CS443: Digital Imaging and Multimedia
Edges and Contours

Spring 2008
Ahmed Elgammal
Dept. of Computer Science
Rutgers University

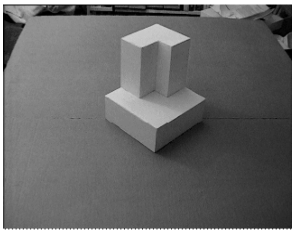
Outlines

- What makes an edge?
- Gradient-based edge detection
- Edge Operators
- From Edges to Contours
- Edge Sharpening

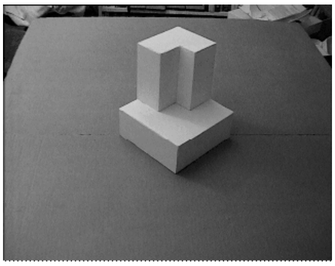
- Sources:
  - Burger and Burge “Digital Image Processing” Chapter 7
  - Forsyth and Ponce “Computer Vision a Modern approach”

What are edges

- What is an edge? A sharp change in brightness
- What generates an edge (Where edges occur) ?
  - Boundaries between objects
  - Reflectance changes (within object)
  - Change in surface orientation (within object)
  - Illumination changes: e.g., cast shadow boundary (within object)



- Edge: A sharp change in brightness
- But which changes we would like to mark as an edge? Meaningful changes. Hard to defined.
- How to tell a semantically meaningful edge from a nuisance edge ?
- Both low level and high level information



Edges in Biological Vision

- We have seen evidence before of edge/bar detectors at different stages of our visual system.

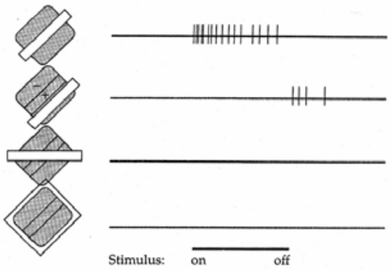


Figure 1.8 Bar stimuli of different orientations (left) and the responses they evoke from a simple cell in primary visual cortex (right).
From D. H. Hubel, Eye, Brain, and Vision, *New York, Scientific American Library, 1988.*

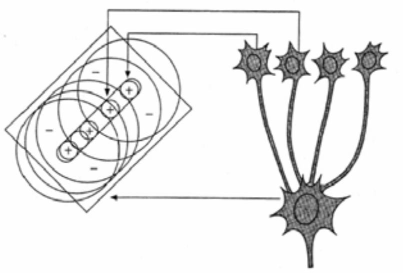


Figure 1.9 Illustration of the idea that simple cells result form the feedforward convergence of a set of center–surround cells.

Adapted from D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," Journal of Physiology, *160, 1962.*

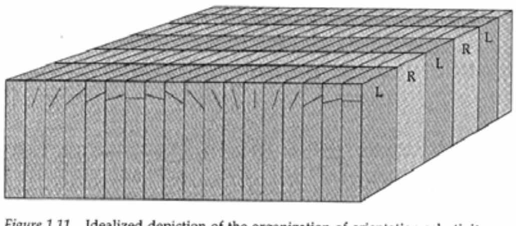
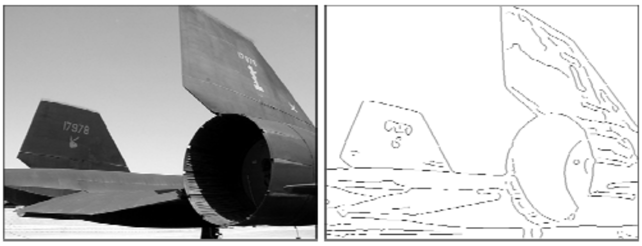


Figure 1.11 Idealized depiction of the organization of orientation selectivity and ocular dominance in primary visual cortex.

Adapted from D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," Journal of Physiology, *160, 1962.*

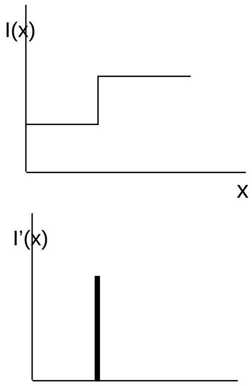
Edge Detection

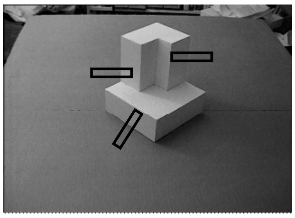
- An image processing task that aims to find edges and contours in images


(a) (b)

Characteristic of an edge

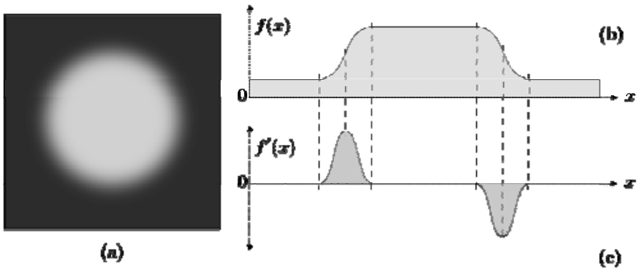
- Edge: A sharp change in brightness
- Ideal edge is a step function in certain direction.



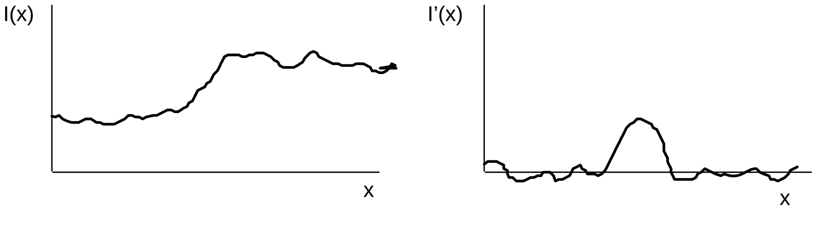


- 1-D edges
- Realistically, edges is a smooth (blurred) step function
- Edges can be characterized by high value first derivative

$$f'(x) = \frac{df}{dx}(x)$$

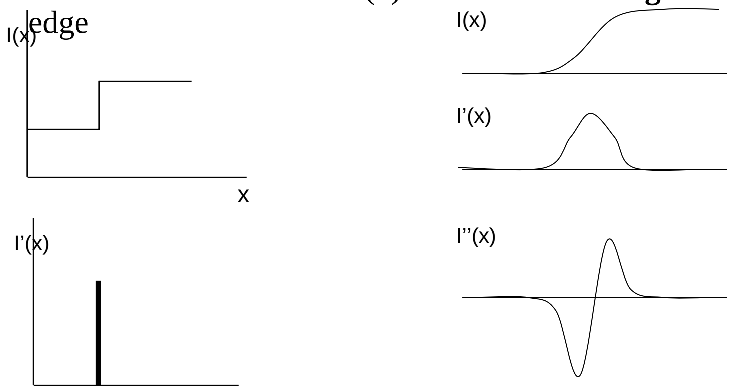


- More realistically, image edges are blurred and the regions that meet at those edges have noise or variations in intensity.
  - blur - high first derivatives near edges
  - noise - high first derivatives within regions that meet at edges



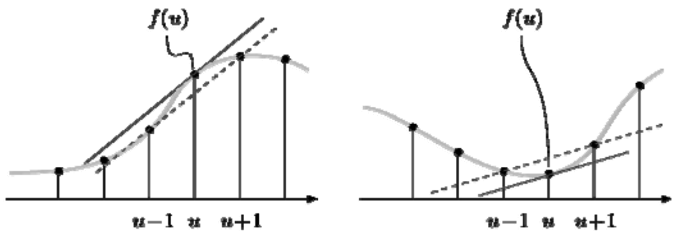
Characteristics of an edge

- Ideal edge is a step function in certain direction.
- The first derivative of I(x) has a **peak** at the edge
- The second derivative of I(x) has a **zero crossing** at the edge



- How can we compute the derivative of a discrete function?

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



Function Gradient

- Let *f*(x,y) be a 2D function. It has derivatives in all directions
  - The gradient is a vector whose direction is in the direction of the maximum rate of change of *f* and whose magnitude is the maximum rate of change of *f* (direction of maximum first derivative)
- If f is continuous and differentiable, then its gradient can be determined from the directional derivatives in any two orthogonal directions - standard to use x and y

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

  - magnitude = $[(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2]^{1/2}$
  - direction = $\tan^{-1}(\frac{\partial f / \partial y}{\partial f / \partial x})$

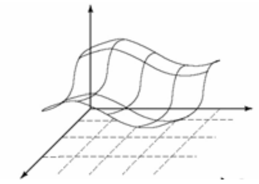


Image Gradient

- Image is a 2D discrete function
- Image derivatives in the horizontal and vertical directions

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

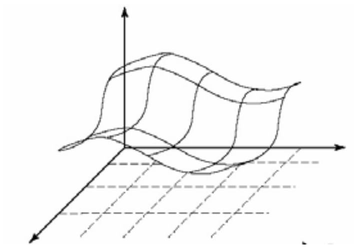
- Image gradient and any given location (u,v)

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

- Gradient Magnitude

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

- Gradient direction



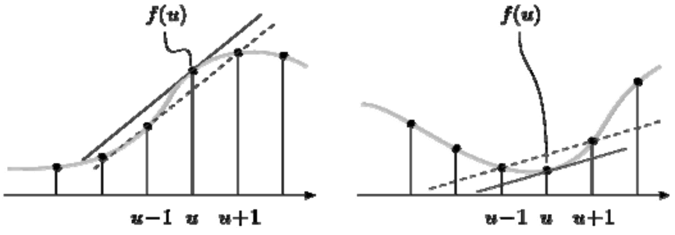
Derivative Filters

- Recall: How can we compute the derivative of a discrete function

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

- This is called finite differences
- Can we make a linear filter that computes this derivative?

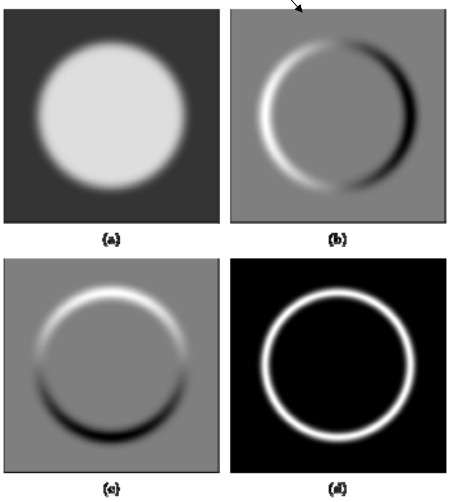
$$H_x^D = \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$



Derivative Filters

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

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



Partial Image derivatives

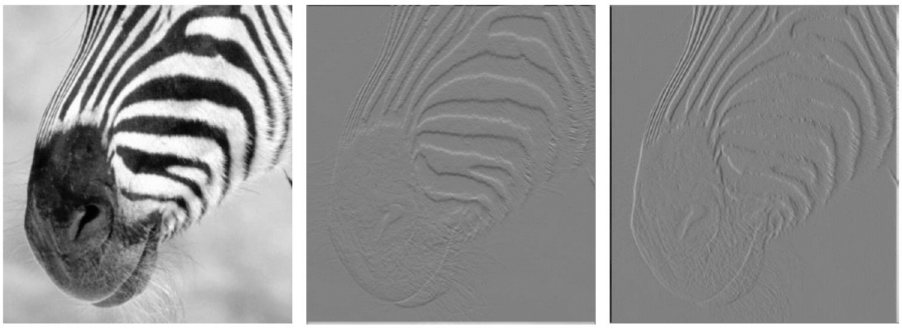
- With a digital image, the partial derivatives are replaced by finite differences:
  - $\Delta_x f = f(x,y) - f(x-1, y)$ $\begin{bmatrix} -1 & 1 \end{bmatrix}$ $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$
  - $\Delta_y f = f(x,y) - f(x, y-1)$
- Alternatives are: $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$
  - $\Delta_{2x} f = f(x+1,y) - f(x-1,y)$
  - $\Delta_{2y} f = f(x,y+1) - f(x,y-1)$
- Robert's gradient
  - $\Delta_+ f = f(x+1,y+1) - f(x,y)$ $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
  - $\Delta_- f = f(x,y+1) - f(x+1, y)$ $\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

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

Prewitt

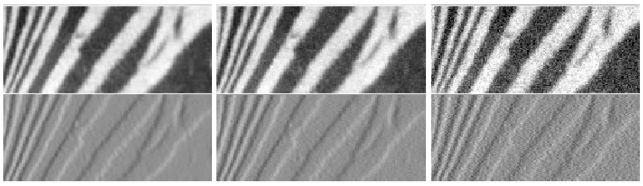
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel



Finite differences and noise

- Finite difference filters respond strongly to noise
  - obvious reason: image noise results in pixels that look very different from their neighbors



- What is to be done?
  - intuitively, most pixels in images look quite a lot like their neighbors
  - this is true even at an edge; along the edge they’re similar, across the edge they’re not
  - suggests that smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors

Edge Operators

Prewitt and Sobel Operators

- Prewitt Operator:

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

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

- Sobel Operator

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

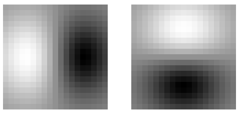
Gaussian Derivative Filters

- So smoothing should help before taking the derivatives.
- Recall: smoothing and differentiation are linear filters
- Recall also: linear filter are associative

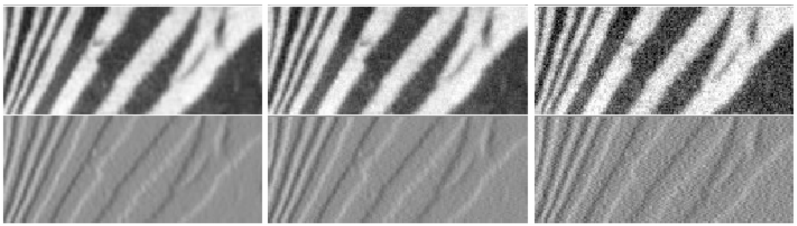
$$K_{\partial/\partial x} * (g * I) = (K_{\partial/\partial x} * g) * I = \frac{\partial g}{\partial x} * I$$

- Smoothing then differentiation ≡ convolution with the derivative of the smoothing kernel.
- If Gaussian is used for smoothing: We need to convolve the image with derivative of the Gaussian

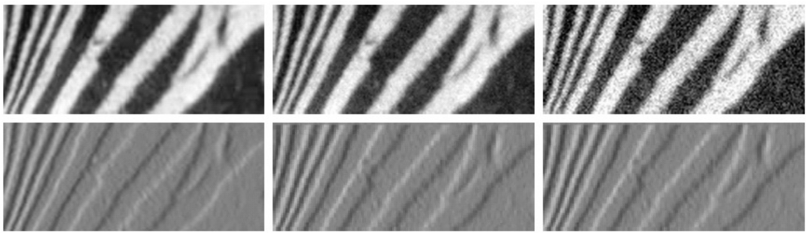
$$\frac{\partial G_\sigma}{\partial x} * I \qquad \frac{\partial G_\sigma}{\partial y} * I$$



Noise σ=3% Noise σ=9%

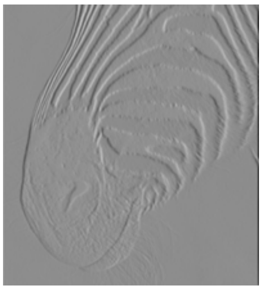


x- derivative - No smoothing

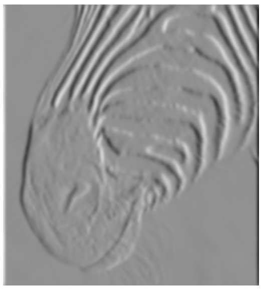


Convolution with x-derivative of a Gaussian (σ=1 pixel)

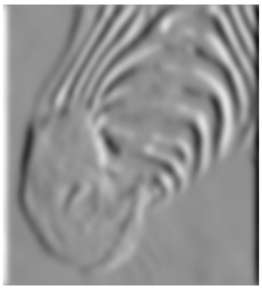
- The scale (σ) of the Gaussian has significant effects on the results - tradeoff…



1 pixel



3 pixels

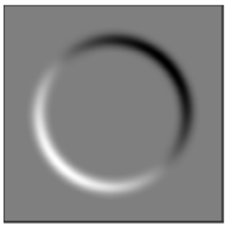


7 pixels

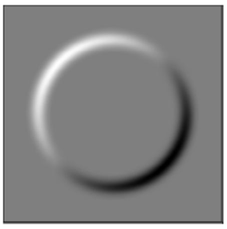
Other operators

- Many other edge operators with different properties
- Roberts operator

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



$D_1 = I * H_1^R$



$D_2 = I * H_2^R$

Gradient-based edge detection:

- Compute image derivatives (with smoothing) by convolution

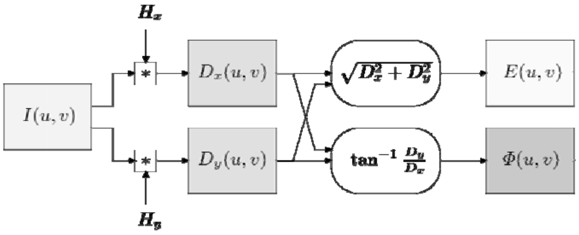
$$D_x(u,v) = H_x * I \quad \text{and} \quad D_y(u,v) = H_y * I$$

- Compute edge strength - gradient magnitude

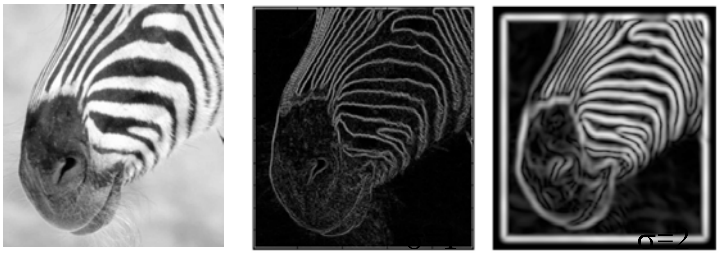
$$E(u,v) = \sqrt{\left(D_x(u,v)\right)^2 + \left(D_y(u,v)\right)^2}$$

- Compute edge orientation - gradient direction

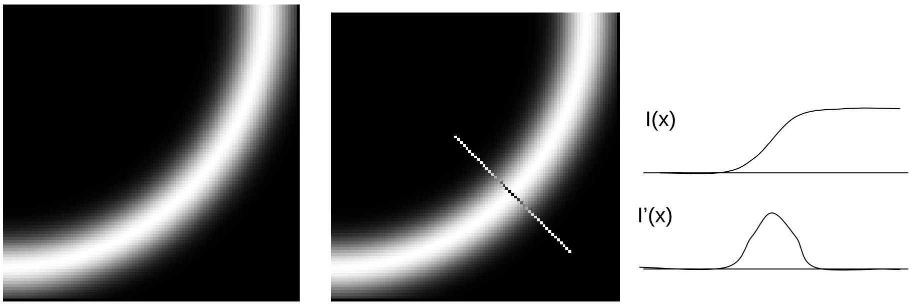
$$\Phi(u,v) = \tan^{-1}\left(\frac{D_y(u,v)}{D_x(u,v)}\right) = \mathrm{ArcTan}\left(D_x(u,v), D_y(u,v)\right)$$



- What’s after computing the gradient magnitude and orientation?
- now mark points where gradient magnitude is particularly large wrt neighbors



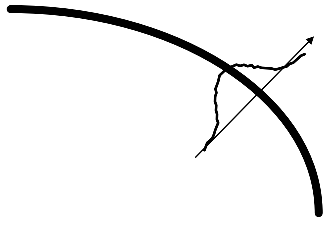
- Problem: The gradient magnitude is large along thick trail; how do we identify the significant points?



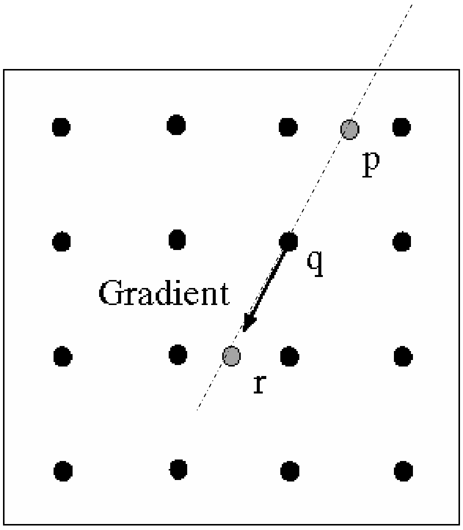
- We wish to mark points along the curve where the magnitude is biggest.
- We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression).
- These points should form a curve.
- There are then two algorithmic issues: at which point is the maximum, and where is the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

Non-maxima suppression

- Non-maxima suppression - Retain a point as an edge point if:
  - its gradient magnitude is higher than a threshold
  - its gradient magnitude is a local maxima in the gradient direction

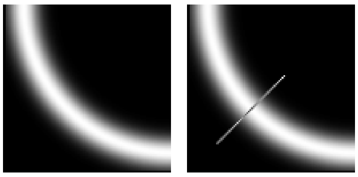


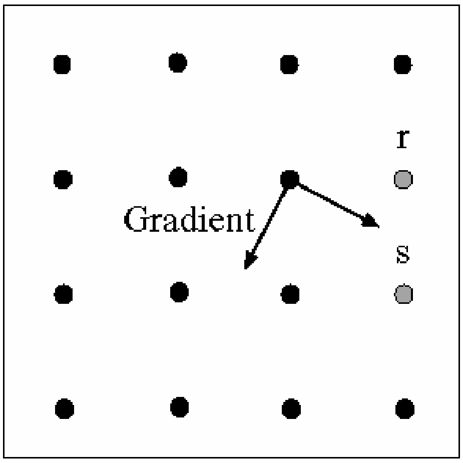
simple thresholding will compute thick edges



Non-maximum suppression

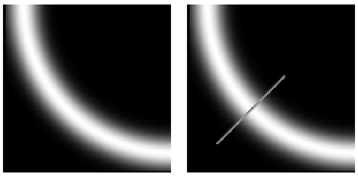
At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.





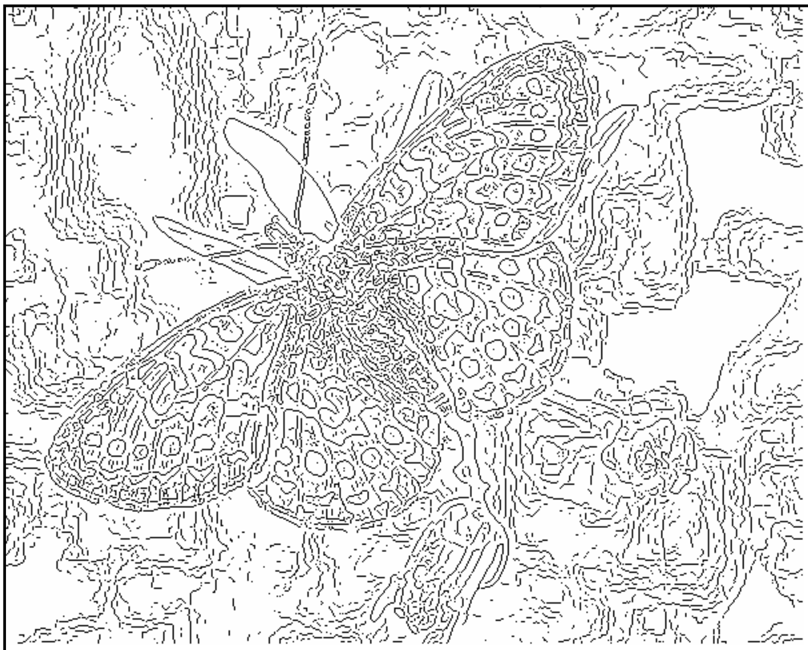
Predicting the next edge point

Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).

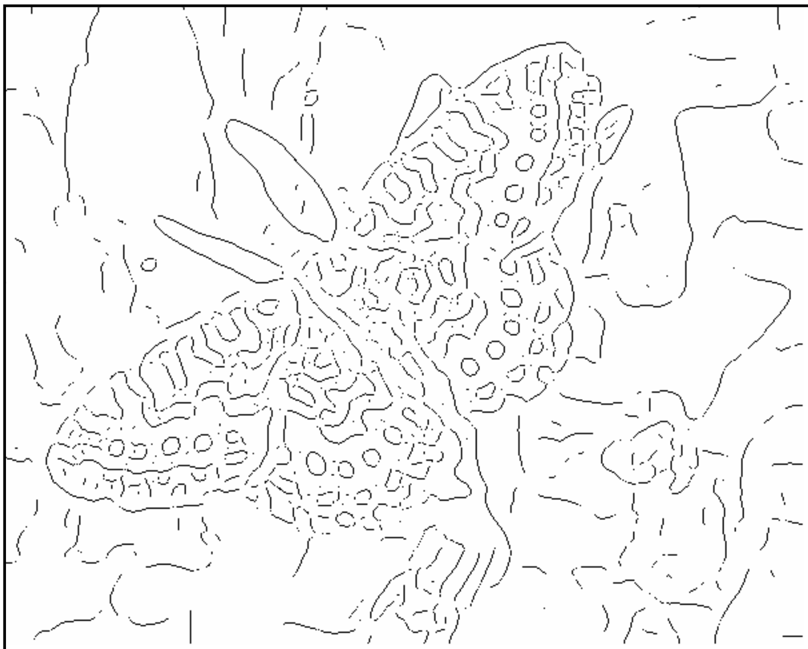


Problem of scale and threshold

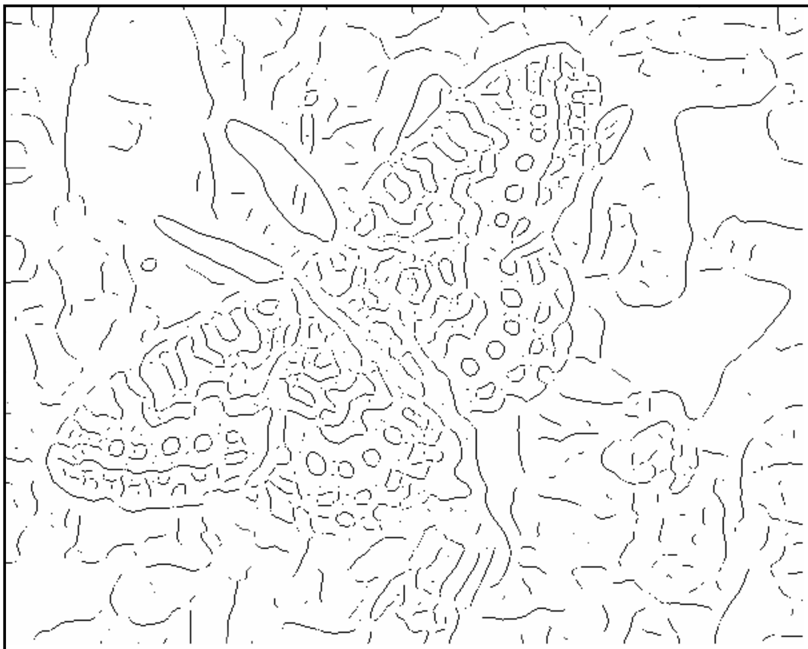
- Usually, any single choice of scale σ does not produce a good edge map
  - a large σ will produce edges form only the largest objects, and they will not accurately delineate the object because the smoothing reduces shape detail
  - a small σ will produce many edges and very jagged boundaries of many objects.
- Threshold:
  - Low threshold : low contrast edges. a variety of new edge points of dubious significance are introduced.
  - High threshold: loose low contrast edges ⇒ broken edges.



fine scale
high
threshold



coarse
scale,
high
threshold



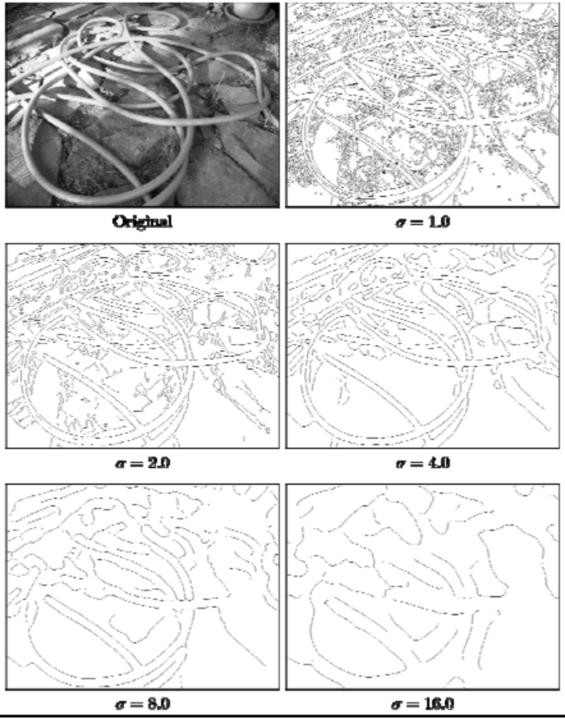
coarse
scale
low
threshold

Hysteresis

- Which Scale:
  - Fine scale: fine details.
  - Coarser scale: fine details disappear.
- Solution: Scale-space approaches
  - detect edges at a range of scales [σ_1, σ_2]
  - combine the resulting edge maps
    - trace edges detected using large σ down through scale space to obtain more accurate spatial localization.
- What Threshold:
  - Low threshold : low contrast edges. a variety of new edge points of dubious significance are introduced.
  - High threshold: loose low contrast edges ⇒ broken edges.
- Solution: use two thresholds
  - Larger threshold: more certain edge, use to start an edge chain
  - Smaller threshold: use to follow the edge chain

Canny Edge detector

- A popular example of a method that operates at different scales and combine the results
  - Minimize the number of false edge points
  - Achieve good localization of edges
  - Deliver only a single mark on each edge
  - Used hystersis to follow edges
  - Typically a single scale implementation is used
  - Available code in ImageJ, matlab and most image processing utilities.



Original

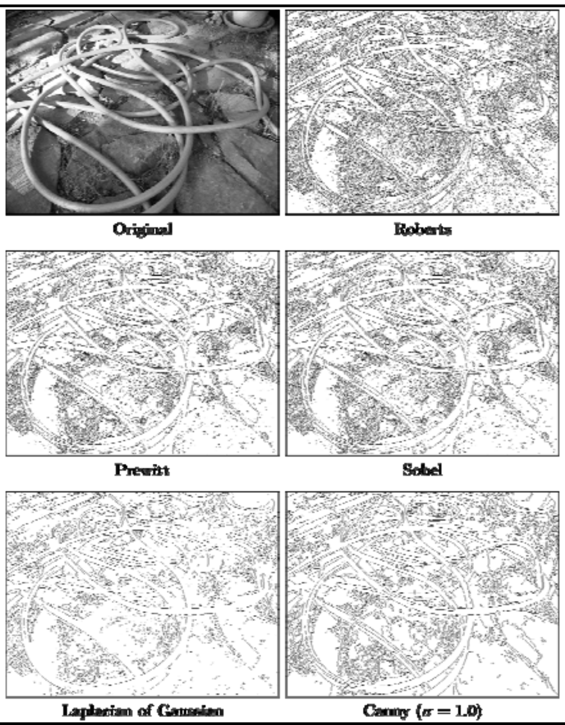
$\sigma = 1.0$

$\sigma = 2.0$

$\sigma = 4.0$

$\sigma = 8.0$

$\sigma = 16.0$



Original

Roberts

Prewitt

Sobel

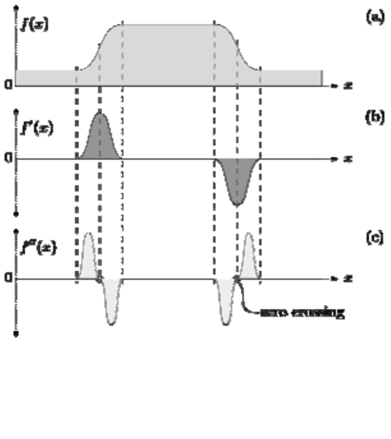
Laplacian of Gaussian

Canny ($\sigma = 1.0$)

Detecting edges based on second derivatives

- Recall: an edge corresponds to a zero crossing at the second derivative
- Laplacian:

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$



Laplace Operator

- Laplacian: $\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$
- Its digital approximation is:

∇^2f(x,y) = [f(x+1,y) - f(x,y)] - [f(x,y) - f(x-1,y)] +
[f(x,y+1)-f(x,y)] - [f(x,y) - f(x,y-1)]

$$\boxed{= [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)] - 4 f(x,y)}$$

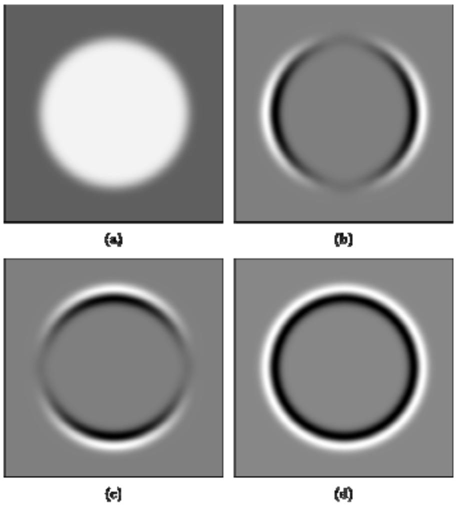
$$\frac{\partial^2 f}{\partial^2 x} \equiv H_x^L = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \quad \text{and} \quad \frac{\partial^2 f}{\partial^2 y} \equiv H_y^L = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

$$H^L = H_x^L + H_y^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\boxed{\begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix}}$$

$$\boxed{\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}}$$

Laplace Operator



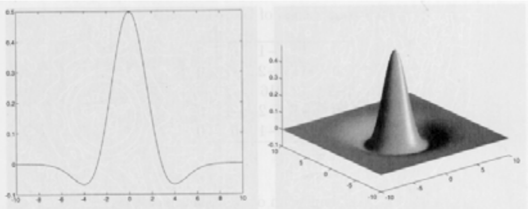
Laplacian of Gaussian

- Laplacian is a linear filter
- Bad idea to apply a Laplacian without smoothing
- If we smooth by a Gaussian before applying a Laplacian:

$$K_{\nabla^2} * (G_\sigma * I) = (K_{\nabla^2} * G_\sigma) * I = \boxed{(\nabla^2 G_\sigma)} * I$$

Laplacian of Gaussian (LoG)

"Mexican Hat"

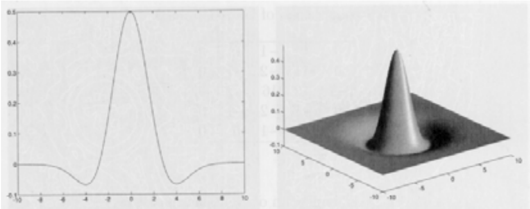


0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Laplacian of Gaussian

- Can be approximated as difference of two Gaussians
- This is called Difference of Gaussians filter DoG

$$\nabla^2 g(x) \approx c_1 e^{-\frac{x^2}{2\sigma_1^2}} - c_2 e^{-\frac{x^2}{2\sigma_2^2}} \qquad \sigma_1 < \sigma_2$$



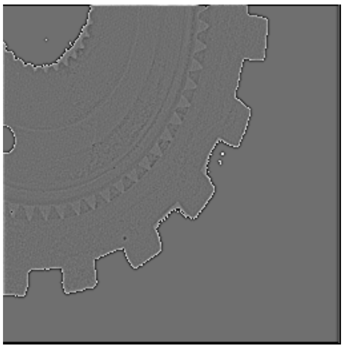
0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

Algorithm (Marr and Hildreth 1980):

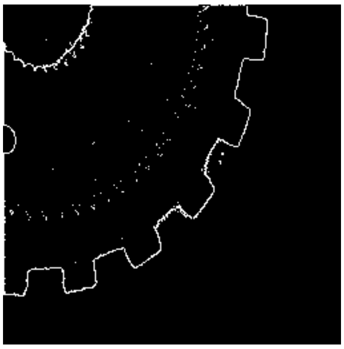
- Convolve the image with a LoG
- Mark the point with zero crossings:
  - these are pixels whose LoG is positive and which have neighbor’s whose LoG is negative or zero
- Check these points to ensure the gradient magnitude is large (to avoid low contrast edges) ⇒ Threshold

- Note : Two parameters: Gaussian scale, contrast threshold

Laplacian of Gaussian

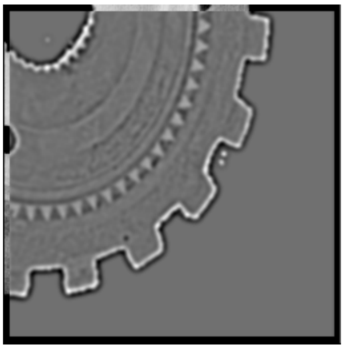


5x5 Mexican Hat - Laplacian of Gaussian

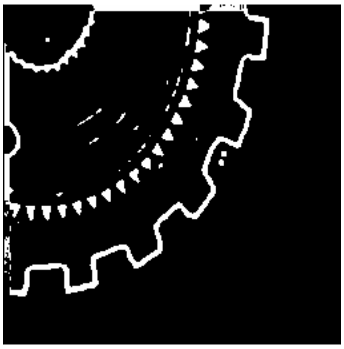


Zero crossings

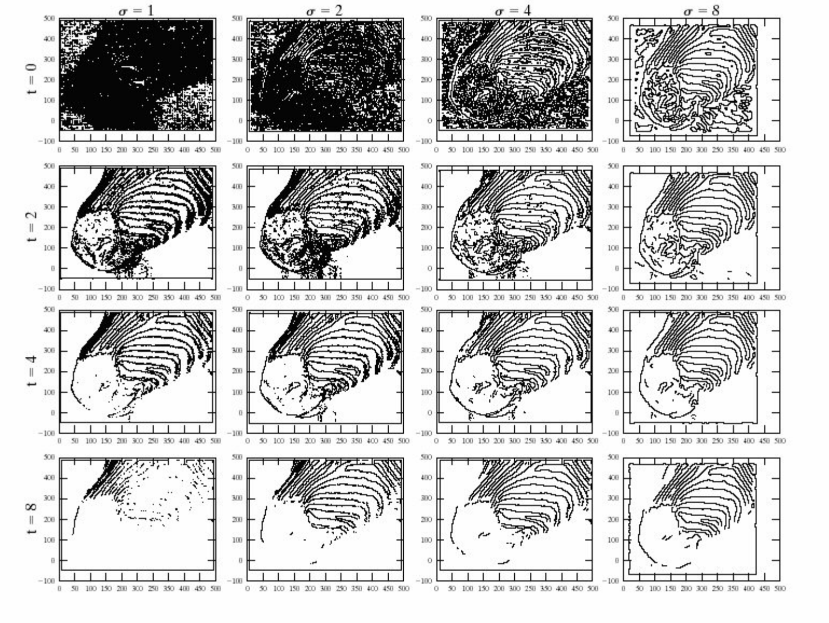
Laplacian of Gaussian



13 x 13 Mexican hat

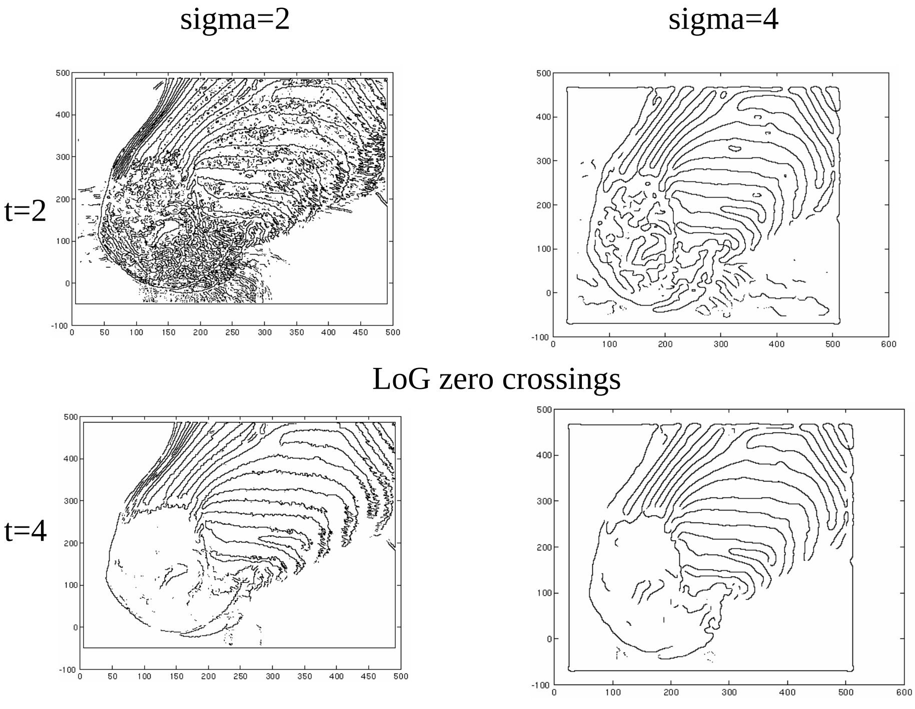


zero crossings



Things to notice:

- As the scale increases, details are suppressed
- As the threshold increases, small regions of edge drop out
- No scale or threshold gives the outline of the head
- Edges are mainly the stripes
- Narrow stripes are not detected as the scale increases.


sigma=2

sigma=4

t=2

LoG zero crossings

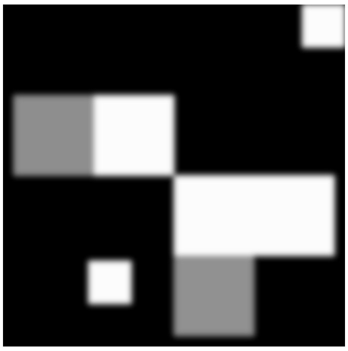
t=4

Problems with the Laplacian approach

- Poor behavior at corners
- Computationally: we need to computer both the LoG and the gradient.

We have unfortunate behavior at corners:

- Zero crossing bulges out at corners
- More than two edges meet: strange behaviors



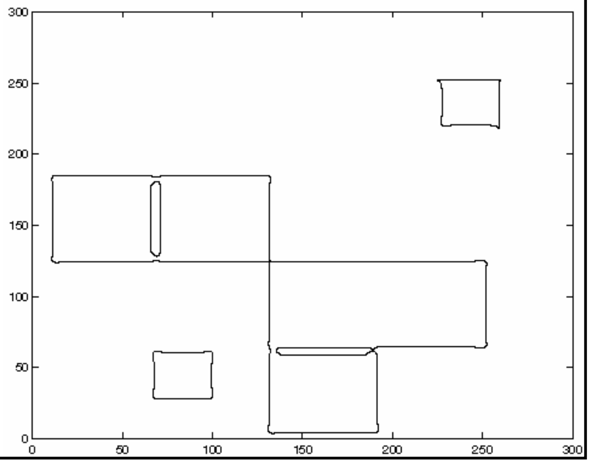
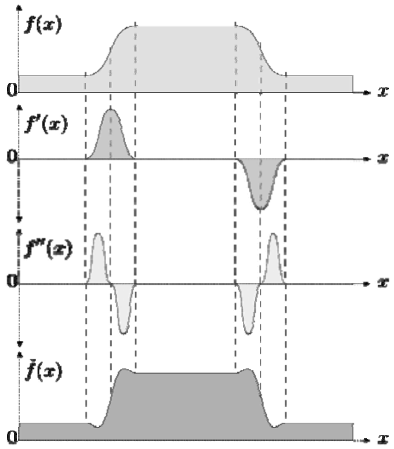


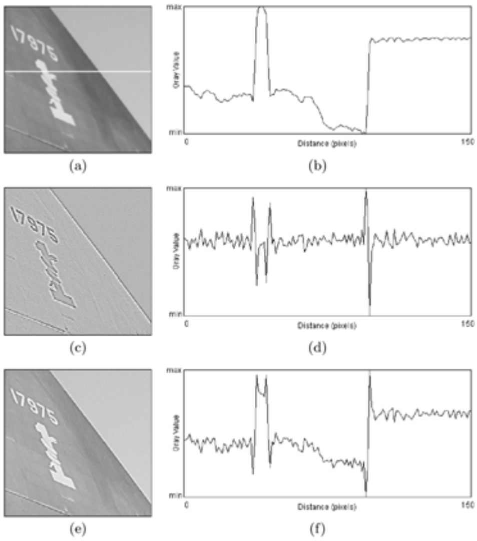
Image Sharpening

- Making images look sharper is common to make up for bluring happened after scanning or scaling
- Amplify high frequency components. What that means?
- High frequencies happen at edges.
- We need to sharpen the edges.

- Edge Sharpening

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





(a) (b)

(c) (d)

(e) (f)

$$\check{I} \leftarrow I - w \cdot (H^L * I)$$

Unsharp Masking (USM)

- Unsharp masking is a technique for edge sharpening!
- Sharpening an image is achieved by combining the image with a smoothed version of it.
- Subtract a smooth version (Gaussian smoothing) from the image itself to obtain an enhanced edge mask:

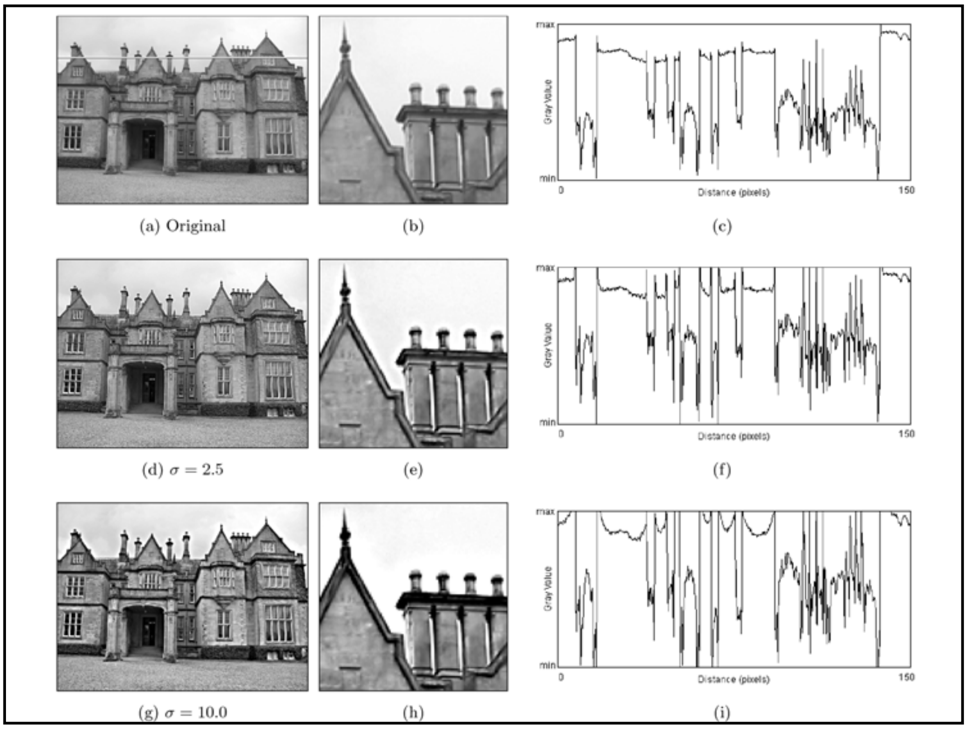
$$M \leftarrow I - (I * \tilde{H}) = I - \tilde{I}$$

- Add the mask to the image with a weight.

$$\check{I} \leftarrow I + a \cdot M$$

- Together:

$$\check{I} \leftarrow I + a \cdot (I - \tilde{I}) = (1 + a) \cdot I - a \cdot \tilde{I}$$



(a) Original

(b)

(c)

(d) $\sigma = 2.5$

(e)

(f)

(g) $\sigma = 10.0$

(h)

(i)