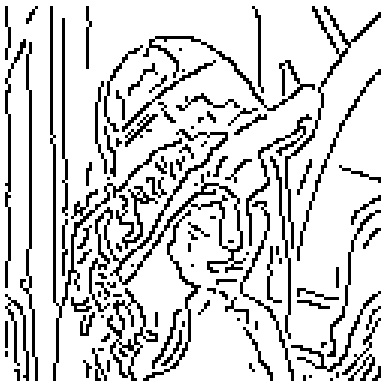
Image Processing
159.731



Canny Edge Detection
Report

Syed Irfanullah, Azeezullah **00297844**
Danh Anh Huynh **02136047**

Canny Edge Detection

INTRODUCTION

Edges

Edges characterize boundaries and are therefore a problem of fundamental importance in image processing. Edges in images are areas with strong intensity contrasts – a jump in intensity from one pixel to the next. Edge detecting an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image.

Canny edge detection algorithm is also known as the optimal edge detector. Canny's intentions were to enhance the many edge detectors in the image.

- The first criterion should have low error rate and filter out unwanted information while the useful information preserve.
- The second criterion is to keep the lower variation as possible between the original image and the processed image.
- Third criterion removes multiple responses to an edge.

Based on these criteria, the canny edge detector first smoothes the image to eliminate noise. It then finds the image gradient to highlight regions with high spatial derivatives. The algorithm then tracks along these regions and suppresses any pixel that is not at the maximum using non-maximum suppression. The gradient array is now further reduced by hysteresis to remove streaking and thinning the edges.

Filter out noise

Convolution

First step to Canny edge detection require some method of filter out any noise and still preserve the useful image. Convolution is a simple mathematic method to many common image-processing operators.

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}	I_{48}	I_{49}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}	I_{58}	I_{59}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}	I_{68}	I_{69}

K_{11}	K_{12}	K_{13}
K_{21}	K_{22}	K_{23}

Figure 1: An example small image (left), kernel (right)

Convolution operation:

Convolution is performed by sliding the kernel or mask over a grey-level image. The kernel starts from the top left corner and moves through entire image within image boundaries. Each kernel position corresponds to a single output pixel. Each pixel is multiplied with the kernel cell value and added together.

Example for pixel value at position 57 below:

$$O_{57} = I_{57}K_{11} + I_{58}K_{12} + I_{59}K_{13} + I_{67}K_{21} + I_{68}K_{22} + I_{69}K_{23}$$

Mathematically we can write:

$$O(i,j) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i+k-1, j+l-1)K(k,l)$$

The output image will have M-m+1 rows and N-n+1 columns, M image rows and N image columns, m kernel rows and n kernel columns.

The output image will be smaller when compared to the original image. This is due to the bottom and right edge pixels, which can't be completely mapped by the kernel therefore m –1 right hand pixels and n-1 bottom pixels can't be used.

Step 1: Gaussian filtering to remove noise

The first step of canny edge detection is to filter out any noise in the original image before trying to locate and detect any edges. The Gaussian filter is used to blur and remove unwanted detail and noise. By calculating a suitable 5 X 5 mask, the Gaussian smoothing can be performed using standard convolution method. A convolution mask is much smaller than the actual image. As a result, the mask is slid over the image, calculating every square of pixels at a time.

Gaussian filter uses 2D distribution to perform convolution. The larger the width of the Gaussian mask, the lower is the detector's sensitivity to noise. The weight of the matrix is concentrated at the centre, therefore any noise appearing in the outside columns and rows will be eliminated, as the weight decreases outward from the centre value. The localization error in the detected edges also increases slightly as the Gaussian width is increased. The increasing of standard deviation reduces or blurs the intensity of the noise.

2D Isotropic Gaussian equation is given below as an example:

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

$$\frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Figure 3 Discrete approximation to Gaussian function with σ=1.4

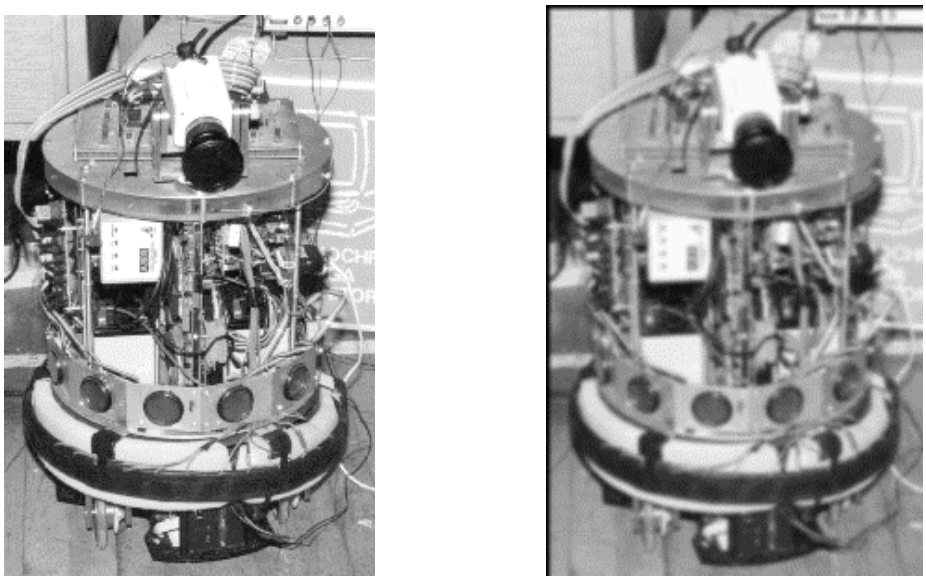


Figure: original image (left), Gaussian filtered image (right)

Step 2: Sobel Operator

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.

The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

Then, the approximate absolute gradient magnitude (edge strength) at each point can be found by the formula below which is simpler to calculate compared to the above exact gradient magnitude.

Approximate gradient magnitude given below:

$$|G| = |Gx| + |Gy|$$

The Sobel operator uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (columns) and the other estimating the gradient in the y-direction (rows).

Sobel Gx and Gy masks shown below each one estimates gradient x direction and y direction respectively:

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

Gx

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

Gy

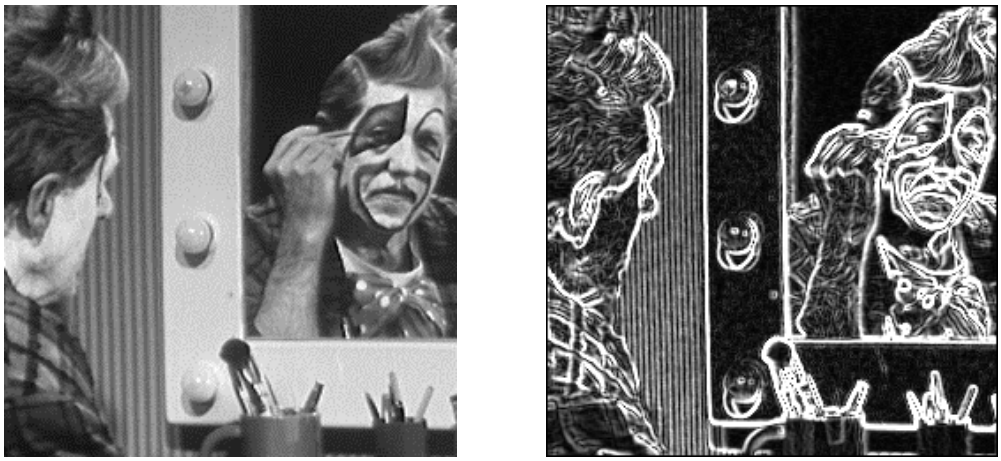


Figure: original image (left), with Sobel operation (right)

Step 3: Finding Gradient angle

Finding the edge direction is trivial once the gradient in the x and y directions are known. However, you will generate an error whenever sum of Gx is equal to zero i.e. Gx value in denominator meaning calculating arctan of infinity. So the edge direction will equal to 90 or 0 degrees depend on Gx value and 0 degrees depend on Gy value.

The formula for finding the edge direction is given below:

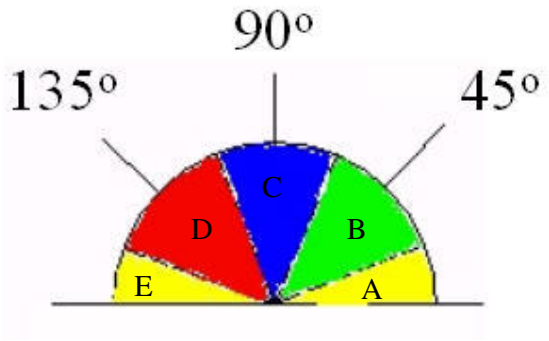
$$\text{theta} = \text{invtan}\ (Gy / Gx)$$

Step 4: Tracing the edge in the image using theta(angle)

Once the edge direction is known, the next step is to relate the edge direction to a direction that can be traced in an image. So if use the 5x5 matrix to calculate the angle of the edge, the smaller the matrix the fewer angles would have in the image.

x	x	x	x	x
x	x	x	x	x
x	x	a	x	x
x	x	x	x	x
x	x	x	x	x

By looking at the centre pixel "**a**", there are four possible directions when describing the surrounding pixels - **0 degrees** (in the horizontal direction), **45 degrees** (along the positive diagonal), **90 degrees** (in the vertical direction), or **135 degrees** (along the negative diagonal), **180 degrees** region is just an mirror region of 0 degrees region. Therefore any edge direction calculated will be round up to the closest angle.

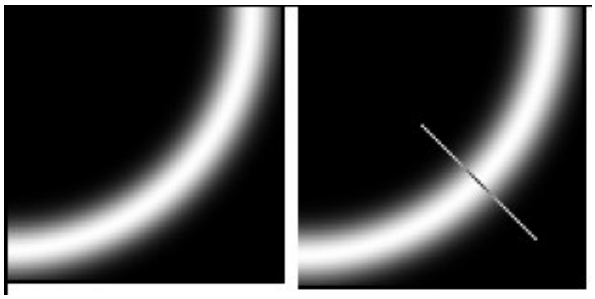


So any edge direction falling within the **A and E** (0 to 22.5 & 157.5 to 180 degrees) is set to 0 degrees. Any edge direction falling in the **D** (22.5 to 67.5 degrees) is set to 45 degrees. Any edge direction falling in the **C** (67.5 to 112.5 degrees) is set to 90 degrees. And finally, any edge direction falling within the **B** (112.5 to 157.5 degrees) is set to 135 degrees.

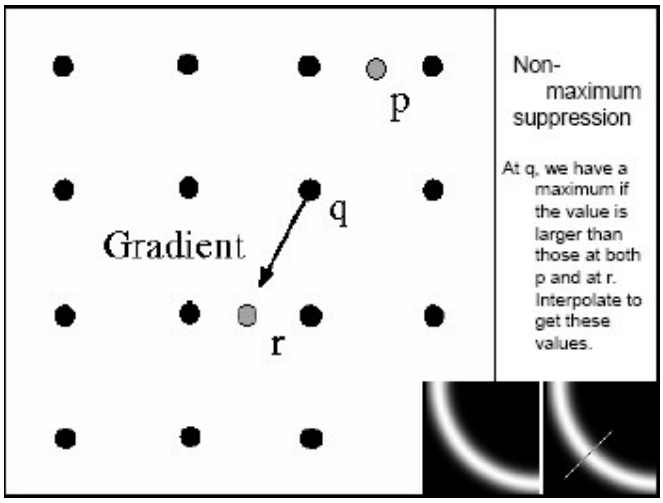
The bigger the matrix the more number of angles one could get, which implies the edge angle will be more precise i.e. will follow the edge better, but on the down side it will be a bigger computation task as now the kernel/mask size is bigger.

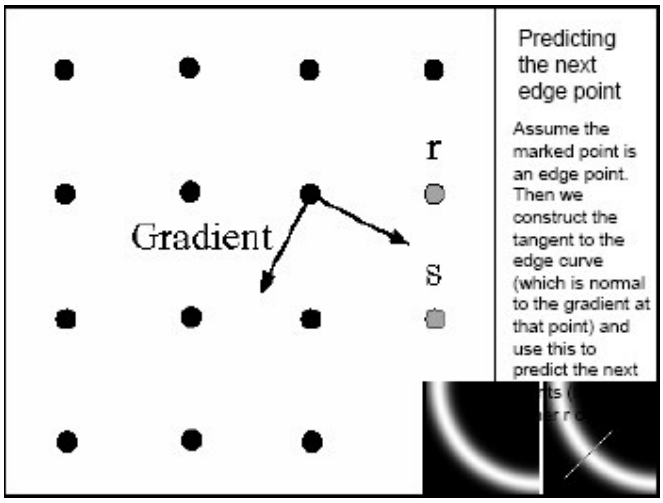
Step 5: Non maximum Suppression

After the edge directions are known, non-maximum suppression is applied. Non-maximum suppression is used to trace along the gradient in the edge direction and compare the value perpendicular to the gradient. Two perpendicular pixel values are compared with the value in the edge direction. If their value is lower than the pixel on the edge then they are suppressed i.e. their pixel value is changed to 0, else the higher pixel value is set as the edge and the other two suppressed with a pixel value of 0.



We wish to mark points along the curve where the magnitude is biggest. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: at which point is the maximum, and where is the next one?

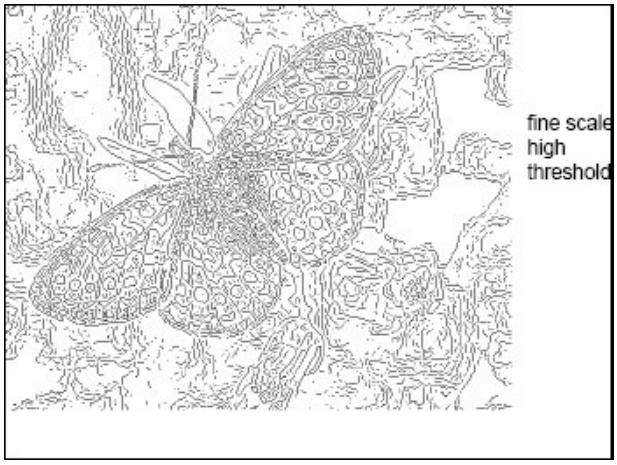


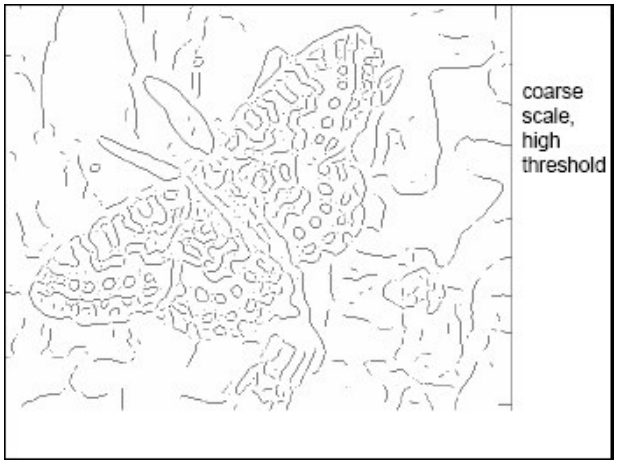


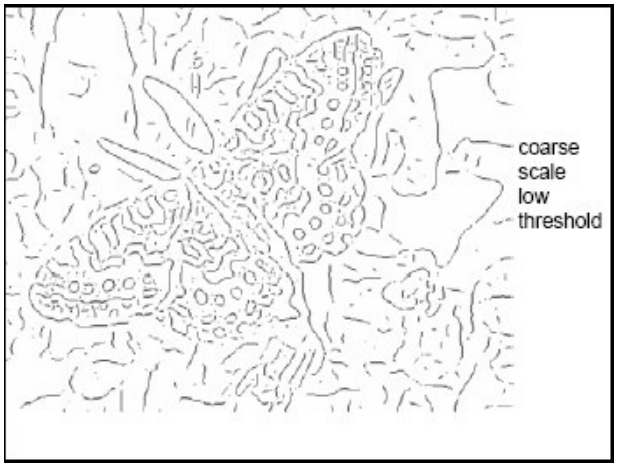
Step 6: Hysteresis

Finally, hysteresis is used as a means of eliminating streaking. Streaking is the breaking up of an edge contour caused by the operator output fluctuating above and below the threshold. If a single threshold, T1 is applied to an image, and an edge has an average strength equal to T1, then due to noise, there will be instances where the edge dips below the threshold. Equally it will also extend above the threshold making an edge look like a dashed line. To avoid this, hysteresis uses 2 thresholds, a high and a low. Any pixel in the image that has a value greater than T1 is presumed to be an edge pixel, and is marked as such immediately. Then, any pixels that are connected to

this edge pixel and that have a value greater than T2 are also selected as edge pixels. If you think of following an edge, you need a gradient of T2 to start but you don't stop till you hit a gradient below T1.

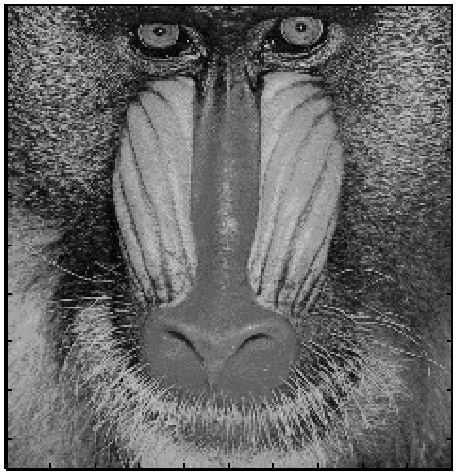




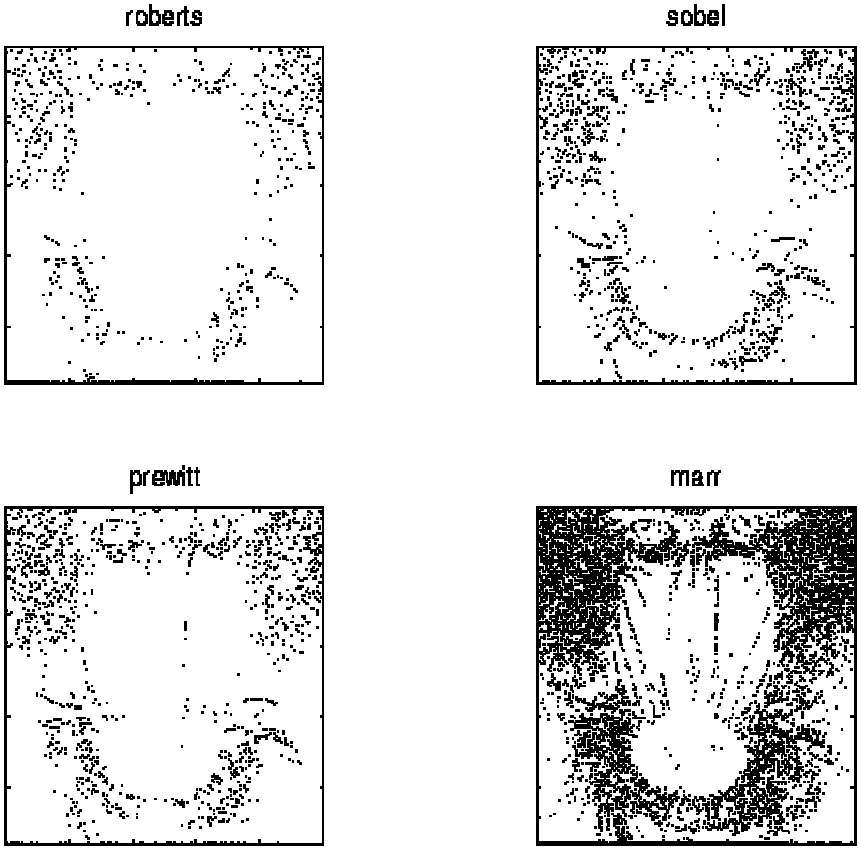


The above output images show the effect of hysteresis thresholding.

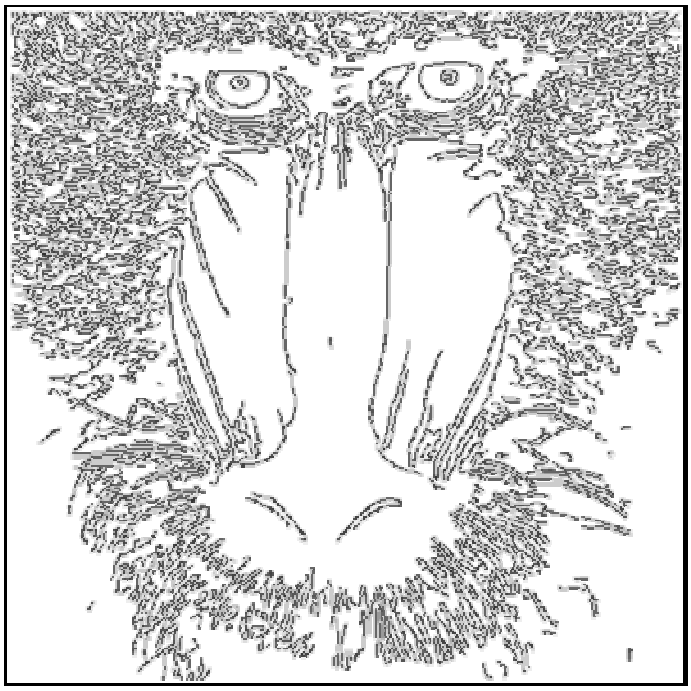
Comparison of different edge detection algorithms



Original Image



Outputs from Roberts, Sobel, Prewitt, Marr edge detection algorithms



Canny edge detection output

Conclusion:

Edge detection applying the Canny edge detection algorithm was discussed. The various steps in performing Canny's algorithm i.e. gaussian, sobel, non-maximum suppression and hysteresis were explained in detail.

References:

1. Classical feature detection
http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/OWENS/LECT6/node2.html

2. Canny interactive demo
http://www.cs.washington.edu/research/imagedatabase/demo/edge/

3. Edge detection tutorial

http://www.pages.drexel.edu/~weg22/edge.html

4. Overview of detection of edges

http://www.cs.wustl.edu/~pless/519/Lecture3.pdf

5. Canny edge detection tutorial

http://www.pages.drexel.edu/~weg22/can_tut.html

6. Convolution

http://www.cee.hw.ac.uk/hipr/html/convolve.html

7. Masks / kernels

http://www.dai.ed.ac.uk/HIPR2/kernel.htm

8. Spatial domains

http://www.dai.ed.ac.uk/HIPR2/spatdom.htm

9. Sobel edge detection algorithm

http://www.dai.ed.ac.uk/HIPR2/sobel.htm

10. Gaussian filtering

http://robotics.eecs.berkeley.edu/~mayi/imgproc/gademo.html