# Edge Detection

Nilanjan Ray

Source: www.imagingbook.com

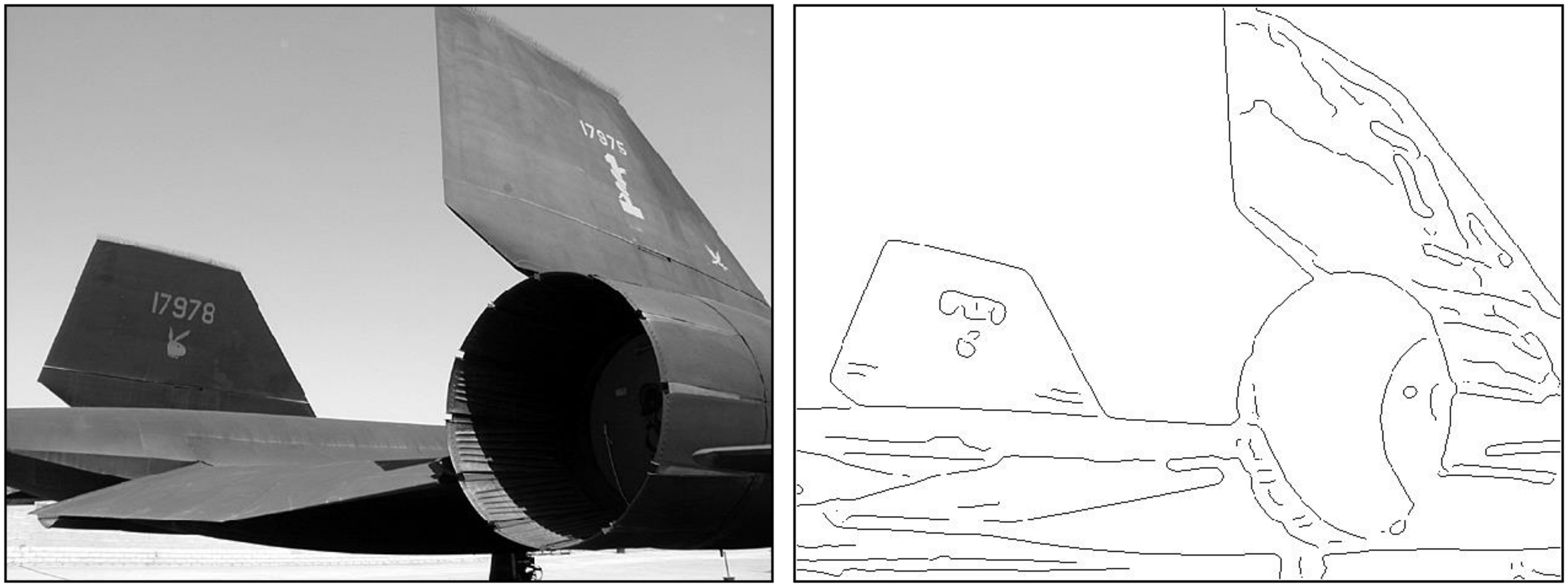
# What are edges?

- Are image positions where local image intensity changes significantly along a particular orientation

- Human visual system extracts sensitive information from edges

- An entire image can be decently "reconstructed" from edges– so in that sense edges are the "information packets" in an image
  - Take a look at: http://www.stanford.edu/class/ee368b/Projects/cnetzer/index.html

# A picture is worth....

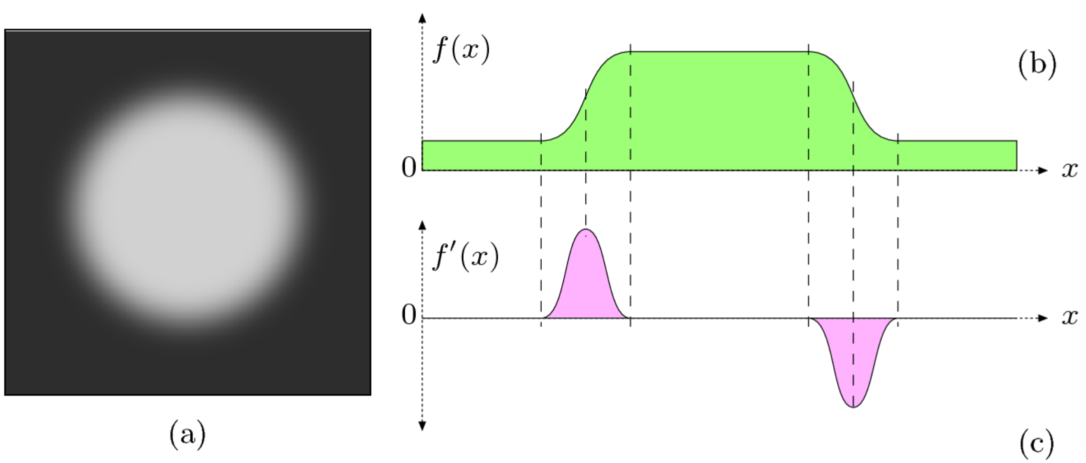



(a) (b)

(a) An image and (b) some of its edges

# Edge detection

- A dominant approach in edge detection is gradient-based
- What is a gradient? $$f'(x) = \frac{df}{dx}(x)$$



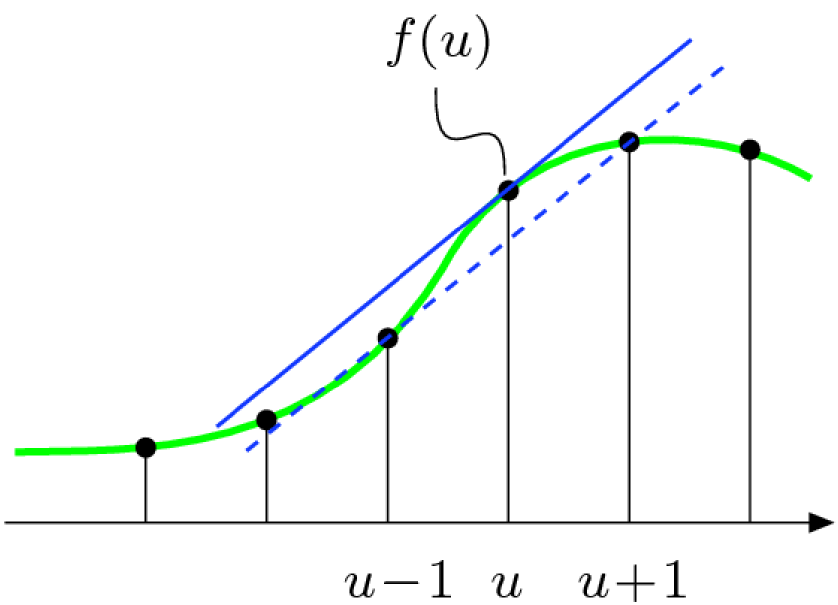

(a) A simple image and (b) its gradient along a cross section

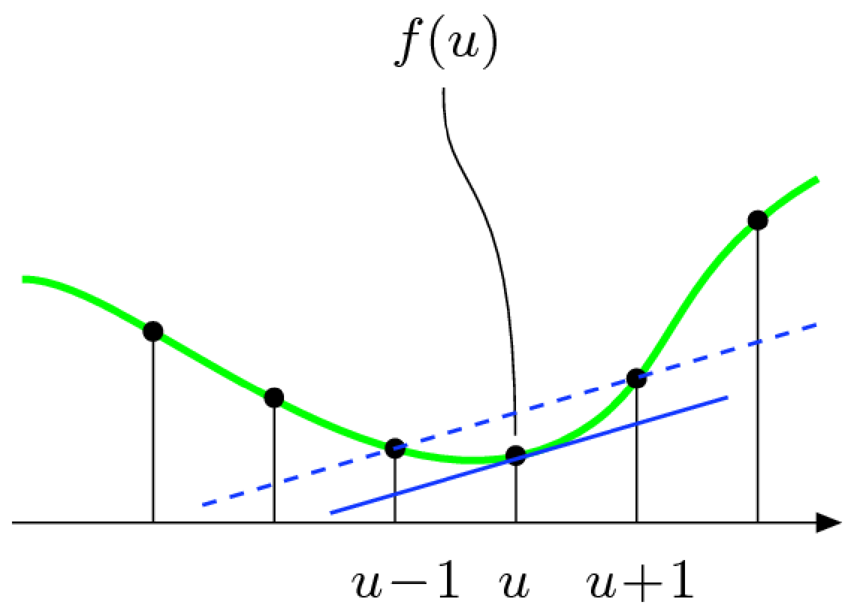
The peaks (valleys) in the gradient are edges here

# Derivatives on functions

- Approximate derivative of a function can be computed as:

$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{2} = 0.5 \cdot \big(f(u+1) - f(u-1)\big)$$



# Derivatives of images

- An image I is a two dimensional function, so we need partial derivatives

$$\frac{\partial I}{\partial u}(u,v) \qquad \text{and} \qquad \frac{\partial I}{\partial v}(u,v)$$

- Gradient of an image is defined as

$$\nabla I(u,v) = \begin{bmatrix} \frac{\partial I}{\partial u}(u,v) \\ \frac{\partial I}{\partial v}(u,v) \end{bmatrix}$$

- Magnitude of image gradient is a good “edge indicator”

$$|\nabla I|(u,v) = \sqrt{\left(\frac{\partial I}{\partial u}(u,v)\right)^2 + \left(\frac{\partial I}{\partial v}(u,v)\right)^2}$$

# Image derivatives by linear filter

- Image derivatives can be computed by linear filtering with masks as:

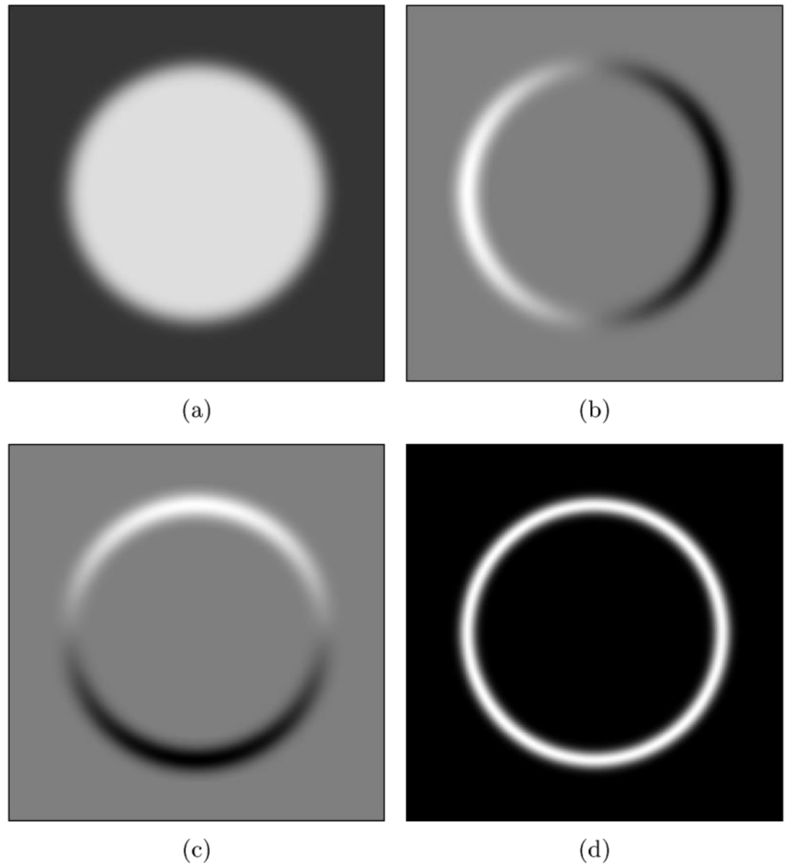
$$H_x^D = \begin{bmatrix} -0.5 & \mathbf{0} & 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix}$$

and

$$H_y^D = \begin{bmatrix} -0.5 \\ \mathbf{0} \\ 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 \\ \mathbf{0} \\ 1 \end{bmatrix}$$

- Once, x and y derivatives are computed, compute gradient magnitude by point operations

# Edge detection by image gradient



(a) A simple image I, (b) x-derivative of I by linear filter, (c) y-derivative of *I* by linear filter, (d) gradient magnitude

# Prewitt edge operator

Linear derivative filter:

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Image gradient computation:

$$\nabla I(u,v) \approx \frac{1}{6} \cdot \begin{bmatrix} \left(I * H_x^P\right)(u,v) \\ \left(I * H_y^P\right)(u,v) \end{bmatrix}$$

# Sobel edge operator

Linear edge operators:

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & \mathbf{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Gradient computation:

$$\nabla I(u,v) \approx \frac{1}{8} \cdot \begin{bmatrix} (I * H_x^S)(u,v) \\ (I * H_y^S)(u,v) \end{bmatrix}$$

# Edge strength and orientation

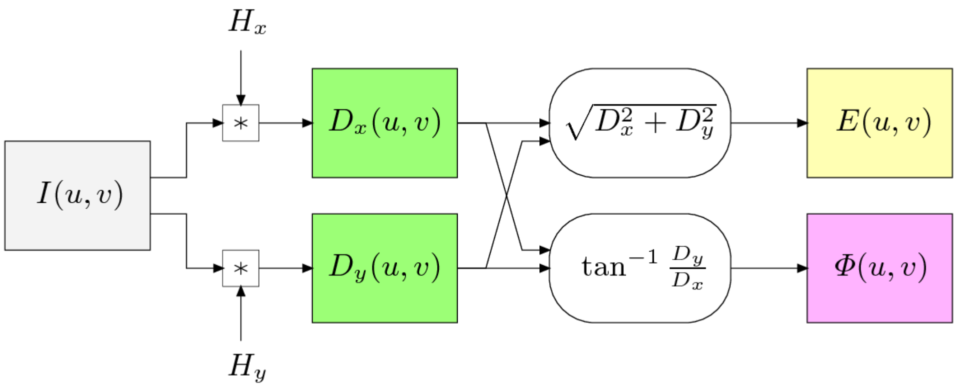
- Edge strength is given by: $E(u,v) = \sqrt{\big(D_x(u,v)\big)^2 + \big(D_y(u,v)\big)^2}$

- Edge orientation is given by:

$$\Phi(u,v) = \tan^{-1}\left(\frac{D_y(u,v)}{D_x(u,v)}\right) = \mathrm{ArcTan}\big(D_x(u,v), D_y(u,v)\big)$$

where $D_x(u,v) = H_x * I \quad \text{and} \quad D_y(u,v) = H_y * I$

Pictorially:



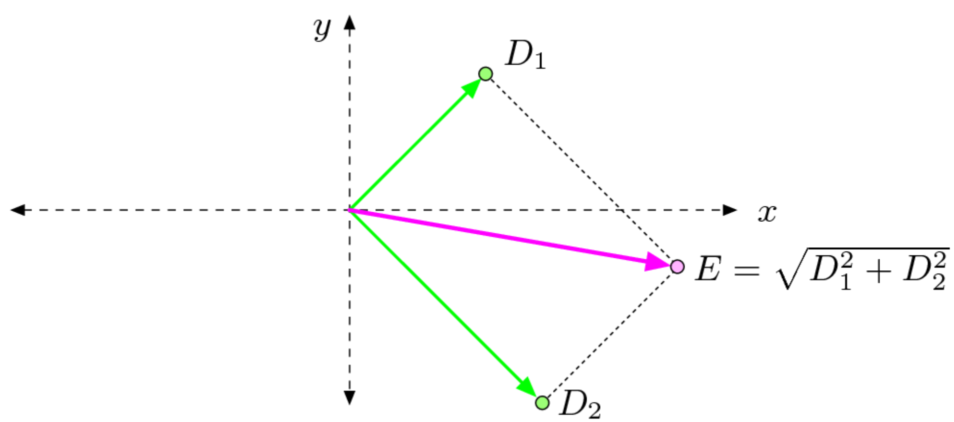
# More linear edge operators

Improved Sobel operator- estimates direction and magnitude better:

$$H_x^{S'} = \frac{1}{32}\begin{bmatrix} -3 & 0 & 3 \\ -10 & \mathbf{0} & 10 \\ -3 & 0 & 3 \end{bmatrix} \quad \text{and} \quad H_y^{S'} = \frac{1}{32}\begin{bmatrix} -3 & -10 & -3 \\ 0 & \mathbf{0} & 0 \\ 3 & 10 & 3 \end{bmatrix}$$

A simple edge operator, called Robert's operator:

$$H_1^R = \begin{bmatrix} 0 & \mathbf{1} \\ -1 & 0 \end{bmatrix} \quad \text{and} \quad H_2^R = \begin{bmatrix} -1 & 0 \\ 0 & \mathbf{1} \end{bmatrix}$$



# Compass operator

Why stop at two directions?

Compass operators compute edges in 8 discrete directions $45^o$ apart:

$$H_3^K = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} \qquad H_7^K = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$$

$$H_0^K = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_4^K = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$H_2^K = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad H_6^K = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$H_1^K = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \qquad H_5^K = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

# Compass operator…

We don't need to compute all the filtering results:

$$\begin{array}{llll} D_0 \leftarrow I * H_0^K & D_1 \leftarrow I * H_1^K & D_2 \leftarrow I * H_2^K & D_3 \leftarrow I * H_3^K \\ D_4 \leftarrow -D_0 & D_5 \leftarrow -D_1 & D_6 \leftarrow -D_2 & D_7 \leftarrow -D_3 \end{array}$$

Why?

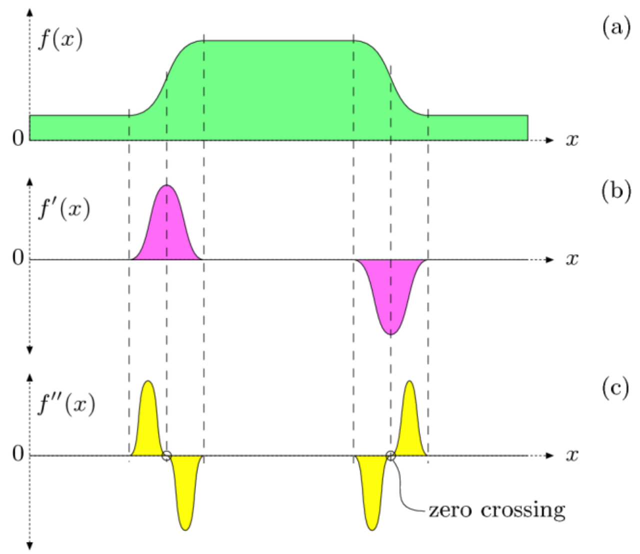
Next compute the edge strength,

$$\begin{aligned} E^K(u,v) &\triangleq \max\big(D_0(u,v), D_1(u,v), \ldots D_7(u,v)\big) \\ &= \max\big(|D_0(u,v)|, |D_1(u,v)|, |D_2(u,v)|, |D_3(u,v)|\big) \end{aligned}$$

and the orientation:

$$\Phi^K(u,v) \triangleq \frac{\pi}{4} \qquad \text{with } \; j = \underset{0 \le i \le 7}{\operatorname{argmax}} \; D_i(u,v)$$

# Second order edge operator




Some linear edge detection methods are based on “zero crossings” of the image second derivative, such as , Laplacian-of-Gaussian edge detector.

# From edges to contour

- OK, we can got the edge strength and the orientation; how do we get a thin edge from this information?
- Let's study Canny edge detection to see one answer to the above question

# Canny edge detection

- Step 1: Smooth image with Gaussian filtering.
- Step 2: Find out edge strength and orientation of the smoothed image.
- Step 3 (non-maximum suppression): At each pixel, determine if it has maximum edge strength along the edge orientation. Mark such pixels as edge pixels.
- Step 4 (hysteresis)
  - Apply a high threshold to the edge strength magnitude to detect a set of edges.
  - Follow the edges perpendicular to the edge orientation. If the pixels survive a low threshold, keep it as an edge pixel.

# Canny method: Step 1

Original image

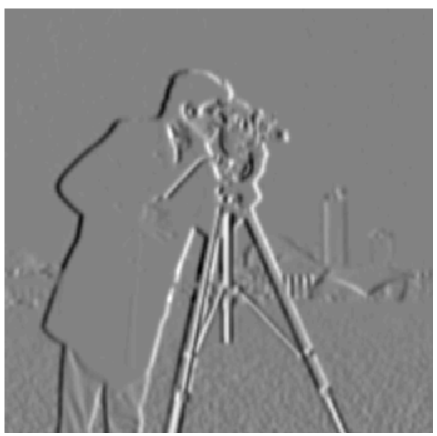
Smoothed image

Smoothing with Gaussian filter: sigma =1

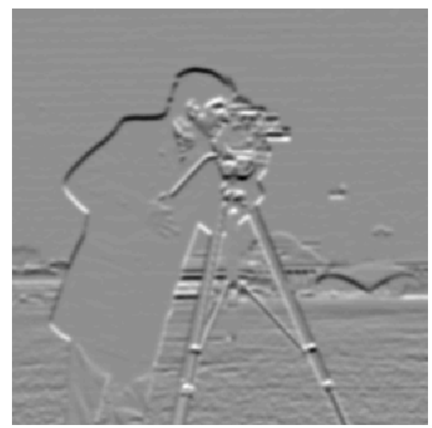
# Canny method: Step 2

Sobel edge filter matrices

$$H = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad V = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$



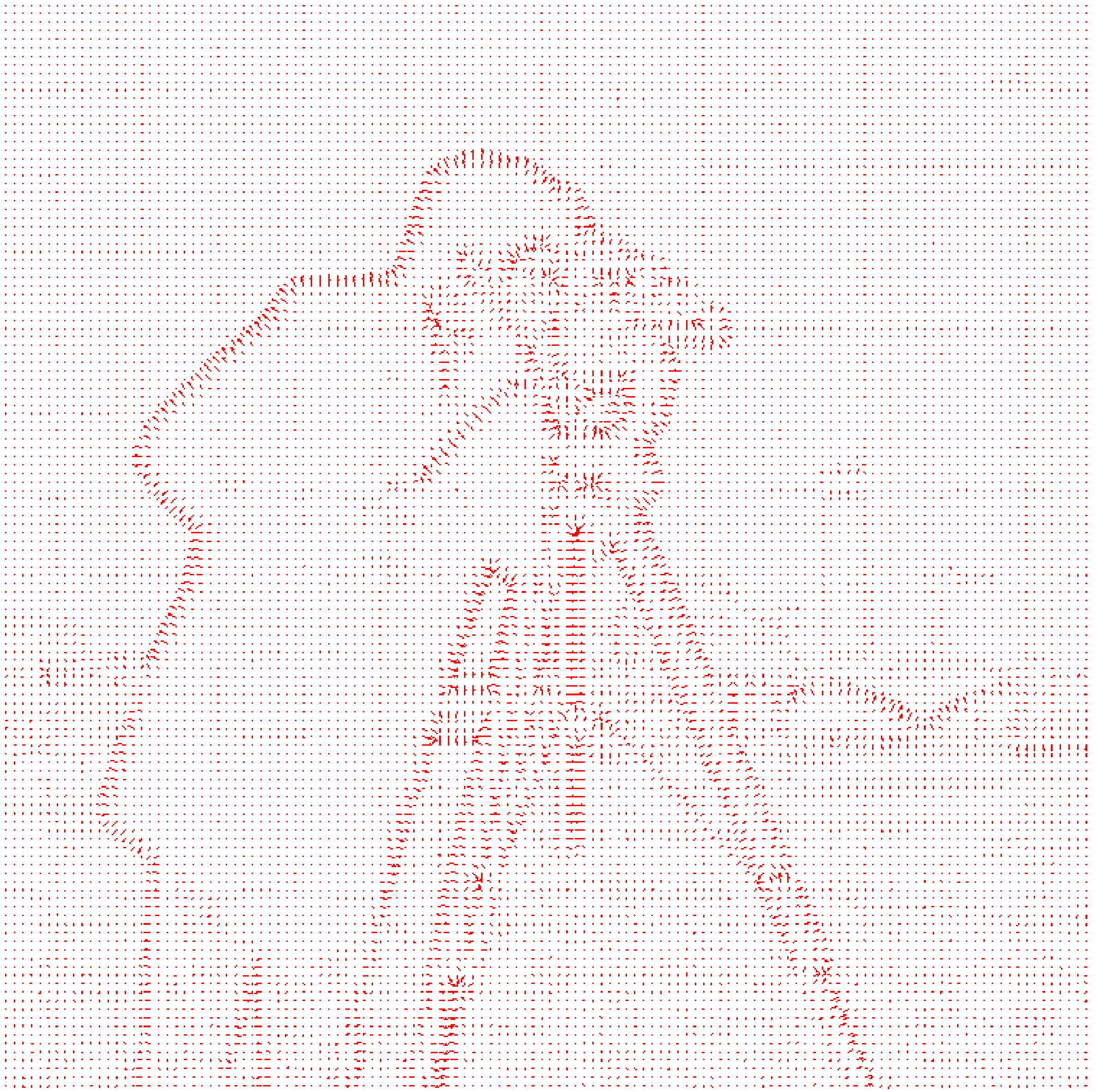
$D_x = I*H$



$D_y = I*V$

# Canny method: Step 2…

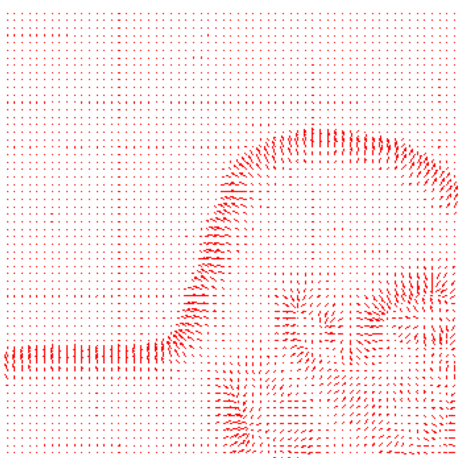
Edge orientation
(shown as arrows
instead of angles)



Edge strength image (E )

# Canny method: Step 3

Edge strength E (Cropped for better visualization)



Edge orientation (Cropped for better visualization)

Non-maximum suppression (aka thinning of edges)

Let’s call this image: S

# Canny method: Step 4

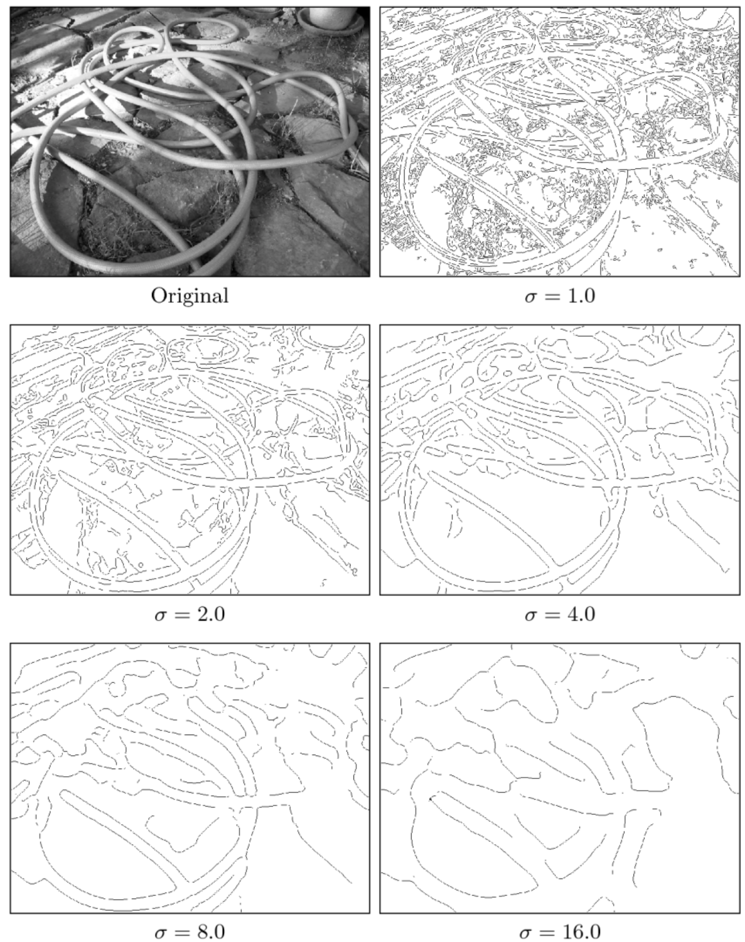
Step 4a: A low threshold on S

Step 4b: apply a high threshold on S

Step 4c: Starting from 4b link/track edges

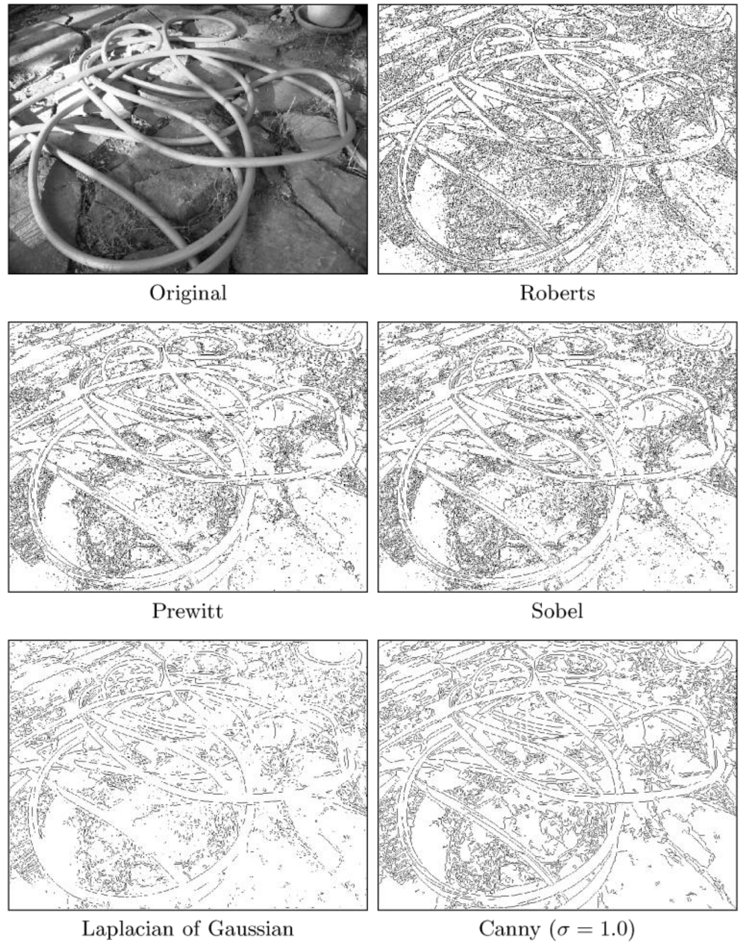
Final result

# Canny edges



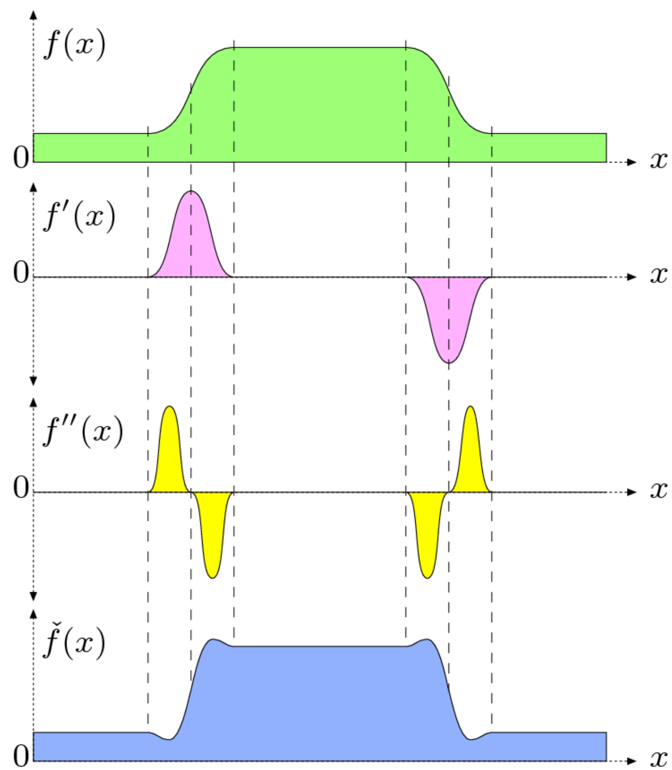

Canny edges at different smoothing levels. A single therhold value is used here.

# Edge detection: visual comparisons




Comparisons of different edge operators

# Edge sharpening




$$\check{f}(x) = f(x) - w \cdot f''(x)$$

*w* is a positive tuning parameter

# Edge sharpening with Laplacian

Laplacian of a function: $$(\nabla^2 f)(x,y) = \frac{\partial^2 f}{\partial^2 x}(x,y) + \frac{\partial^2 f}{\partial^2 y}(x,y)$$

Discrete linear second derivative operators:

$$\frac{\partial^2 f}{\partial^2 x} \equiv H_x^L = \begin{bmatrix} 1 & -\mathbf{2} & 1 \end{bmatrix} \quad \text{and} \quad \frac{\partial^2 f}{\partial^2 y} \equiv H_y^L = \begin{bmatrix} 1 \\ -\mathbf{2} \\ 1 \end{bmatrix}$$

Discrete Laplacian operator: $$H^L = H_x^L + H_y^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -\mathbf{4} & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

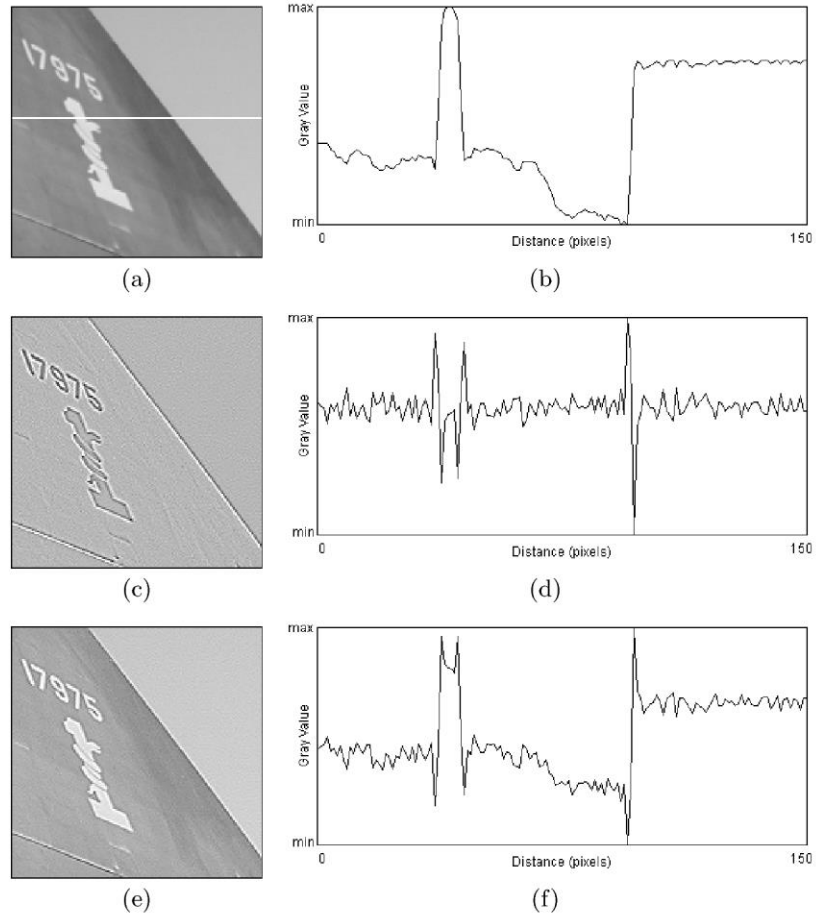
Other discrete approximations to Laplacian: $$H_8^L = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -\mathbf{8} & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad H_{12}^L = \begin{bmatrix} 1 & 2 & 1 \\ 2 & -\mathbf{12} & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

# Edge sharpening with Laplace filter

Original image

Laplacian of original image

Edge sharpened image



# Unsharp masking

Another edge sharpening method. Works better than Laplacian edge sharpening on noisy images. Why?

Computed in two steps:

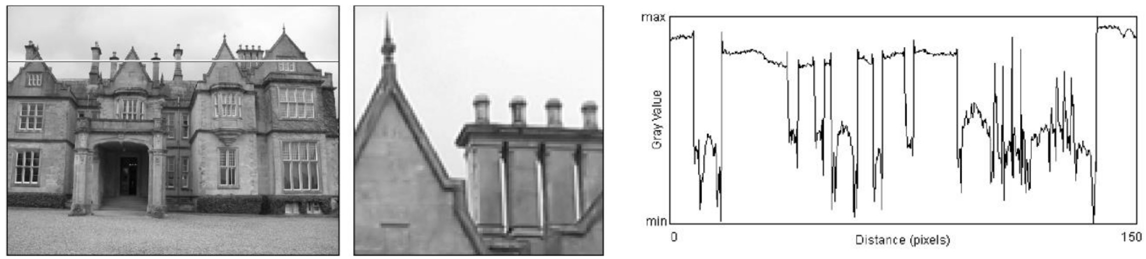
Step 1: $M \leftarrow I - (I * \tilde{H}) = I - \tilde{I}$

Step 2: $\check{I} \leftarrow I + a \cdot M$

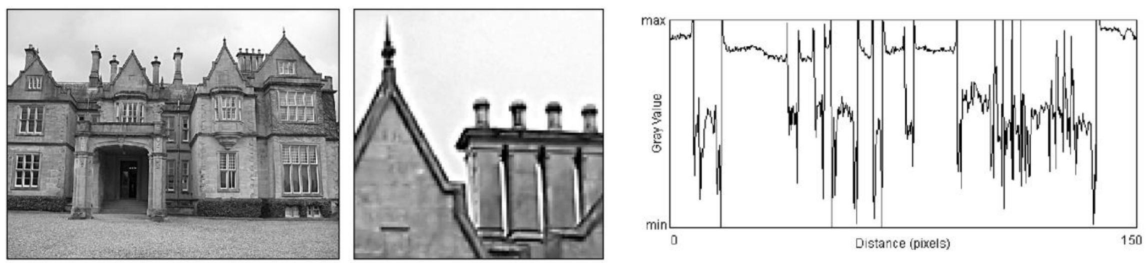
A slightly modified step 2 often works better:

$$\check{I}(u,v) \leftarrow \begin{cases} I(u,v) + a \cdot M(u,v) & \text{for } |\nabla I|(u,v) \geq t_c \\ I(u,v) & \text{otherwise.} \end{cases}$$

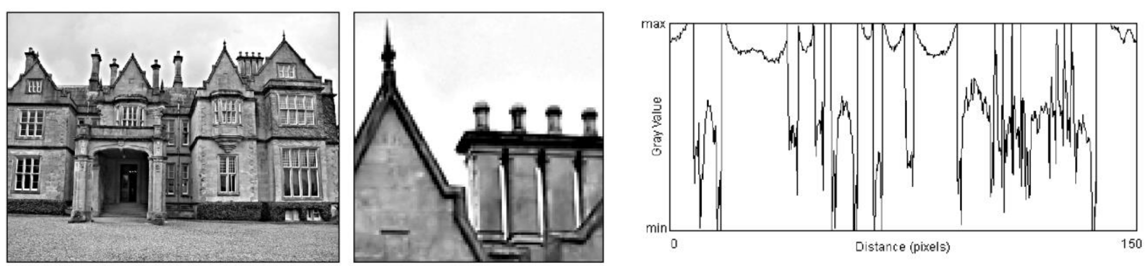
# Unsharp masking: example




(a) Original (b) (c)




(d) $\sigma = 2.5$ (e) (f)




(g) $\sigma = 10.0$ (h) (i)

# Edges: scales

- Back to basics: edges can only be detected at a scale
- Multi-scale edge detection
  - Elder-Zucker edge detection (Local scale control for edge detection and blur estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 7, 699-716, 1998.)
  - Lindeberg edge detection (Tony Lindeberg: Edge Detection and Ridge Detection with Automatic Scale Selection. International Journal of Computer Vision 30(2): 117-156,1998)
  - …