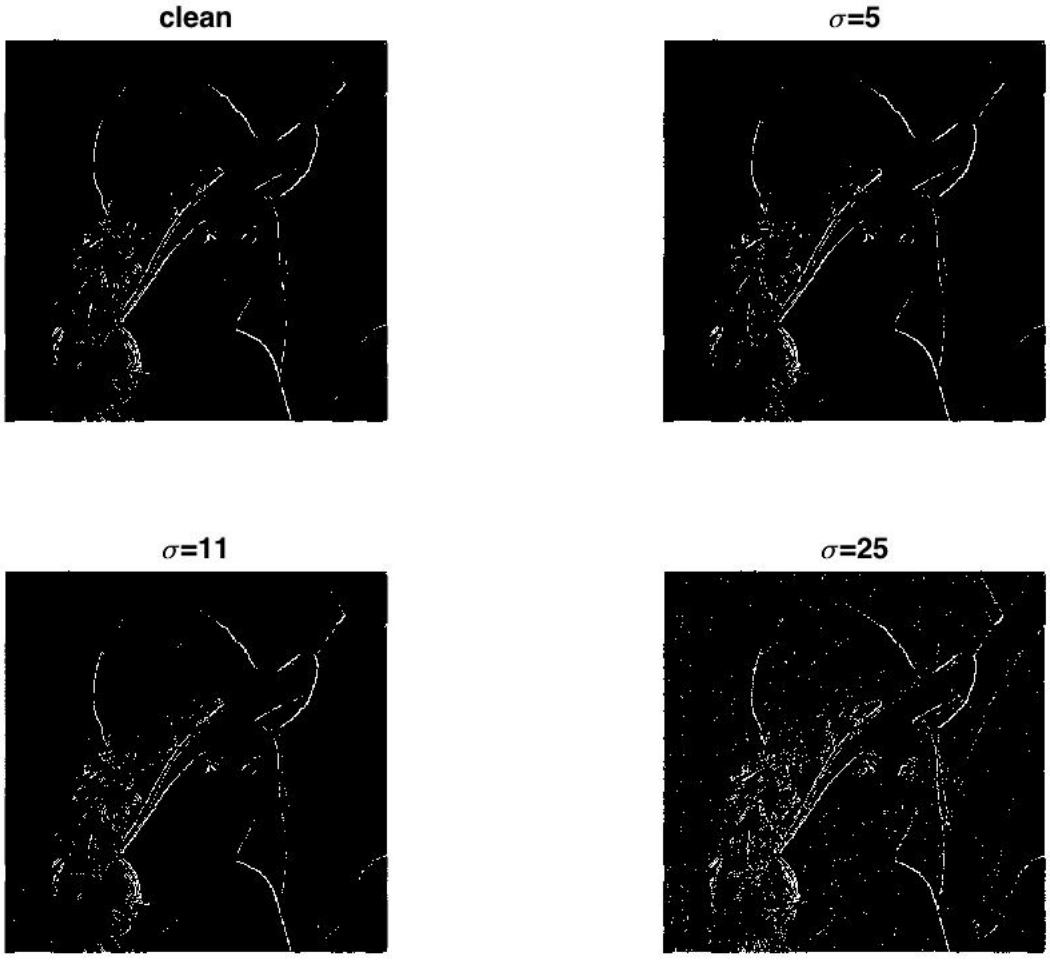

Figure 16: lena.png applying the template method with the L2 norm and a threshold T=100

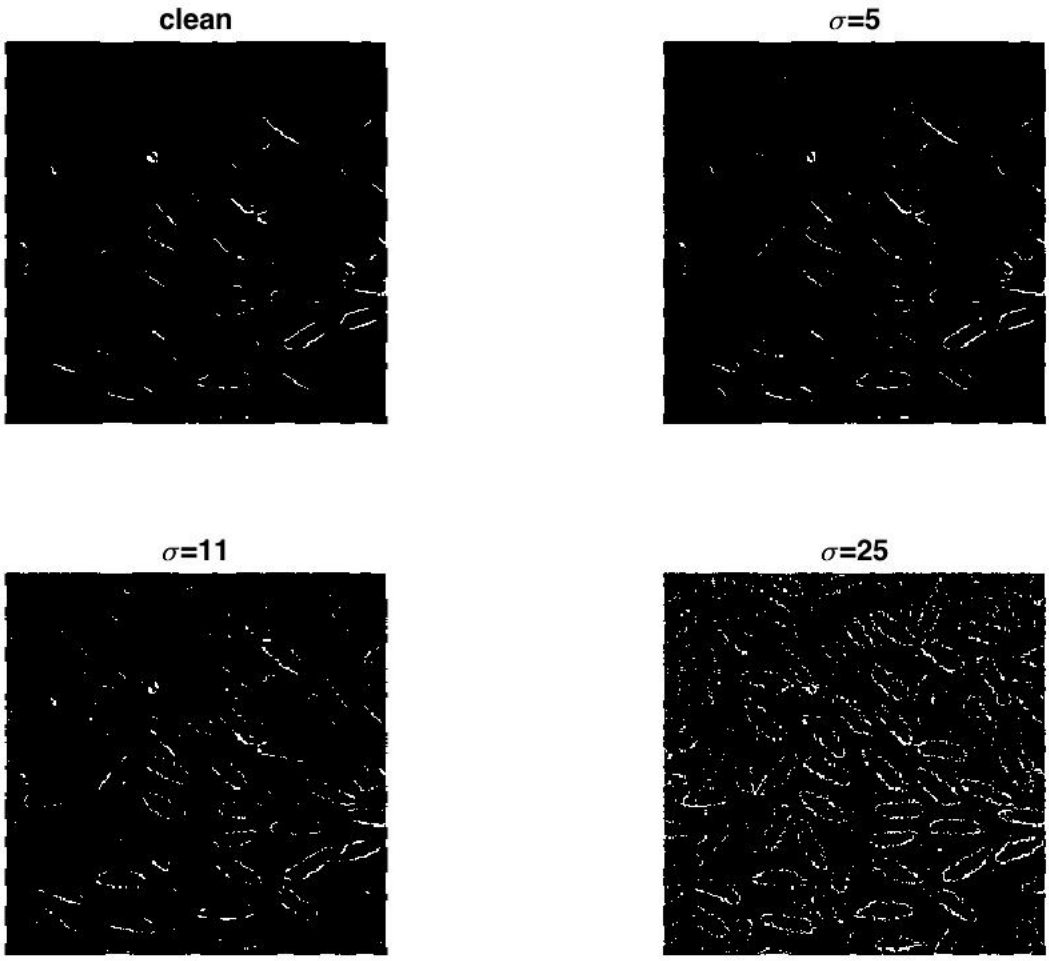

Figure 17: rice.png applying the template method with the L2 norm and a threshold T=100

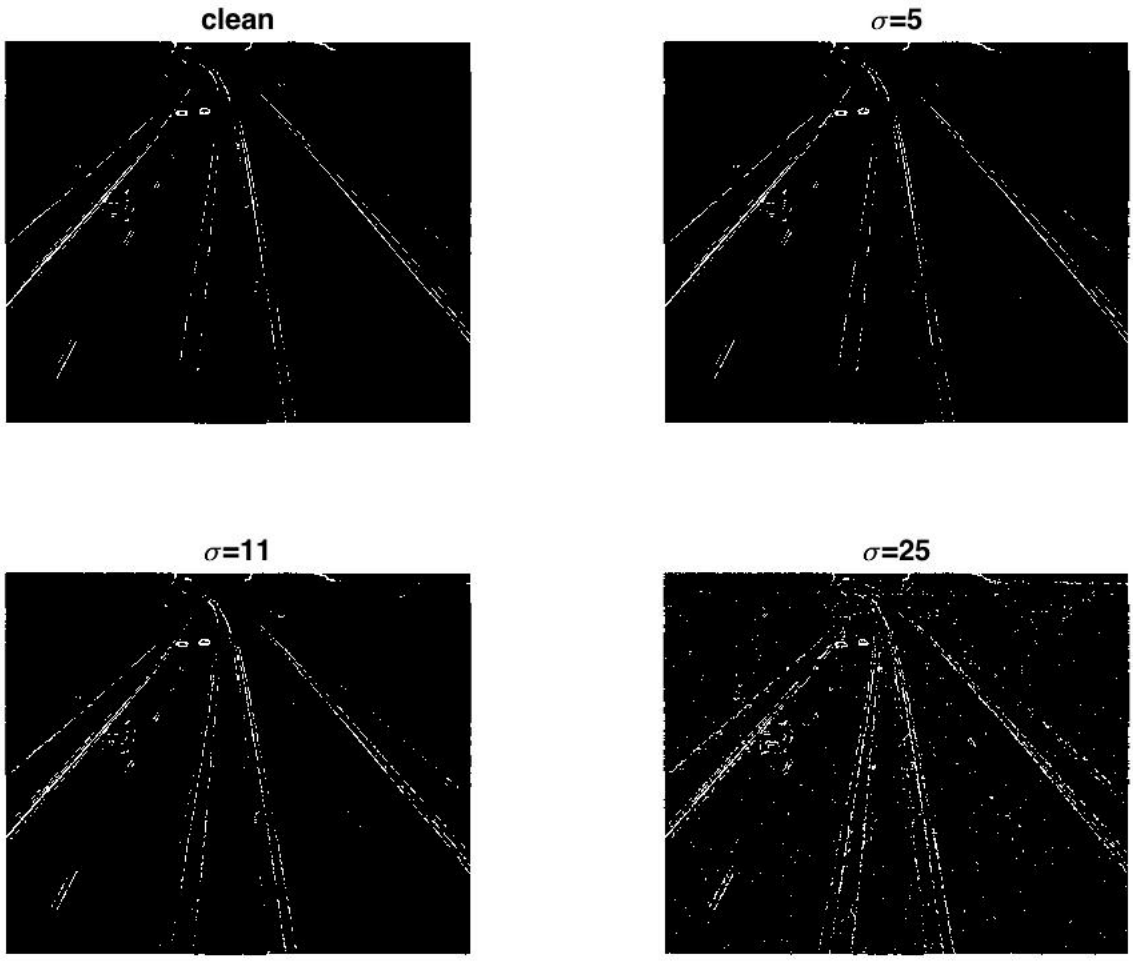

Figure 18: road.png applying the template method with the L2 norm and a threshold T=100

Compass operator

The normal way to detect edges and contours is to find the areas of images where a sharp intensity variation between pixels exists, and computing the *gradient* is good method to detect the variations. In *Compass operator*, eight directions are tried in this method, and we only pick up the one with the largest magnitude.

Here, a function named *compass_operator* is created, with two input parameters (i.e. the input image x, and the threshold T). By doing a convolution of the image and each compass operator separately, we will get eight intermediate results, and before using the threshold, at each pixel, we assign the one with largest magnitude among the eight intermediate results to the output.

Still testing with different threshold, at first without doing the normalization, the results were quite different from the previous method *Template method*, a higher threshold seems to give more useful edge and contour information. This difference may be caused by the implementation of operators in distinct magnitude. Instead of using a nearly normalized operator, in *Compass operator*, each direction's operator is in a much larger magnitude, which consequently has effect like multiplying a large integer (at least more than 1) to each pixel. Then, each pixel is not in range 0 to 255. However, when arriving a relatively high magnitude (the peak value), after that, if we still increase threshold, we will get less edge and contour information as previous experiment.

Then, we did a normalization to the resulting image, and set every pixel's range from 0 to 255. And the same law of change for the result was gotten as previous, when changing the threshold.

Compared to the *Template method*, the *Compass operator* seems to offer more edges and contours information.

With T=30

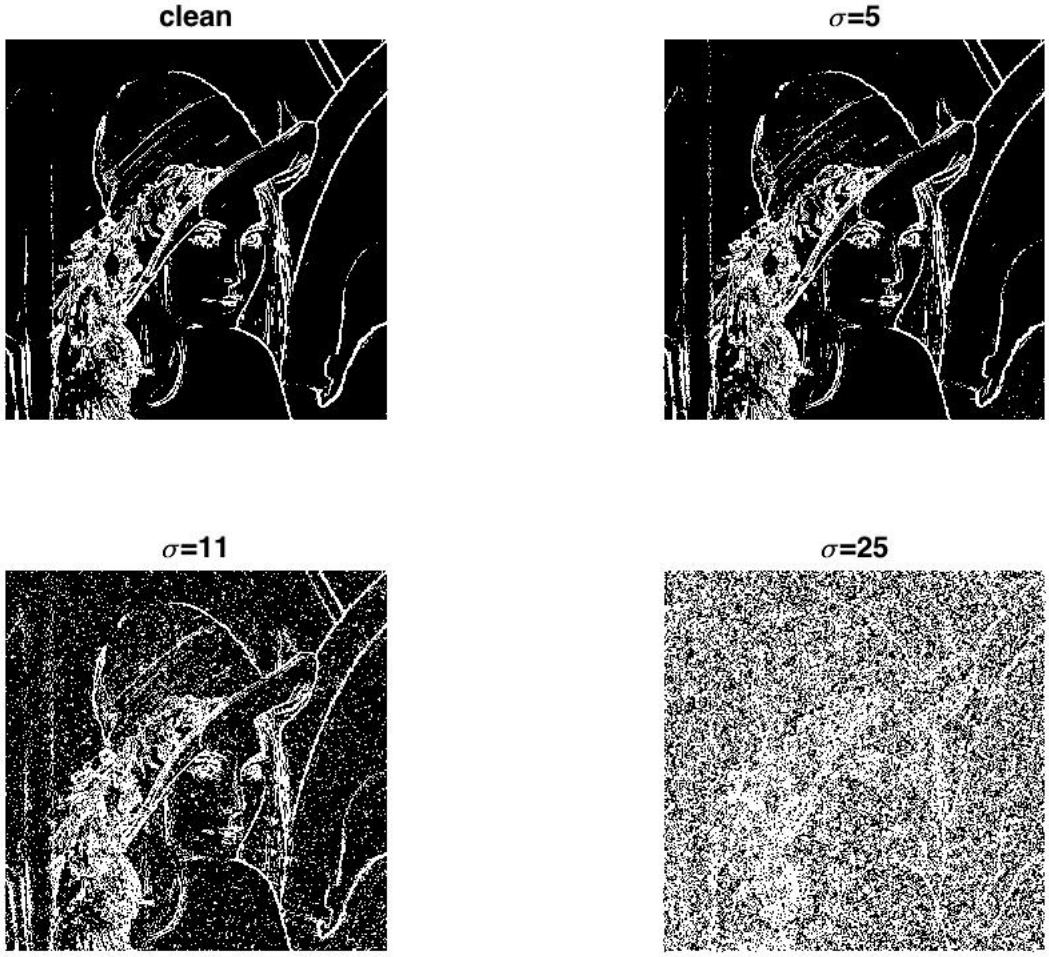

Figure 19: lena.png applying the Compass Operator with a threshold T=30

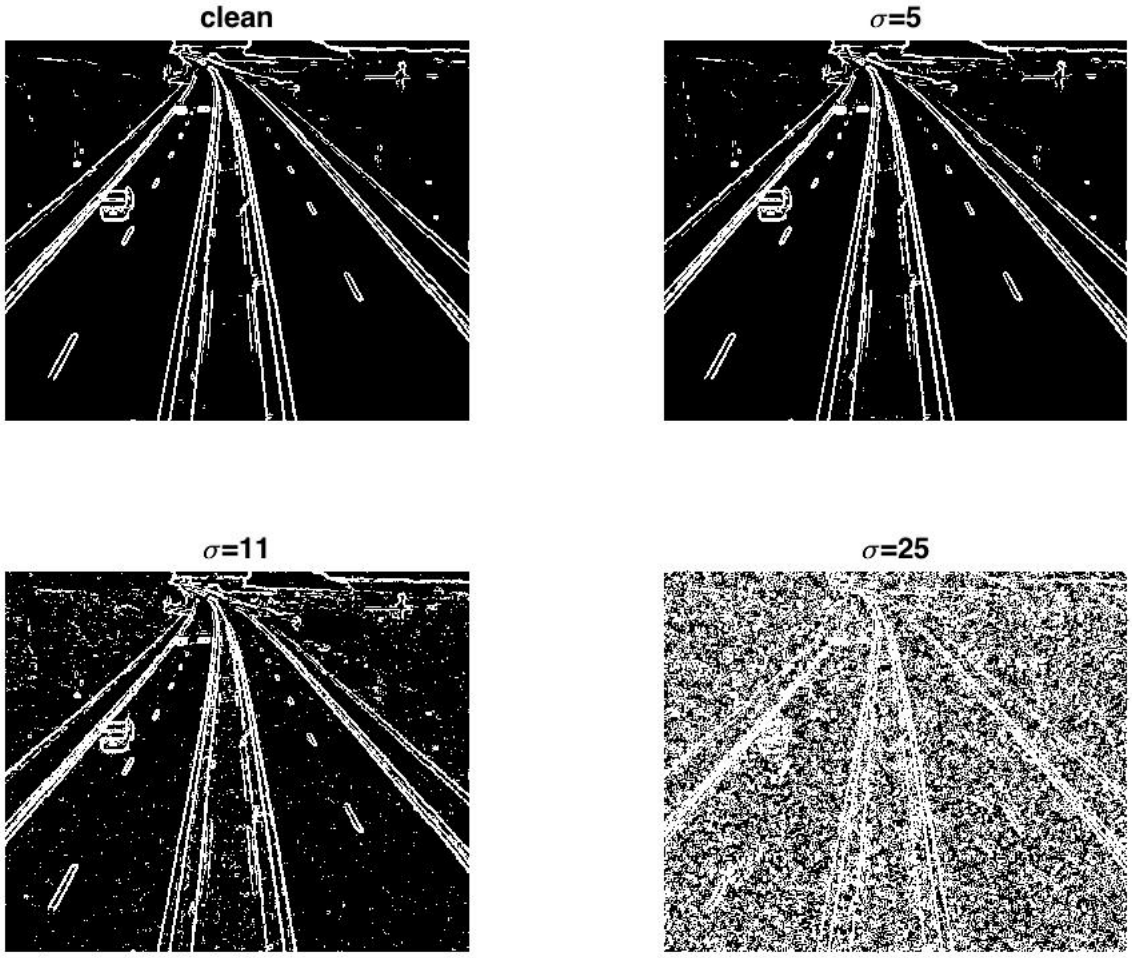

Figure 20: road.png applying the Compass Operator with a threshold T=30

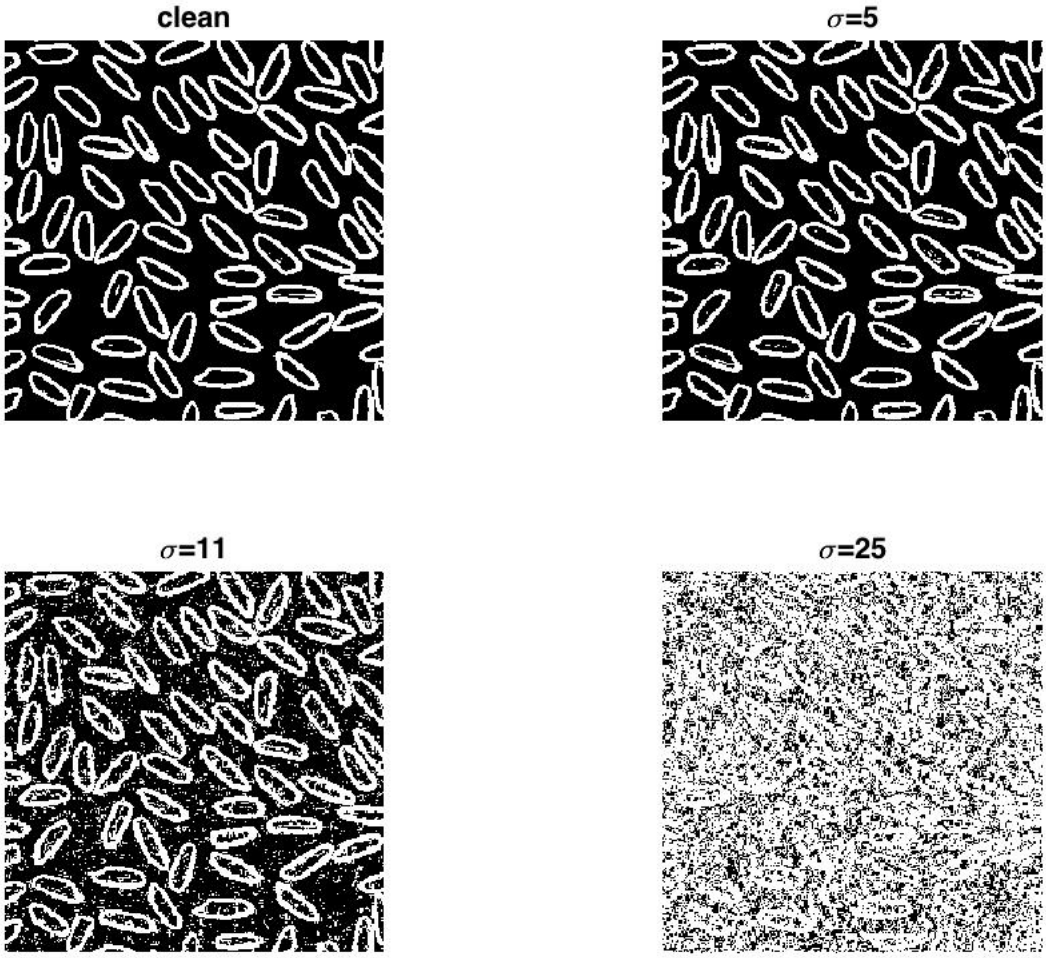

Figure 21: road.png applying the Compass Operator with a threshold T=30

With T=70

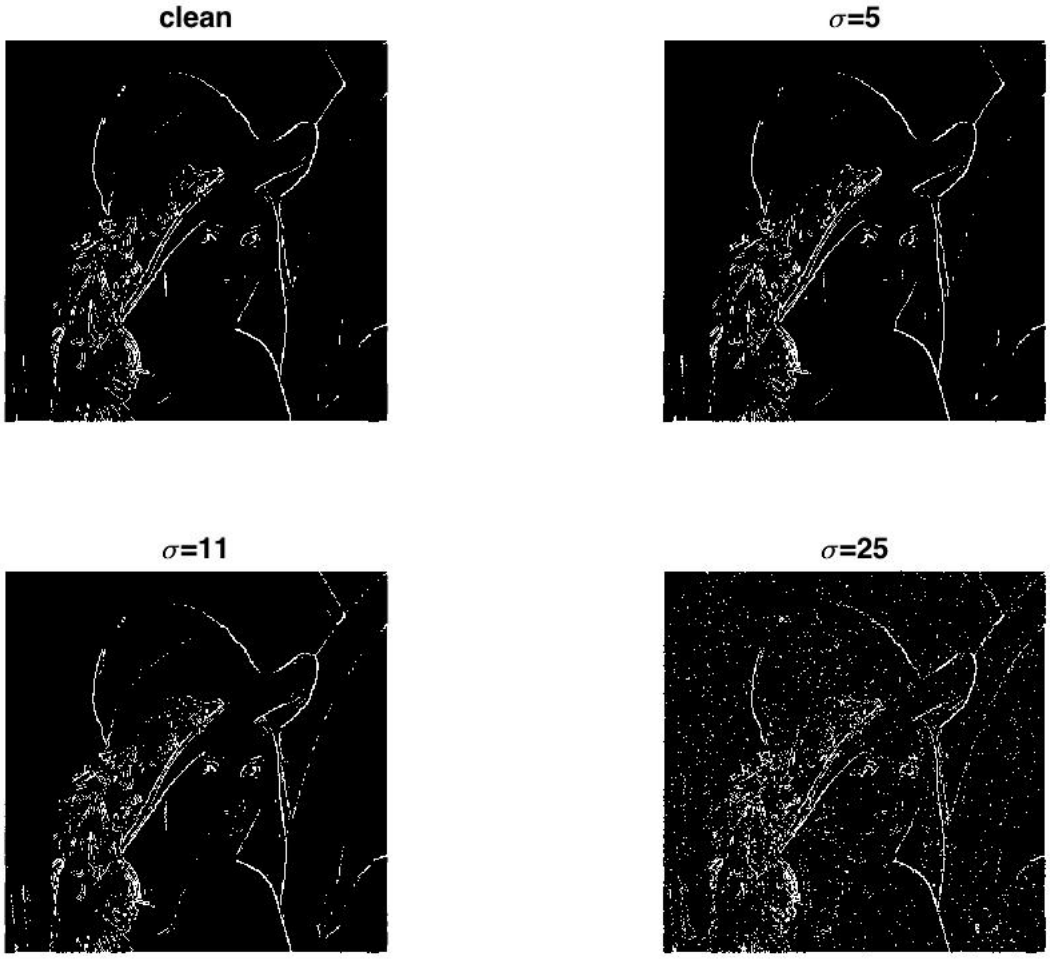

Figure 22: lena.png applying the Compass Operator with a threshold T=70

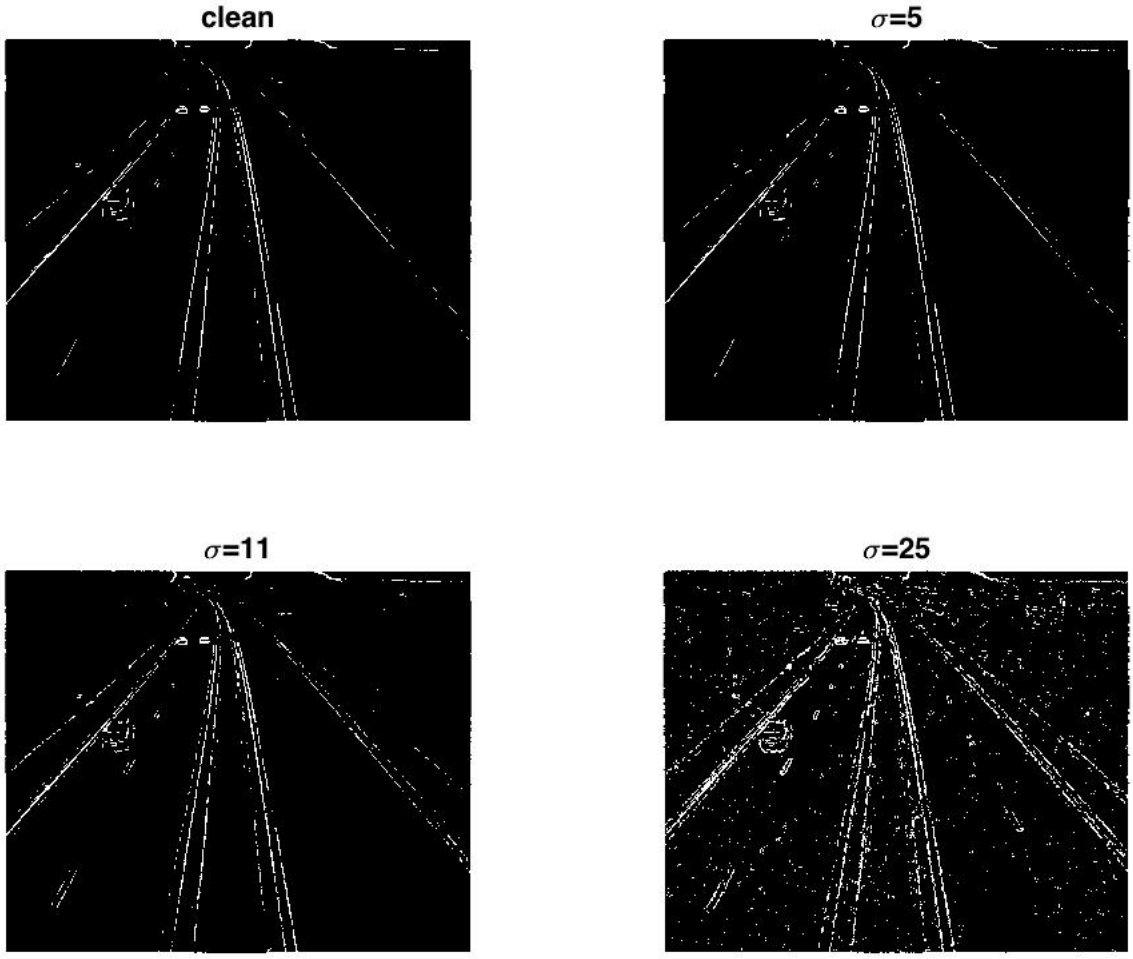

Figure 23: road.png applying the Compass Operator with a threshold T=70

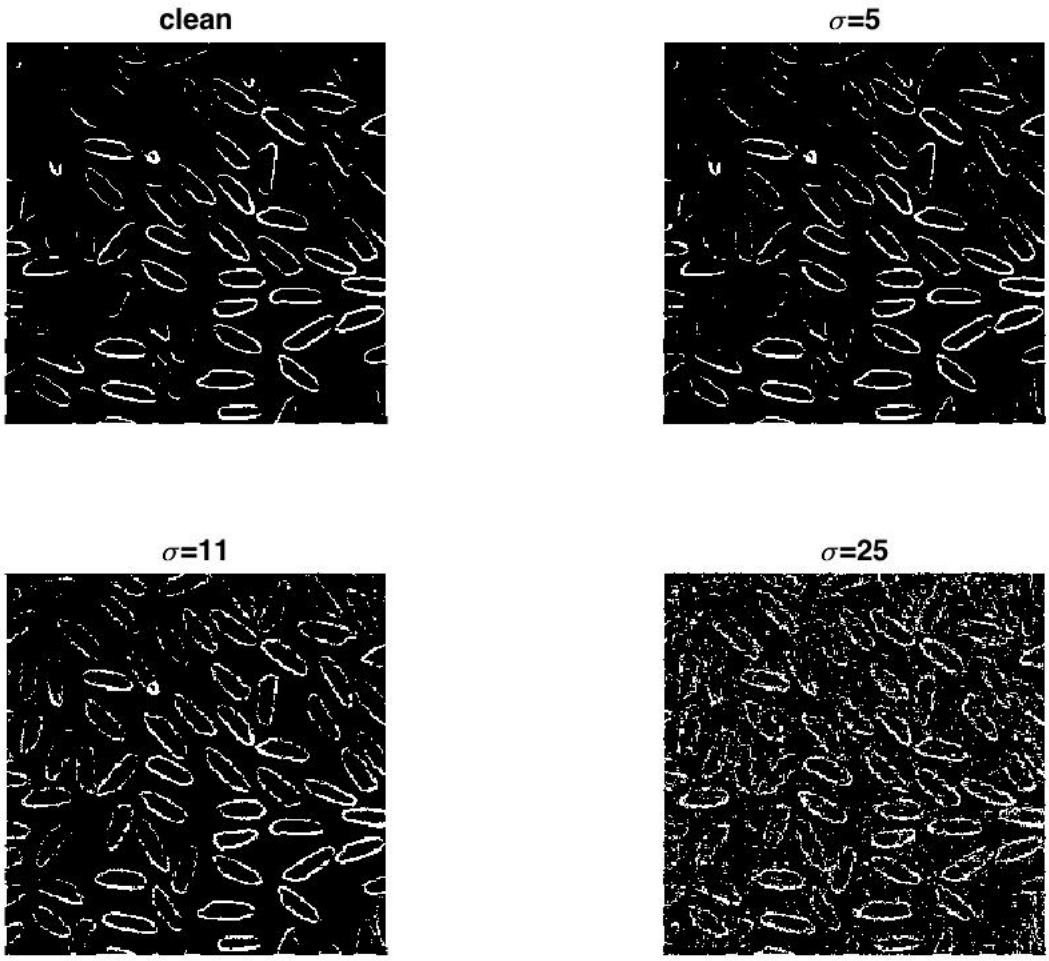

Figure 24: road.png applying the Compass Operator with a threshold T=70

With T=100

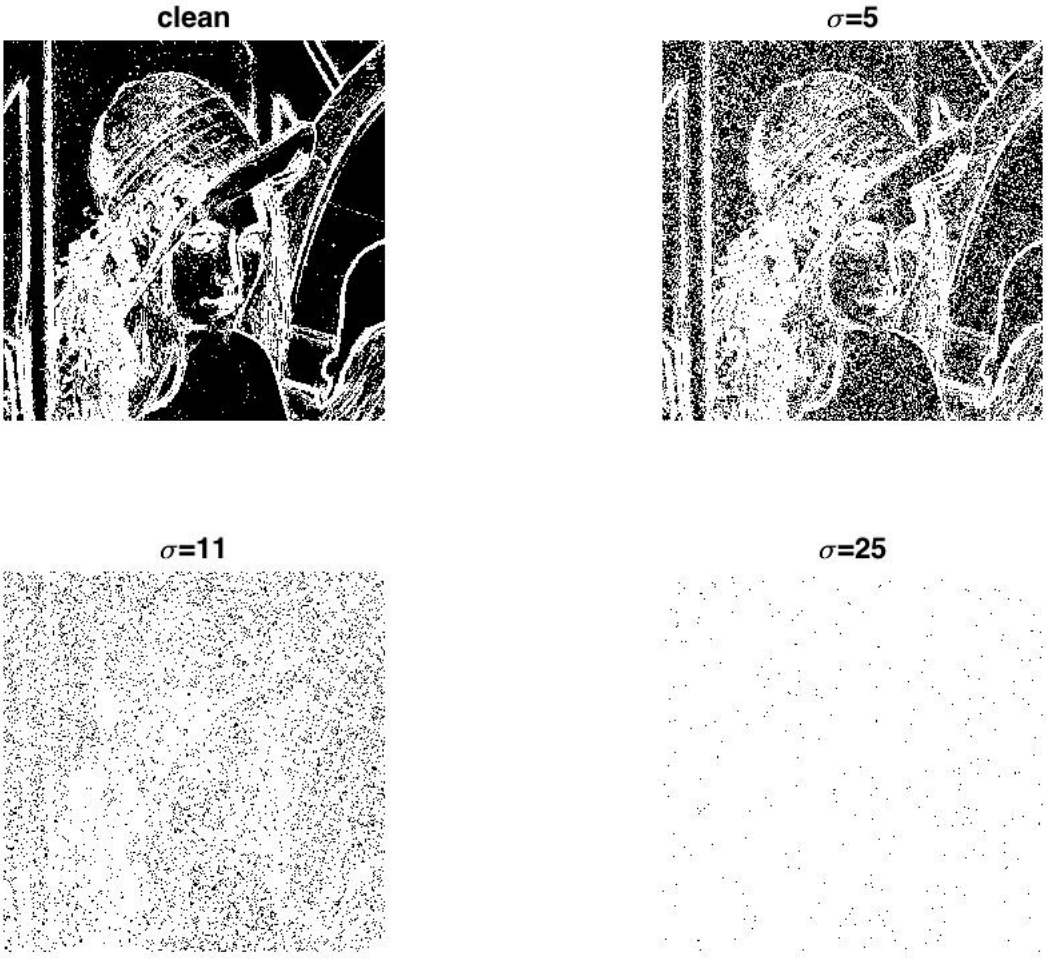

Figure 25: lena.png applying the Compass Operator with a threshold T=100

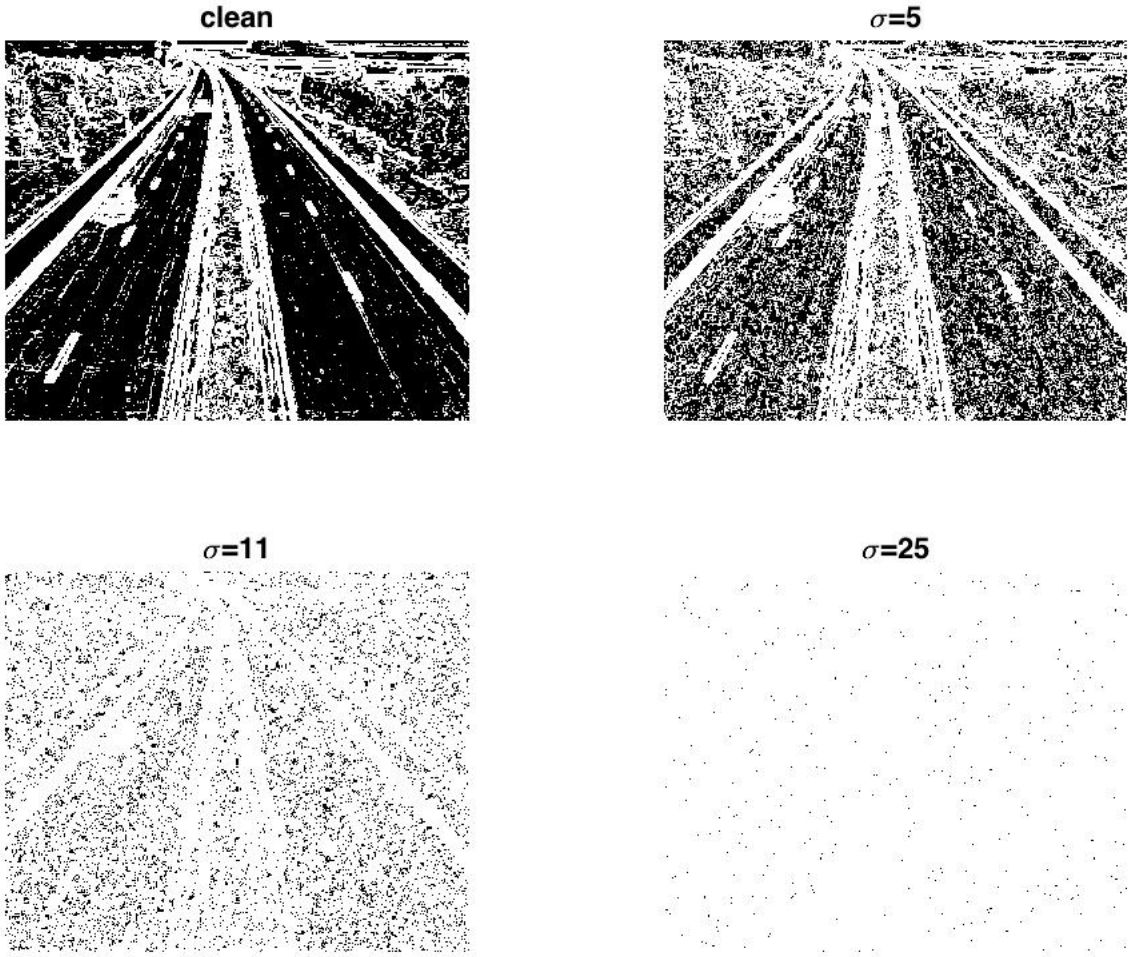

Figure 26: road.png applying the Compass Operator with a threshold T=100

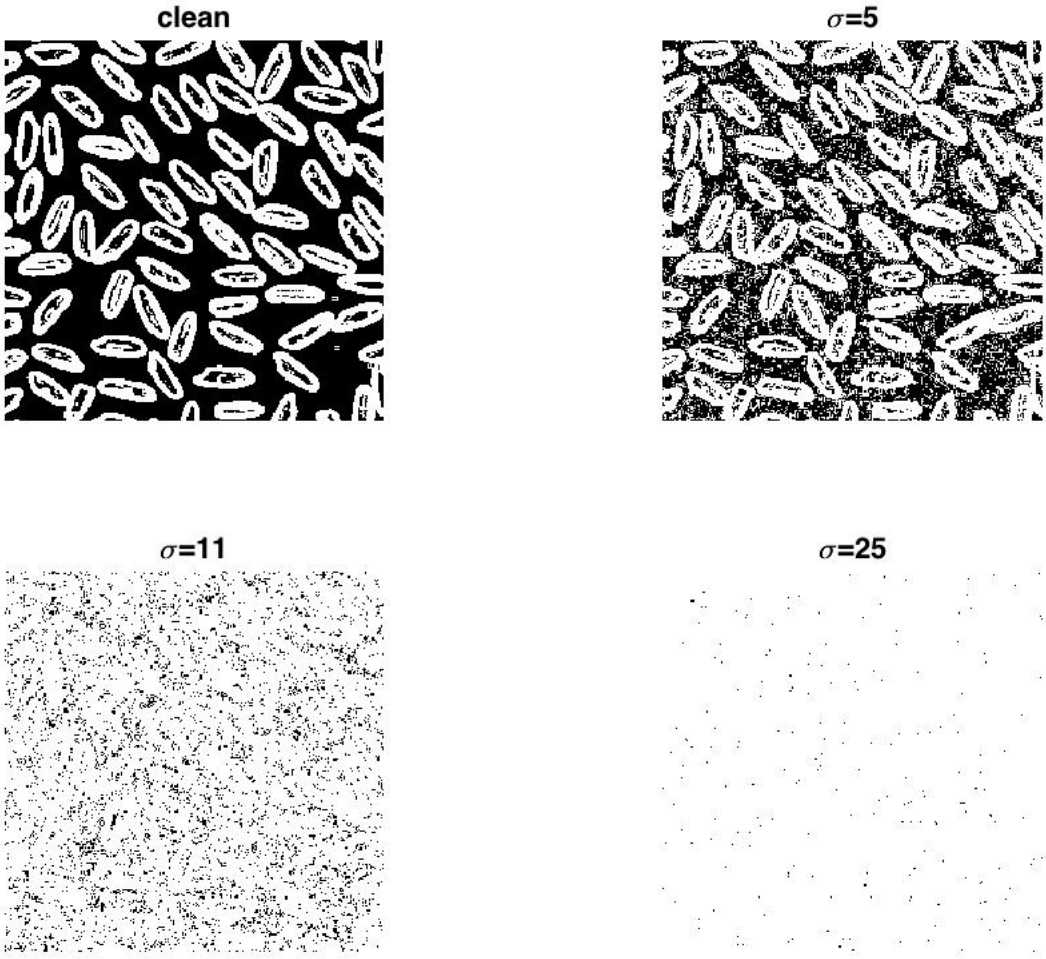

Figure 27: road.png applying the Compass Operator with a threshold T=100

Laplace operator

As *Laplacian* acts like a high-pass filter, and it is quite sensitive to noise, *Gaussian smooth* is used. The combination of *Gaussian* and *Laplacian* makes the *LoG* method, which can be used to detect edges and contours by zero crossing. A function named *laplace_operator* was created in our project, with three input parameter (the input image x, σ, and a threshold T).

Figure 28 to Figure 45 are the result with different σ and threshold T. Compared to the previous two methods, *Laplace operator* seems to offer precise edge and contour information, as we can see thinner solid lines. What's more, a smaller σ in a laplace operator seem to depict more contour information.

With $\sigma = 2$

With T=1

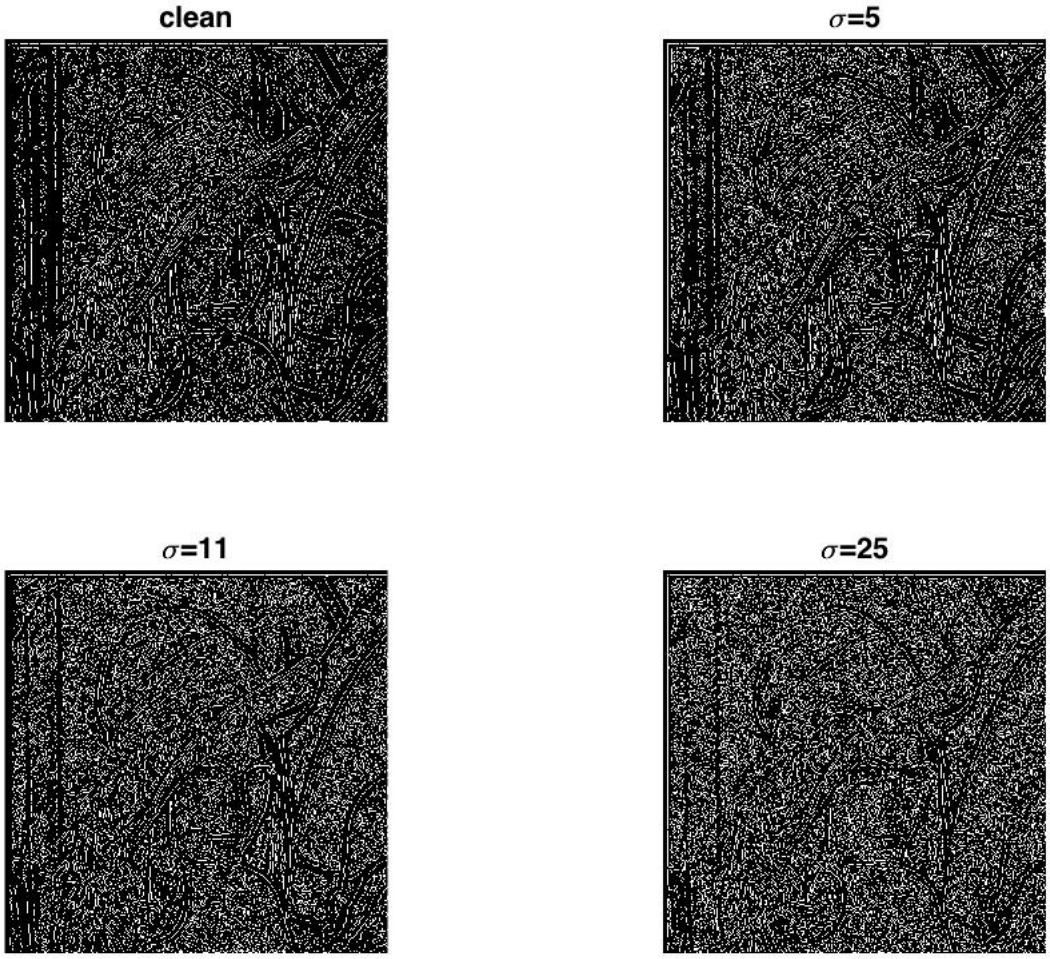

Figure 28: lena.png applying the Laplace operator with $\sigma = 2$

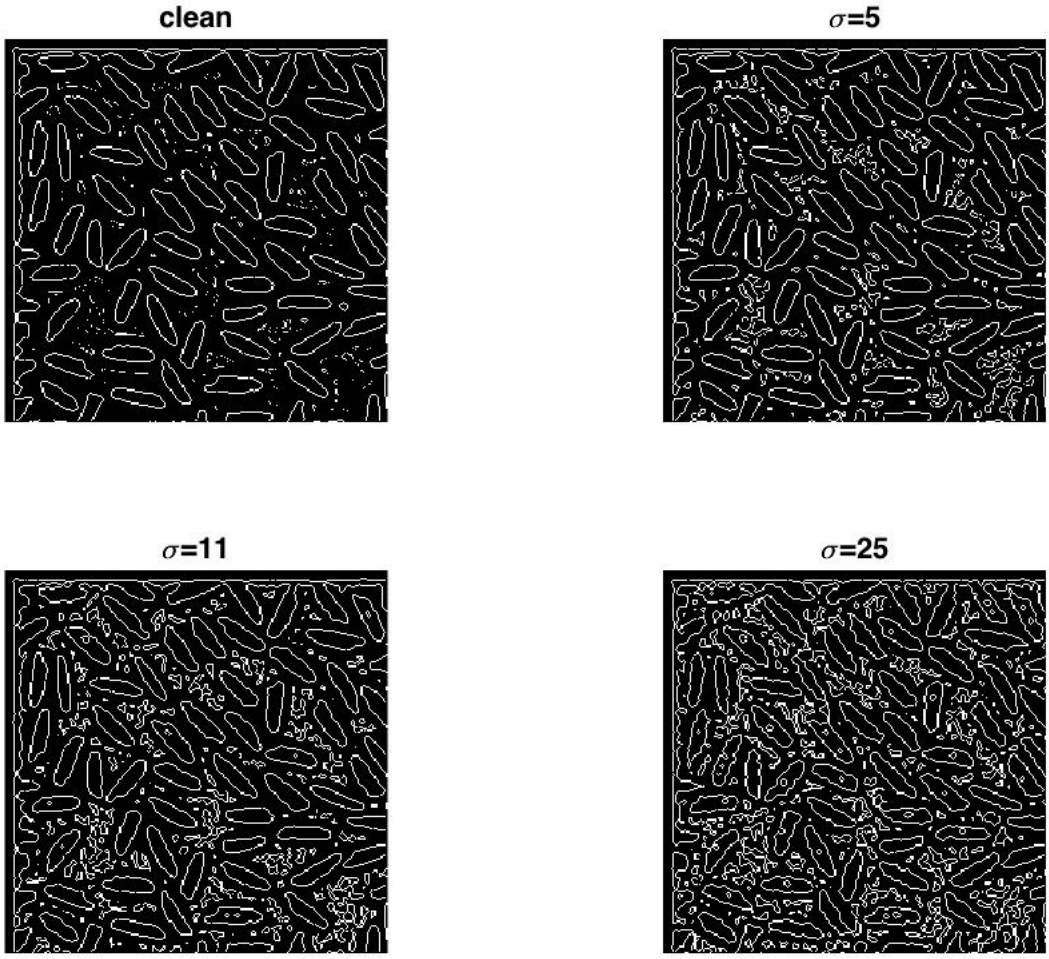

Figure 29: rice.png applying the Laplace operator with $\sigma = 2$

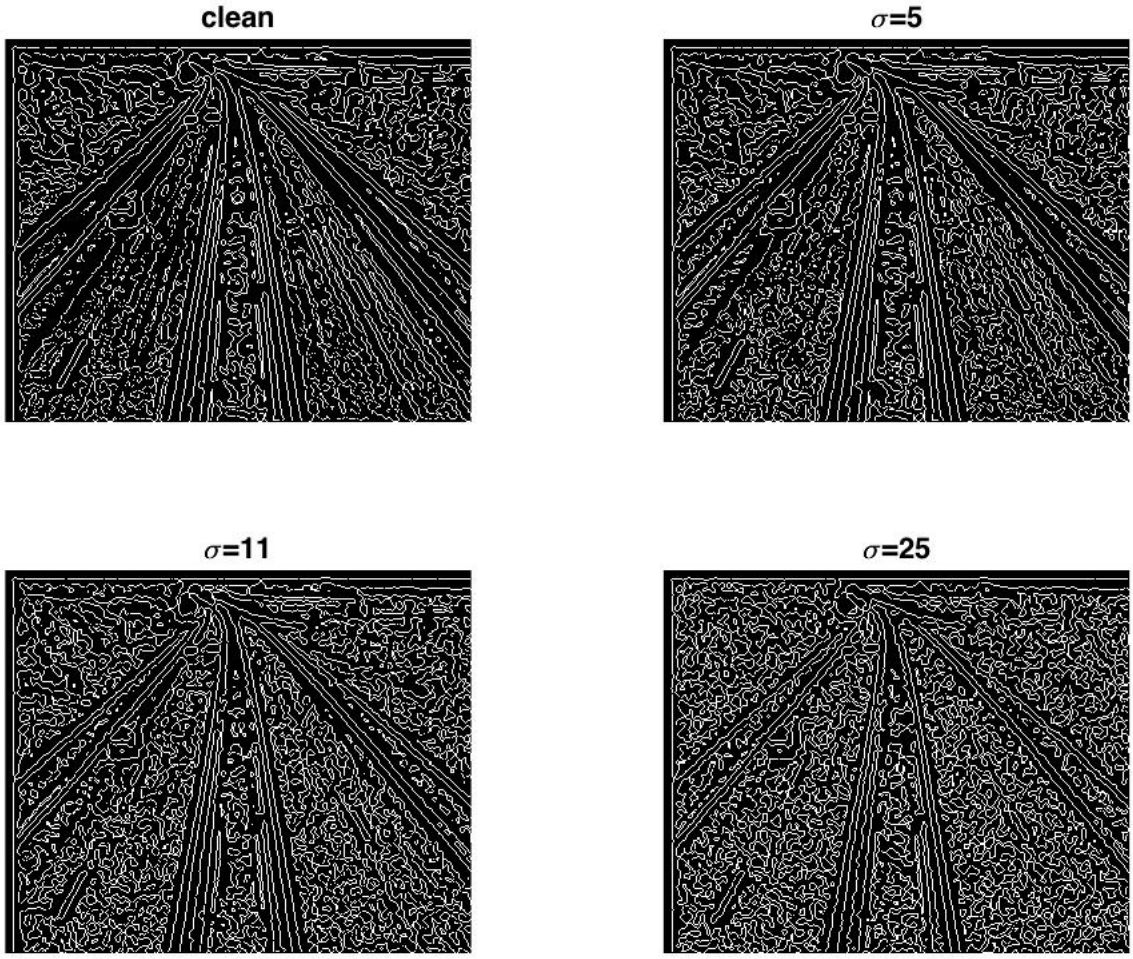

Figure 30: road.png applying the Laplace operator with $\sigma = 2$

With T=15

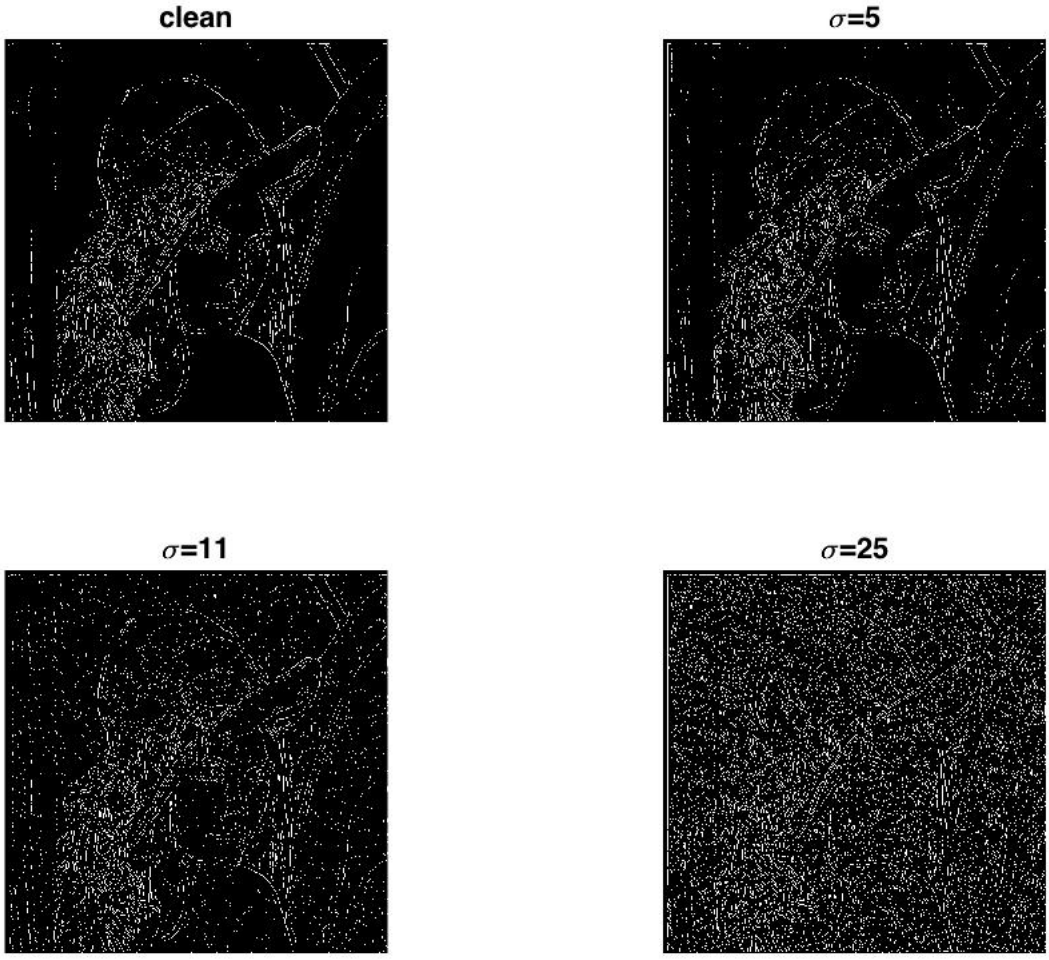

Figure 31: lena.png applying the Laplace operator with $\sigma = 2$

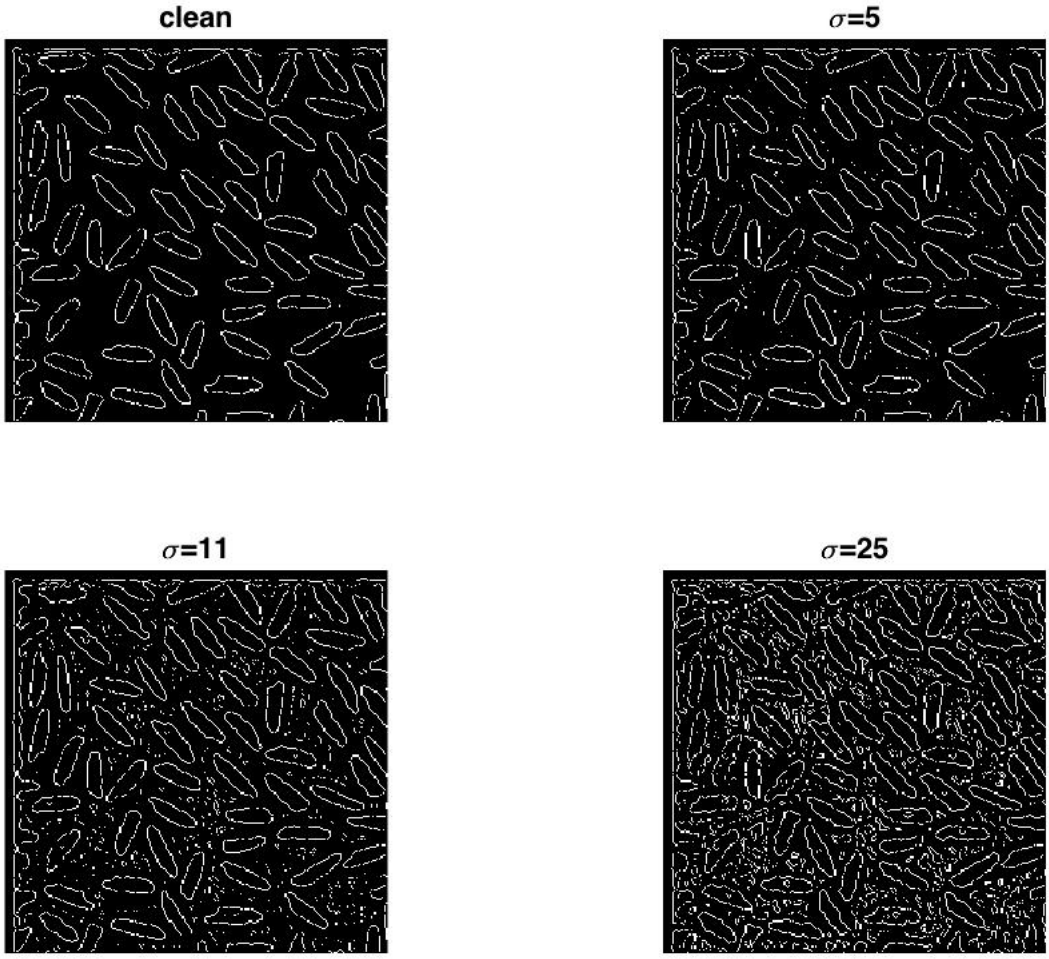

Figure 32: rice.png applying the Laplace operator with $\sigma = 2$

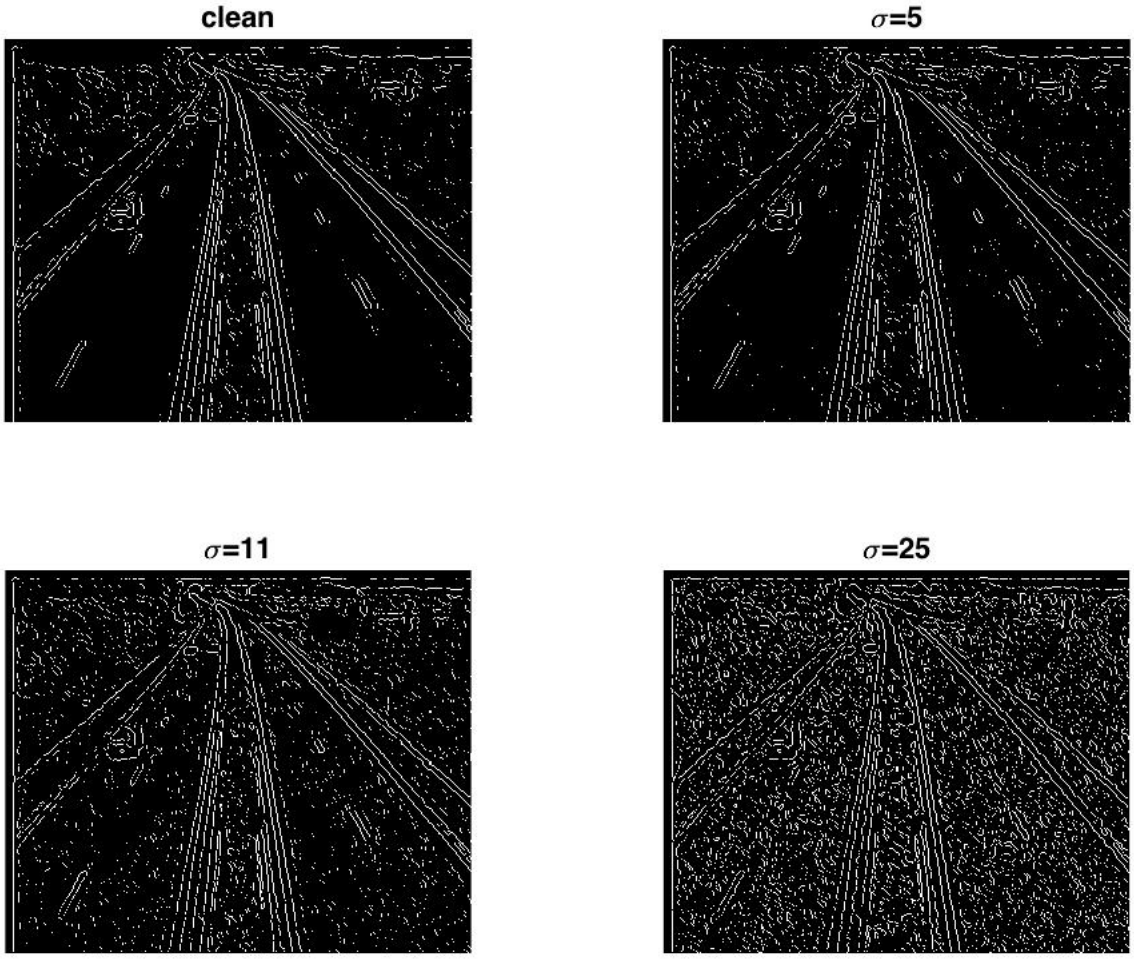

Figure 33: road.png applying the Laplace operator with $\sigma = 2$

With $\sigma = 2.5$

With T=1

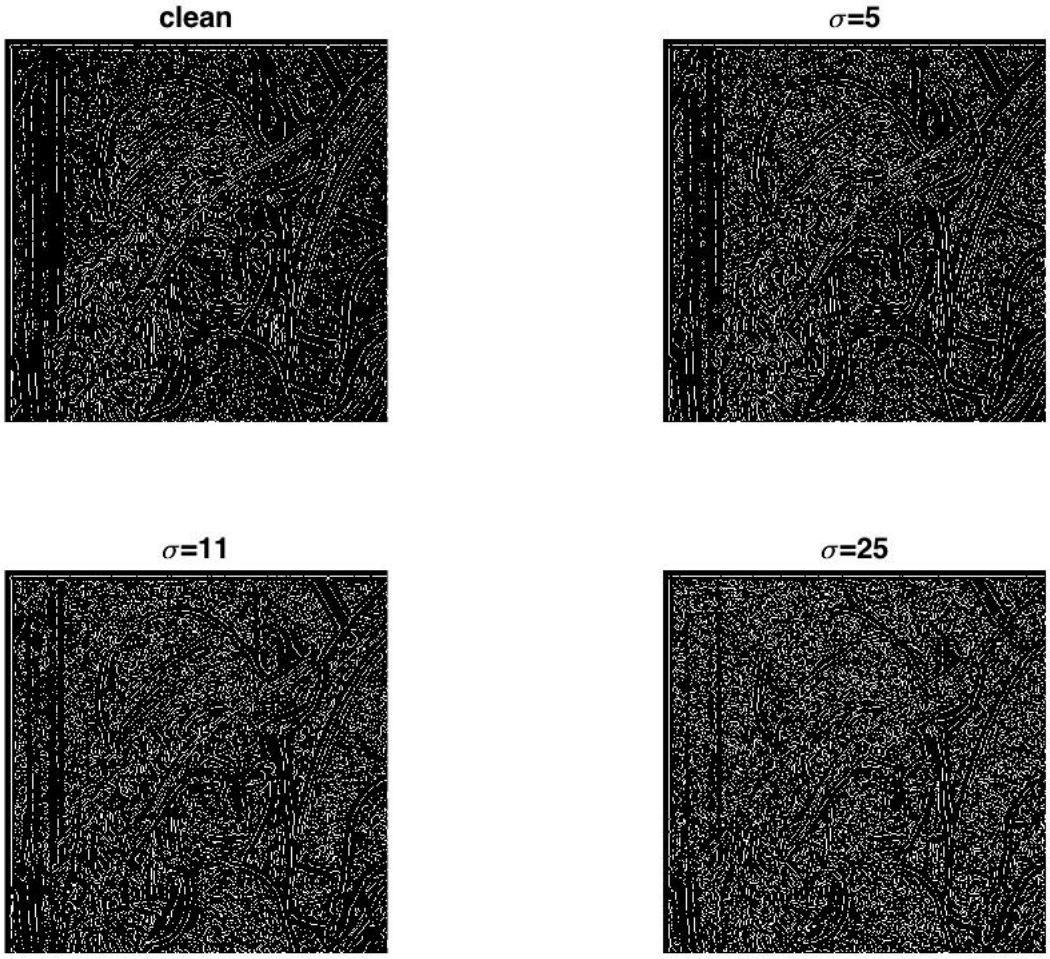

Figure 34: lena.png applying the Laplace operator with $\sigma = 2.5$

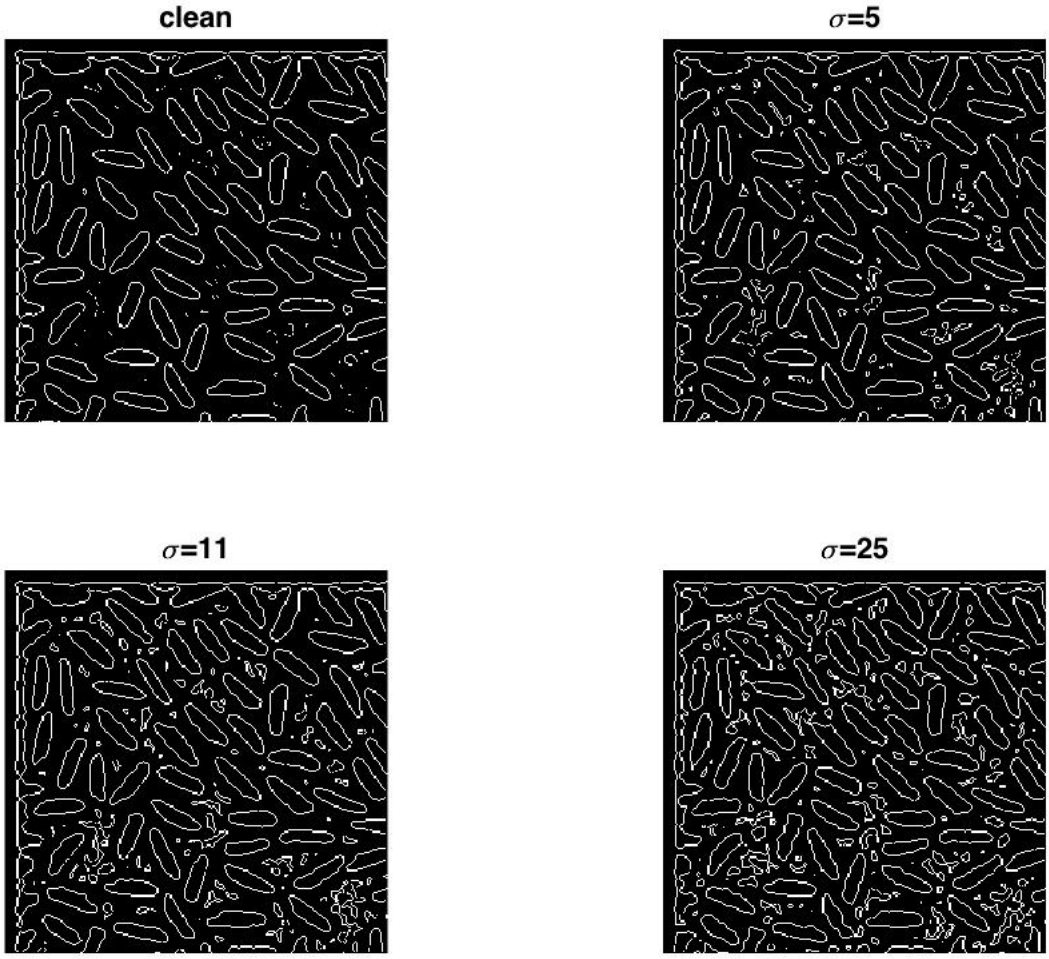

Figure 35: rice.png applying the Laplace operator with $\sigma = 2.5$

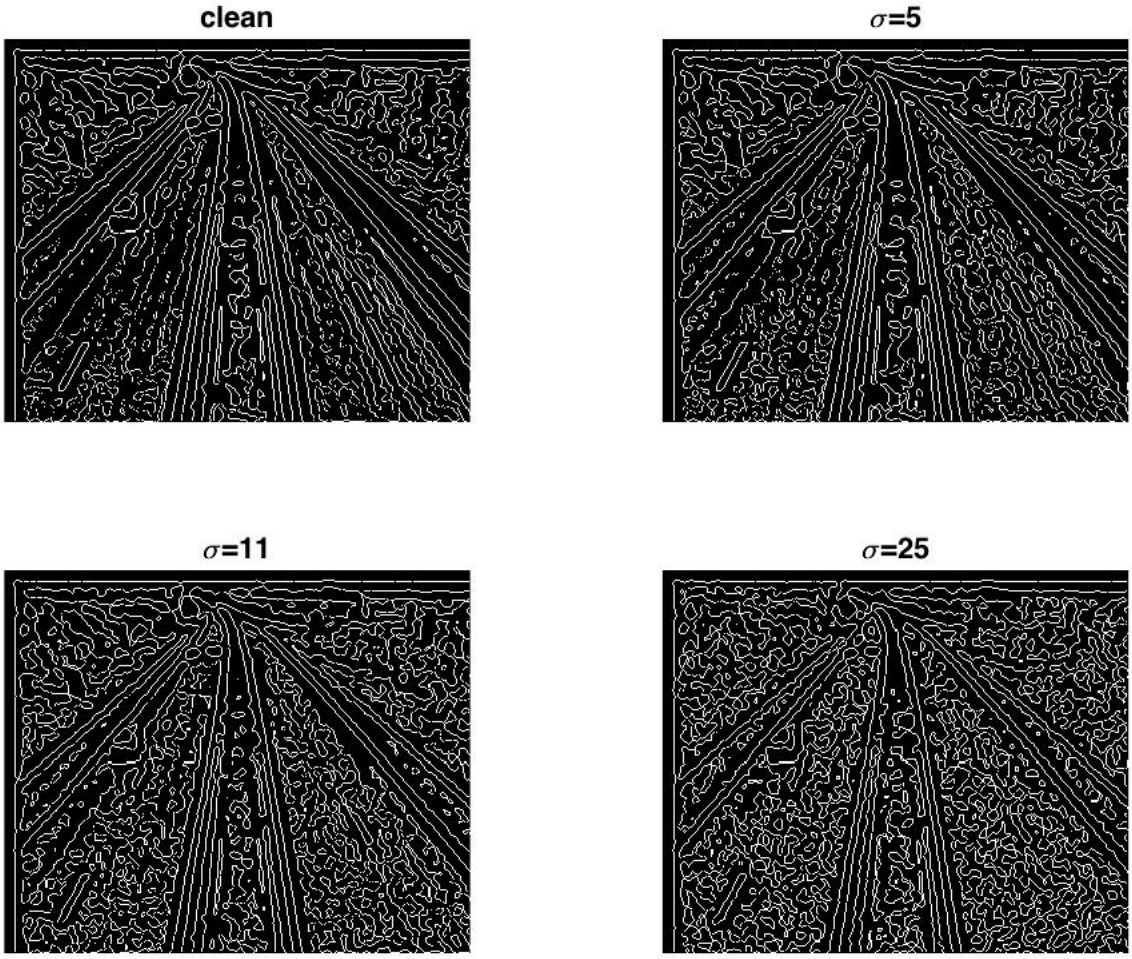

Figure 36: road.png applying the Laplace operator with $\sigma = 2.5$

With T=15

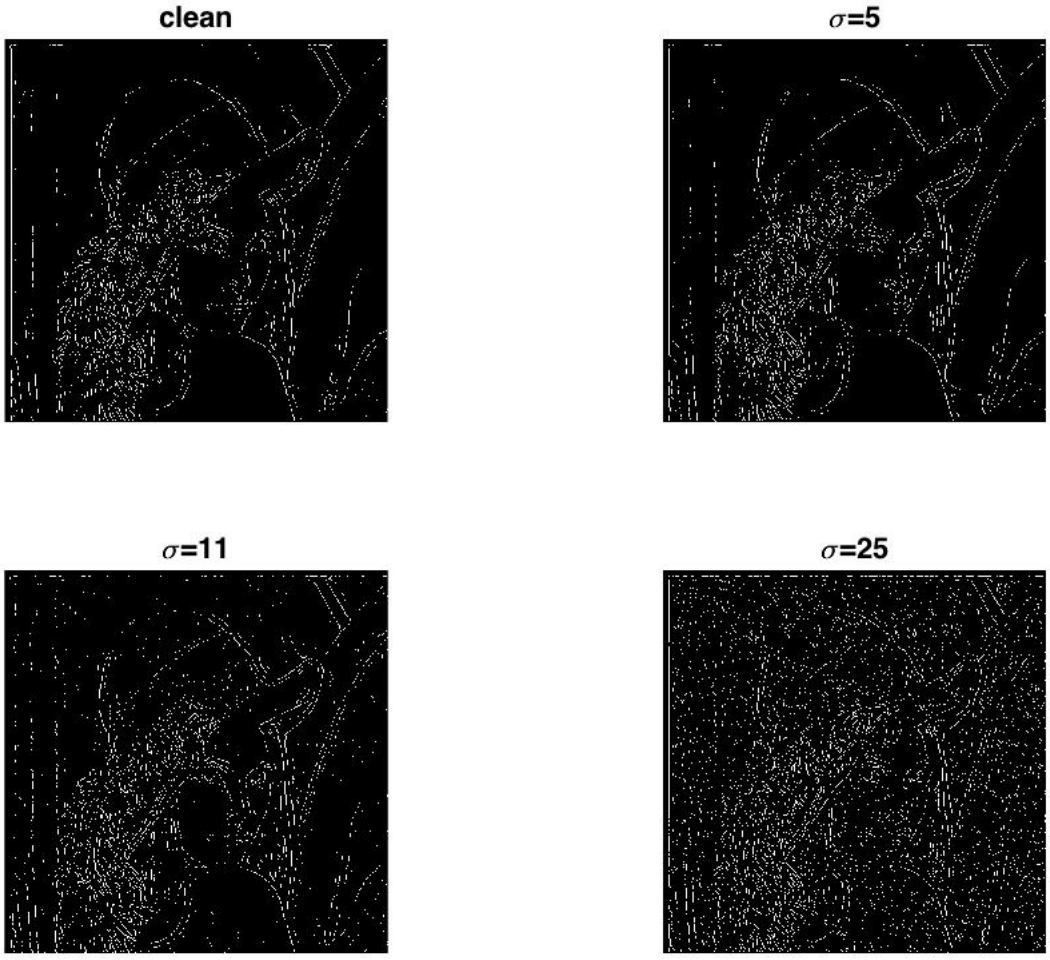

Figure 37: lena.png applying the Laplace operator with $\sigma = 2.5$

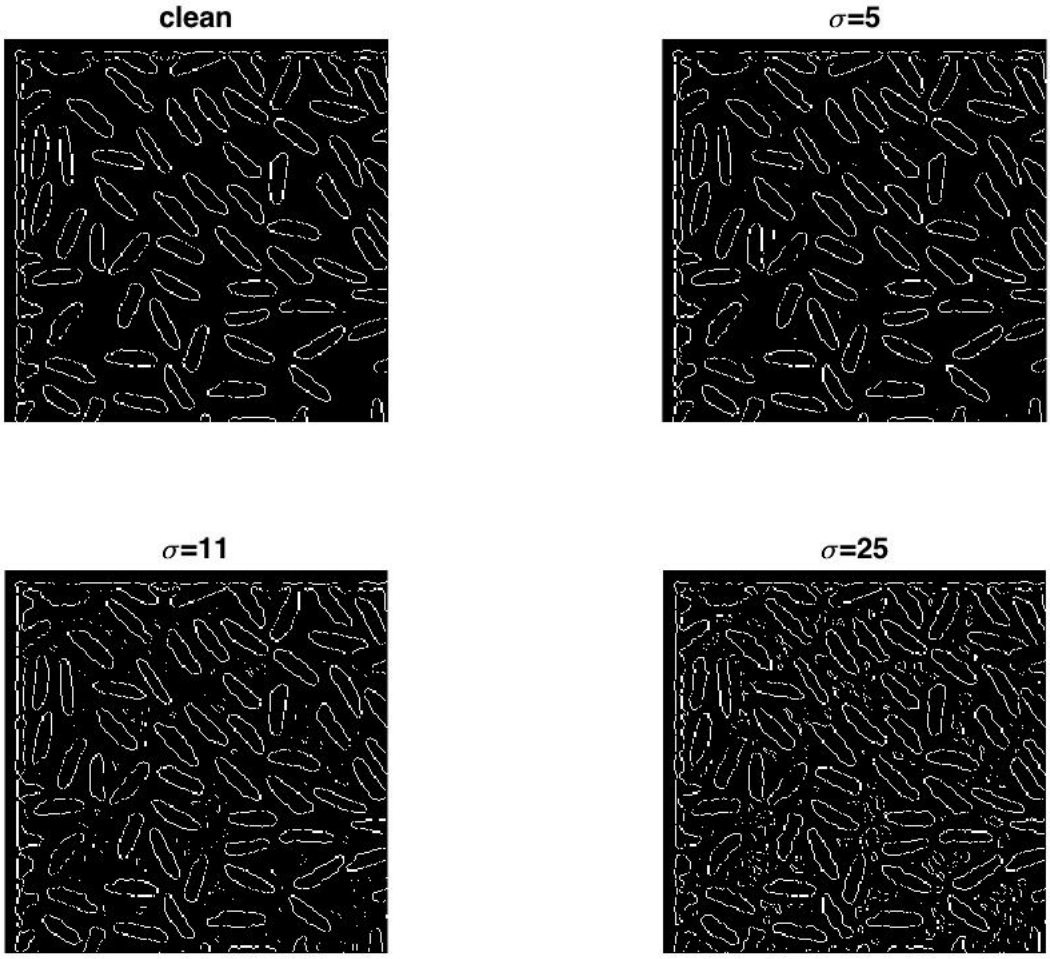

Figure 38: rice.png applying the Laplace operator with $\sigma = 2.5$

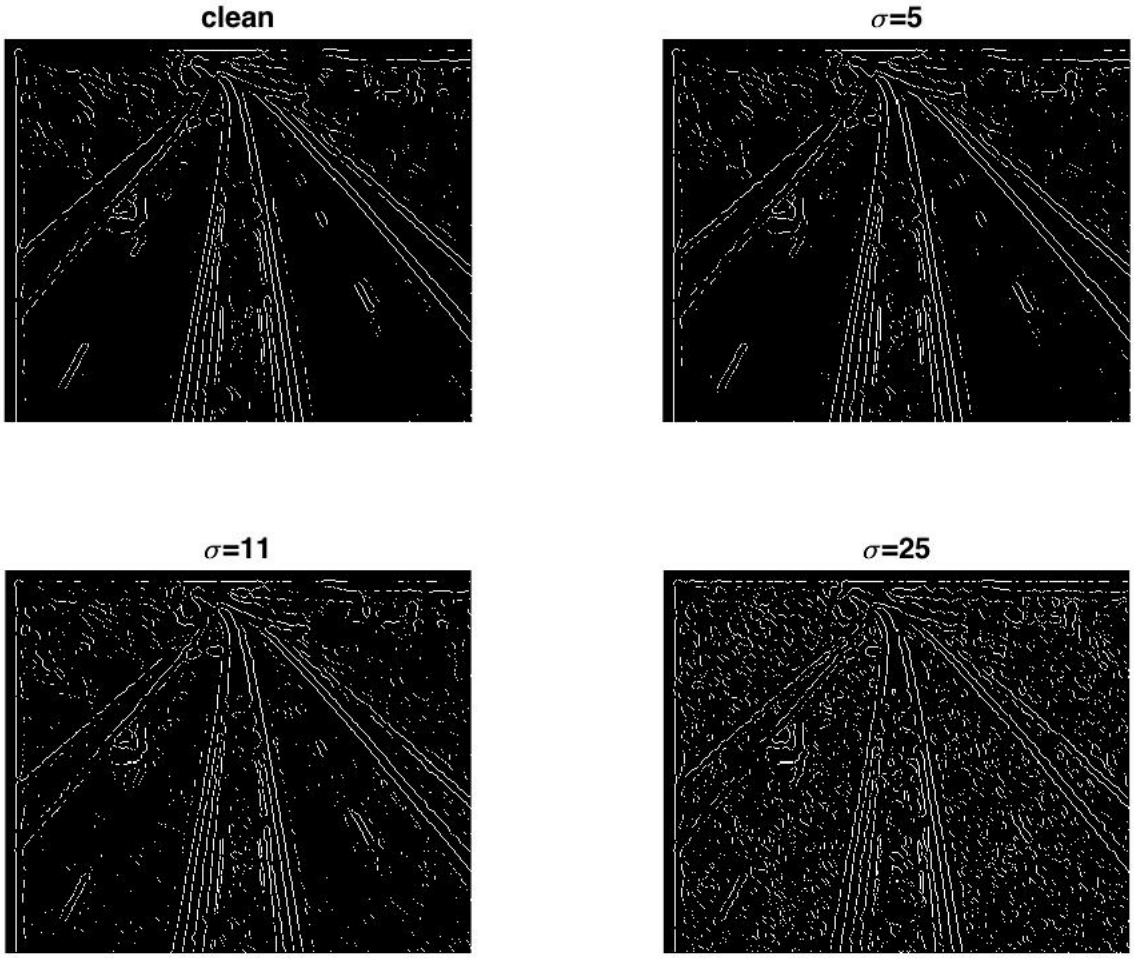

Figure 39: road.png applying the Laplace operator with $\sigma = 2.5$

With $\sigma = 3$

With T=1

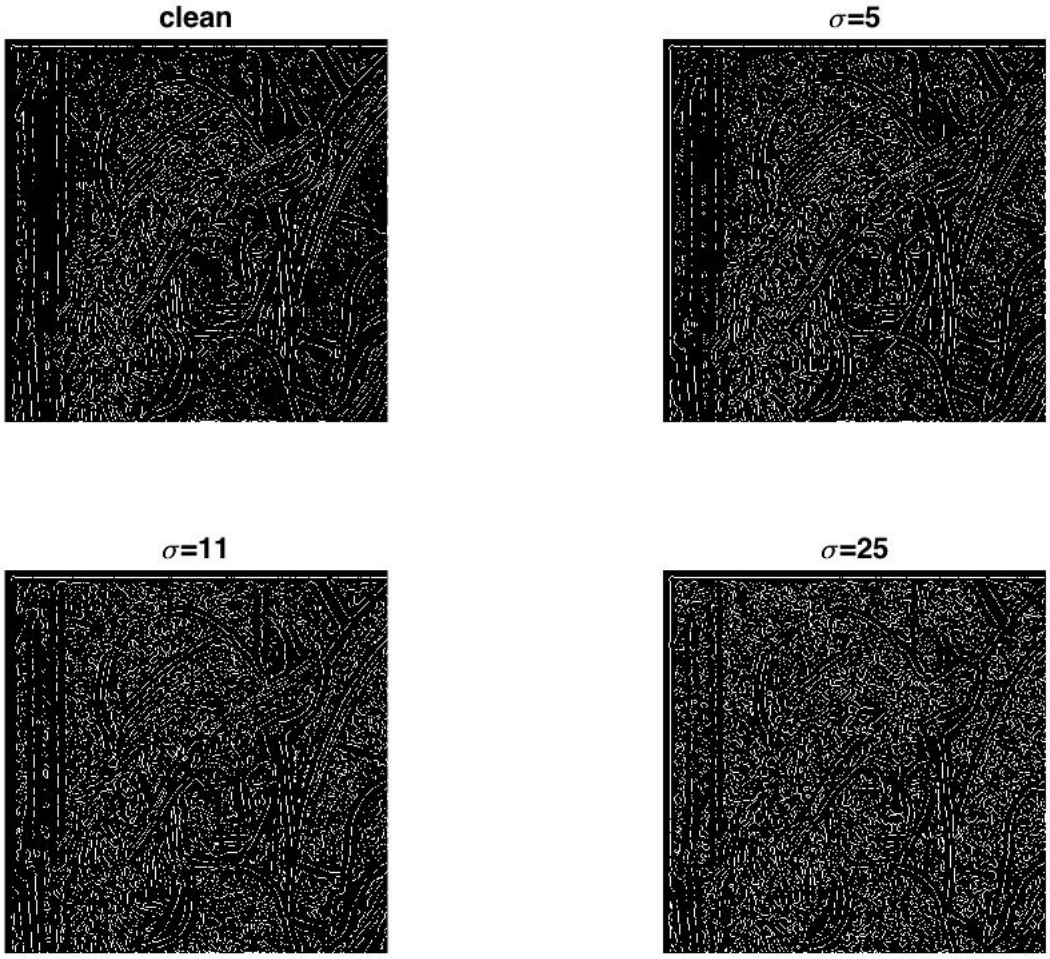

Figure 40: lena.png applying the Laplace operator with $\sigma = 3$

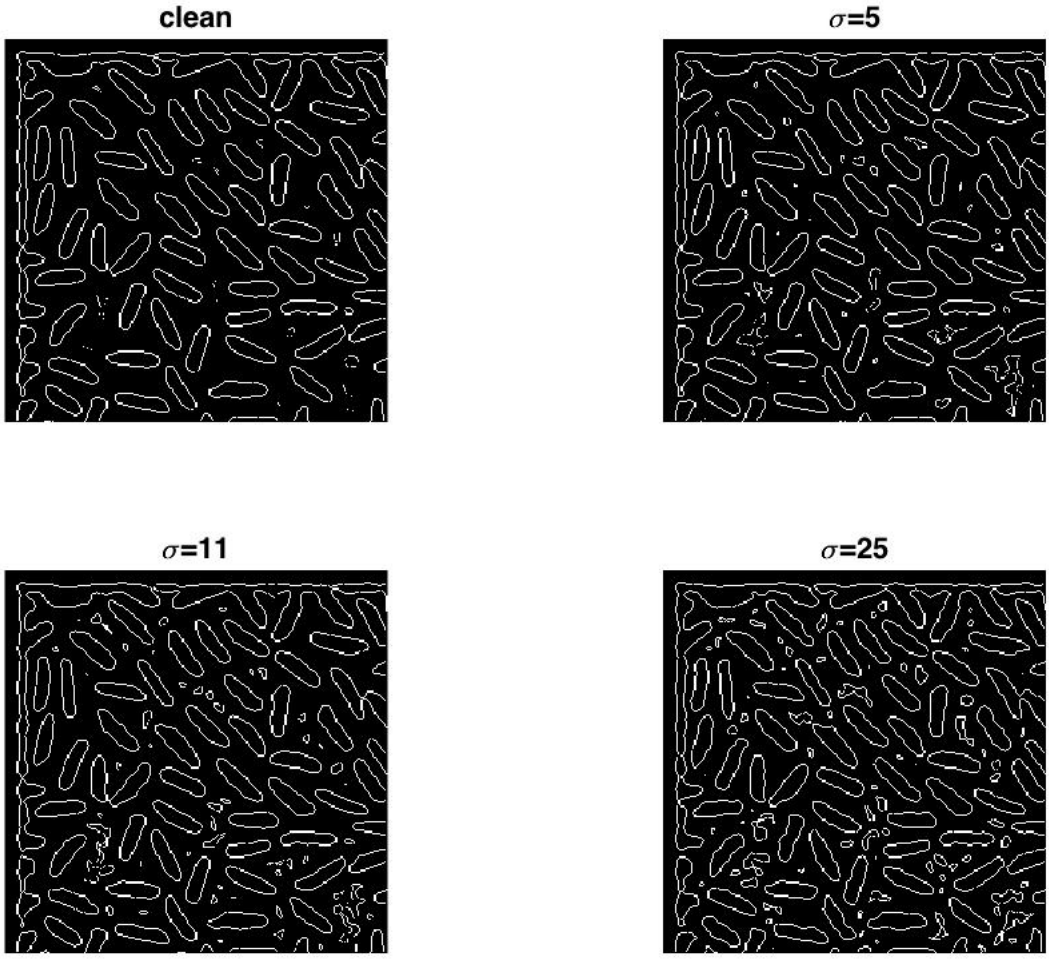

Figure 41: rice.png applying the Laplace operator with $\sigma = 3$

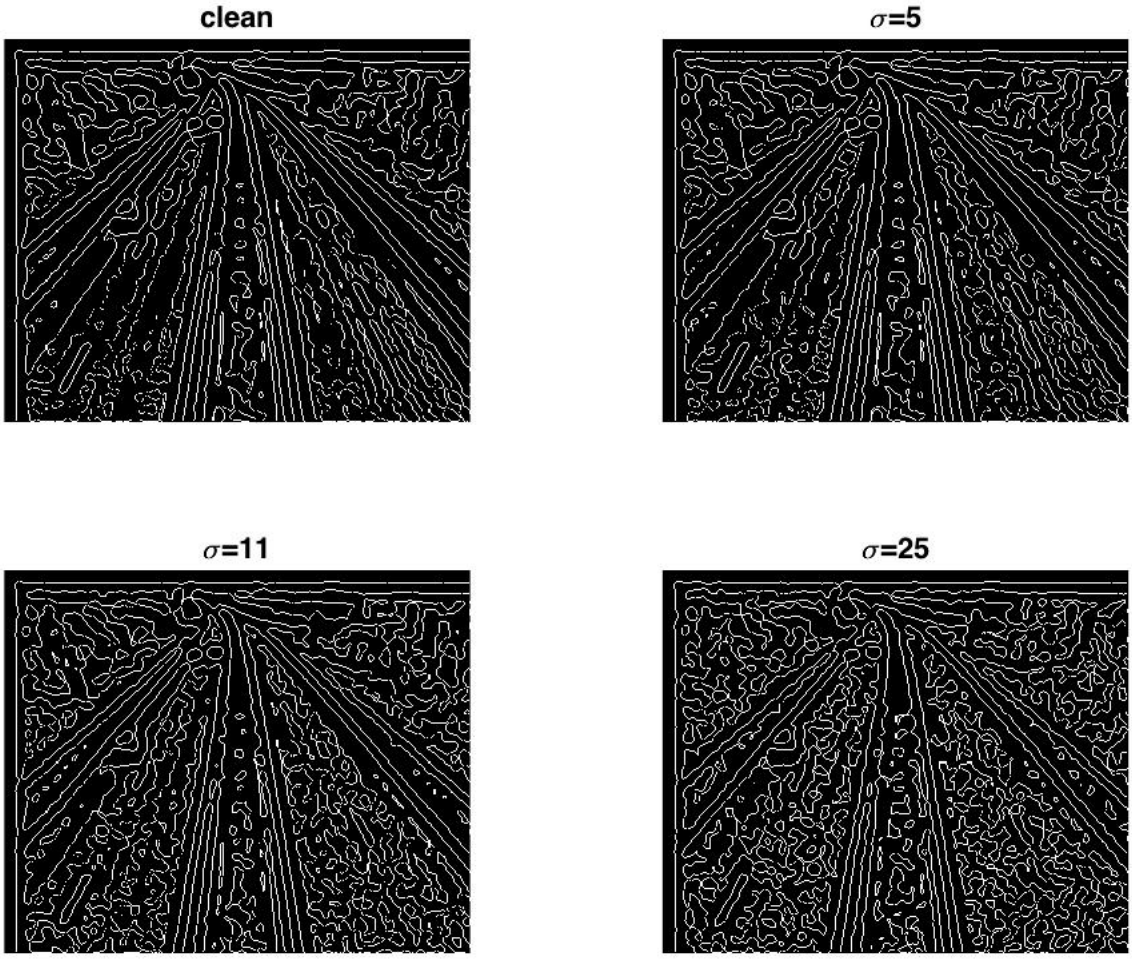

Figure 42: road.png applying the Laplace operator with $\sigma = 3$

With T=15

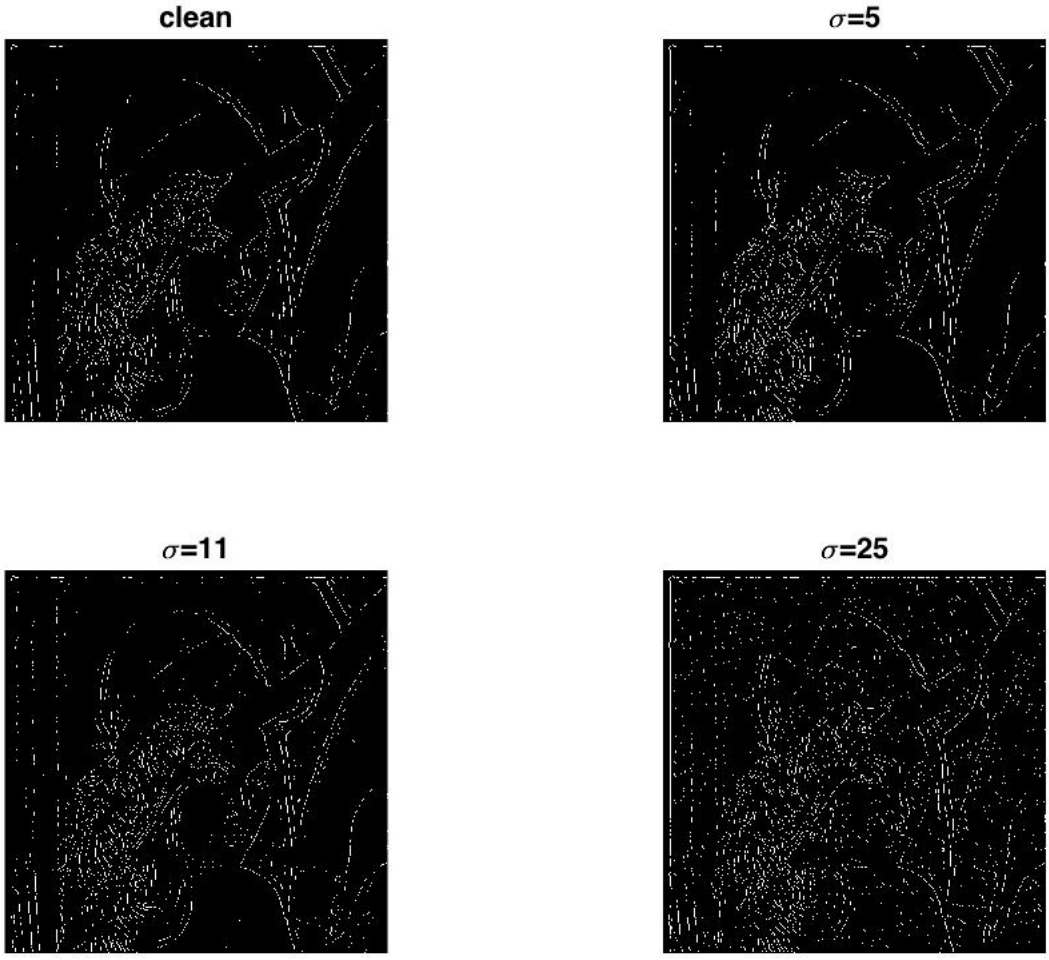

Figure 43: lena.png applying the Laplace operator with $\sigma = 3$

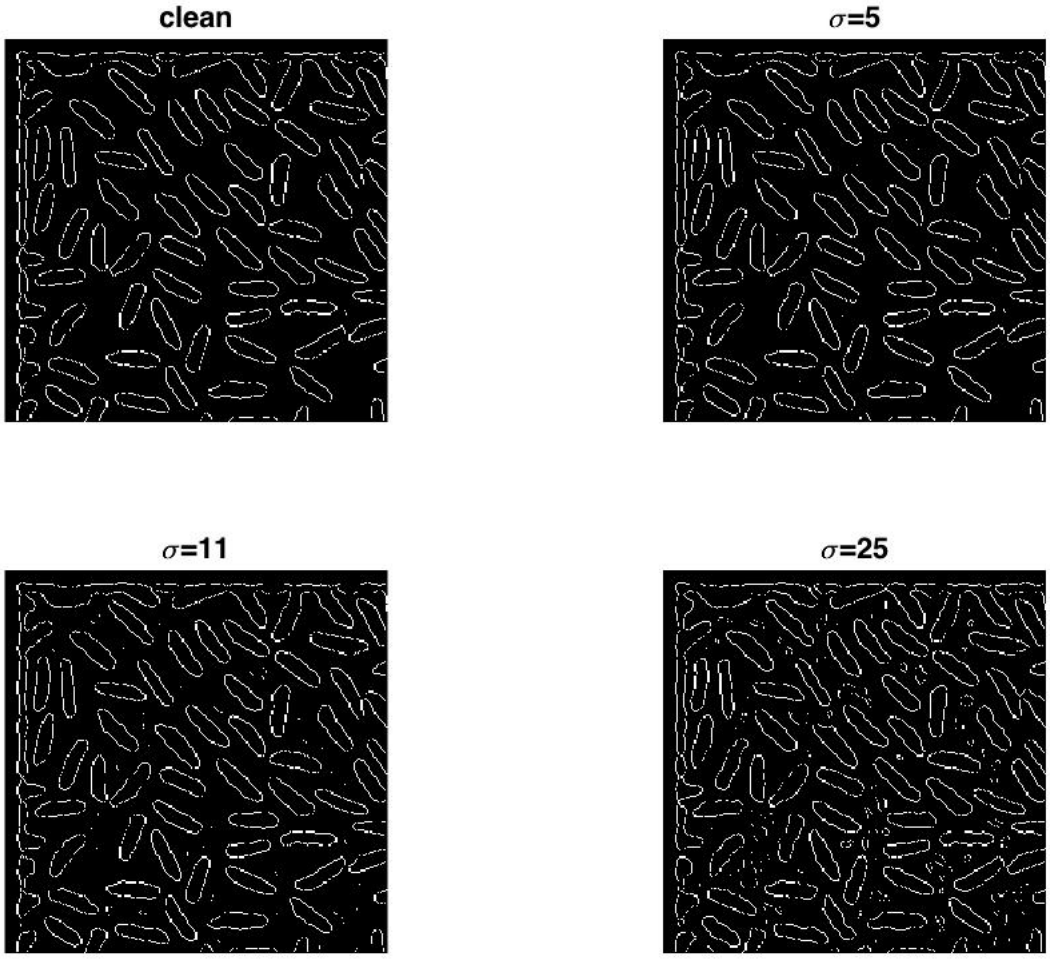

Figure 44: rice.png applying the Laplace operator with $\sigma = 3$

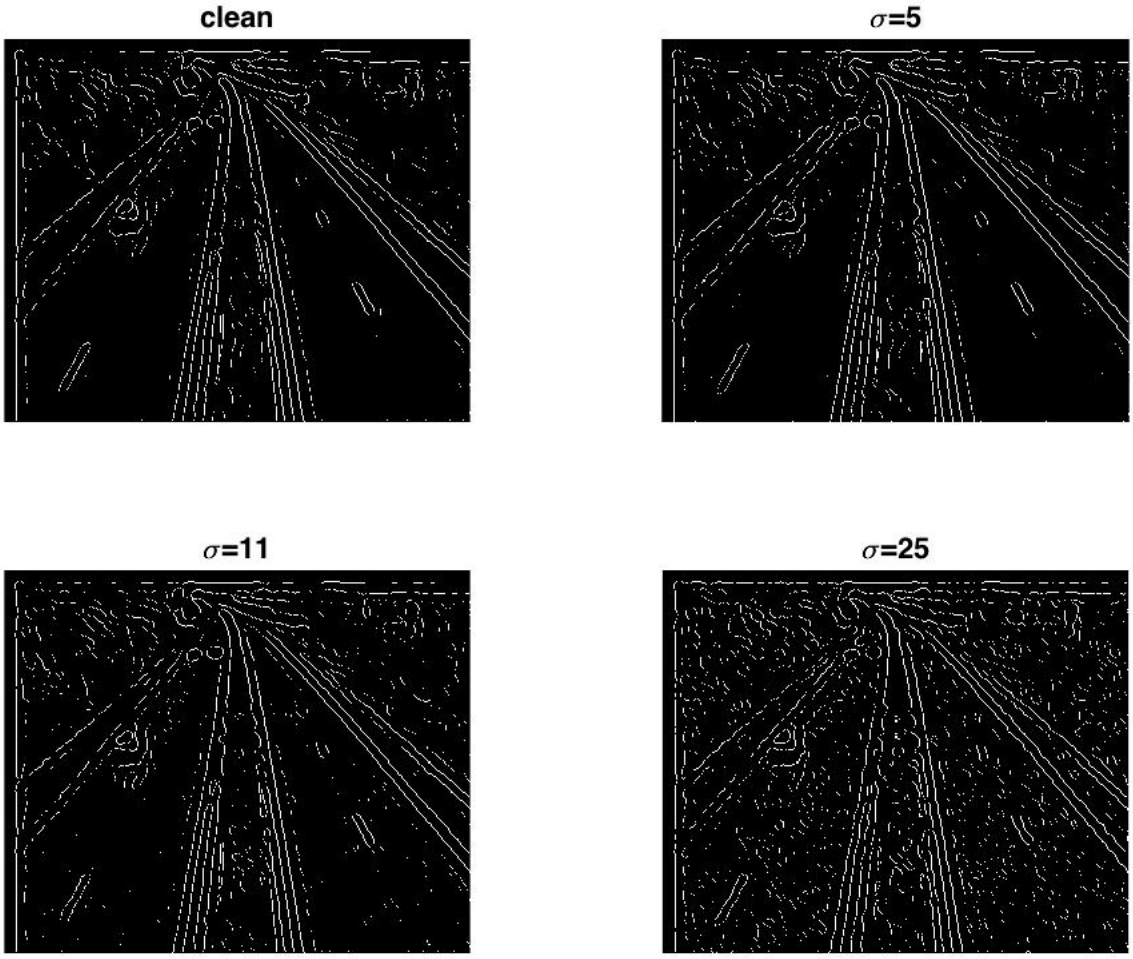

Figure 45: road.png applying the Laplace operator with $\sigma = 3$

Frei-Chen method

This method uses templates not only to detect contours but also other structures likes lines or ripples. We created a function, where those templates are already implemented, that takes as inputs the original image x and the threshold T and return an image with the highlighted contours.

In this function, we do the convolution of x with every template, then re-sized it to its original size, assuming the wanted result of the convolution is at the center of the resulting image.

We compute the matrix magnitude y of the projection of the nine resulting matrices on the two resulting matrices using the two first templates f0 and f1 (representing the edges structures).
We find the maximum elements of y and normalized y.

Then we used the threshold method to highlight the edges in white and the rest in black.
Compared to the Compass operator the Frei Chen method gives the same amount of information when applying the same threshold (for both method, 0 ¡ T ¡ 255) but with finer precision. Therefore the Frei Chen method gives better results.

With T=30

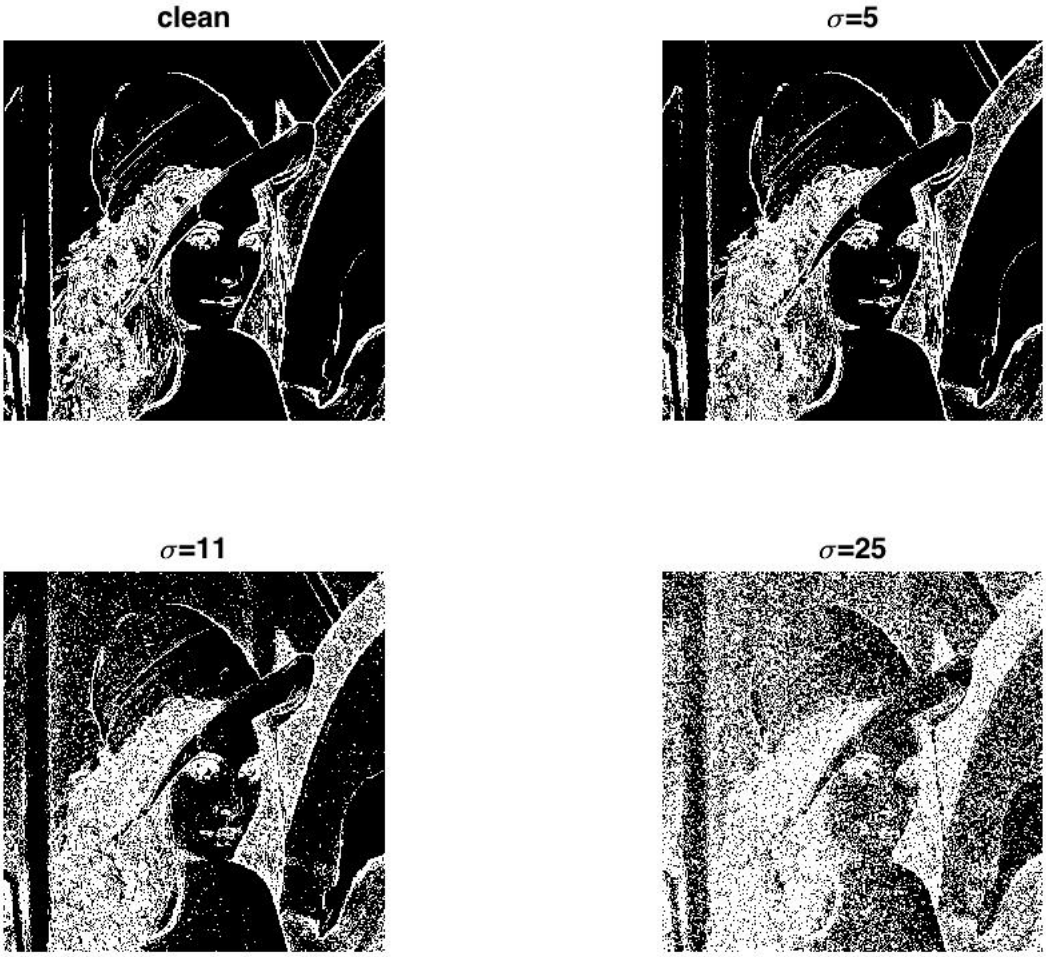

Figure 46: lena.png applying the Frei-Chen method with a threshold T=30

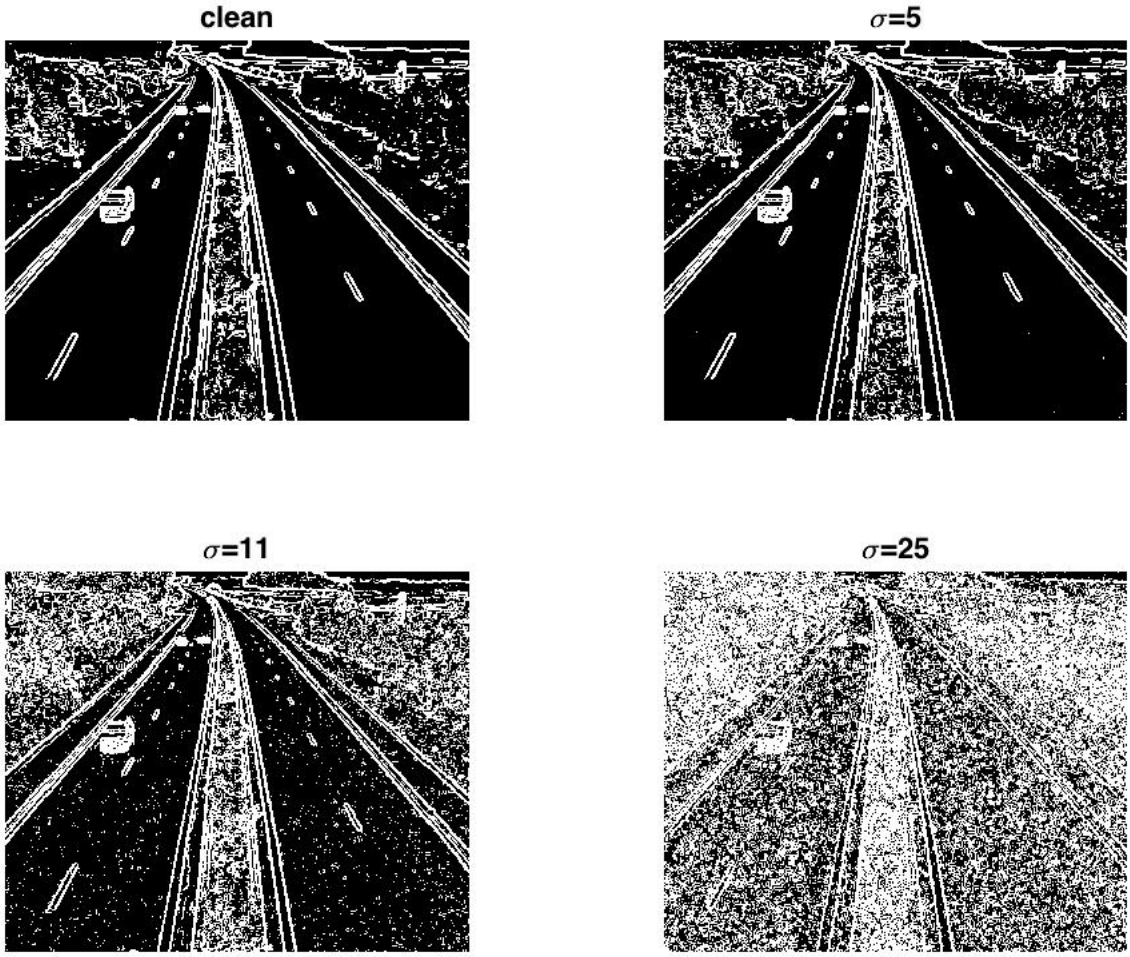

Figure 47: road.png applying the Frei-Chen method with a threshold T=30

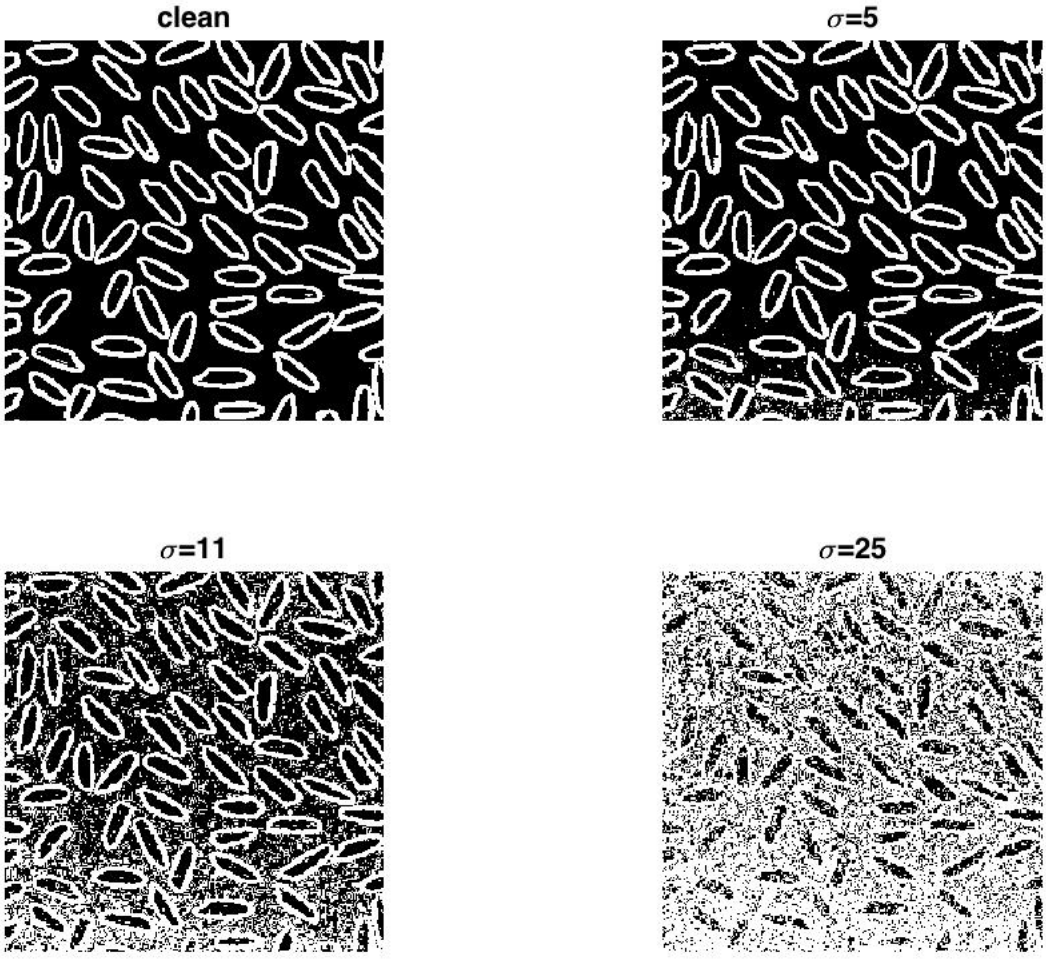

Figure 48: rice.png applying the Frei-Chen method with a threshold T=30

With T=70

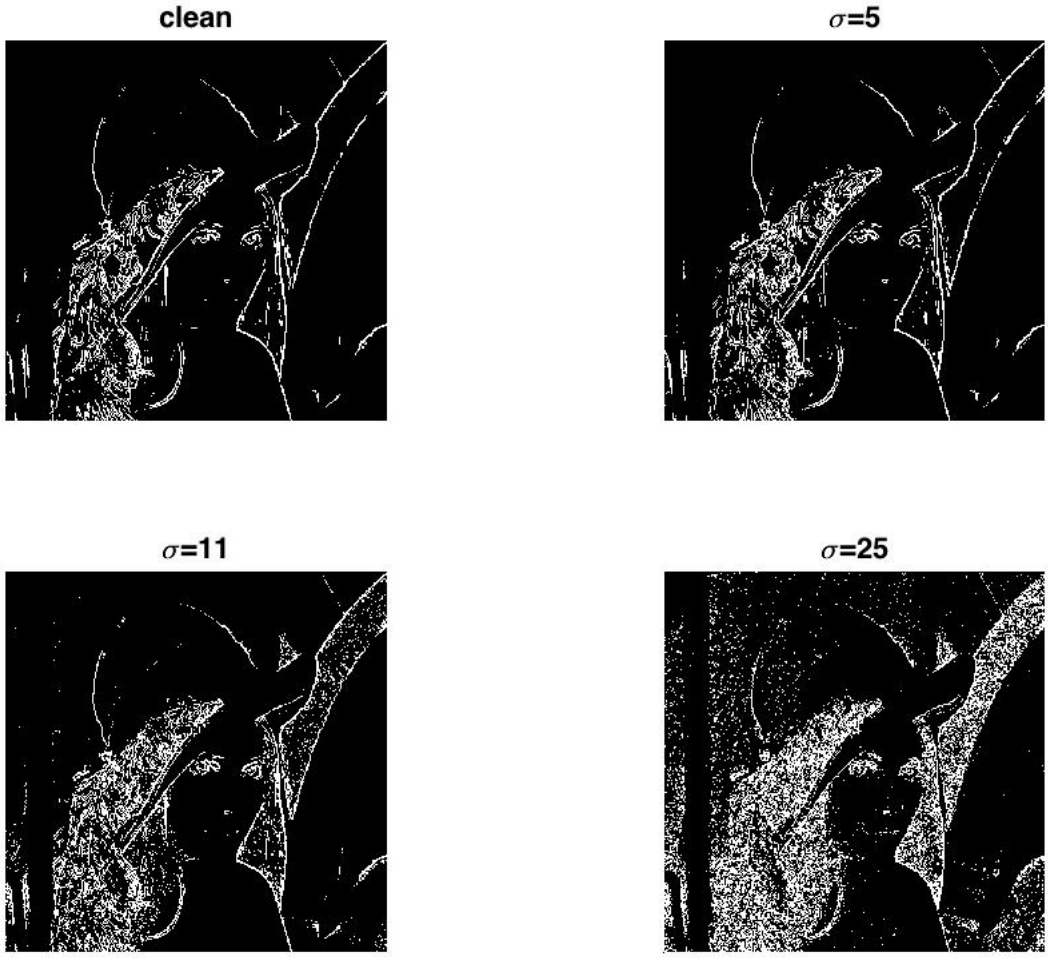

Figure 49: lena.png applying the Frei-Chen method with a threshold T=70

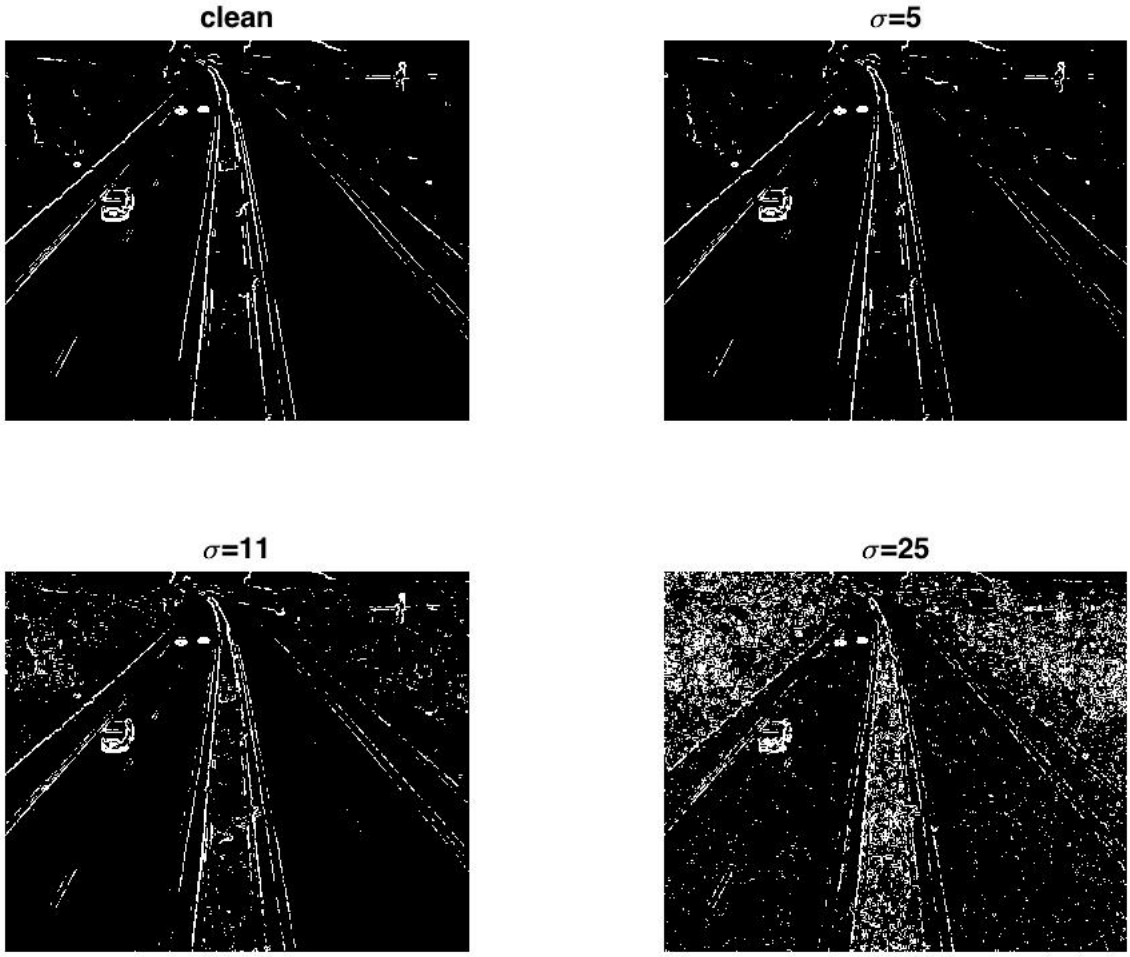

Figure 50: road.png applying the Frei-Chen method with a threshold T=70

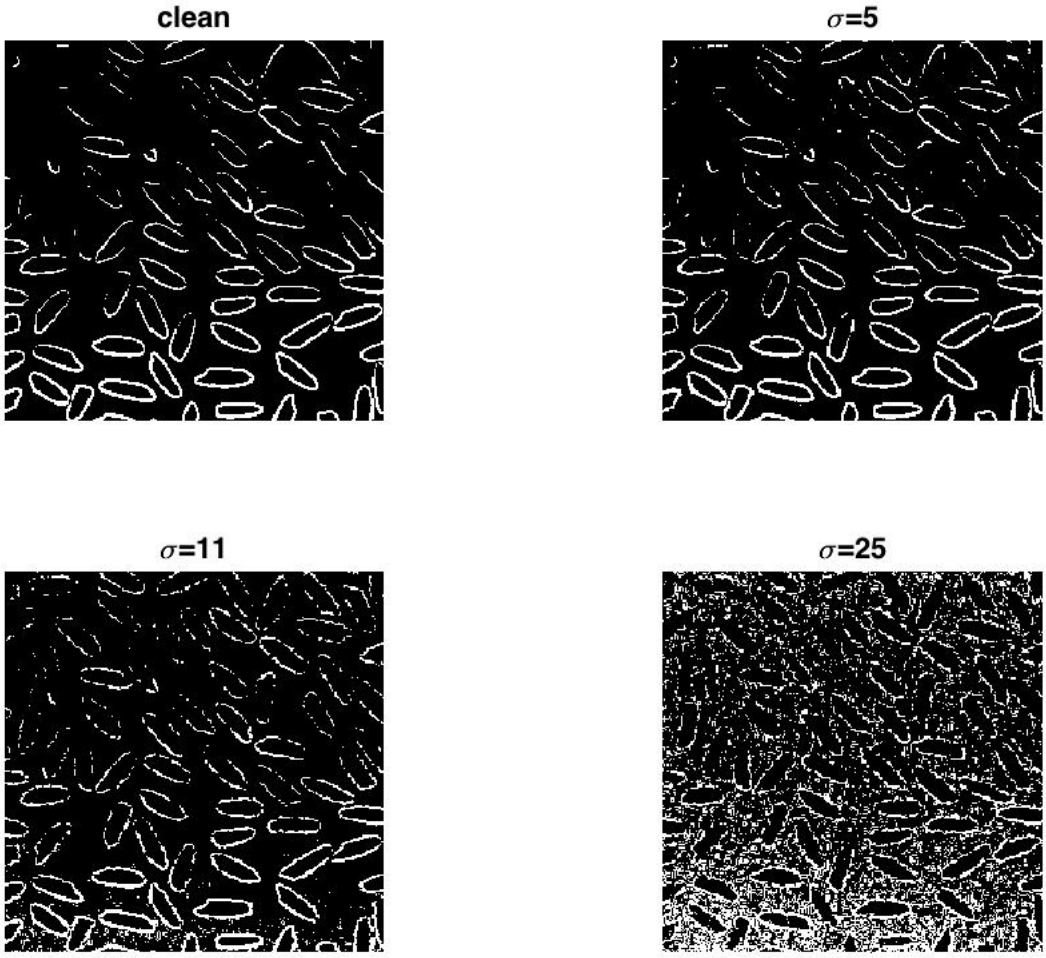

Figure 51: rice.png applying the Frei-Chen method with a threshold T=70

With T=100

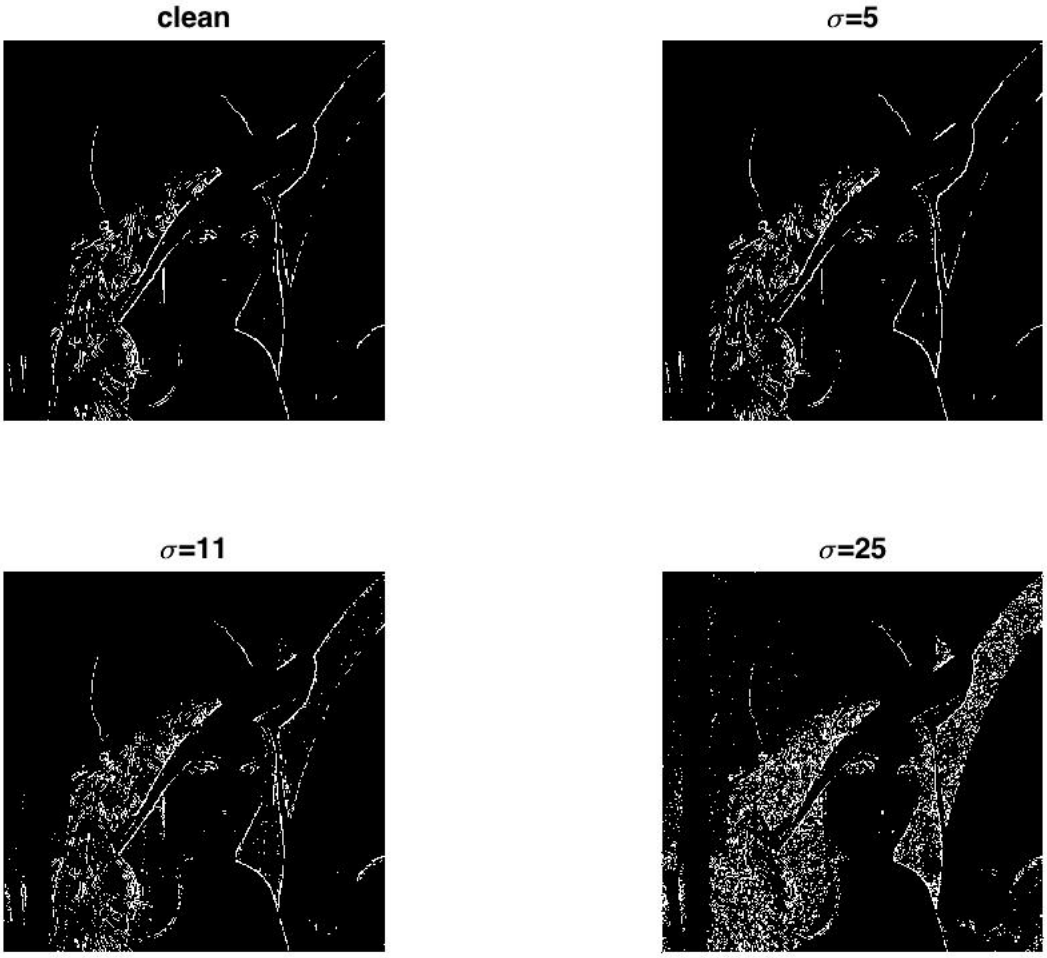

Figure 52: lena.png applying the Frei-Chen method with a threshold T=100

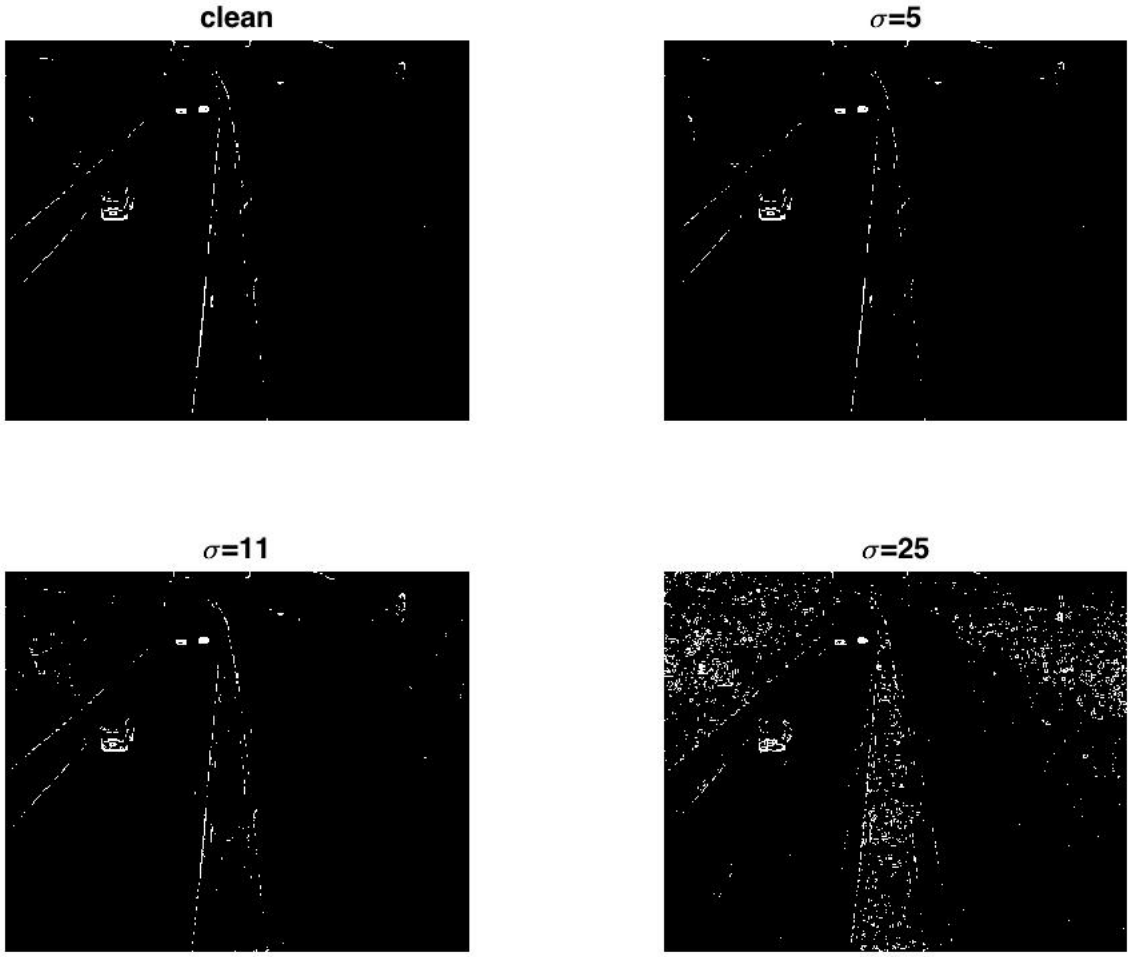

Figure 53: road.png applying the Frei-Chen method with a threshold T=100

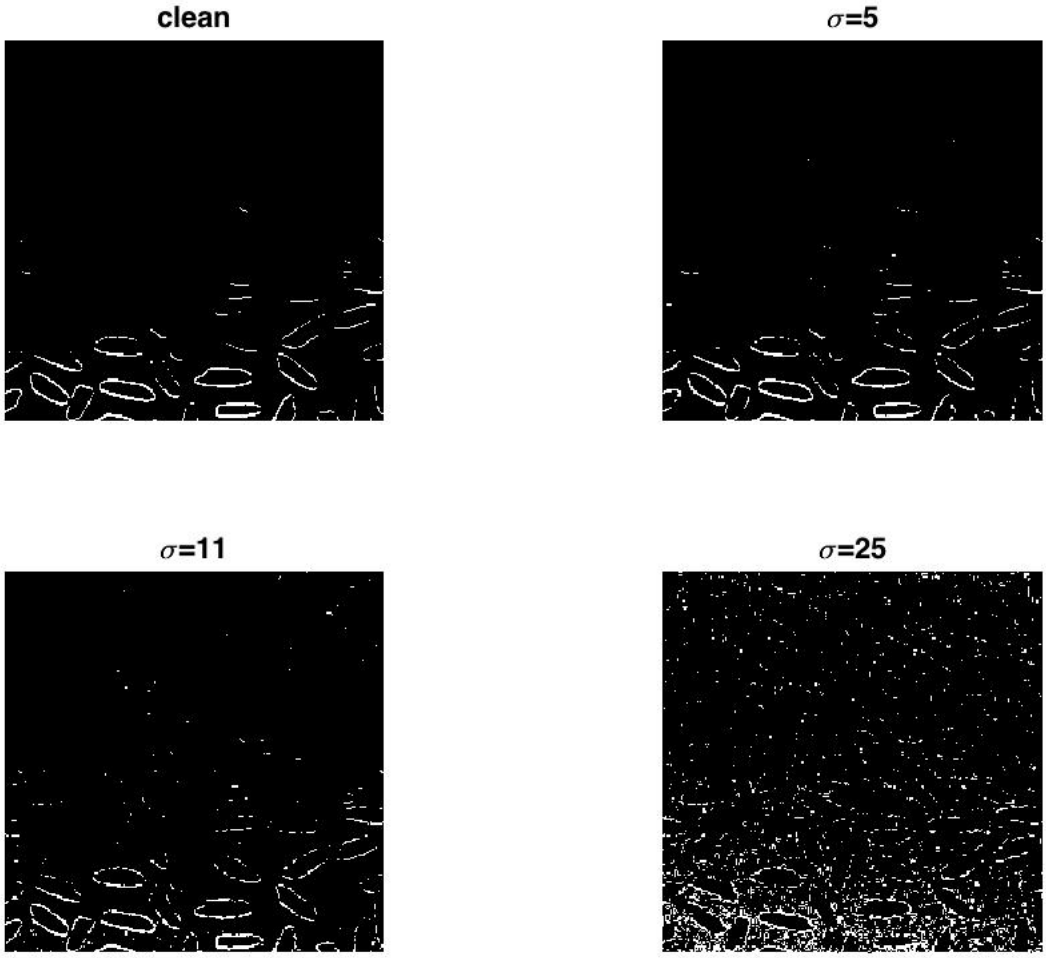

Figure 54: rice.png applying the Frei-Chen method with a threshold T=100

Evaluation

efficiency

We can divided in two groups the results corresponding to two different type of edges.

The Template, Compass, and Frei Chen methods give a similar type of edges, those methods tend to find the major edges of the images, resulting in edges of different thickness. This thickness is regulated by the Threshold parameter but if we tend towards fine edges, edges information are lost.

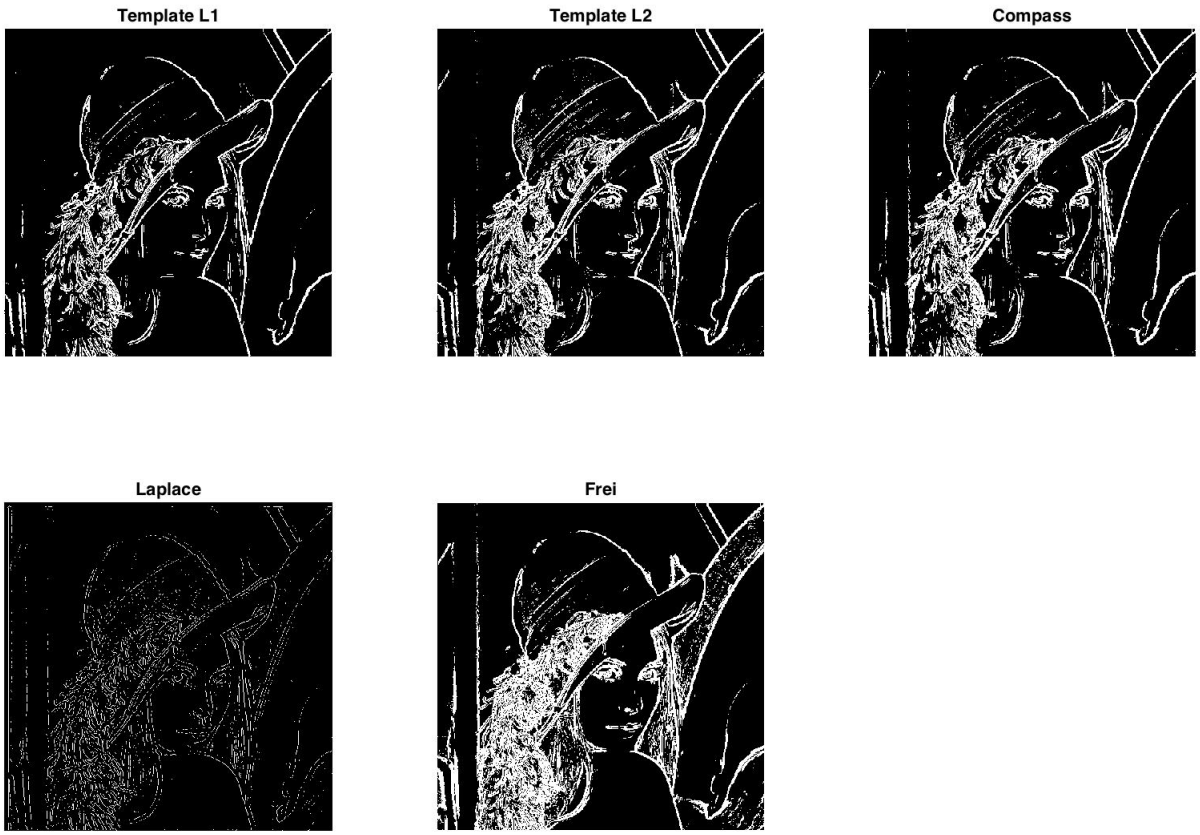

Figure 55: Comparison of the edges result for the clean lena image with T=30

In term of efficiency for the Template method , the result with the L1-norm gives less information about the edges (less edges are represented) than the the result with the L2-norm. For example look at the fig.55,the contour of the mouth is more detailed for the L2 -norm than for the L1-norm. The Frei Chen method and Compass operator results in more edges represented but they are thicker resulting in a loss of precision of the edges.

The methods are differently affected by noise, for the Template and Compass method the result is like a uniformally distributed noise have been applied after the edges detection. For a lot of noise the edges are barely noticeable. Here is the ranking from the less sensitive to the more sensitive to the added gaussian noise : - Template with the L1-norm; - Template with the L2-norm; - Compass operator where the edges are unreadable;

The Frei Chen however, is differently affected by noise, the resulting image with the edges doesn't have a uniform distribution of some noise. The effect of the noise on the edges detection seems to result in a negative of the original image. The edges when it defines the limit between a dark area and a light area is noticeable even with a lot of noise but the edges inside both areas are unreadable.

The Laplace method gives an other type of resulting edges defined like the precise pixel where the LOG goes through zero with a difference between the negative and positive values of the LOG greater than the given threshold. If the threshold is high, it results in representing only sharp edges. As σ increase, the quantity of edge information also increases.

The effect on the noisy input images on those methods results in also noisy resulting edges where the noise is uniform on the image.

The efficiency depends on what is aimed, if it is precise contours the Laplace operator is preferred. If it is the major edges usually defining he major components of the images, the Template, the Compass and the Frei Chen methods give good results. Still, their efficiency depends on the noise on the image and each method has it s own dependency to noise.

complexity

For the comparison of complexity of these methods, we will do a brief analysis of the number of operations done by each method.

Template method in *L1-norm* basically has two convolutions, three power operation, and one sum as well as comparisons with threshold for each pixel. *Template method* in *L2-norm* has two convolutions, two operations for getting magnitude, and one sum as well as comparisons with threshold for each pixel.

Compass operator with 8 different operators, basically contains eight convolutions, one maximum operation, as well as comparisons with threshold for each pixel.

Laplace operation has one operation for creating the LoG, one operation for convolution, as well as doing the zero crossing by comparison for neighboring pixels. It also includes a threshold part.

Frei-Chen method includes 9 different operators, thus it has nine convolutions, ten power operations, two sums, and one division. The same threshold part is implemented as previous.

To sum up, *Frei-chen method* seems to be the most complex one with the largest number of operations. Compared to *Template method*, *Compass operator* more of less does the same thing, but instead of two operators, it includes eight different direction's operators. Consequently, *Compass operator* increases complexity than *Template method*. And *Laplace operator* seems to be a simple one, with one convolution of the image and LoG(or you can view it as one convolutions of a low-pass filter and then a Laplacian).