Computer Vision

Spring 2006 15-385,-685

Instructor: S. Narasimhan

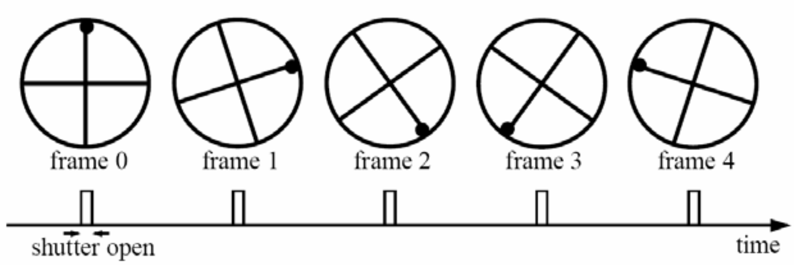
Wean 5403

T-R 3:00pm – 4:20pm

Aliasing - Really bad in video

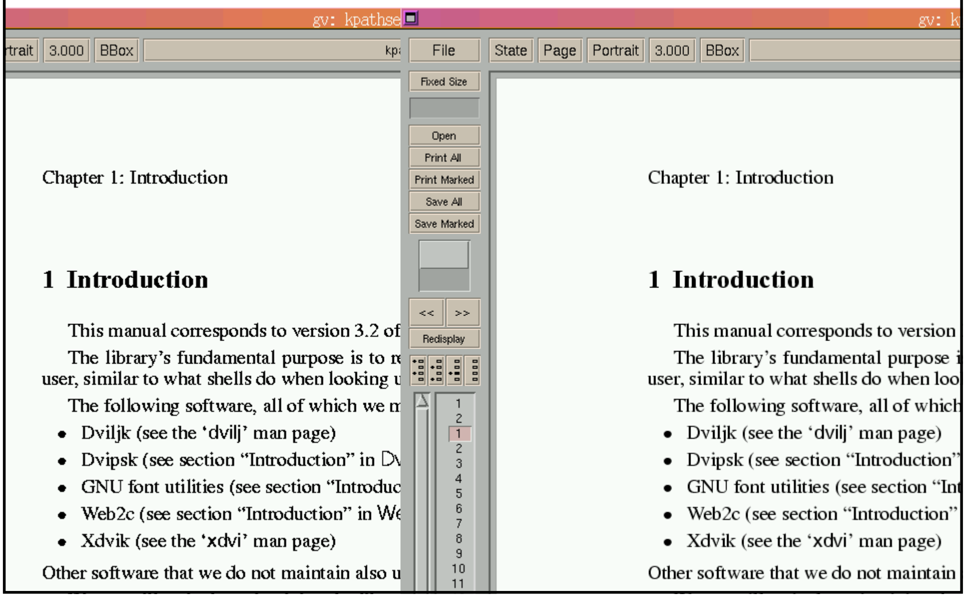
Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

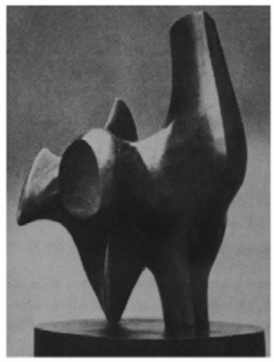
Text Aliasing



Edge Detection

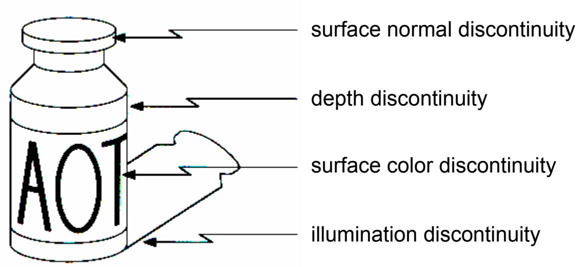
Lecture #7

Edge Detection



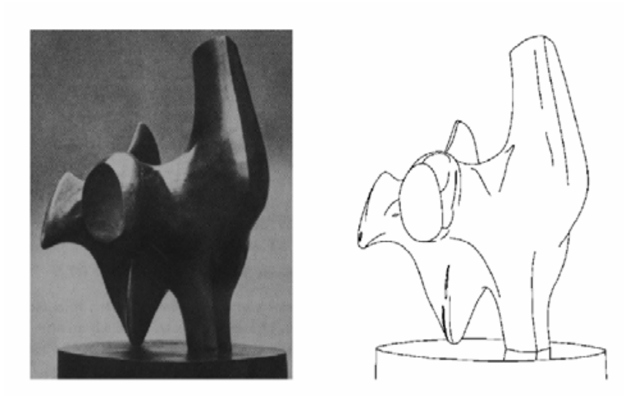
- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

Origin of Edges

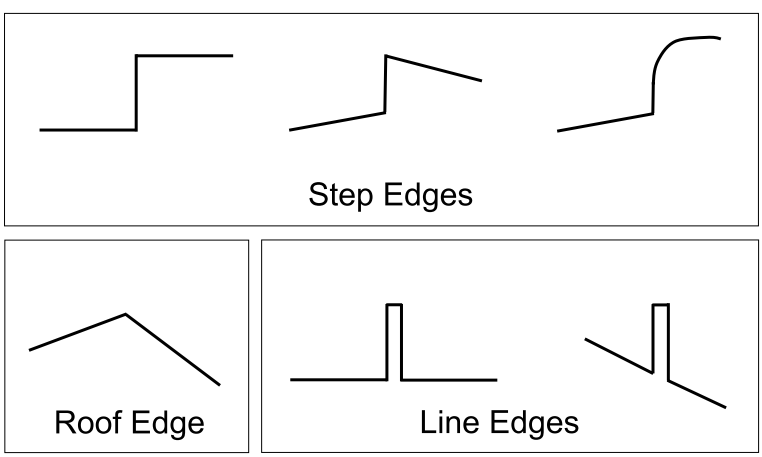


- Edges are caused by a variety of factors

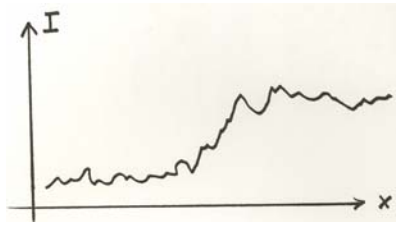
How can you tell that a pixel is on an edge?



Edge Types



Real Edges



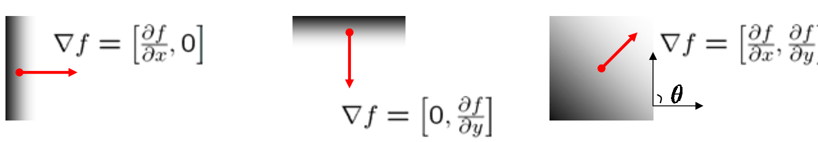
Noisy and Discrete!

We want an **Edge Operator** that produces:

- Edge Magnitude
- Edge Orientation
- High Detection Rate and Good Localization

Gradient

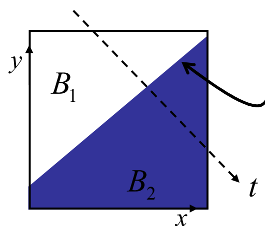
- Gradient equation: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- Represents direction of most rapid change in intensity



- Gradient direction: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
- The *edge strength* is given by the gradient magnitude

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

Theory of Edge Detection



Ideal edge

$$L(x,y) = x\sin\theta - y\cos\theta + \rho = 0$$

$$B_1 : L(x,y) < 0$$

$$B_2 : L(x,y) > 0$$

Unit step function:

$$u(t) = \begin{cases} 1 & \text{for } t > 0 \\ \frac{1}{2} & \text{for } t = 0 \\ 0 & \text{for } t < 0 \end{cases} \qquad u(t) = \int_{-\infty}^{t} \delta(s)\,ds$$

Image intensity (brightness):

$$I(x,y) = B_1 + (B_2 - B_1)\,u(x\sin\theta - y\cos\theta + \rho)$$

Theory of Edge Detection

- Image intensity (brightness):

$$I(x,y) = B_1 + (B_2 - B_1)\,u(x\sin\theta - y\cos\theta + \rho)$$

- Partial derivatives (gradients):

$$\frac{\partial I}{\partial x} = +\sin\theta\,(B_2 - B_1)\,\delta(x\sin\theta - y\cos\theta + \rho)$$

$$\frac{\partial I}{\partial y} = -\cos\theta\,(B_2 - B_1)\,\delta(x\sin\theta - y\cos\theta + \rho)$$

- Squared gradient:

$$s(x,y) = \left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2 = \left[(B_2 - B_1)\,\delta(x\sin\theta - y\cos\theta + \rho)\right]^2$$

Edge Magnitude: $\sqrt{s(x,y)}$

Edge Orientation: $\arctan\left(\frac{\partial I}{\partial y} \Big/ \frac{\partial I}{\partial x}\right)$ (normal of the edge)

Rotationally symmetric, non-linear operator

Theory of Edge Detection

- Image intensity (brightness):

$$I(x,y) = B_1 + (B_2 - B_1)u(x\sin\theta - y\cos\theta + \rho)$$

- Partial derivatives (gradients):

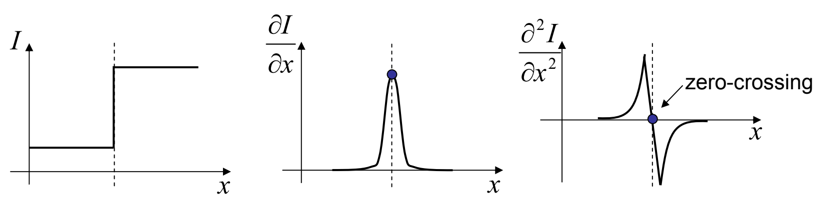
$$\frac{\partial I}{\partial x} = +\sin\theta(B_2 - B_1)\delta(x\sin\theta - y\cos\theta + \rho)$$

$$\frac{\partial I}{\partial y} = -\cos\theta(B_2 - B_1)\delta(x\sin\theta - y\cos\theta + \rho)$$

- Laplacian:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = (B_2 - B_1)\delta'(x\sin\theta - y\cos\theta + \rho)$$

Rotationally symmetric, linear operator



Discrete Edge Operators

- How can we differentiate a ***discrete*** image?

Finite difference approximations:

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i,j+1}\right) + \left(I_{i+1,j} - I_{i,j}\right)\right)$$

$$\frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i+1,j}\right) + \left(I_{i,j+1} - I_{i,j}\right)\right)$$

$I_{i,j+1}$	$I_{i+1,j+1}$
$I_{i,j}$	$I_{i+1,j}$

ε

Convolution masks :

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\begin{array}{|c|c|}\hline -1 & 1 \\ \hline -1 & 1 \\ \hline\end{array} \qquad \frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\begin{array}{|c|c|}\hline 1 & 1 \\ \hline -1 & -1 \\ \hline\end{array}$$

Discrete Edge Operators

- Second order partial derivatives:

$$\frac{\partial^2 I}{\partial x^2} \approx \frac{1}{\varepsilon^2}\left(I_{i-1,j} - 2I_{i,j} + I_{i+1,j}\right)$$

$$\frac{\partial^2 I}{\partial y^2} \approx \frac{1}{\varepsilon^2}\left(I_{i,j-1} - 2I_{i,j} + I_{i,j+1}\right)$$

$I_{i-1,j+1}$	$I_{i,j+1}$	$I_{i+1,j+1}$
$I_{i-1,j}$	$I_{i,j}$	$I_{i+1,j}$
$I_{i-1,j-1}$	$I_{i,j-1}$	$I_{i+1,j-1}$

- **Laplacian** :

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Convolution masks :

$$\nabla^2 I \approx \frac{1}{\varepsilon^2}\begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline\end{array} \quad \text{or} \quad \frac{1}{6\varepsilon^2}\begin{array}{|c|c|c|}\hline 1 & 4 & 1 \\ \hline 4 & -20 & 4 \\ \hline 1 & 4 & 1 \\ \hline\end{array}$$

(more accurate)

The Sobel Operators

- Better approximations of the gradients exist
  - The *Sobel* operators below are commonly used

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline\end{array} \qquad \frac{1}{8}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline\end{array}$$

s_x $\qquad$ s_y

Comparing Edge Operators

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

Good Localization
Noise Sensitive
Poor Detection

Roberts (2 x 2):

0	1
-1	0

1	0
0	-1

Sobel (3 x 3):

-1	0	1
-1	0	1
-1	0	1

1	1	1
0	0	0
-1	-1	1

Sobel (5 x 5):

-1	-2	0	2	1
-2	-3	0	3	2
-3	-5	0	5	3
-2	-3	0	3	2
-1	-2	0	2	1

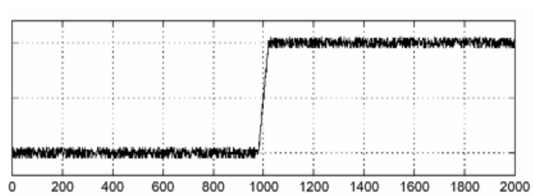
1	2	3	2	1
2	3	5	3	2
0	0	0	0	0
-2	-3	-5	-3	-2
-1	-2	-3	-2	-1

Poor Localization
Less Noise Sensitive
Good Detection

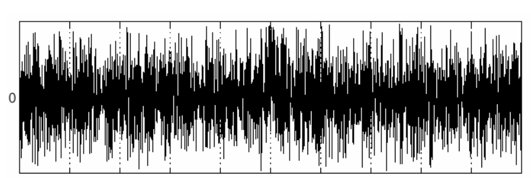
Effects of Noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

$f(x)$

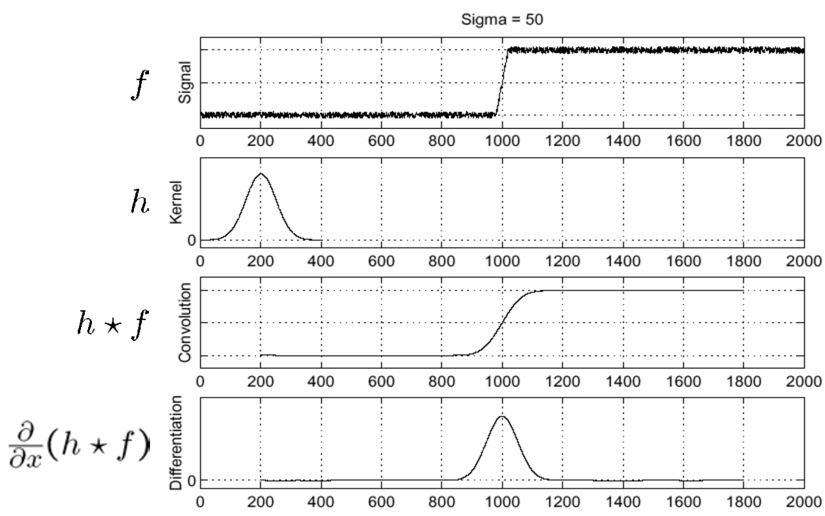


$\frac{d}{dx}f(x)$



Where is the edge??

Solution: Smooth First



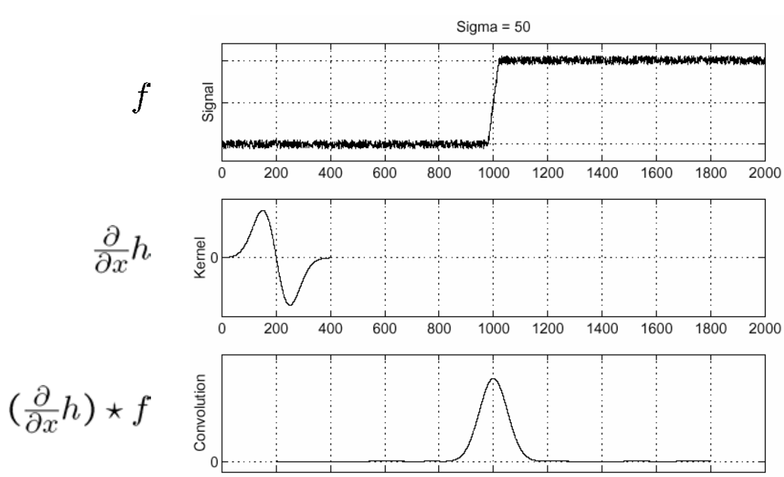
Where is the edge?

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

Derivative Theorem of Convolution

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

…saves us one operation.



Laplacian of Gaussian (LoG)

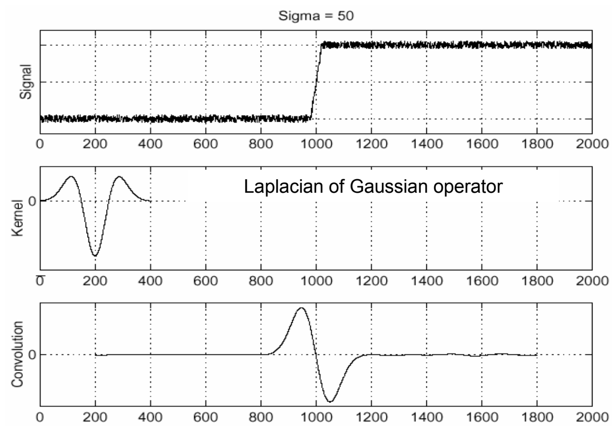
$$\frac{\partial^2}{\partial x^2}(h * f) = \boxed{\left(\frac{\partial^2}{\partial x^2}h\right)} * f$$

Laplacian of Gaussian

f

$\frac{\partial^2}{\partial x^2}h$

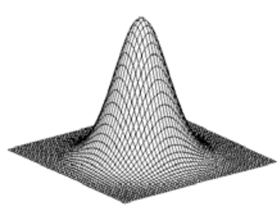
$(\frac{\partial^2}{\partial x^2}h) \star f$

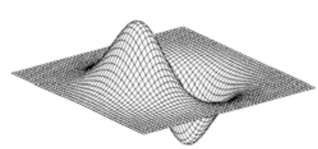


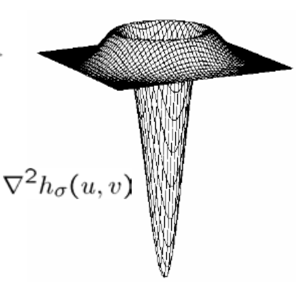
Where is the edge?

Zero-crossings of bottom graph !

2D Gaussian Edge Operators







$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2}e^{-\frac{u^2+v^2}{2\sigma^2}}$$

$$\frac{\partial}{\partial x}h_\sigma(u,v)$$

$$\nabla^2 h_\sigma(u,v)$$

Gaussian

Derivative of Gaussian (DoG)

Laplacian of Gaussian
Mexican Hat (Sombrero)

- ∇^2 is the **Laplacian** operator:

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

Canny Edge Operator

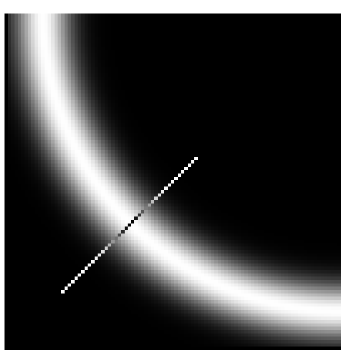
- Smooth image *I* with 2D Gaussian: $G * I$
- Find local edge normal directions for each pixel

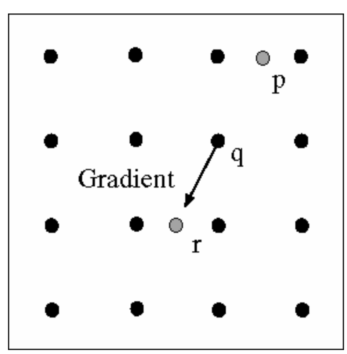
$$\overline{\mathbf{n}} = \frac{\nabla(G * I)}{|\nabla(G * I)|}$$

- Compute edge magnitudes $|\nabla(G * I)|$
- Locate edges by finding zero-crossings along the edge normal directions (**non-maximum suppression**)

$$\frac{\partial^2 (G * I)}{\partial \overline{\mathbf{n}}^2} = 0$$

Non-maximum Suppression





- Check if pixel is local maximum along gradient direction
  - requires checking interpolated pixels p and r

The Canny Edge Detector

original image (Lena)

The Canny Edge Detector

magnitude of the gradient

The Canny Edge Detector

After non-maximum suppression

Canny Edge Operator

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

- The choice of σ depends on desired behavior
  - large σ detects large scale edges
  - small σ detects fine features

Difference of Gaussians (DoG)

- Laplacian of Gaussian can be approximated by the difference between two different Gaussians

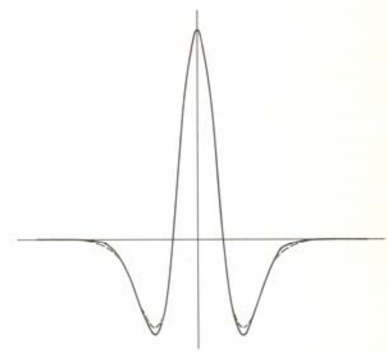
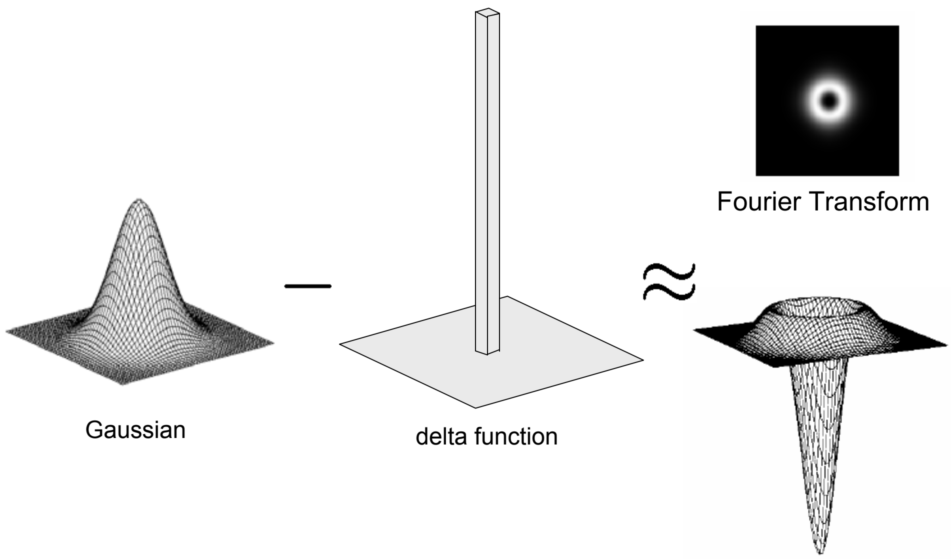


Figure 2–16. The best engineering approximation to $\nabla^2 G$ (shown by the continuous line), obtained by using the difference of two Gaussians (DOG), occurs when the ratio of the inhibitory to excitatory space constraints is about 1:1.6. The DOG is shown here dotted. The two profiles are very similar. (Reprinted by permission from D. Marr and E. Hildreth, "Theory of edge detection," *Proc. R. Soc. Lond. B 204,* pp. 301–328.)

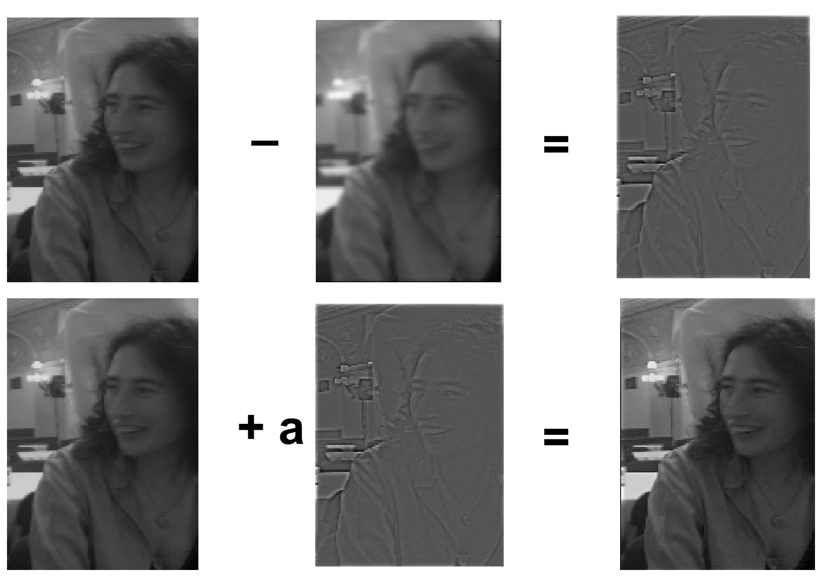
DoG Edge Detection

(a) $\sigma = 1$ (b) $\sigma = 2$ (b)-(a)

Gaussian – Image filter



Unsharp Masking



MATLAB demo

```
g = fspecial('gaussian',15,2);
imagesc(g)
surfl(g)
gclown = conv2(clown,g,'same');
imagesc(conv2(clown,[-1 1],'same'));
imagesc(conv2(gclown,[-1 1],'same'));
dx = conv2(g,[-1 1],'same');
imagesc(conv2(clown,dx,'same');
lg = fspecial('log',15,2);
lclown = conv2(clown,lg,'same');
imagesc(lclown)
imagesc(clown + .2*lclown)
```

Edge Thresholding

- Standard Thresholding:

$$E(x,y) = \begin{cases} 1 & \text{if } \|\nabla f(x,y)\| > T \text{ for some threshold } T \\ 0 & \text{otherwise} \end{cases}$$

  - Can only select “strong” edges.
  - Does not guarantee “continuity”.

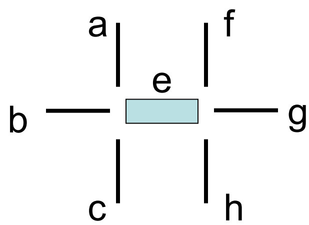
- Hysteresis based Thresholding (use two thresholds)

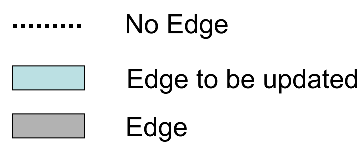
$$\begin{array}{rl} \|\nabla f(x,y)\| \geq t_1 & \text{definitely an edge} \\ t_0 \geq \|\nabla f(x,y)\| < t_1 & \text{maybe an edge, depends on context} \\ \|\nabla f(x,y)\| < t_0 & \text{definitely not an edge} \end{array}$$

Example: For “maybe” edges, decide on the edge if neighboring pixel is a strong edge.

Edge Relaxation

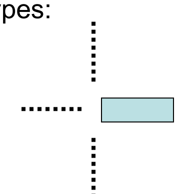
- Parallel – Iterative method to adjust edge values on the basis of neighboring edges.



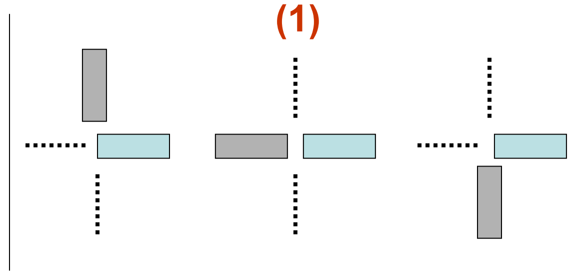


- Vertex Types:

(0)



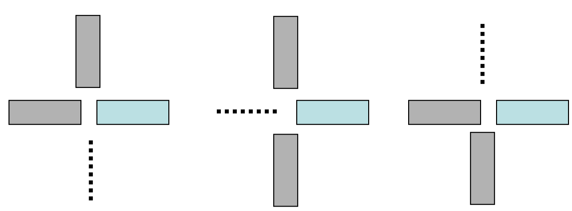
(1)



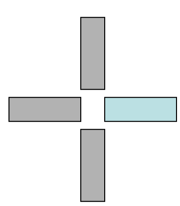
Edge Relaxation

- Vertex Types (continued):

(2)



(3)



Edge Relaxation Algorithm

- Action Table:

Edge Type	Decrement	Increment	Leave as is
	0 - 0	1 - 1	0 - 1
	0 - 2	1 - 2	2 - 2
	0 - 3	1 - 3	2 - 3
			3 - 3

- Algorithm:

Step 0: Compute Initial Confidence of each edge e:

$$C^0(e) = \frac{Magnitude\ of\ e}{Maximum\ Gradient\ in\ image}$$

Step 1: Initialize $k = 1$

Step 2: Compute Edge Type of each edge e

Step 3: Modify confidence $C^k(e)$ based on $C^{k-1}(e)$ and Edge Type

Step 4: Test to see if all $C^k(e)'s$ have CONVERGED to either 1 or 0. Else go to Step 2.

Edge Relaxation

Fig. 3.22 Edge relaxation results. (a) Raw edge data. Edge strengths have been thresholded at 0.25 for display purposes only. (b) Results after five iterations of relaxation applied to (a). (c) Different version of (a). Edge strengths have been thresholded at 0.25 for display purposes only. (d) Results after five iterations of relaxation applied to (c).

Next Class

- Image Resampling
- Multi-resolution Image Pyramids