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 4

Design of Canny Edge Detector

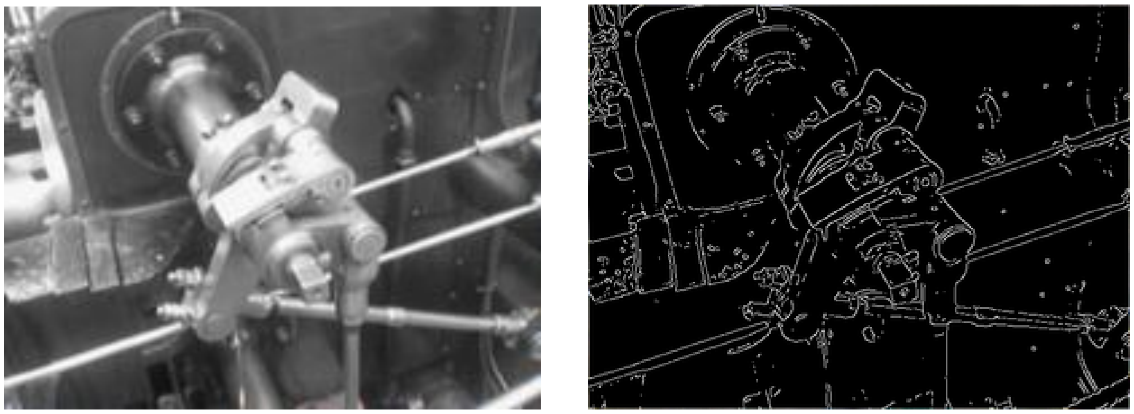
Purpose:

In this lab, we will design Canny Edge Detector with sobel operator with SystemC, and will use Vista as a tool for simulation and verification.

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

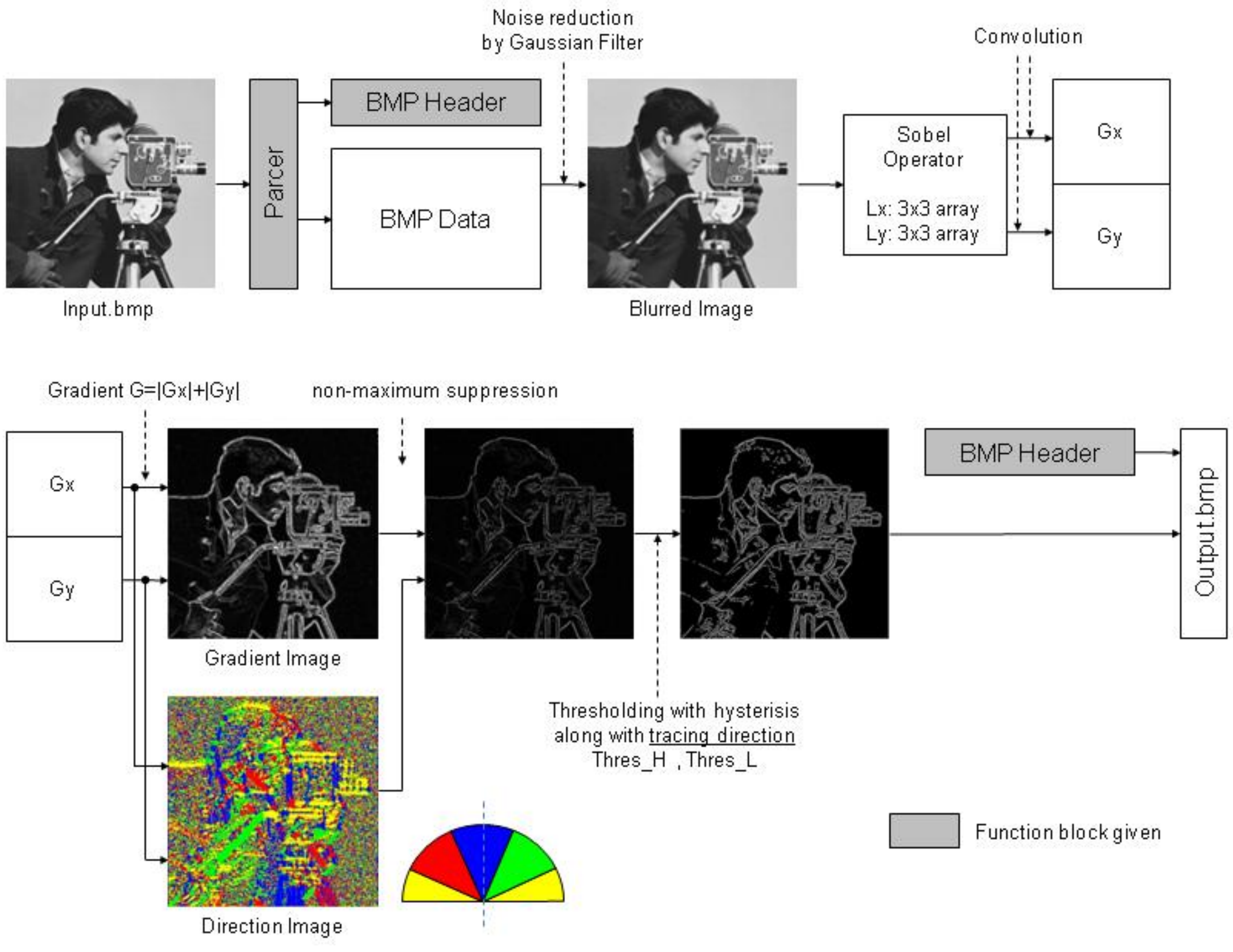


Figure 2 Data-flow of Edge Detection

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

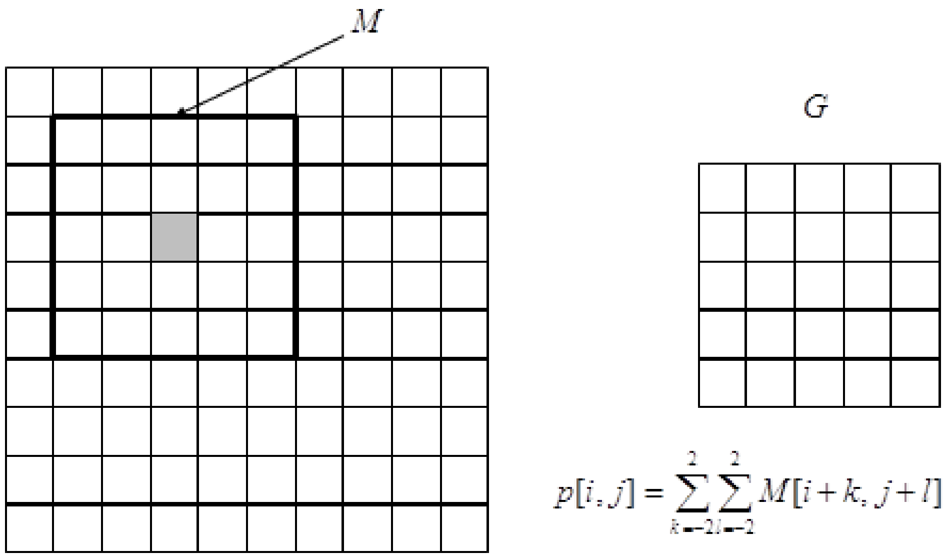
2.1. Noise Reduction

The first step is to filter out any noise in the original image before trying to locate and detect any edges. And because the Gaussian filter can be computed using a simple mask, it is used exclusively in the Canny algorithm. The Gaussian filter should be well defined because the noise reduction step does not only eliminate noise component, but it also results in degradation of the image quality. Once a suitable mask has been calculated, the Gaussian smoothing can be performed using standard convolution methods. A convolution mask is usually much smaller than the actual image. As a result, the mask is slid over the image, manipulating a square of pixels at a time. The larger the width of the Gaussian mask, the lower is the detector's sensitivity to noise. The localization error in the detected edges also increases slightly as the Gaussian width is increased. The Gaussian mask is shown in Figure 3 will be used in this lab.

$$\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Figure 3 A Gaussian Mask G

Figure 4 shows a process to get smooth image. For a certain pixel *p[i,j]*, M will be selected from the original image. And then, new pixel value is calculated with matrix *M* and *G*.



$$p[i,j] = \sum_{k=-2}^{2}\sum_{l=-2}^{2} M[i+k, j+l]\cdot G[k,l]$$

Figure 4 An example of the process to get smooth image

2.2. Gradient Information

After smoothing the image and eliminating the noise, the next step is to find the edge strength by taking the gradient of the image. The Sobel operator performs a 2-D spatial gradient measurement on an image. Then, the approximate absolute gradient magnitude (edge strength) at each point can be found. The Sobel operator uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (Vertical edge) and the other estimating the gradient in the y-direction (Horizontal edge). They are shown in Figure 5.

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(a) (b)

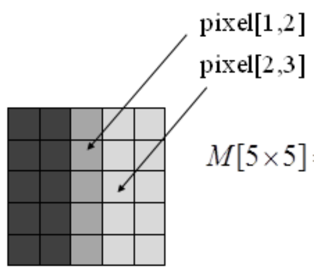
Figure 5 Sobel operators (a) SobelX, (b) SobelY

$$Gx[i,\, j] = \sum_{k=-1}^{1}\sum_{l=-1}^{1} M[i+k,\, j+l]\cdot SobelX\,[k,l]$$

$$Gy[i,\, j] = \sum_{k=-1}^{1}\sum_{l=-1}^{1} M[i+k,\, j+l]\cdot SobelY[k,l]$$

Figure 6 formulas of Gx and Gy

Using the formulas in Figure 6, calculate Gx and Gy value. If Gx is large value, it means that the pixel has large gradient of x-direction (Vertical edge). If Gy is large value, it means that the pixel has large gradient of y-direction (Horizontal edge).



$$M[5\times5]=\begin{bmatrix} 0 & 0 & 30 & 100 & 100 \\ 0 & 0 & 30 & 100 & 100 \\ 0 & 0 & 30 & 100 & 100 \\ 0 & 0 & 30 & 100 & 100 \\ 0 & 0 & 30 & 100 & 100 \end{bmatrix} \quad Gx=\begin{bmatrix} . & . & . & . & . \\ . & 120 & 400 & 280 & . \\ . & 120 & 400 & 280 & . \\ . & 120 & 400 & 280 & . \\ . & . & . & . & . \end{bmatrix} \quad Gy=\begin{bmatrix} . & . & . & . & . \\ . & 0 & 0 & 0 & . \\ . & 0 & 0 & 0 & . \\ . & 0 & 0 & 0 & . \\ . & . & . & . & . \end{bmatrix}$$

(a) (b) (c) (d)

Figure 7 An example of Sobel processes (a) An example image, (b) Pixel values of the (a) (c) Gx (d) Gy

Figure 7 shows an example of the calculation. The 5x5 example image Figure 7(a) has vertical edge which means x-direction gradients. The pixel values of the image are same to Figure 7(b). After applying the formulas in Figure 6, the results of Gx and Gy are shown in Figure 7(c) and Figure 7(d). Let’s take a look more with two specific pixels. For pixel[1,2], the Gx value is 400 and Gy value is 0. It means that it has only x-direction gradient which is vertical edge. For pixel[2,3], it also has only x-direction gradient.

Based on the Gx and Gy, we can calculate the magnitude of the gradient and the direction information. The magnitude, or edge strength, of the gradient is then approximated using this formula. It has been proved that it has good performance while it decreases computational complexity. In this formula, we can use the constant value to fit the maximum level to byte size. The maximum level of an pixel is 255, so the maximum value of Gx and Gy is 255*4. Then, the maximum value of G is 255*8, and we can set a constant 1 to 8 to fit the level to 255 which is full pixel level of our system. However, this case is quite hard to exist in real images such as pictures. For several real images, if we use a maximum constant 8 to fit the pixel level, almost magnitudes of the gradient will be very small values. Therefore, we can adjust the constant value according to the images we used. Make sure that the magnitude should not be over than full pixel level which is 255 in our system. In this lab, you should use 2 as the value of α to acquire larger gradient component. It will be easier to compare the images in eyes.

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

Also, we can get the direction information of the edge using this formula.

$$\text{Theta} = \tan^{-1}(Gy/Gx)$$

However, arctan function cannot be implemented in hardware, and we will approximate it. The purpose to get the direction information is that we determine the direction of the edge among 0 degree, 45 degree, 90 degree and 135 degree which are in the horizontal direction, the vertical direction and two diagonal directions as shown in Figure 8. Thus, we can get one of four directions without arctan function. Please refer to below approximation.

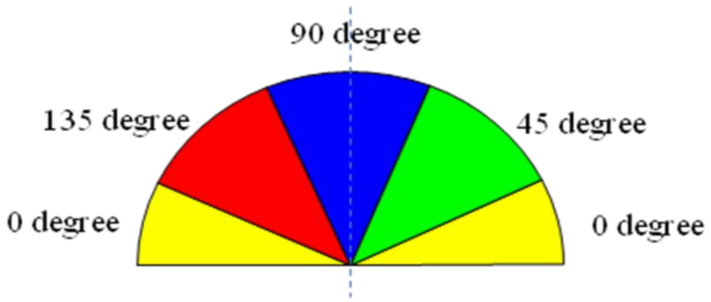


Figure 8 Directions of edges

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

2nd step:    Gx ≥0,  Gy ≤ 0.4*Gx → degree 0
                     0.4*Gx < Gy ≤ 2.4*Gx → degree 45
                     2.4*Gx < Gy → degree 90

             Gx <0,  Gy ≤ -0.4*Gx → degree 0
                     -0.4*Gx < Gy ≤ -2.4*Gx → degree 135
                     -2.4*Gx < Gy → degree 90
```

Please note that the direction in Figure 8 is not a normal information or direction of the gradient, but is a direction of edge. For the example image of the Figure 7, its normal direction or direction of the gradient is 0 degree which means vertical edge, and its edge direction is 90 degree which means vertical edge as shown in Figure 8. For the specific example of pixel[1,1] in Figure 7, Gx is 400 and Gy is 0, so the normal direction is 0 degree using the simplified version of arctan formula. In the source code given, the pixel which has 0 degree normal direction will be displayed as blue color as shown in Figure 8.

2.3. Non-maximum suppression

Due to the multiple response, edge magnitude M(x,y) may contain wide ridges around the local maxima. Non-maxima suppression removes the non-maxima pixels preserving the connectivity of the contours. For each position (x,y), step in the two directions perpendicular to edge orientation Theta(x,y). And, denote the initial pixel (x,y) by C, the two neighboring pixels in the perpendicular directions by A and B. **(1)** If M(C) ≥ M(A) and M(C) ≥ M(B), discard the pixel of A and B by setting M(A)=0 and M(B)=0. **(2)** Otherwise (if the M(A) > M(C) or M(B) > M(C)), discard the pixel (x,y) by setting M(C)=0.

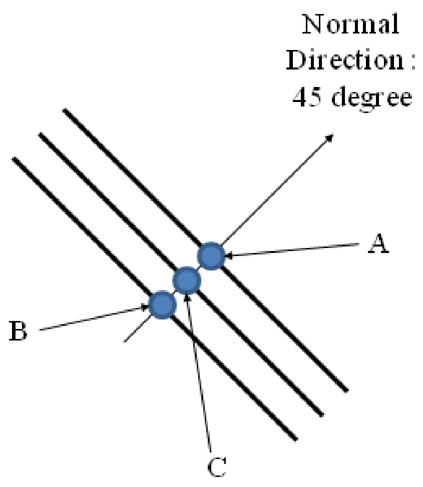


Figure 9 Illustration of the non-maximum suppression.

2.4. Hysteresis thresholding

The output of the non-maxima suppression still contains noisy local maximum. To increase its performance, some of additional approaches can be used. In this lab, hysteresis thresholding method is used. Its contrast (edge strength) may be different in different points of the contour. Thus, careful thresholding of M(x,y) is needed to remove these weak edges while preserving the connectivity of the contours. Hysteresis thresholding receives the output of the non-maximum suppression. This algorithm uses two thresholds, T_high and T_low. **(1)** A pixel (x,y) is called strong if M(x,y) if larger or equal to T_high, **(2)** weak if M(x,y) is smaller or equal to T_low, **(3)** candidate pixels otherwise. In each position of (x,y), discard the pixel (x,y) if it is weak; output the pixel if it is strong. If the pixel is a candidate, it should check two neighbor pixels on its edge directions. If the starting candidate pixel (x,y) is **(3a)** connected to a strong pixel or **(3b)** a candidate pixel which has already been regarded as a strong pixel, output this candidate pixel; otherwise, do not output the candidate pixel. For the step 3a and 3b, you need to use the final result image which has been already processed (The name of the array is regZ in the code given). You should use the threshold values below for this step.

High threshold : 20
Low threshold : 5

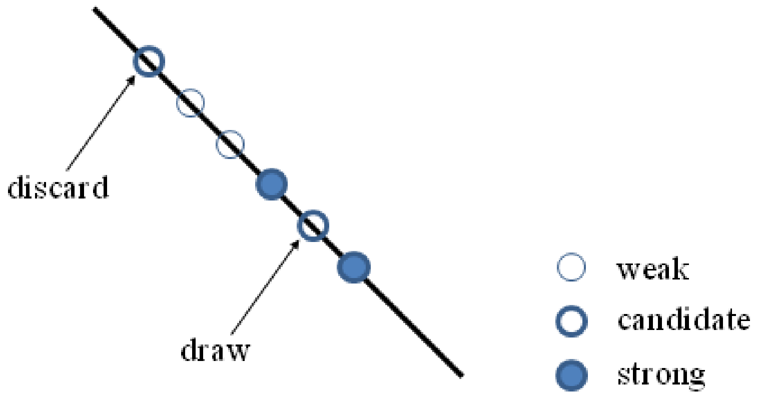


Figure 10 Illustration to the hysteresis thresholding.

3. Structure

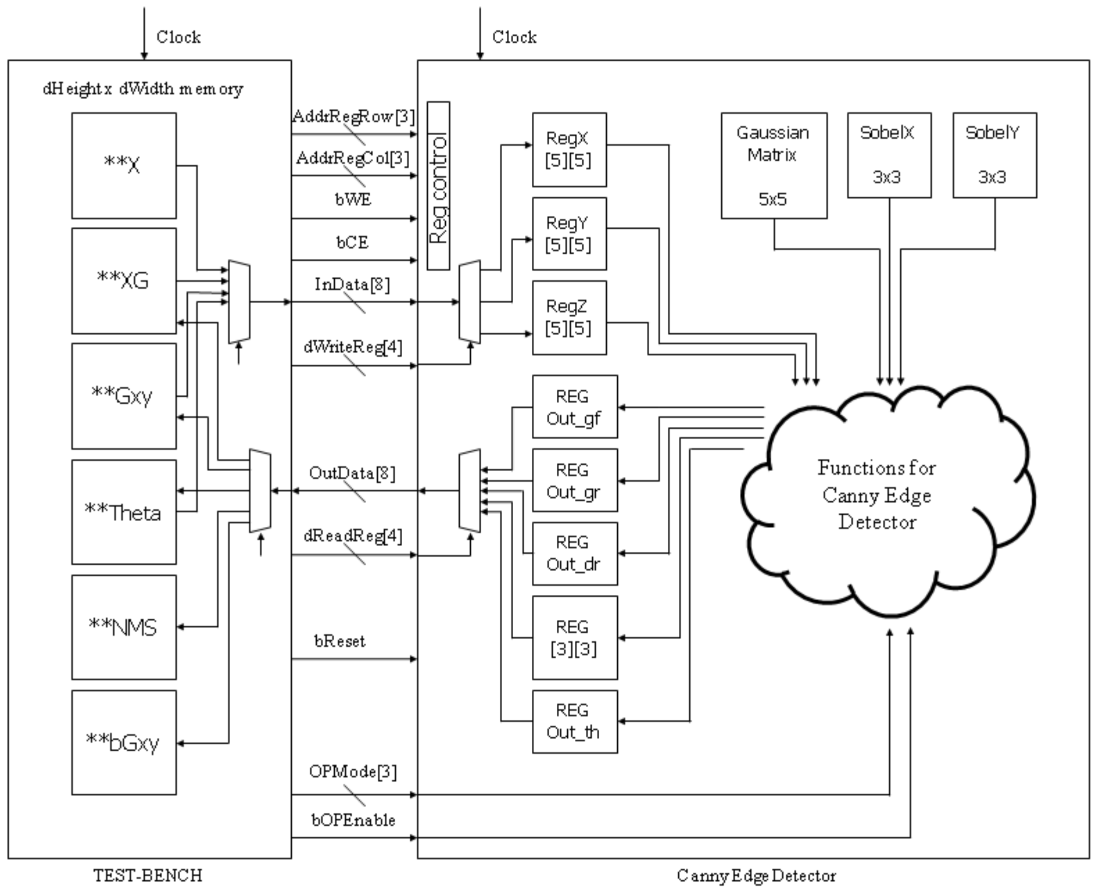


Figure 11 Data Flow of top module and test bench

Figure 11 shows the data flow of top module and test bench.

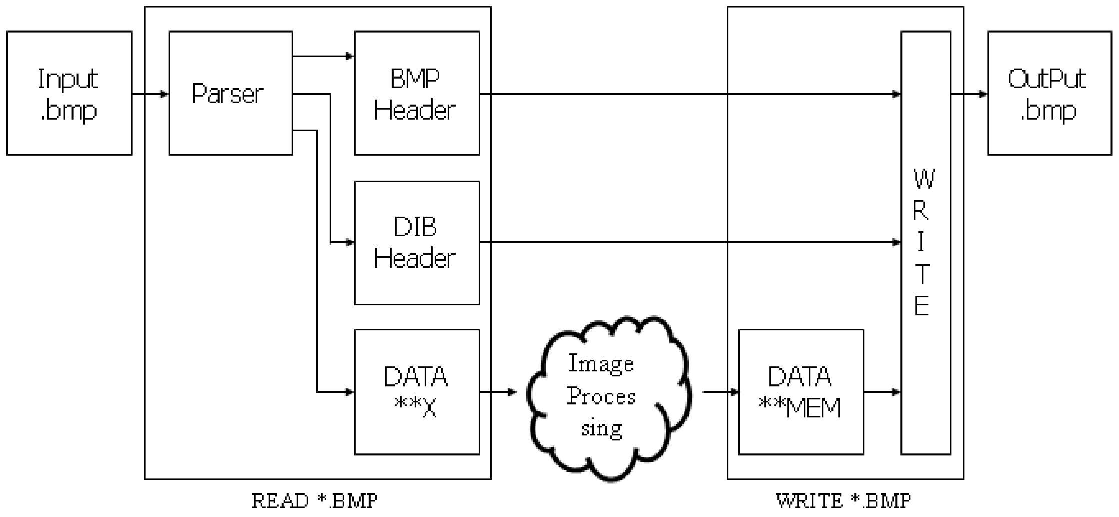


Figure 12 FILE in/out process

Figure 12 shows how to access file in and out. We will use this function to load *.bmp and save *.bmp file while we simulate all of processes whenever you want to write bmp file.

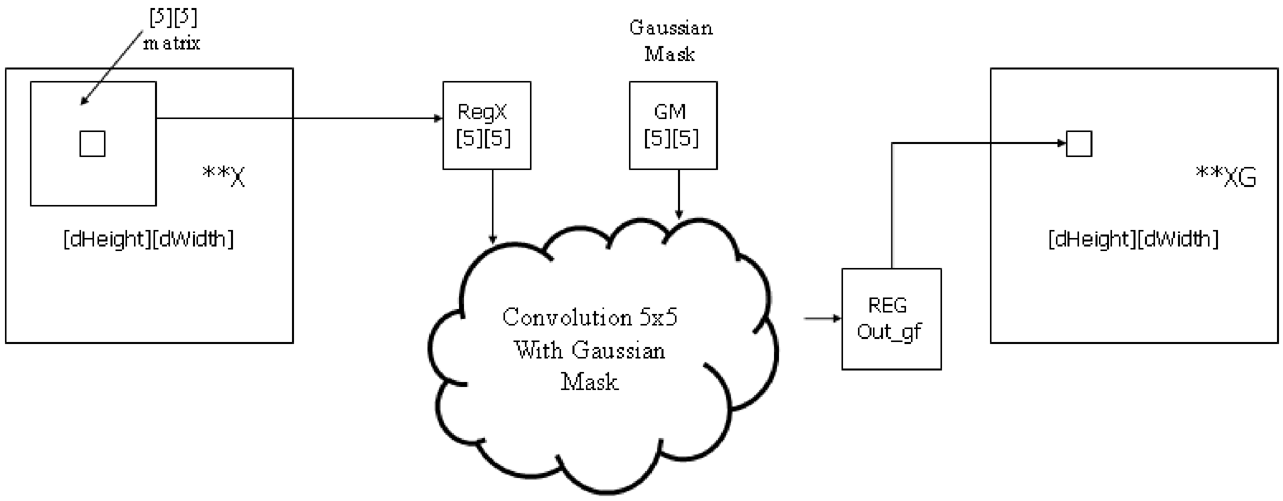


Figure 13 Data flow of Noise Reduction process

Figure 13 shows the data flow of noise reduction process. In the first step, the top module receives 5x5 data of the original image(**X) from the test-bench. And, the array of data is convoluted with Gaussian mask for noise reduction at noise reduction phase. Finally, the data is loaded into register Out_gf, and it goes out to other memory area(**XG) in the test bench.

4. 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 ‘Lab4_code.tar.gz’ file from the lecture website. You will design modules in these files after you decompress the file.

a. Make working folder for this lab.

```
HOME]$ mkdir ECEN468/Lab4/SRC                (make work folder)
HOME]$ cd ECEN468/Lab4/SRC                   (move to work folder)
Move the file ‘Lab4_code.tar.gz’ into the working folder
ECEN468/Lab4/SRC]$ tar xvfz Lab4_code.tar.gz (decompress the file)
```

Please check the files decompressed. It should contain the files below.

Canny_Edge.cpp, Canny_Edge.h, test.h, test.cpp, main.cpp and cman_200.bmp

b. Open Vista and make a project with adding the files. The ‘cman_200.bmp’ is the input file image. Must not include the input image file into the project.

c. Build your project and simulate it. After simulation is completed, you will see the output bmp files in your directory as well as the output message at Vista command window. Please check the bmp files.

Description of the bmp files

1.OutputOrigin.bmp : After loading an input image and storing into local memory, it reads the image from the local memory.
2.OutputGauss.bmp : The image Gaussian filtered
3.OutputGradient.bmp : The gradient components of the image Gaussian filtered
4.OutputAngle.bmp : The direction component based on the gradient dx and dy
5.OutputNMS.bmp : The image after Non-Maximum Suppression process
6.OutputThres.bmp : Final image after Hysteresis Thresholding process

Only first two images(Origin and Gauss images) will be shown correctly. For the remaining processes, you should insert your codes into the Canny_Edge.cpp.

d. Complete the functions Write_Data() and Read_Data() by referring Figure 11.

e. Add your code into the Canny_Edge.cpp for Sobel operation. You may refer to the modules and test-bench for Gaussian filtered image. Please show your result to TA within your regular lab hour to get credit for lab evaluation.

f. Add your code into the Canny_Edge.cpp for NMS. Please show your result to TA within your regular lab hour to get credit for lab evaluation.

g. Add your code into the Canny_Edge.cpp for Hysteresis Thresholding process. Please show your result to TA within your regular lab hour to get credit for lab evaluation.

5. How to evaluate your results.

For your convenience, it will show matching ratio between your results and reference images at the end of simulation. However, please check your results with output images (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.

Requirements:

1. Your design modules should satisfy constraints below.
   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 (input : cman_200.bmp) (10 points)
      (Please refer to the reference output results given and matching ratio at the end of the simulation.)
   b. Canny_Edge.cpp with detailed comments (5 points)
   c. Only the parts modified in your testbench, if you have modified. (2 points)
   d. Question 1. You will see lots of double lines for edges in your final result images if your results are correct. What is the reason? What is a solution to solve this issue? (5 points)
   e. An additional simulation (10 points)
      - Please use different input image (kodim06_200.bmp given) in test.cpp
      - Change two thresholds in Canny_Edge.cpp. (high : 35, low : 1)
      - Add the output images in your reports.
        (Please refer to the reference output results given and matching ratio at the end of the simulation.)
   f. Show your final results to TA. (5 points)

Note : Please send **ONLY Canny_Edge.cpp** to TA's email address. (3 points)