SGN-12006 2016-02 – BASIC COURSE IN IMAGE AND VIDEO PROCESSING

TAMPERE UNIVERSITY OF TECHNOLOGY - TUT

PROJECT REPORT: CANNY EDGE DETECTOR

DANIEL KOSLOPP – 267637 – 12/2016

INTRODUCTION

The project consists in the realization of a graphical interface for *Canny* Edge Detector. The user should be able to apply it to any image, with any *Canny* parameters. The user should also be able to specify a certain window in the image (by selecting the region with the mouse) and apply the edge detector only there.

EDGE DETECTION

In image processing, a big effort is put on extracting attributes from the images and one of the most difficult tasks is segmentation. Segmentation are based mainly in two properties: similarity and discontinuity. The latter one is where edges detectors are applied. (Gonzalez & Woods, 2008)

Edge detectors are local image processing methods designed to detect edge pixels. To do so, three fundamentals steps are needed. (Gonzalez & Woods, 2008)

1. *Noise reduction from image smoothing*: Even little visual noise may have huge impact on derivative functions, and they are the key for edge detection.
2. *Detection of edge points*: This step locates all potential points candidates to become edges.
3. *Edge localization*: Select from all candidates only the true members of the edges.

In the sequence of this report some basic techniques will be shortly described and compared with the one used in the software which this report is about.

EDGE DETECTION TECHNIQUES

Gradient Operators

The first-order derivative (gradient) may be used to obtain edges. The easiest way to apply the gradient to an image is to use masks and filter it. The simplest gradient mask is show bellow:

-1	
1	

-1	1

1. Gradient

These masks detect borders only on *x* and *y* direction. If diagonal edges are of interest, it is also possible to apply *Roberts cross-gradient operators*:

-1	0
0	1

0	-1
1	0

2. Roberts

Masks 2x2 are not as useful to obtain direction as 3x3 that are symmetric about the center point. Such masks are called *Prewitt* and *Sobel*:

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

3. Prewitt

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

4. Sobel

As mentioned earlier, smoothing the image improve the result since the derivatives are very sensitive to noise. In fact, Sobel mask have better noise-suppression, thus, is usually preferable when compared to Prewitt (Gonzalez & Woods, 2008).

The masks mentioned above are used to obtain the gradient of the edges and correspond to the second step on the process (the first one is smoothing). In other words, when applying this masks potential edges are found, but the image should be further processed to obtain the true ones.

Hence the next step is to compute the magnitude of the gradient by one of the following equations:

$$M(x,y) = \sqrt{g_x^2 + g_y^2} \approx |g_x^2| + |g_y^2|$$

The magnitude combined with threshold is used to finally obtain the true edges.

Canny Edge Detector

The simple way to obtain edges mentioned above does not make any attempt to improve the results based on noise content or edge characteristics, therefore, they usually do not provide good enough results. The *Canny* method is a more advanced technique that takes it into account. The approach is based on three objectives (Gonzalez & Woods, 2008):

1. *Low error rate:* All edges should be found.
2. *Edge points should be well localized:* The locations should be as close as possible to the true edges.
3. *Single edge point response:* The detector should return only one point for each true edge.

Canny showed that the first derivative of the Gaussian closely approximates the operator that optimizes the localization of the edge. In this sense the first step of the technique is to obtain a smoothed image $f(x,y)$ convolving as:

$$f_s(x,y) = G(x,y) \circledast f(x,y), \text{where } G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Then, the magnitude $M(x,y)$ and the direction $\alpha(x,y)$ are calculated as follows, where g_y and g_x are the gradient of f_s for each pixel.

$$M(x,y) = \sqrt{g_x^2 + g_y^2}$$

$$\alpha(x,y) = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

The magnitude usually contains wide edges around the local maxima. To thin these ridges *non-maxima suppression* is applied. This step basically selects the pixel with biggest magnitude in a local edge.

The final phase is to threshold the image to select the true edges. If the threshold is done using a single value, there are great chances that noise edges will become false positives and/or true edges will be discarded depending on the value used. To improve this, the *Canny* Edge Detector uses *hysteresis thresholding*.

Hysteresis thresholding uses two thresholding levels to form *weak* and *strong* edge pixels' images. All *strong* pixels that are not contained on the *weak* are accepted as true. After that, true edges are linked from the *strong* image using the *weak* version by 8-connectivity for example.

Resuming, the *Canny Edge Detector* consists basically in four steps (Gonzalez & Woods, 2008):

1. Smooth the input image with Gaussian filter.
2. Compute the gradient magnitude and angles images.
3. Apply non-maxima suppression to the gradient magnitude.
4. Use hysteresis thresholding and connectivity analysis to detect and link edges.

The algorithm to execute this process is detailed in the following section.

CANNY EDGE DETECTOR ALGORITHM

The Interface

The software was developed in MATLAB and consists in a GUI interface showed in Figure 1. In the GUI, the user can select the image from "*Select Figure*" button and input parameters for "*Lower Threshold*", "*Higher Threshold*" and "*Sigma*" in the corresponding fields. To select a specific region in the image where the edges should be detected, the user should first click in the

left image area (original image), a cross pointer should appear and then the region can be selected. To finally detect the edges, the user should press the button "*Detect Edges*".

If no specific region was selected, the software will detect edges in the whole image. The result is displayed in the right side.

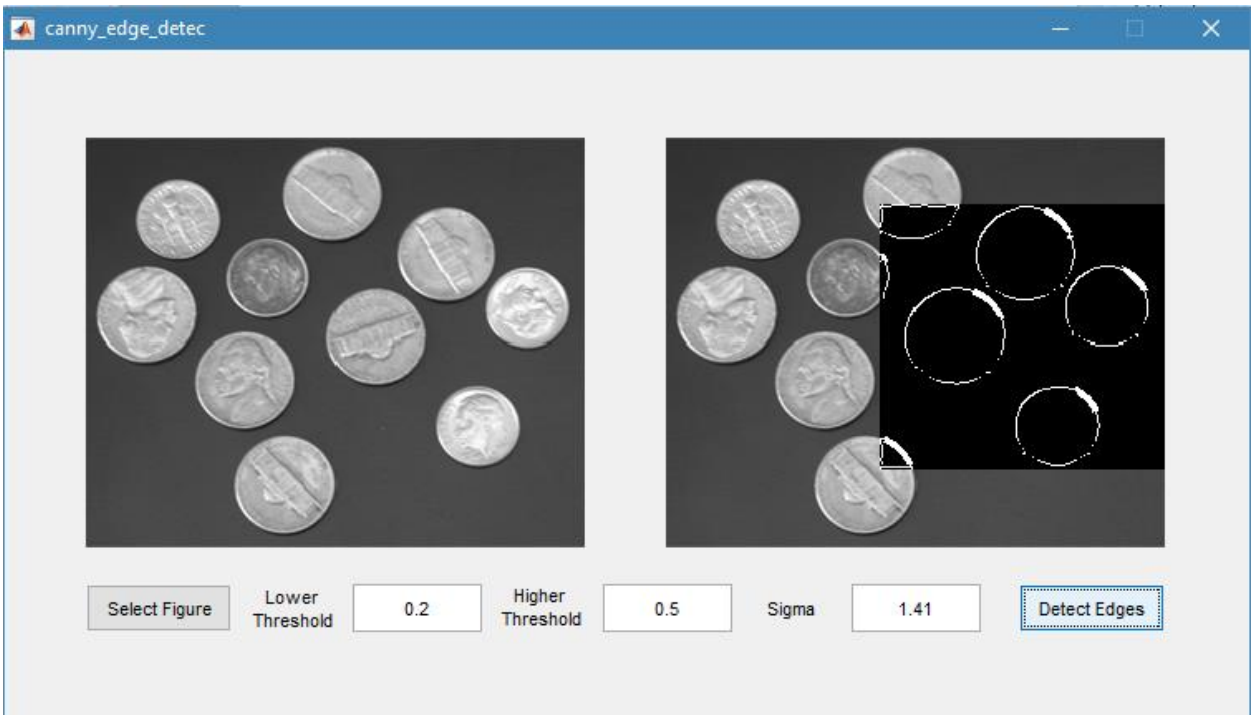

Figure 1. GUI Interface.

Executing the Software and Handling User Inputs

To open the software, the file "canny_edge_detec.m" should be open in MATLAB and executed. This file handles the inputs and user actions on the GUI. The following topics describe the functions in this files:

- "canny_edge_detec": Used by MATLAB to initialize the GUI.
- "canny_edge_detec_OpeningFcn": Is executed during initialization and set that no region is selected, thus the software is initially configured to detect edges in the whole image.
- "select_figure_Callback": When the "*Select Figure"* button is pressed, this function is called and opens the interface for selecting the figure in the drive and shows it.

- “original_img_ButtonDownFcn”: When the user clicks on the original image, this function is called and waits for the user to determine a specific region for the edge detection. It processes the regions selected to be inside limits and sets the values corrected and the corresponding variable to indicate that a region was selected.
- “detect_edges_Callback”: When the “*Detect Edges”* button is pressed, this functions reads the values of thresholds and sigma from the corresponding fields, verify if a region is or not selected, and calls the function “canny_edge” accordingly to finally detect the edges and show it.

Since MATLAB can handle all possible errors without crashing the software, no error detection for handling bad user’s input was developed. For example, if “Lower Threshold” is bigger than ““Higher Threshold”, MATLAB will signalize the error and no detection will be made, but after fixing the input problem the software continues to work.

Detecting Edges with *Canny* Algorithm

When the function “canny_edge” is called by the GUI, it essentially applies the four steps needed to find the edges described previously.

The function obtains first the Gaussian mask by calling “gauss_mask_2d”. The mask size is proportional to the sigma value. The filtered image is therefore obtained convolving the mask with the original image.

The gradients of the filtered image are calculated calling “grad_x” and “grad_y” for the directions *x* and *y* respectively. Then the magnitude image and gradient direction are found.

To thinner the ridges edges found in the magnitude image, a non-maxima suppression function “suppress” is called. It detects the maxima points of magnitudes in a 3x3 region accordingly with the direction of the edges (0°, 45°, -45° and 90°).

The thinner image is hence normalized to apply the hysteresis threshold by the function “hyst_thresh”. This function find all the *strong* pixels and its positions. Finally, the true edges are found by 8-connectivity and returned to the GUI.

RESULTS

To compare the results, the same figure is shown with different input parameters. Comparing the Figures 2 to 4 it is possible to see that lower thresholds results in more noise edges but the edges are not as broken as in higher threshold values.

Likewise, the effect of the sigma value can be noticed looking at Figure 2 and 5. Both use the same thresholds values but the last one has bigger sigma. As consequence, the original image is more heavily filtered and less noise is accepted as true.

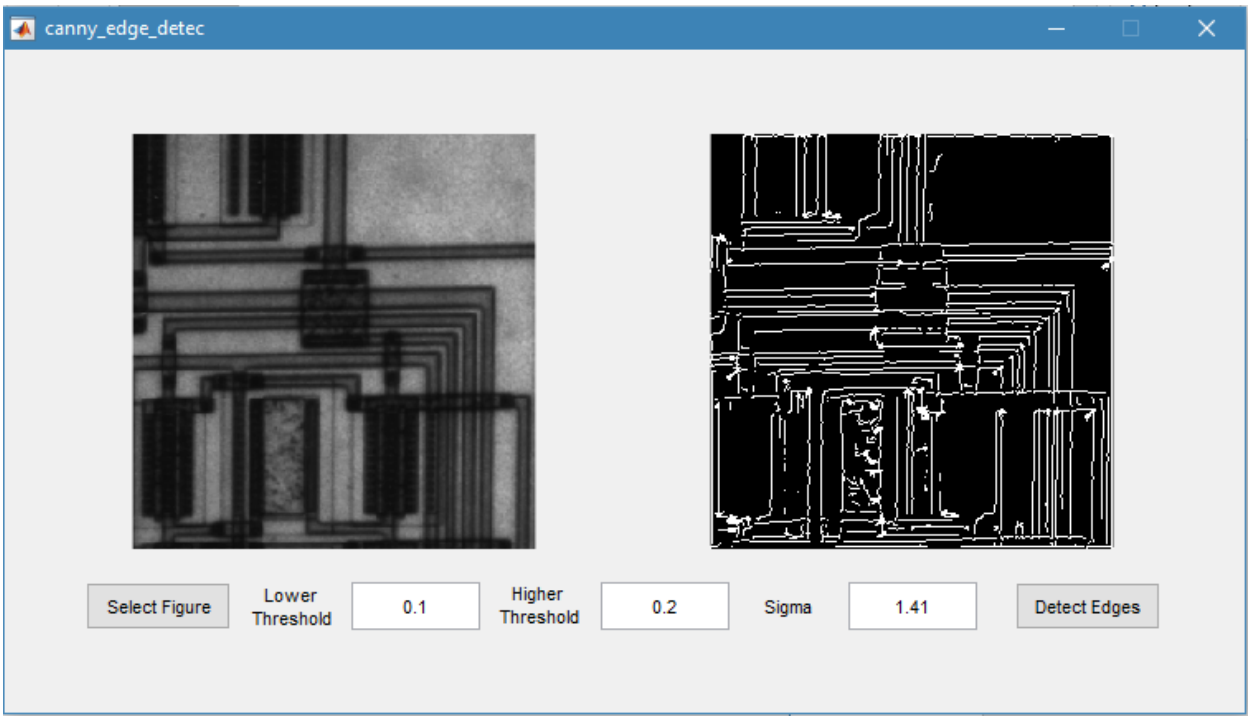

Figure 2. Result for Lower Threshold = 0.1, Higher Threshold = 0.2 and Sigma = 1.41.

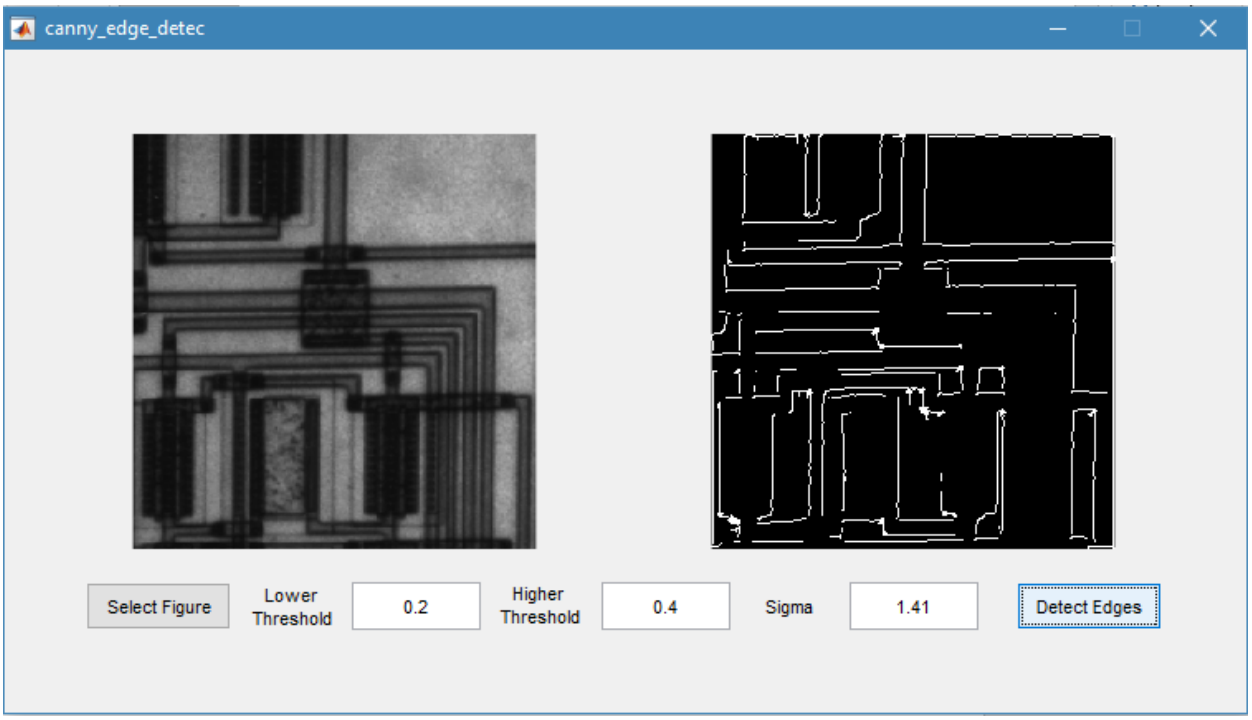

Figure 3. Result for Lower Threshold = 0.2, Higher Threshold = 0.4 and Sigma = 1.41.

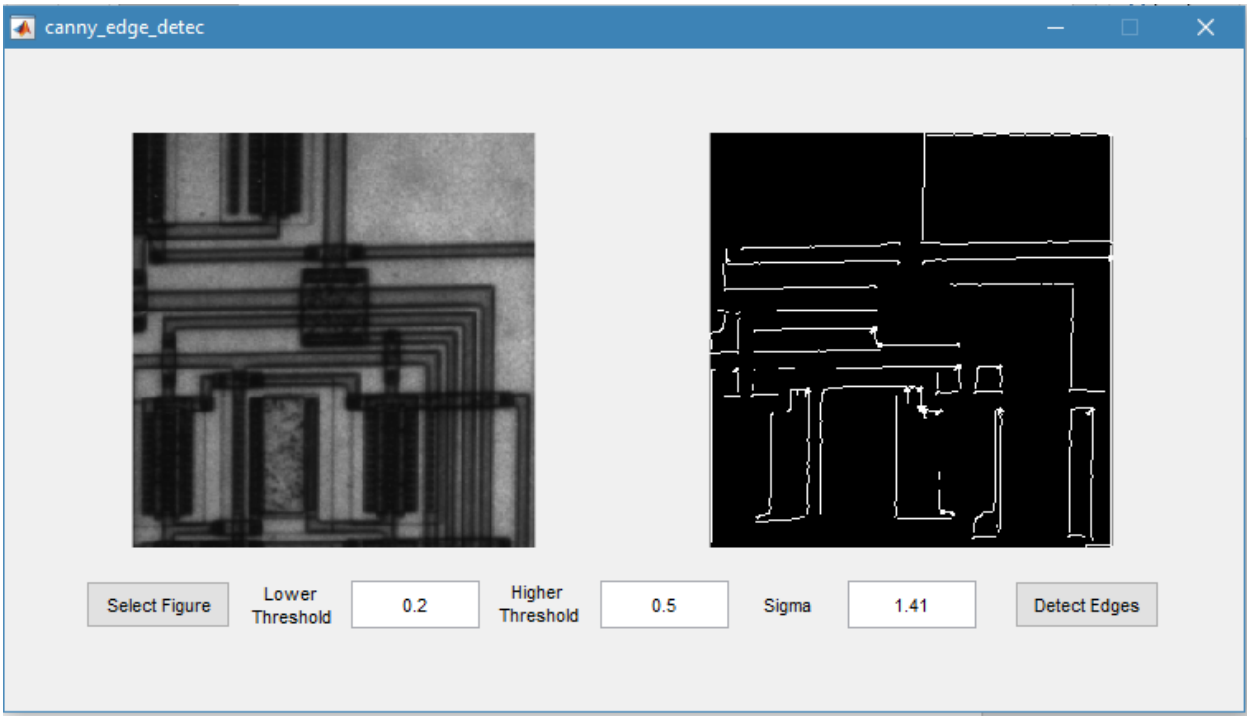

Figure 4. Result for Lower Threshold = 0.2, Higher Threshold = 0.5 and Sigma = 1.41.

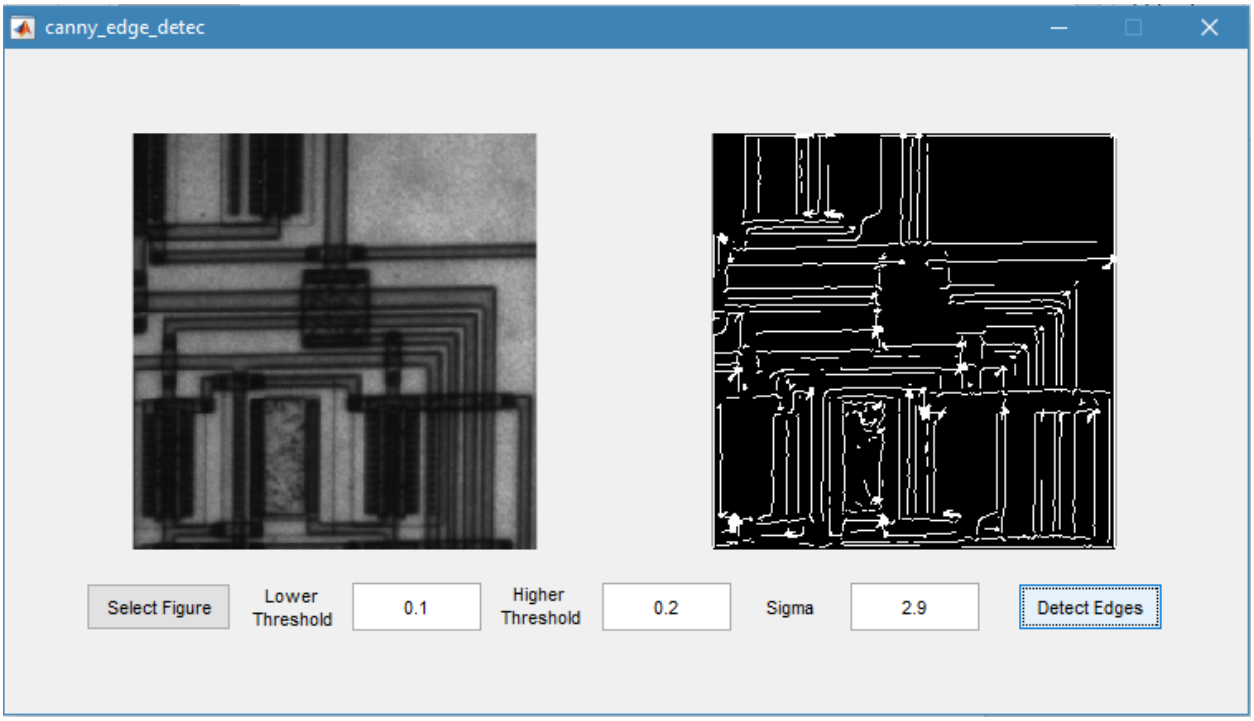

Figure 5. Result for Lower Threshold = 0.1, Higher Threshold = 0.2 and Sigma = 2.9.

CONCLUSION

The *Canny* Edge Detector is one of the mostly applied algorithm used for this purposed due to good results it provides specially in noisy images. The application here developed could provide a good approximation of the *Canny* algorithm and demonstrate the parameter's effect on the result.

Although the results were satisfactory in the author's opinion, when comparing with professional applications like the built-in MATLAB function "edge", it is clear that the result could be improved.