ECEN 468 Advanced Logic Design
Department of Electrical and Computer Engineering
Texas A&M University

(Lab exercise created by Jaeyeon Won and Jiang Hu)

Lab 10

Design of Canny Edge Detector

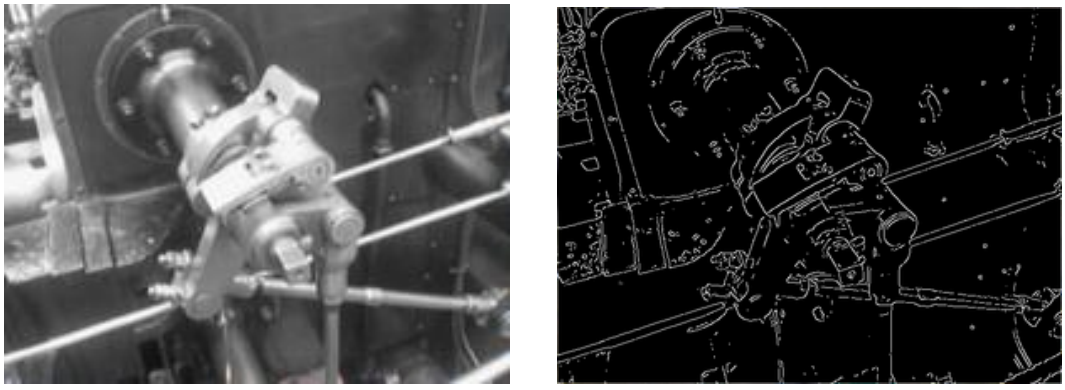
Purpose:

In this lab, we will design Canny Edge Detector with Verilog, and will use Design Analyzer as a tool for synthesis as well as Lab7 to Lab9. Also, your module will be verified using functional simulation and gate simulation with the netlist synthesized.

Preparation:

1. Brief introduction to Edge Detector

Edge detection refers to the process of identifying and locating sharp discontinuities in an image. The discontinuities are abrupt changes in pixel intensity which characterize boundaries of objects in a scene. Classical methods of edge detection involve convoluting the image with an operator (2-D filter), which is constructed to be sensitive to large gradients in the image while returning values of zero in uniform



(a) (b)

Figure 1 An example of Canny Edge Detection (reference : Wikipedia)

regions. There is an extremely large number of edge detection available, each designed to be sensitive to certain types of edges. Variables involved in the selection of an edge detection operator include edge orientation and noise environment. The geometry of the operator determines a characteristic direction in which is most sensitive to edges. Operators can be optimized to look for horizontal, vertical, or diagonal edges. Also, edge detection is difficult in noisy images, since both the noise and the edges contain high-frequency content. Attempts to reduce the noise result in blurred and distorted edges.

There are many ways to perform edge detection. However, the majority of different methods may be grouped into two categories which are gradient method and Laplacian method. In general, Laplacian method results in higher quality and complexity. Thus, we will use gradient method to detect edges. The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image.

2. Flow of the algorithm

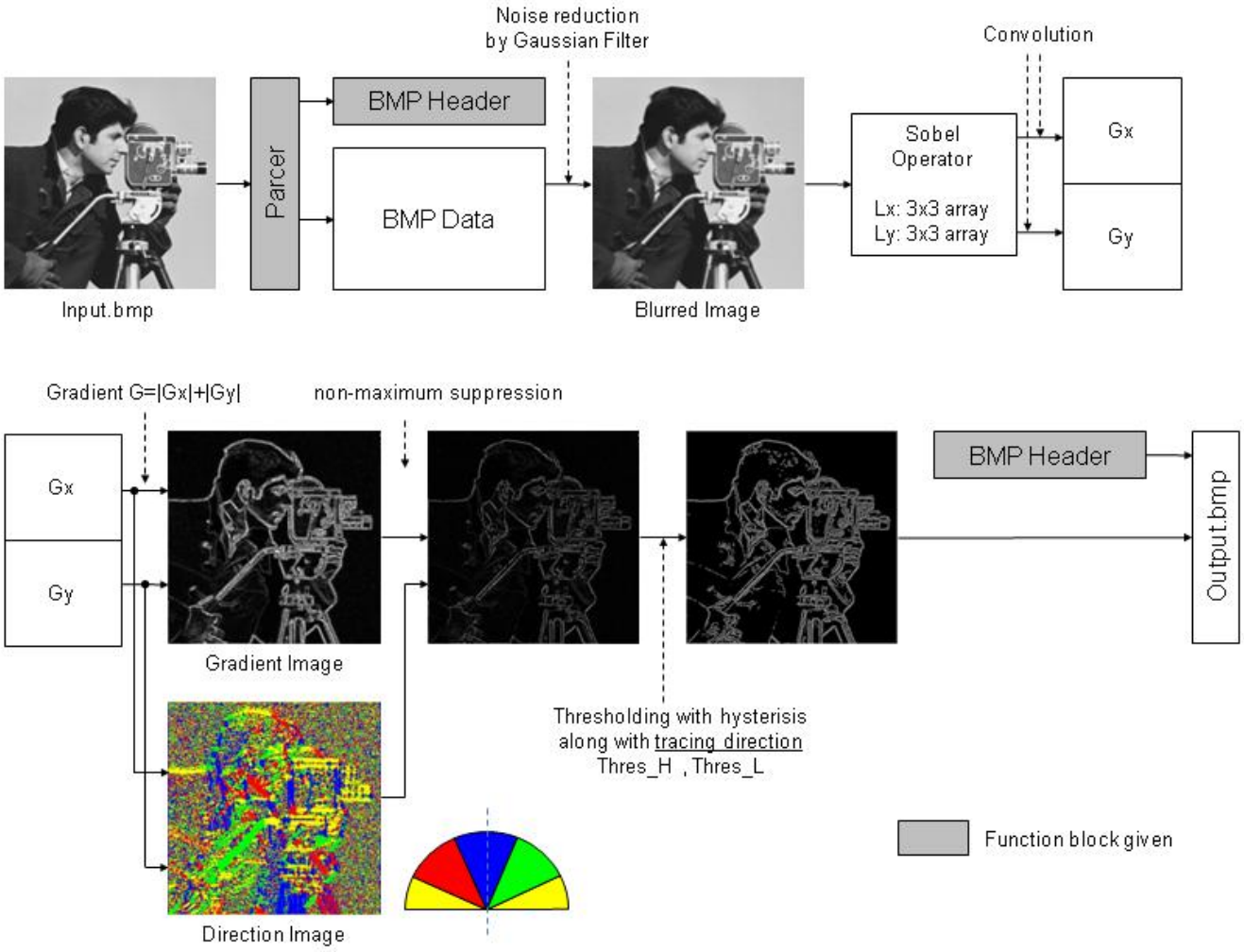


Figure 2 Data-flow of Edge Detection

In order to implement the Canny edge detector algorithm, a series of steps should be followed as shown in Figure 2.

2.1. Noise Reduction

Please refer to the Lab 4 manual for general concept.

For this lab, we will use different Gaussian filter to minimize the error by approximation. Please use the matrix as a Gaussian Filter for this lab.

$$\frac{1}{128}\begin{bmatrix} 1 & 3 & 4 & 3 & 1 \\ 3 & 7 & 10 & 7 & 3 \\ 4 & 10 & 16 & 10 & 4 \\ 3 & 7 & 10 & 7 & 3 \\ 1 & 3 & 4 & 3 & 1 \end{bmatrix}$$

Figure 3 5x5 Gaussian Filter

2.2. Gradient Information

Please refer to the Lab 4 manual for general concept.

Please use the equation to calculate gradient component and set 8 as the value α.

$$|G| = (|Gx|+|Gy|)/\alpha \quad (\alpha\text{: constant for matching full pixel level})$$

Also, we will use different values to compare two directional gradient components. Please use 0.5 and 2.5 instead of 0.4 and 2.4. It will be easier to implement to hardware.

1st step: If(Gy<0) { Gx = Gx * (-1) Gy = Gy * (-1) }

2nd step: Gx ≥0, Gy ≤ 0.5*Gx → degree 0
0.5*Gx < Gy ≤ 2.5*Gx → degree 45
2.5*Gx < Gy → degree 90

Gx <0, Gy ≤ -0.5*Gx → degree 0
-0.5*Gx < Gy ≤ -2.5*Gx → degree 135
-2.5*Gx < Gy → degree 90

2.3. Non-maximum suppression

Please refer to the Lab 4 manual for general concept.

2.4. Hysteresis thresholding

You should use the threshold values below for this step.

High threshold : 15
Low threshold : 10

Please refer to the Lab 4 manual for general concept.

3. Structure

Please refer to the Lab 4 manual for general concept.

4. Useful Verilog HDL skills

4.1. Array

a. One-dimensional Array

We have implemented two-dimensional array sc_uint <8> regX [5][5] in SystemC. Almost Verilog simulators do not provide two-dimensional expression. Thus, you may implement it with one-dimensional expression, and use it with indexing. For example, if we want to implement the 5x5 array, you can declare reg [8] regX[25] and index is from 0 to 24. To be supported by all kinds of simulators and synthesizable, it is recommended to use one dimensional array. But, you may use the two-dimensional array expression for the functional simulation.

Index = Row number*5 + Column number

b. Two-dimensional Array

The latest version of the simulator VCS provides the expression of two-dimensional array. You can declare reg [8] regX[5][5]. For using this syntax, you will use the option +v2k when you simulate it.

```
ECEN468/Lab10/SRC]$ vcs    tb.v    +v2k
```

4.2. reg signed

```
reg [8] RegX[25];              // unsigned 25 registers with 8 bits width
reg signed [8] RegX[25];       // signed 25 registers with 8 bits width
```

4.3. Shift Right instead of divide operation

When you synthesize modules, cell libraries may not have appropriate libraries for divide operation. To support all libraries, it is recommended to use shift right operation with the sign '>>'.

- Divided by 2 → (variable) >> 1
- Divided by 4 → (variable) >> 2
- i.e.) var <= var / 128; → var <= (var>>7);

5. Functional Simulation

Once the design is completed it must be tested. The functionality of the design module can be tested by applying a testbench and checking the results. The testbench module can instantiate the design module and directly drive the signals in the design module. The testbench can be complied along with the design module. At the end of compilation the simulation results will be displayed.

Download 'Lab10_code.tar.gz' file from the lecture website. The file contains CannyEdge.v, tb.v, generic.sdb, osu018_stdcells.v and osu018_stdcells.db. You will design modules in these files after you decompress the file.

a. Make working folder for this lab.

HOME]$ mkdir ECEN468/Lab10/SRC (make work folder)
HOME]$ cd ECEN468/Lab10/SRC (move to work folder)
Move the file 'Lab10_code.tar.gz' into the working folder
ECEN468/Lab10/SRC]$ tar xvfz Lab10_code.tar.gz (decompress the file)

Please check the files decompressed. It should contain the files below.
CannyEdge.v, tb.v, generic.sdb, osu018_stdcells.v and osu018_stdcells.db

b. Insert your code into the CannyEdge.v. Your module should satisfy the requirements

c. You may change the test-bench given. If you have modified, please write down why and what parts you have modified in your report.

d. Start Functional Simulation

ECEN468/Lab10/SRC]$ vcs tb.v

ECEN468/Lab10/SRC]$ vcs tb.v +v2k (if you use two-dimensional array)

If it shows compile errors, please check your codes again (go to step b) or make sure that your previous modules are correct. If compilation is complete, go to next step to get the results.

ECEN468/Lab10/SRC]$ simv

If it is not same to your estimated results, please check your codes again (go to step b).

Now, please show your result to TA within your regular lab hour to get credit for lab evaluation.

6. Starting design analyzer

To start design analyzer, please follow steps. For your convenience, please start *design_analyzer* from separate directory with verilog files(*.v) because it generates a lot of intermediate files while it goes further.

Make directory for design analyzer. Now, you should be in ECEN468/Lab10/SRC.

ECEN468/Lab10/SRC]$ mkdir DV_WORK	(make work folder)
ECEN468/Lab10/SRC]$ cd DV_WORK	(move to the work folder)
ECEN468/Lab10/SRC/DV_WORK]$ design_vision &	(execute Design Analyzer)

7. Procedures of synthesis process:

Please refer to the previous manuals.

8. Gate Simulation

a) Copy Verilog netlist and sdf file to work folder.

ECEN468/Lab10/SRC/DV_WORK]$ cp CannyEdge_gate.v **../.**	(copy it to work folder)
ECEN468/Lab10/SRC/DV_WORK]$ cp CannyEdge.sdf **../.**	(copy it to work folder)
ECEN468/Lab10/SRC/DV_WORK]$ cd **..**	(move to work folder)

b) Follow the steps below to make your testbench for gate simulation

Now we will make testbench for gate simulation. For gate simulation, we need CannyEdge_gate.v (netlist), CannyEdge.sdf (SDF file), osu018_stdcells.v and testbench.

Please make sure those files should be in the current working folder. If not, please let TA know.

ECEN468/Lab10/SRC]$ cp tb.v gate_tb.v (copy new testbench)

Please check that the below lines are in gate_tb.v

On top of the file,

```
`timescale 1ns/10ps
`include "CannyEdge_gate.v"
`include "osu018_stdcells.v"
```

On the bottom of the file, **(This line should be inside module)**

```
Initial
        $sdf_annotate("CannyEdge.sdf", CannyEdge_01);
```

Delete the line `include “CannyEdge.v” in your testbench

If you want to generate the waveform file, change the name of dump file to “wave_gate.dump” in your testbench. Please note that the size of the file will be huge, and be careful when you use the generation. You may delete the file after completion of this lab for your linux server capacity allowed.

d) Start gate simulation

ECEN468/Lab10/SRC]$ vcs gate_tb.v

If there is no syntax error, then do next step.

ECEN468/Lab10/SRC]$ simv

Please compare the result with the functional simulation result. If it is not same to your estimation, start simulation again from the functional simulation after you modify design modules.

Now, please show your result to TA within your regular lab hour to get credit for lab evaluation.

9. How to use the comparator given

To compare your results with the reference outputs given, you may use the another testbench file given. All names of output files(images) should not be changed. Please check your results with output pictures (bmp files) first. If the results do not look good enough, try to fix your implementation regardless of matching ratio of the comparator. When you have correct result, but you are not sure if your results are correct, please refer to the matching ratio of the comparator.

ECEN468/Lab10/SRC]$ vcs tb_comp.v +v2k

Requirements:

1. General requirements.
   a. It should control Canny edge detector correctly.
   b. Detailed comments.
   c. Please do not change the names of the signals given and the other functions not mentioned in the report. (You will lose some points if you change any names of the input or output signals.)
   d. You may modify the test-bench given to get the results only if it is resaonable. Please write detailed comments if you have modified.
   e. You score will be given based on your result, not your code.
   f. Late penalty : 20% of total score will be deducted on each subsequent weekday after due date.

2. Submit hardcopy of your report to TA. The report should include contents below
   a. The output images using functional simulation (input : kodim22_200.bmp) (10 points)
      (Please refer to the reference output results given and a comparator.)
   b. The output images using gate simulation (input : kodim22_200.bmp) (10 points)
      (Please refer to the reference output results given and a comparator.)
   c. CannyEdge.v with detailed comments (2 points)
   d. Only the parts modified in tb.v and gate_tb.v, if you have modified. (2 points)
   e. Calculate the maximum clock speed of the test-bench you used if you have changed clock period of the testbench. Otherwise, calculate the frequency given in the testbench. (6 points)
      (You will get scores based on the maximum clock speed.)
   f. Show your final results to TA. (5 points)

Note : Please send **the files ONLY** to TA's email address. (5 points)

CannyEdge.v, tb.v, CannyEdge_gate.v, gate_tb.v, CannyEdge.sdf

---