Image Features (II)

Dr. Gerhard Roth

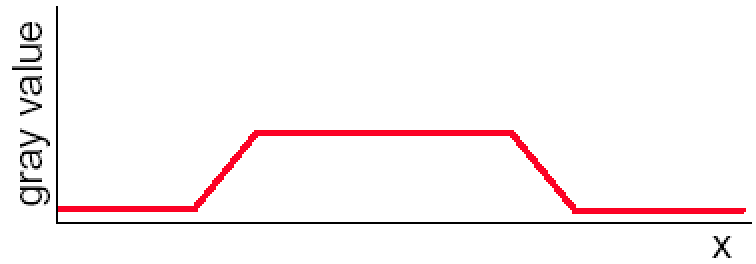
COMP 4102A

Winter 2014

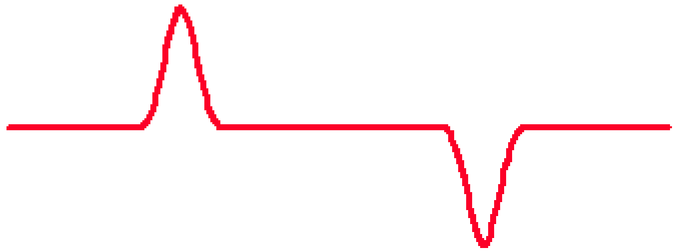
Version 2

Edge Detection using Derivatives

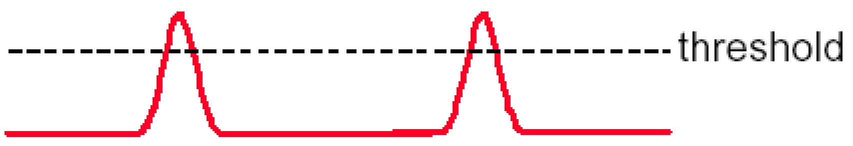
1-D image $f(x)$



1^{st} derivative $f'(x)$



$|f'(x)|$ threshold



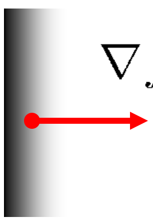
Pixels that pass the threshold are edge pixels

Image gradient

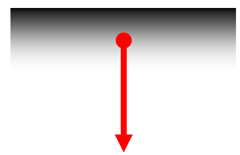
The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

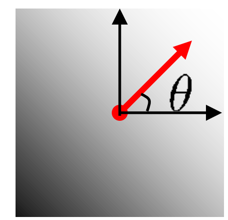
The gradient points in the direction of most rapid change in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient direction is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

- Gradient direction is along the normal to the edge

The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Finite Difference for Gradient

Discrete approximation:

$$I_x(i,j) = \frac{\partial f}{\partial x} \approx f_{i+1,j} - f_{i,j}$$

$$I_y(i,j) = \frac{\partial f}{\partial y} \approx f_{i,j+1} - f_{i,j}$$

Convolution kernels:

$$\begin{bmatrix} -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Magnitude (strength) $G(i,j) = \sqrt{I_x^2(i,j) + I_y^2(i,j)}$

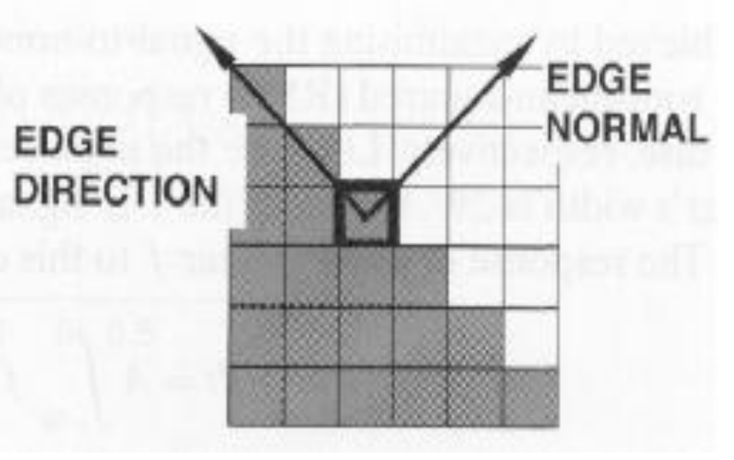
aprox. magnitude $G(i,j) \approx |I_x| + |I_y|$

direction $\arctan(I_y / I_x)$

Edge Detection Using the Gradient

Properties of the gradient:

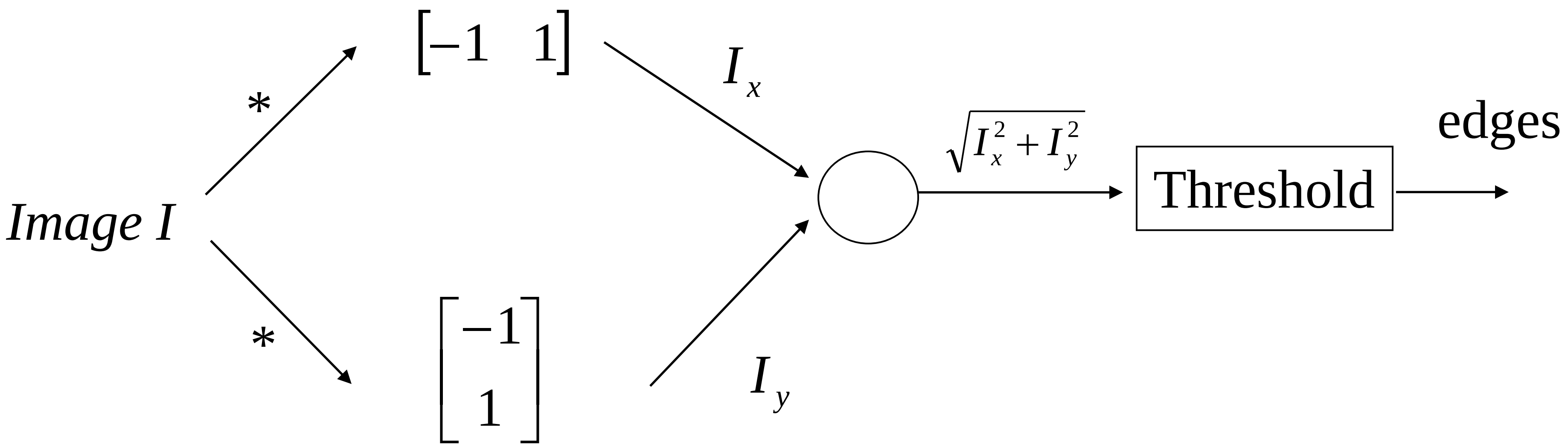
- The magnitude of gradient provides information about the strength of the edge
- The direction of gradient is always perpendicular to the direction of the edge



Simple edge detector:

- Compute derivatives in x and y directions (Edge enhancement)
- Find gradient magnitude (Edge localization)
- Threshold gradient magnitude (Edge localization)

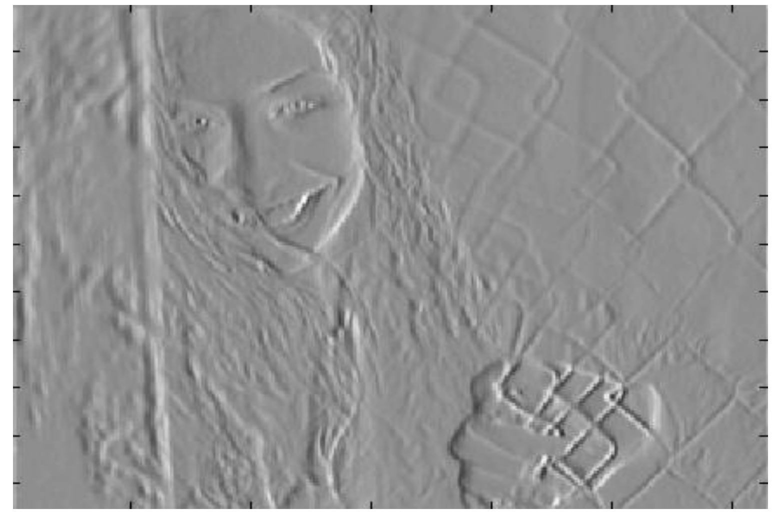
Edge Detection Algorithm



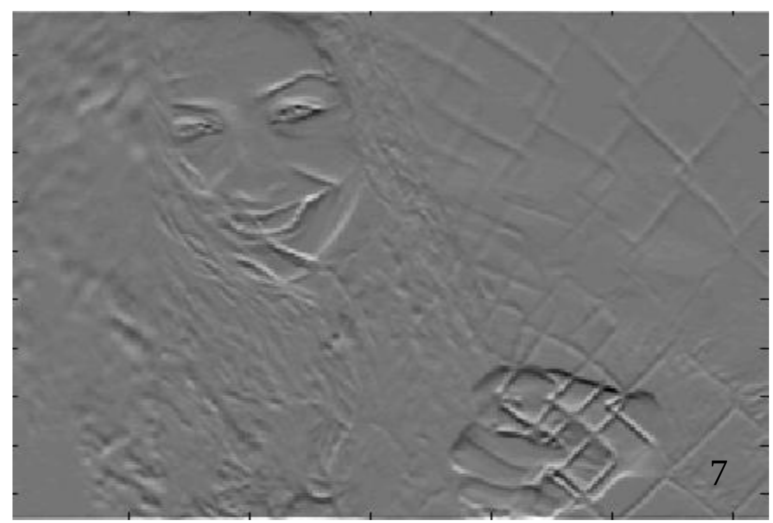
Edge Detection Example

I

I_x



I_y



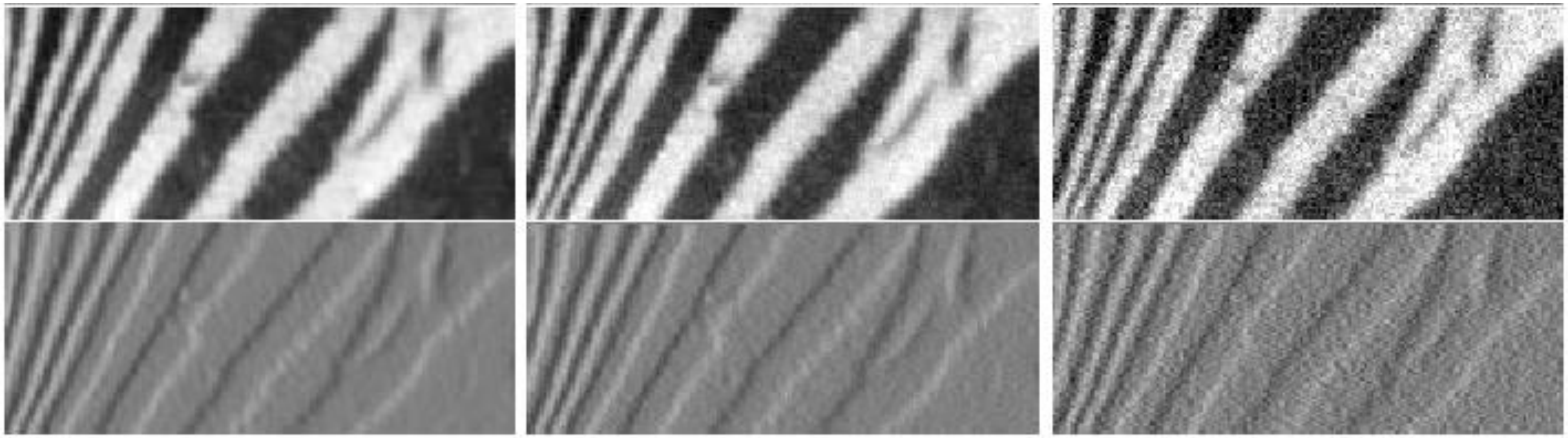
Edge Detection Example

$$G(i,j)=\sqrt{I_x^2(i,j)+I_y^2(i,j)}$$

I

$$G(i,j) > Threshold = \tau$$

Finite differences responding to noise



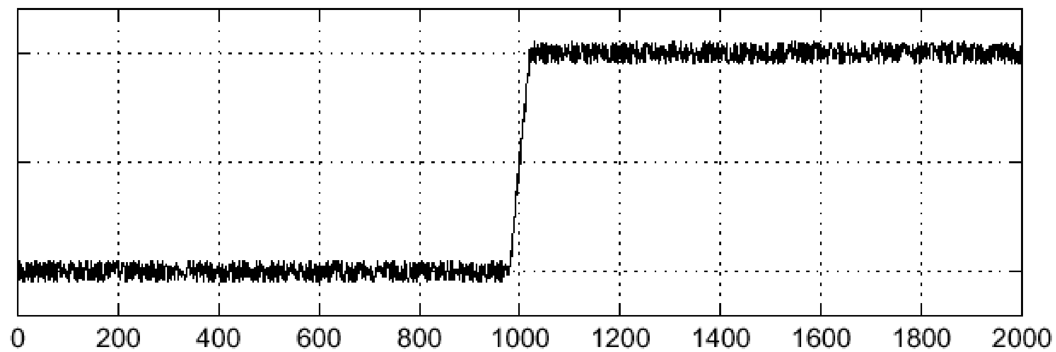
Increasing noise ->
(this is zero mean additive gaussian noise)

Effects of noise

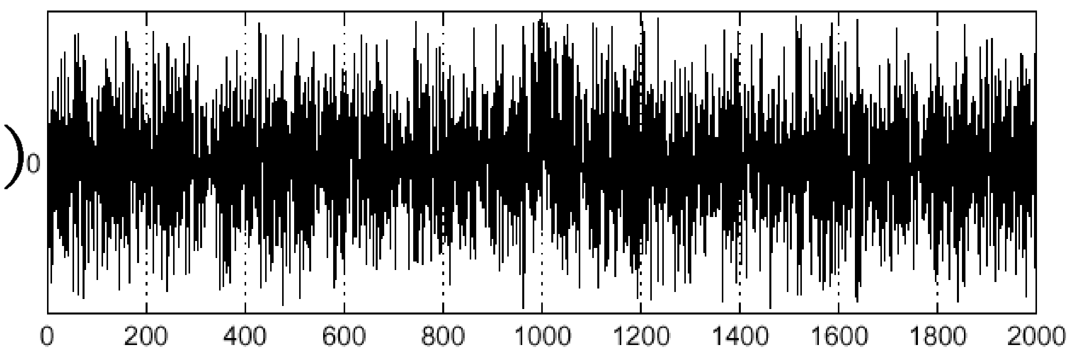
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

$f(x)$



$\frac{d}{dx}f(x)$



Where is the edge?

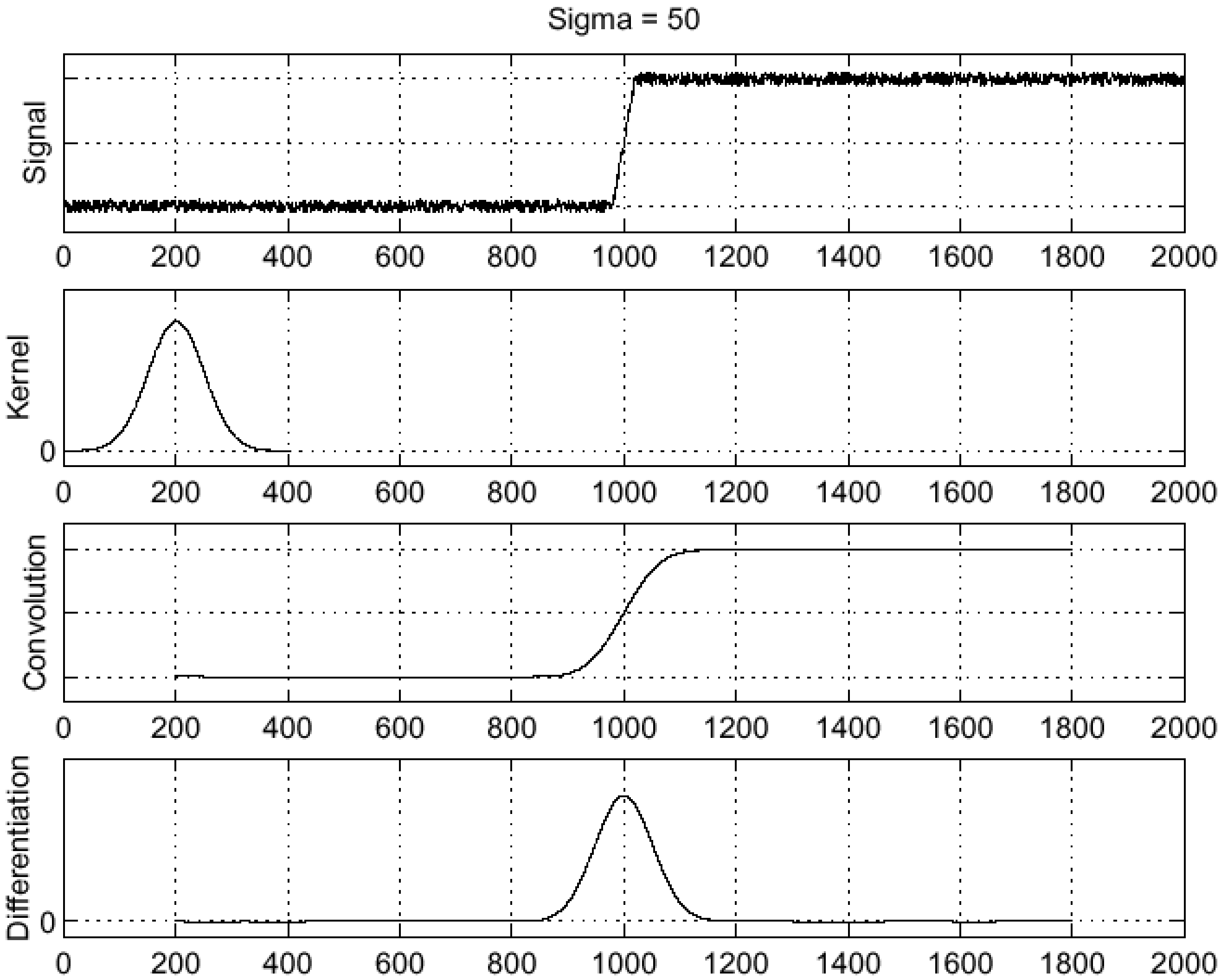
Solution: smooth first

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



Where is edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Sobel Edge Detector (smooth and diff.)

Approximate derivatives with central difference

Convolution kernel

$$I_x(i,j) = \frac{\partial f}{\partial x} \approx f_{i-1,j} - f_{i+1,j}$$

$$\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

Smoothing by adding 3 column neighbouring differences and give more weight to the middle one

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

Convolution kernel for I_y

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

Sobel Operator Example

$$\begin{array}{|c|c|c|}\hline a_1 & a_2 & a_3 \\ \hline a_4 & \boxed{a_5} & a_6 \\ \hline a_7 & a_8 & a_9 \\ \hline \end{array} * \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

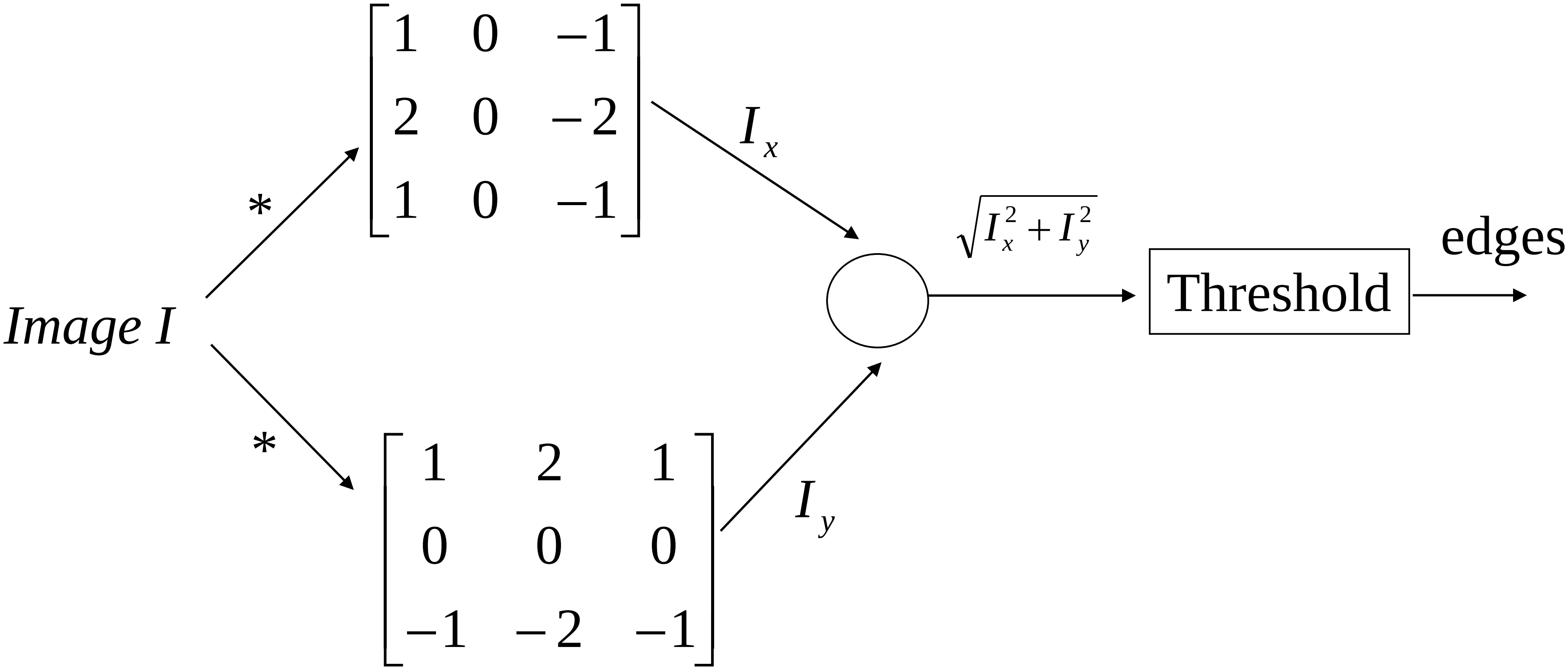
$$\begin{array}{|c|c|c|}\hline a_1 & a_2 & a_3 \\ \hline a_4 & \boxed{a_5} & a_6 \\ \hline a_7 & a_8 & a_9 \\ \hline \end{array} * \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

The approximate gradient at a_5

$$I_x = (a_1 - a_3) + 2(a_4 - a_6) + (a_7 - a_9)$$

$$I_y = (a_1 - a_7) + 2(a_2 - a_8) + (a_3 - a_9)$$

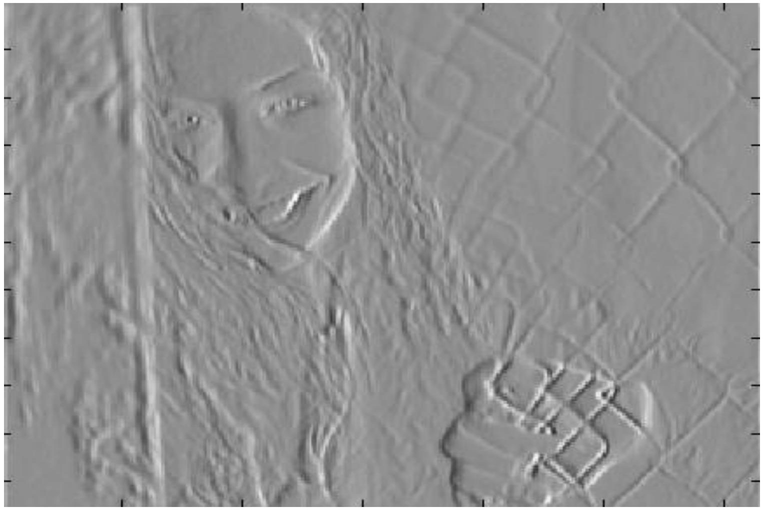
Sobel Edge Detector



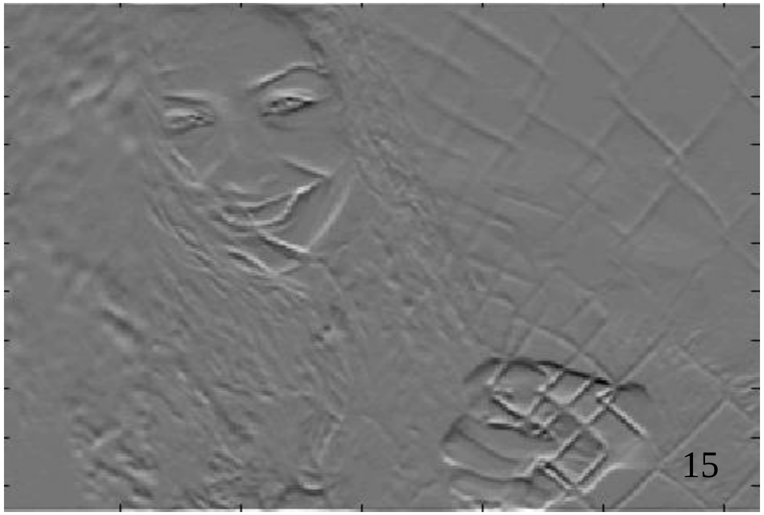
Sobel Edge Detector

I

$\frac{d}{dx} I$



$\frac{d}{dy} I$



Sobel Detector

$$G(x,y) = \sqrt{S_x^2 + S_y^2}$$

I

$$G(x,y) \geq Threshold = 25$$

Simple Edge Detection Summary

Input: an image I and a threshold τ.

1. Noise smoothing: $I_s = I * h$
   (e.g. h is a Gaussian kernel)

2. Compute two gradient images I_x and I_y by convolving I_s with gradient kernels (e.g. Sobel operator).

3. Estimate the gradient magnitude at each pixel

$$G(i, j) = \sqrt{I_x^2(i, j) + I_y^2(i, j)}$$

4. Mark as edges all pixels (i, j) such that $G(i, j) > \tau$

Evaluating Edge Detectors

- Want good detection
  - Minimize false positives (spurious edge points) caused by noise and also false negatives (missing edge points)
  - Find all the edges!
- Want good localization
  - The edges detected should be as closely as possible to the true edges
  - Find the edges accurately!
- Use these two criteria to evaluate edge detectors
- Detection and localization are traded off against each other (hard to have both good)

Canny Edge Detection

- Simple edge detection has poor localization
  - Thresholding gradient image produces many edge pixels
  - Edges are therefore wider than one pixel (undesirable)
  - Reason for this is that the localization step is simple thresholding of the gradient image
- Canny edge detector has very similar smoothing and enhancement but localizes by
  - Thinning edges to one pixel by non-maxima suppression
  - Uses hysteresis thresholding instead of simple thresholding
- Is optimal for a simple edge and noise model
  - And is very commonly used (a function in Opencv)
- Links edge pixels together into contours

Canny Smoothing and Enhancement

Algorithm CANNY_ENHANCER

The input is I, an intensity image corrupted by noise. Let G be a Gaussian with zero mean and standard deviation σ.

[5] This can be done for any edge class, not only step edges. All integrals evaluated using the step edge model change; see Canny's original article (Further Readings) for details.

[6] And 90% of the single-response criterion, although this figure is rather complicated to achieve.

[7] Notice that this algorithm *implements edge descriptors through a set of images*. An alternative would be to define a data structure gathering all the properties of an edge.

1. Apply Gaussian smoothing to I (algorithm LINEAR_FILTER of Chapter 3 with a Gaussian kernel discretising G), obtaining $J = I * G$.

2. For each pixel (i, j):

(a) compute the gradient components, J_x and J_y (Appendix, section A.2);

(b) estimate the edge strength

$$e_s(i, j) = \sqrt{J_x^2(i, j) + J_y^2(i, j)}$$

(c) estimate the orientation of the edge normal

$$e_o(i, j) = \arctan \frac{J_y}{J_x}$$

The output is a *strength image*, E_s, formed by the values $e_s(i, j)$, and an *orientation image*, E_o, formed by the values $e_o(i, j)$.

Smoothing to Reduce Noise

(a) Original

(b) Smoothed

Non-maxima suppression

- Edge strength is gradient magnitude and edge orientation is gradient orientation

The input is the output of CANNY_ENHANCER, that is, the edge strength and orientation images, E_s and E_o. Consider the four directions $d_1 \ldots d_4$, identified by the 0°, 45°, 90° and 135° orientations (with respect to the horizontal axis image reference frame).

For each pixel (i, j):

1. find the direction, $\hat{d}_k$, which best approximates the direction $E_o(i, j)$ (the normal to the edge);

2. if $E_s(i, j)$ is smaller than at least one of its two neighbors along $\hat{d}_k$, assign $I_N(i, j) = 0$ (suppression); otherwise assign $I_N(i, j) = E_s(i, j)$.

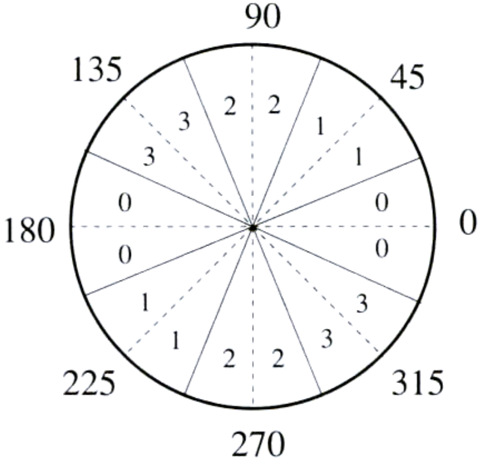
The output is an image, $I_N(i, j)$, of the thinned edge points (that is, $E_s(i, j)$ after suppressing nonmaxima edge points).

Gradient Orientation

Reduce angle of Gradient θ[i,j] to one of the 4 sectors

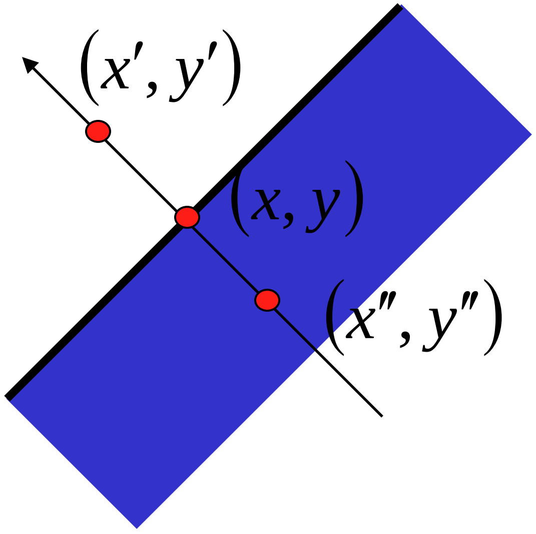
Check the 3x3 region of each G[i,j]

If the value at the center is not greater than the 2 values along the gradient, then G[i,j] is set to 0



Non-Maximum Suppression

Suppress the pixels in 'Gradient Magnitude Image' which are not local maximum



$$G(x,y) = \begin{cases} G(x,y) & \text{if } G(x,y) >= G(x',y') \\ & \& \, G(x,y) >= G(x'',y'') \\ 0 & \text{otherwise} \end{cases}$$

(x',y') and (x'',y'') are the neighbors of (x,y) in G along the direction normal to an edge

Non-Maxima Suppression

Thin edges by keeping large values of Gradient

- not always at the location of an edge
- there are many thick edges

0	0	0	0	1	1	1	3
0	0	0	1	2	1	3	1
0	0	2	1	2	1	1	0
0	1	3	2	1	1	0	0
0	3	2	1	0	0	1	0
2	3	2	0	0	1	0	1
2	3	2	0	1	0	2	1

Non-Maxima Suppression

0	0	0	0	1	1	1	3
0	0	0	1	2	1	3	1
0	0	2	1	2	1	1	0
0	1	3	2	1	1	0	0
0	3	2	1	0	0	1	0
2	3	2	0	0	1	0	1
2	3	2	0	1	0	2	1

local maxima

removed

equal so stays

Non-Maxima Suppression

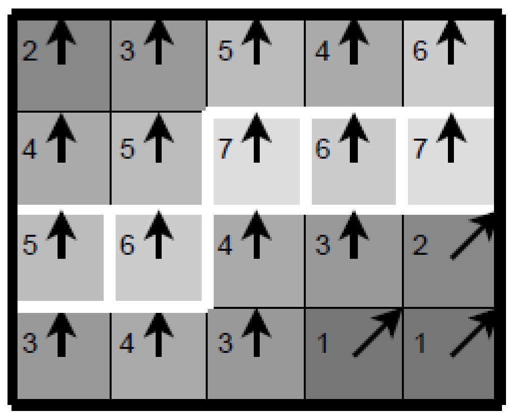
0	0	0	0	0	0	0	3
0	0	0	0	2	1	3	0
0	0	2	1	2	0	0	0
0	0	3	0	0	0	0	0
0	3	2	0	0	0	0	0
0	3	0	0	0	1	0	1
0	3	0	0	1	0	2	0

false edges

The suppressed magnitude image will contain many false edges caused by noise or fine texture

Non-Maxima Suppression

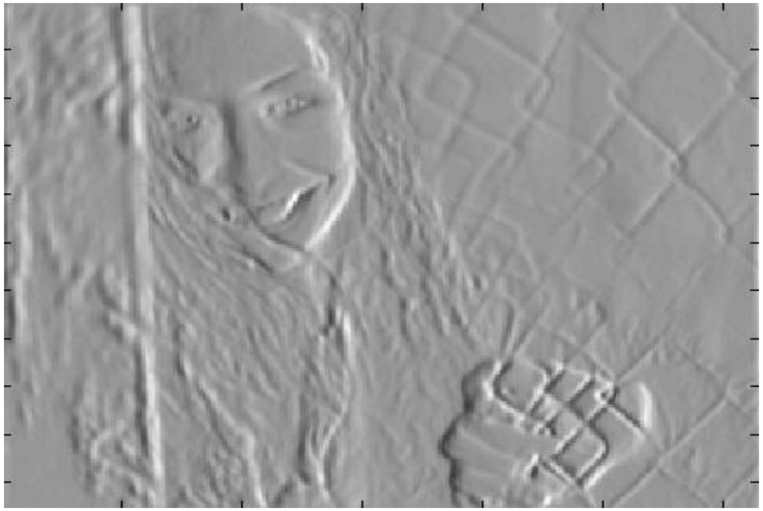
- Edge strength and direction show for each of the edge pixels
- The white pixels are what remain, the rest have their gradient magnitude set to zero



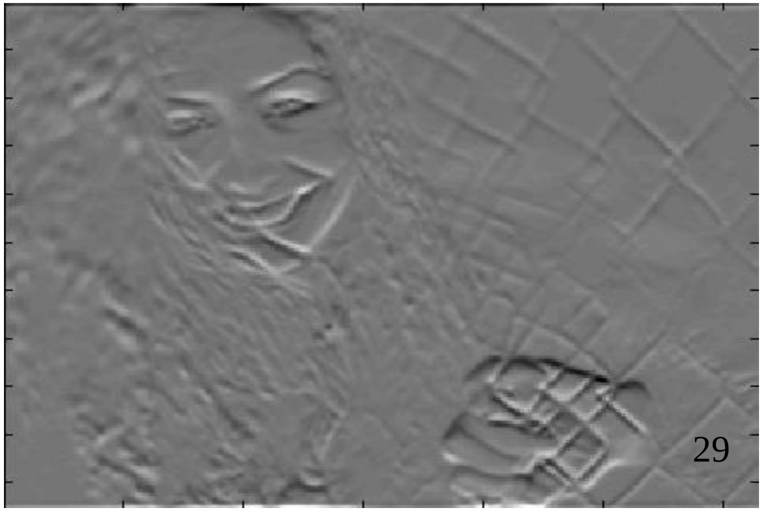
Canny Edge Detector

I

S_x



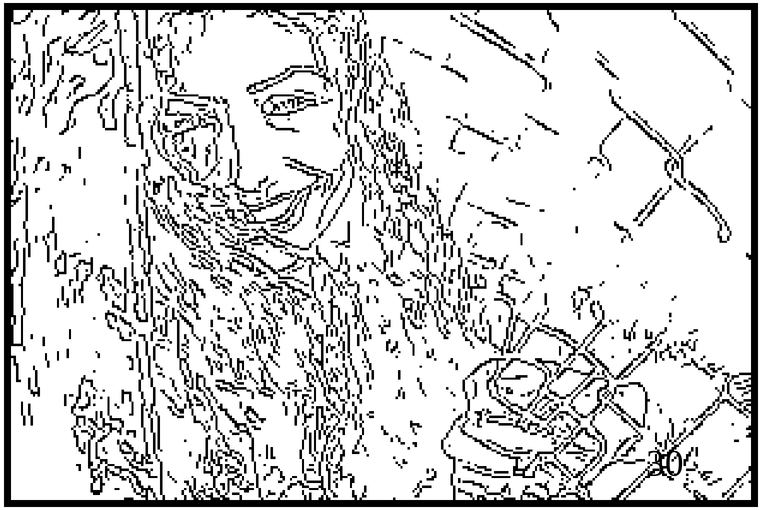
S_y



Non-Maximum Suppression

$$|\Delta S| = \sqrt{S_x^2 + S_y^2}$$

M



$$M \geq Threshold = 25$$

Non-Maxima Suppression

Original image

Gradient magnitude

Non-maxima suppressed

Non-Maxima Suppression

(a) Smoothed

(b) Gradient magnitudes

Suppressed edges as function of σ_f

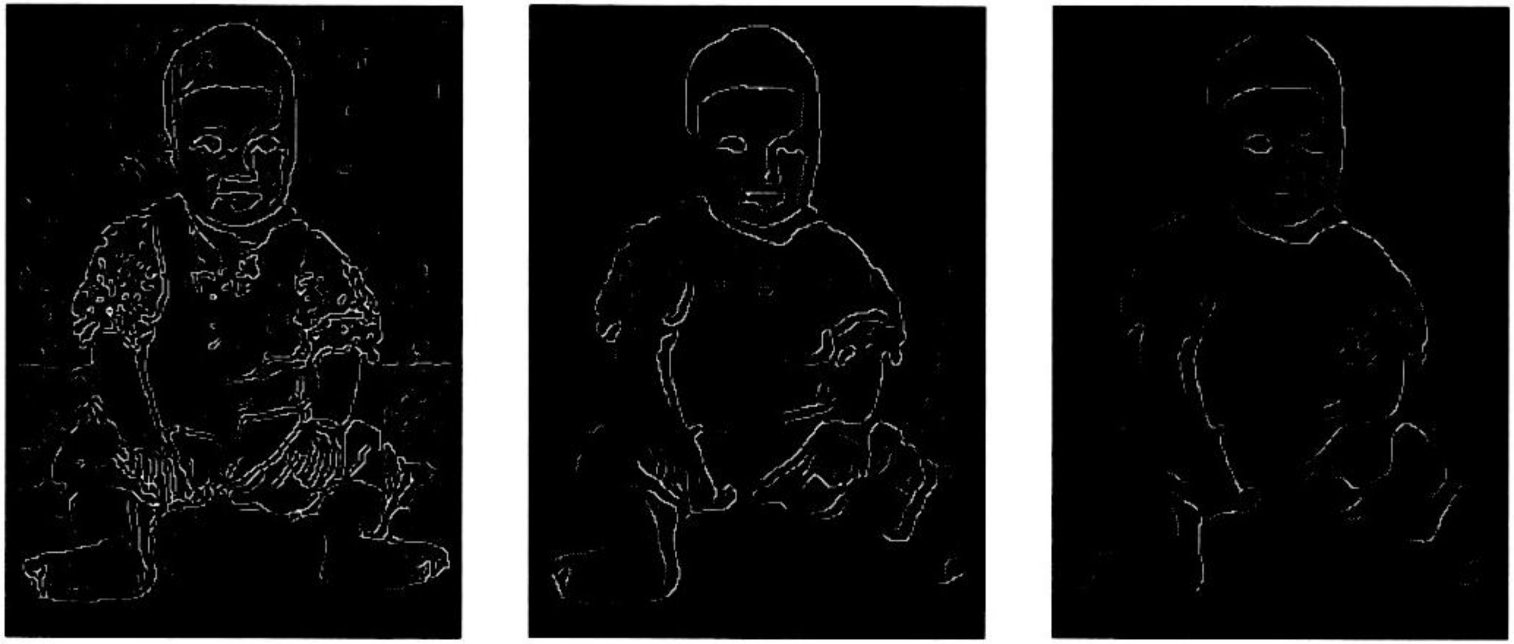


Figure 4.5 Strength images output by CANNY_ENHANCER run on Figure 4.1, after nonmaximum suppression, showing the effect of varying the filter's size, that is, the standard deviation, σ_f, of the Gaussian. Left to right: $\sigma_f = 1, 2, 3$ pixel.

Hysteresis Thresholding

- Hysteresis = where two or more physical quantities bear a relationship that depends on prior history
- Have two thresholds for edges (lower and higher)
    - Once we start an edge chain with higher threshold then continue as long as edge pixels pass lower threshold
- Tends to jump over weaker edges and make better and longer edge chains
- Use hysteresis thresholding on edge pixels that pass non-maxima suppression test

Hysteresis Thresholding

If the gradient at a pixel is above ‘High’, declare it an ‘edge pixel’

If the gradient at a pixel is below ‘Low’, declare it a ‘non-edge-pixel’

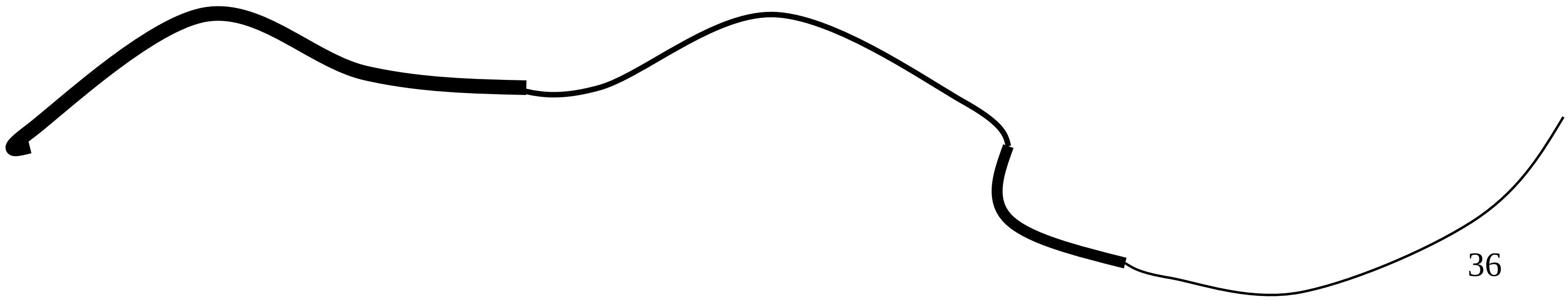
If the gradient at a pixel is between ‘Low’ and ‘High’ then declare it an ‘edge pixel’ if and only if it is connected to an ‘edge pixel’ directly or via pixels between ‘Low’ and ‘ High’

Edge pixels are linked into contours, which are connected edge pixels that pass this test

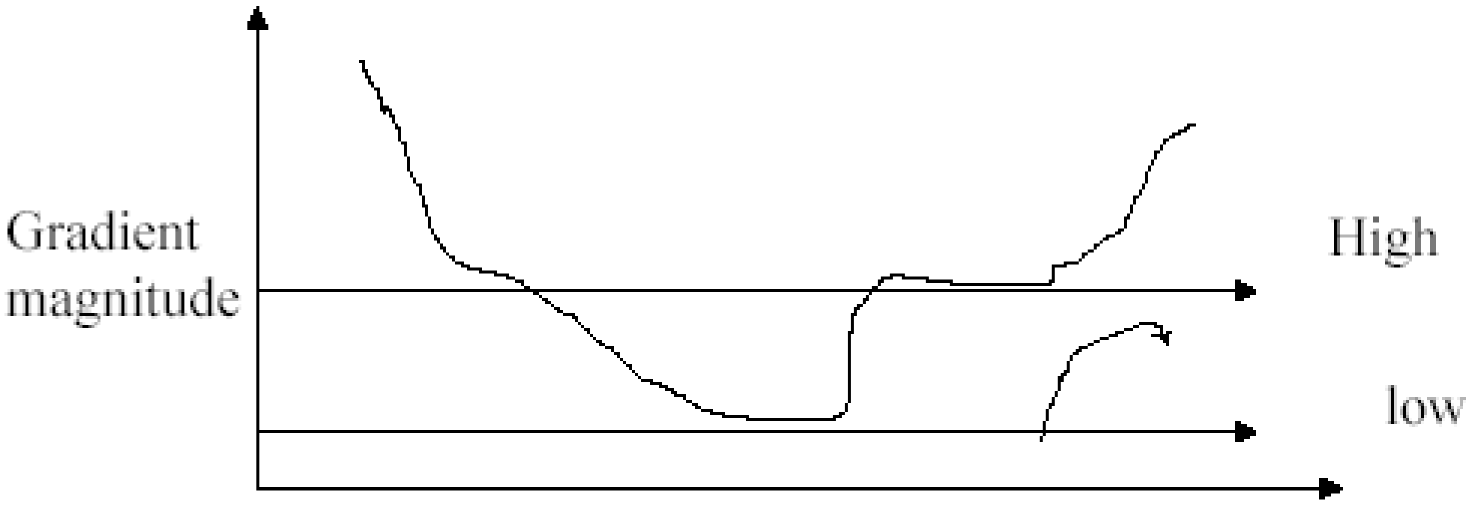
Hysteresis

Check that maximum value of gradient value is sufficiently large

- drop-outs? use **hysteresis**
  - use a high threshold to start edge curves and a low threshold to continue them.



Hysteresis Thresholding



T_2=2 (High threshold) T_1=1 (Low Threshold)

0	0	0	0	0	0	0	3
0	0	0	0	2		3	0
0	0	2		2	0	0	0
0	0	3	0	0	0	0	0
0	3	2	0	0	0	0	0
0	3	0	0	0	0	0	0
0	3	0	0	0	0	2	0

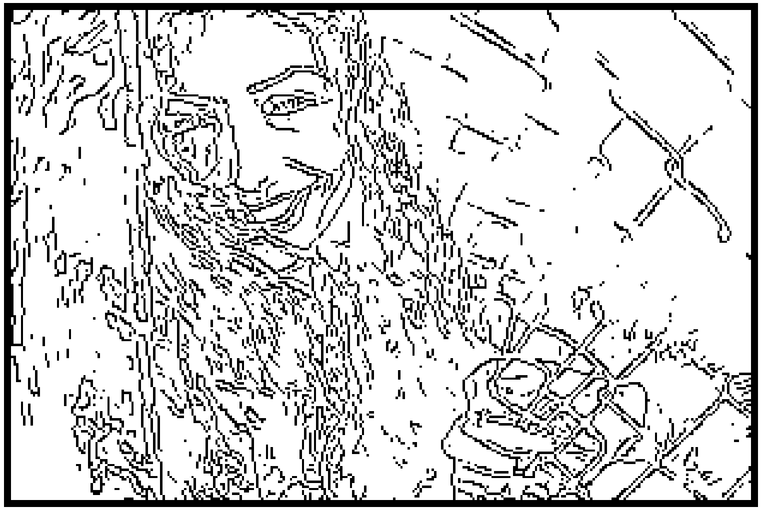
gaps filled from T_1

0	0	0	0	0	0	0	3
0	0	0	0	2	1	3	0
0	0	2	1	2	0	0	0
0	0	3	0	0	0	0	0
0	3	2	0	0	0	0	0
0	3	0	0	0	1	0	1
0	3	0	0	1	0	2	0

- A T_2 contour has pixels along the green arrows
- Linking: search in a 3x3 of each pixel and connect the pixel at the center with the one having greater value
- Search in the direction of the edge (direction of Gradient)

Hysteresis Thresholding

M



$M \geq Threshold = 25$

Keep the important long edges, removes noisy small edges

$High = 35$

$Low = 15$

Hysteresis Thresholding

gap is gone

Original image

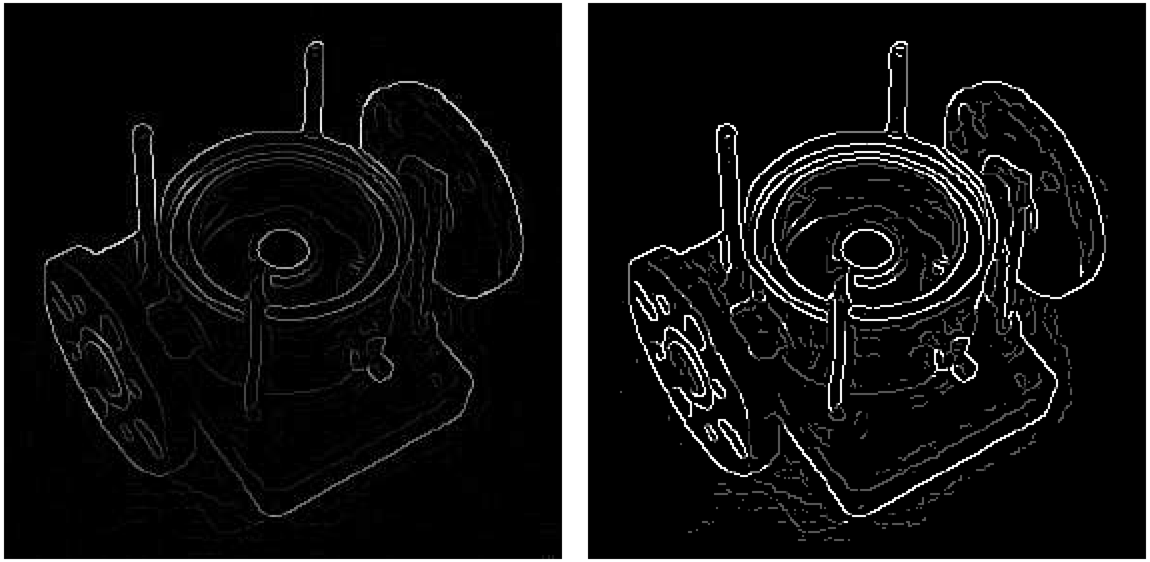
Strong + connected weak edges

Strong edges only

Weak edges

Hysteresis Thresholding

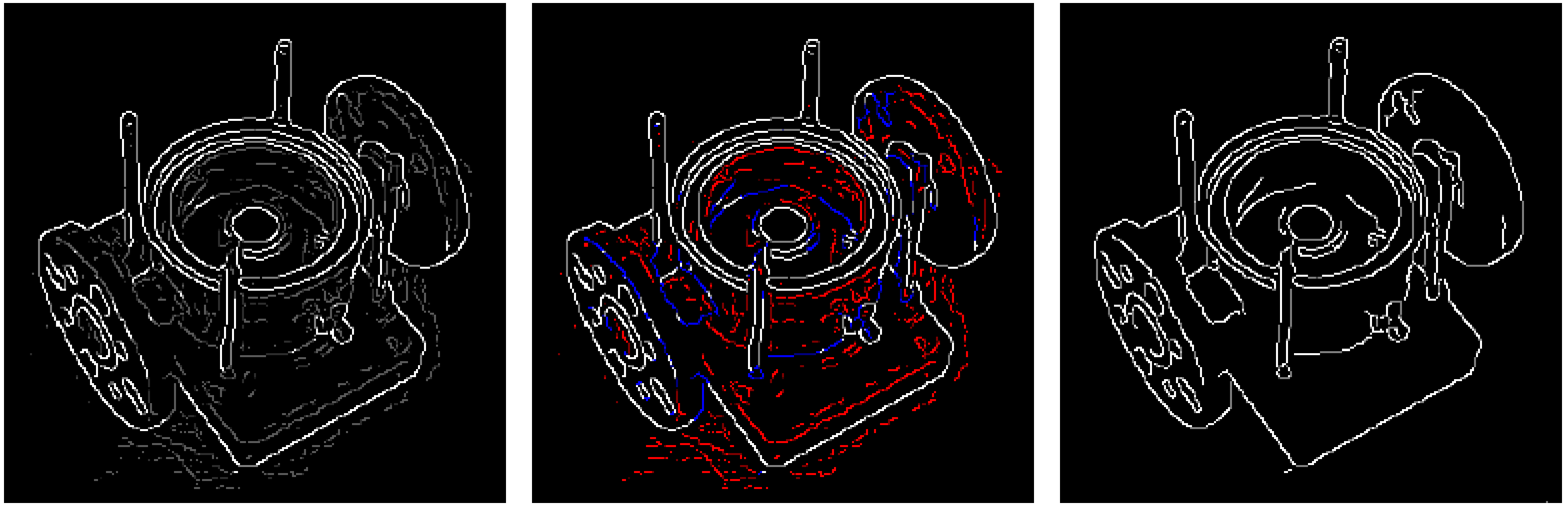
- Strong edges in white (higher threshold)
- Weak edges in blue (lower threshold)



(a) Edges after non-maximum suppression

(b) Double thresholding

Hysteresis Thresholding



(a) Double thresholding (b) Edge tracking by hysteresis (c) Final output

Figure 7: *Edge tracking and final output. The middle image shows strong edges in white, weak edges connected to strong edges in blue, and other weak edges in red.*

The Canny edge detector

original image (Lena)

The Canny edge detector

norm of the gradient

The Canny edge detector

thinning

(non-maximum suppression)

The Canny edge detector

Final edges

Scale of the edge detector

- Remember that first step for canny edge detectors is a smoothing step by a Gaussian with a sigma?
  - Most edge detectors smooth with a given size kernel
- Smoothing eliminates noise, makes edges smoother and removes fine detail
- This sigma (std. deviation of the Gaussian smoothing kernel) is called the scale
- We can choose any scale we want
  - large scale removes small details, but shifts edges slightly
  - small scale has more details and less edge shift

Localization detection/tradeoff

- To evaluate an edge detector we consider the detection and localization capabilities
  - Can we detect only important edges and do it accurately
- We can adjust filter spatial scale (sigma of Gaussian smoothing) but we can not simultaneously improve detection and localization
  - Larger scale improves detection (less noise) but makes localization worse
  - Smaller scale improves localization (more accurate) but makes detection worse
- This is a fundamental tradeoff!

Effect of varying scale (Gaussian size)

Larger Gaussian means better edges but they are slightly shifted and distorted

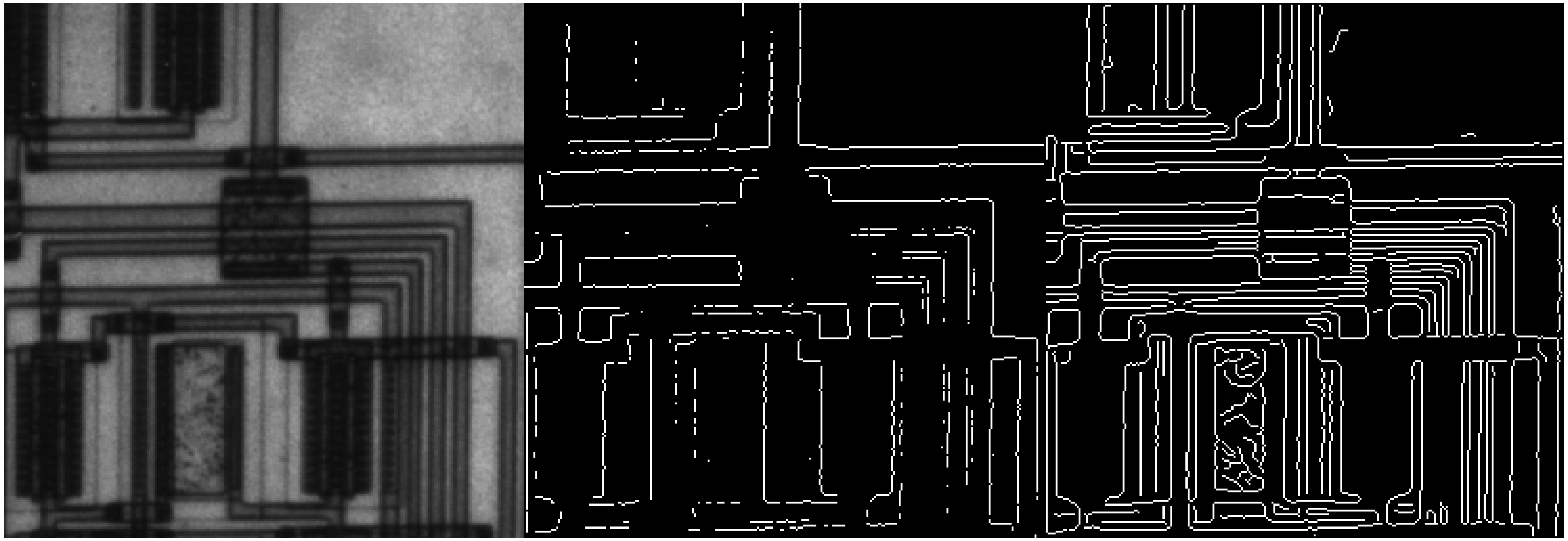
Comparing Edge Detectors

- Want good detection
  - Minimize false positives (spurious edge points) caused by noise and also false negatives (missing edge points)
  - Want to find all the important edges and no more!
- Want good localization
  - The edges detected should be as close as possible to the true edges
  - Want to find the location of the important edges accurately!
- Use these two criteria to evaluate edge detectors

Edge Contour following

- Edge contours are sequences of connected edge pixels
  - By connected we mean they are pixel neighbours
- Canny edges then do contour following
  - Can find sequences of connected edges (all of them)
  - Opencv routine findcontours (very useful)
- Then we can go through each contour
  - Many different features can be extracted for each contour
  - Simple examples are #pixels, smallest bounding circle, etc.
  - Can also fit a polygon to contours within a certain max. error
  - Opencv routine approxpolydp can do this for given contour
  - Contours are very useful features for classification
  - But they are not always completely visible (occlusions)

Comparison of edge detectors

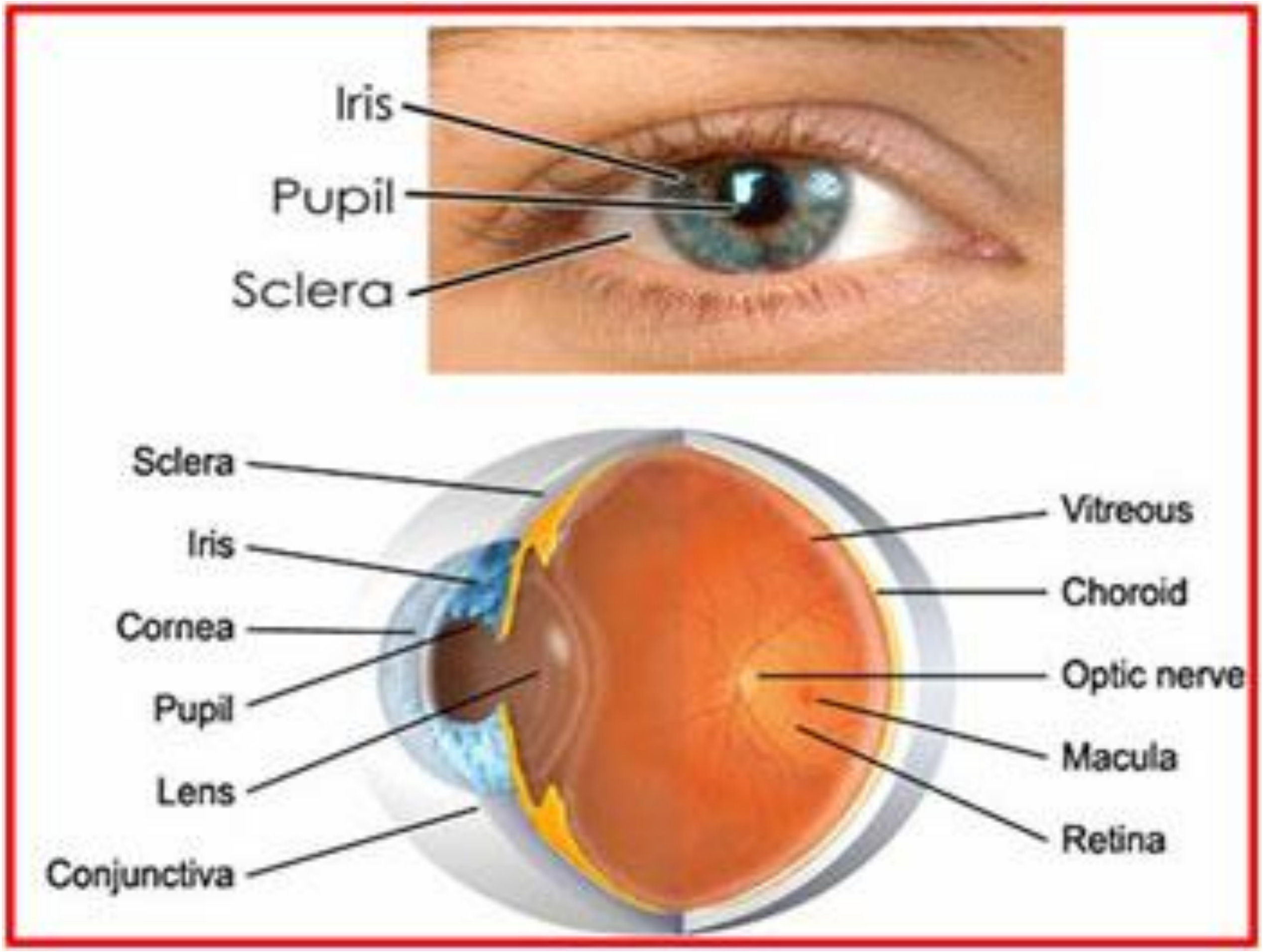


'prewitt' ‘canny'

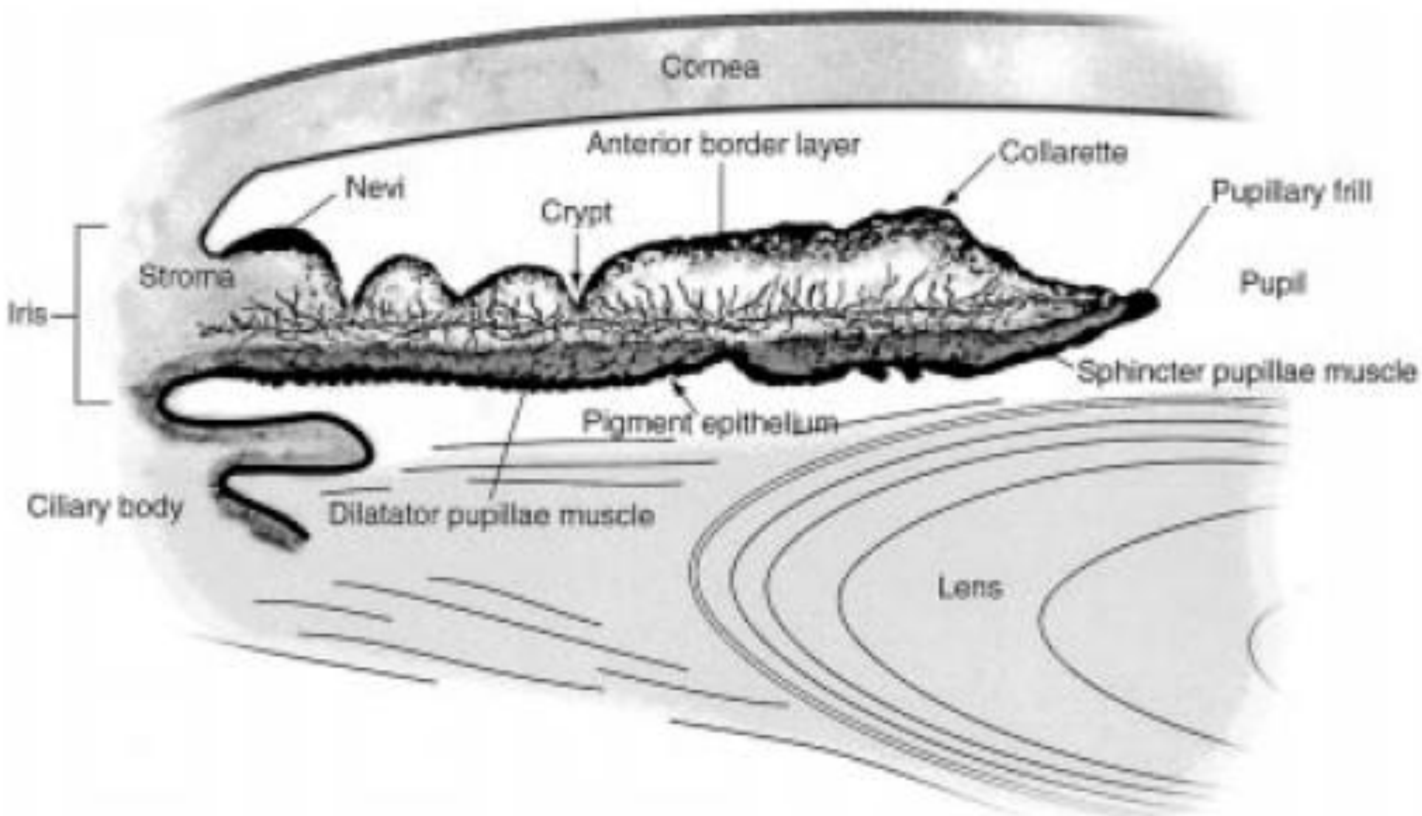
Edge Detection in Transportation

http://www.mobileye-vision.com/default.asp?PageID=214

Iris/Retina of the Eye

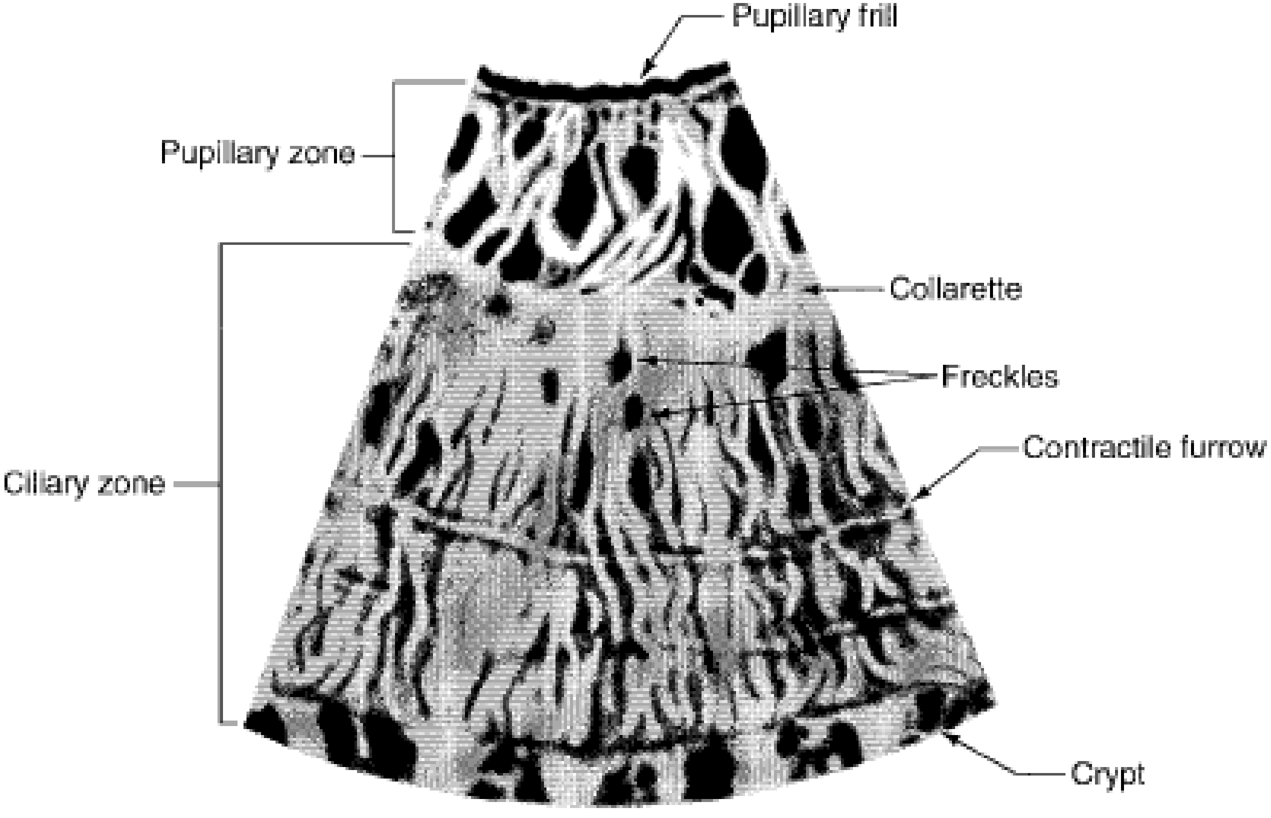


Edge Detection in Iris Recognition



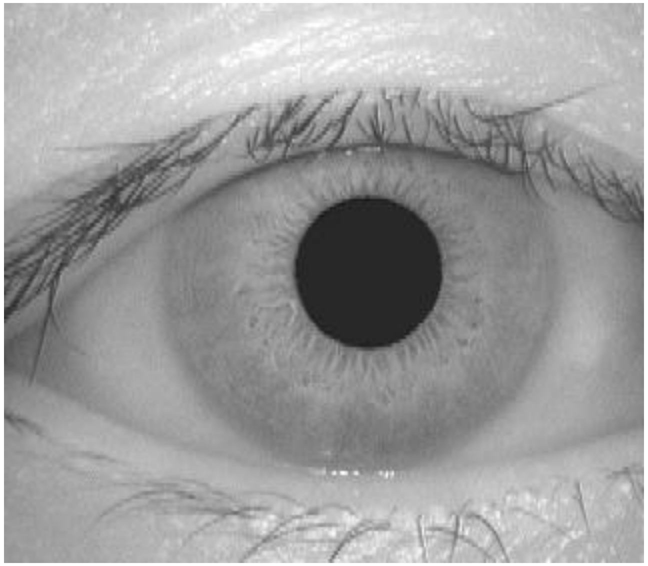
Structure of iris seen in a transverse section

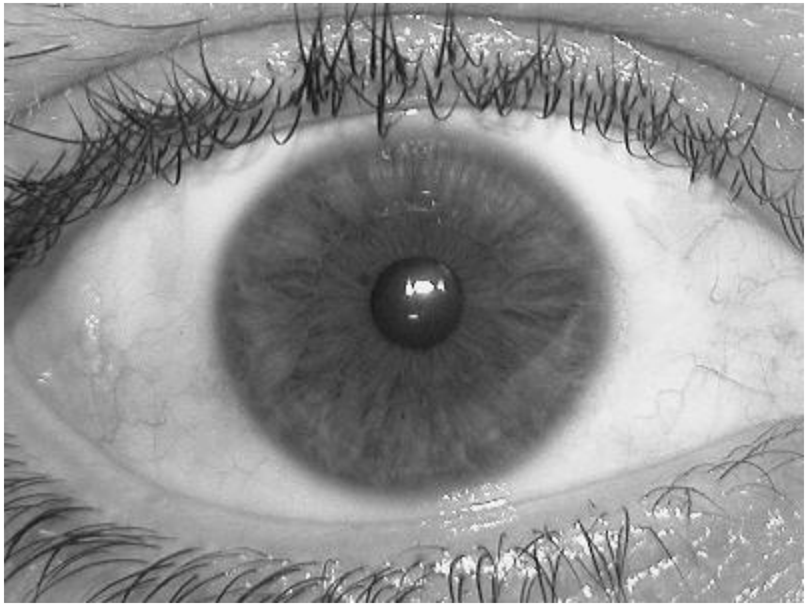
Visual Appearance of Human Iris



Structure of iris seen in a frontal section

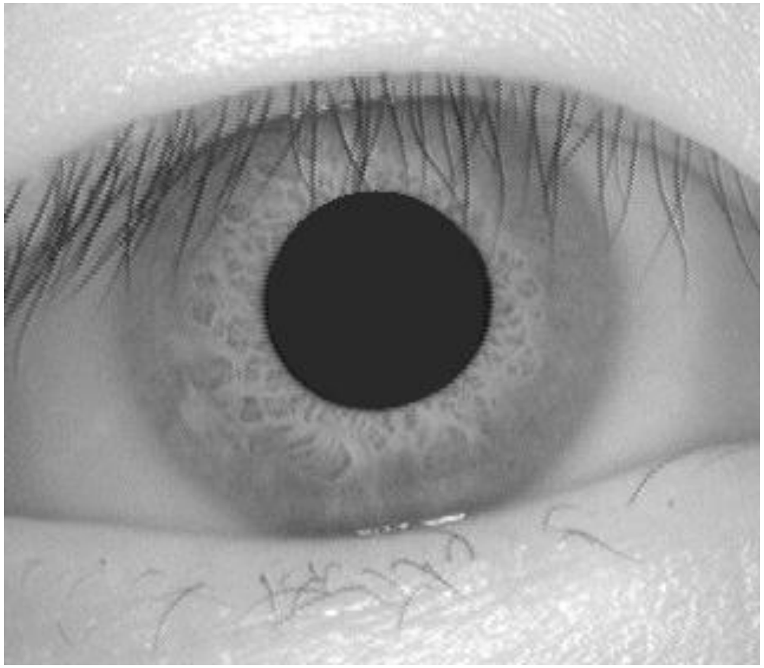
Iris Image Examples

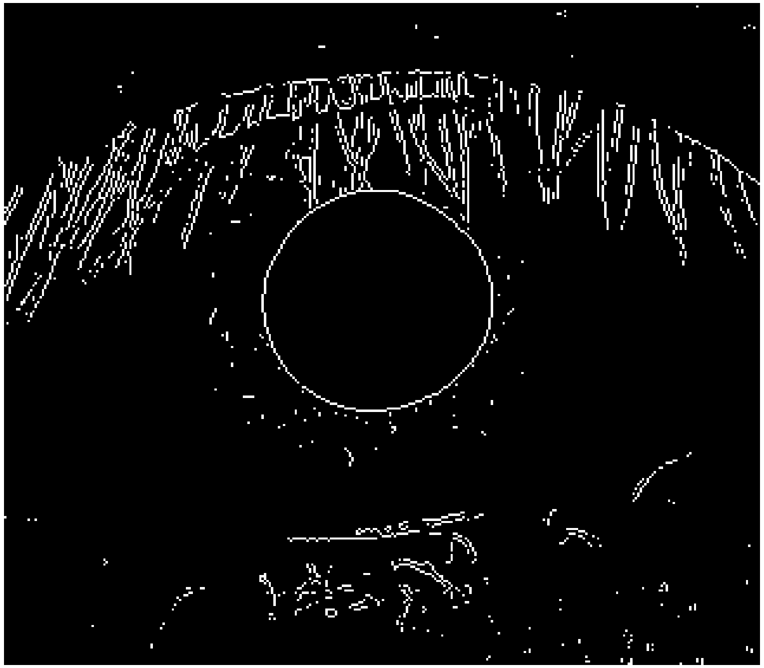




Iris Recognition System

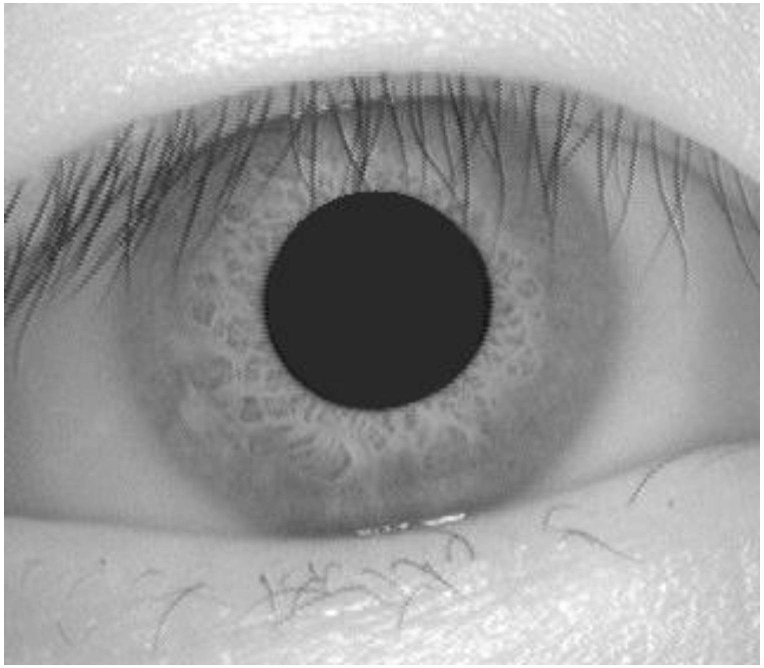
Edge Detection for Pupil Localization

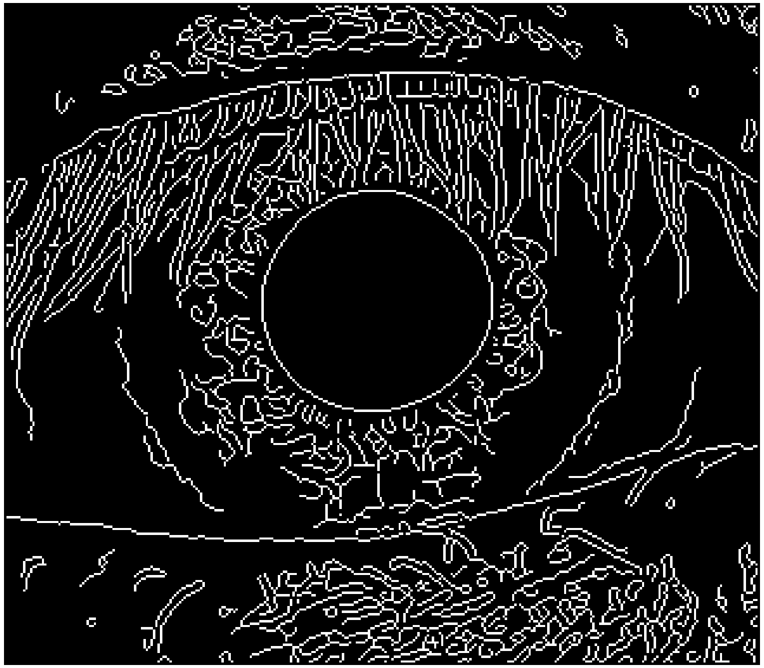




Sobel Edge Detector

Change Edge Detection Operator





Canny Edge Detector

Robust Pupil Detection

After edge detection, we can remove the small edges and fit a circle or ellipse to locate the pupil