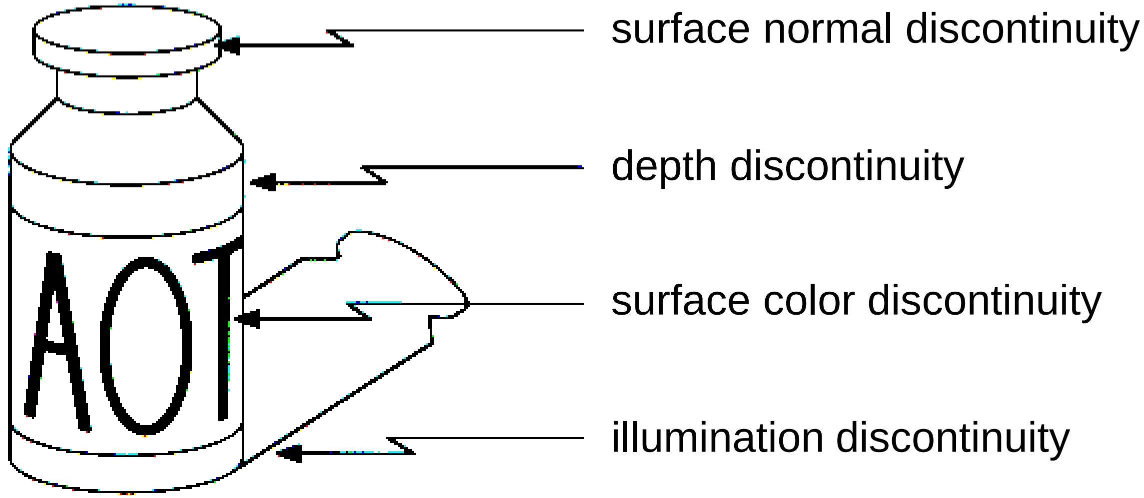
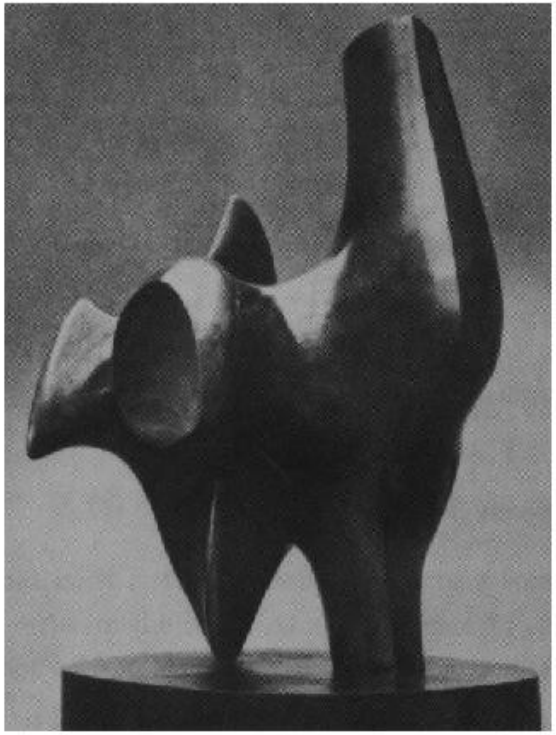
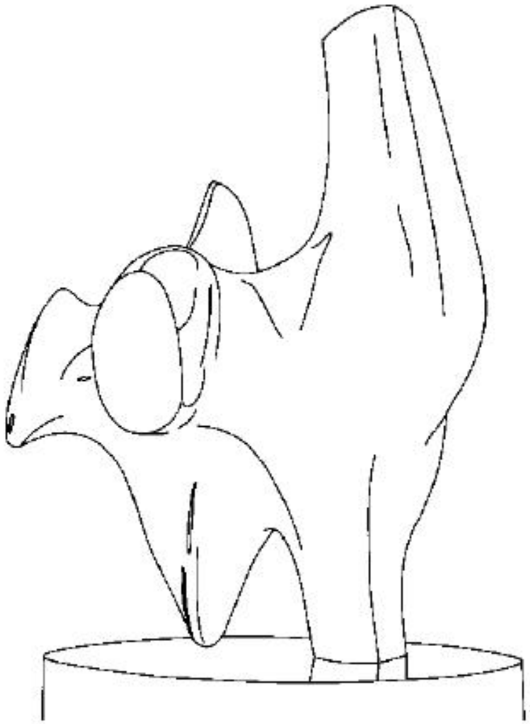
Edge Detection

Origin of Edges



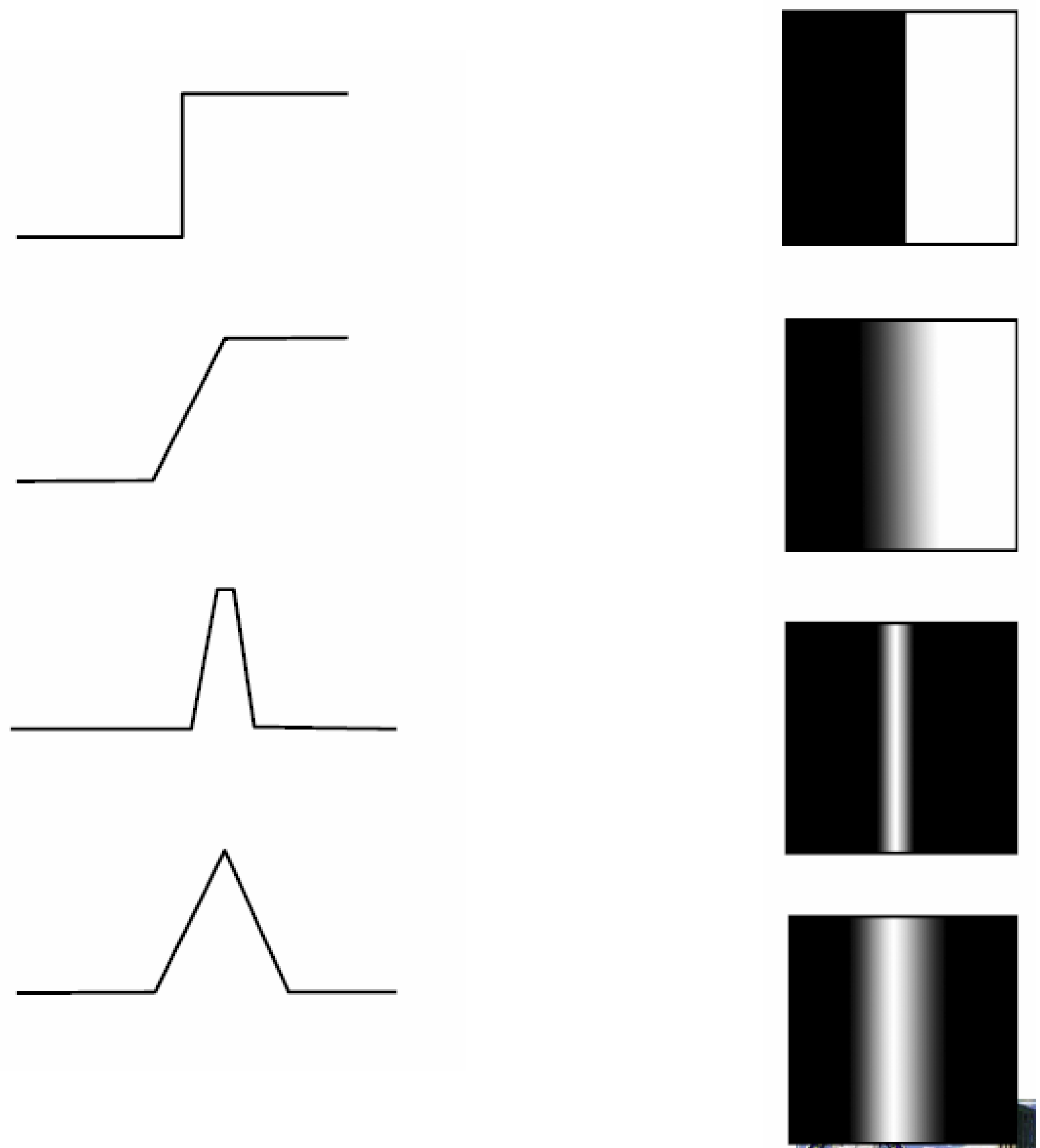
- Edges are caused by a variety of factors

Edge detection

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

Profiles of image intensity edges



Edge is Where Change Occurs

- Change is measured by derivative in 1D
- Biggest change, derivative has maximum magnitude
- Or 2^{nd} derivative is zero.

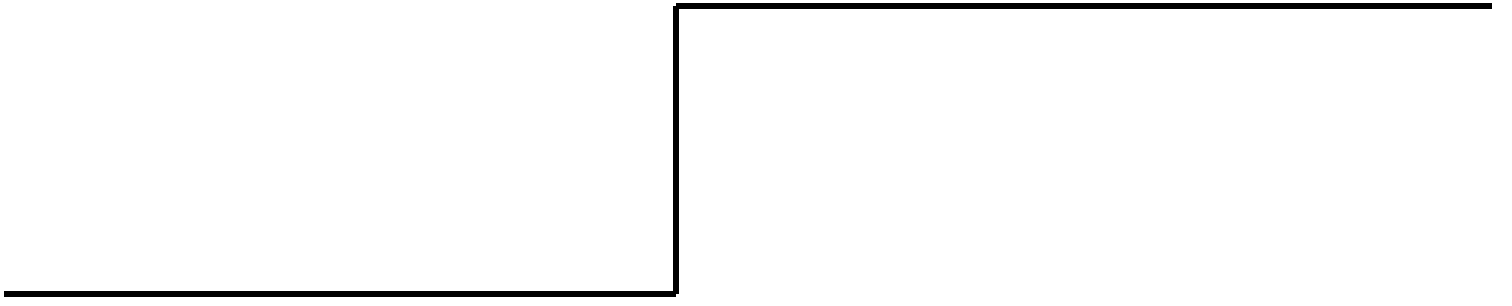
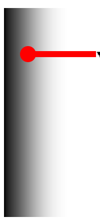


Image gradient

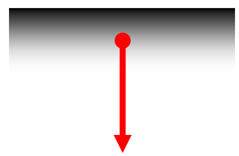
- The gradient of an image:

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

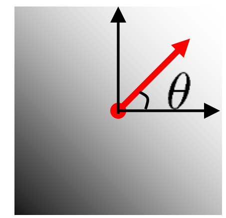
- The gradient points in the direction of most rapid change in intensity



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



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



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

The gradient direction is given by:

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

- how does this relate to the direction of the edge?

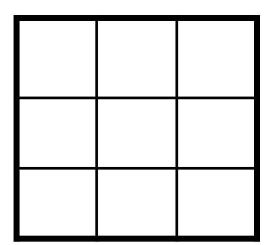
The *edge strength* is given by the gradient magnitude

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

The discrete gradient

- How can we differentiate a *digital* image f[x,y]?
  - Option 1: reconstruct a continuous image, then take gradient
  - Option 2: take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x, y] \approx f[x+1, y] - f[x, y]$$



H

The Sobel operator

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

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$s_x \qquad\qquad s_y$$

- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient value, however

Gradient operators

(a)

Δ_1

0	1
-1	0

Δ_2

1	0
0	-1

(b)

Δ_1

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

Δ_2

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

(c)

Δ_1

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

Δ_2

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

(d)

Δ_1

-3	-1	1	3
-3	-1	1	3
-3	-1	1	3
-3	-1	1	3

Δ_2

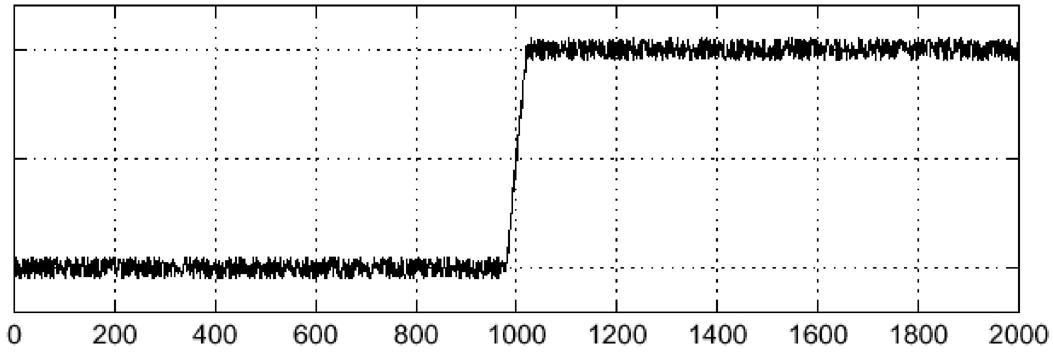
3	3	3	3
1	1	1	1
-1	-1	-1	-1
-3	-3	-3	-3

(a): Roberts' cross operator (b): 3x3 Prewitt operator
(c): Sobel operator (d) 4x4 Prewitt operator

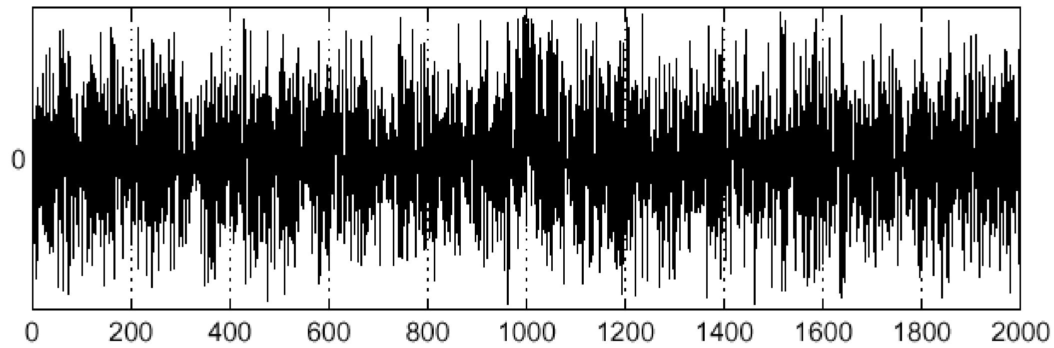
Effects of noise

- 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?

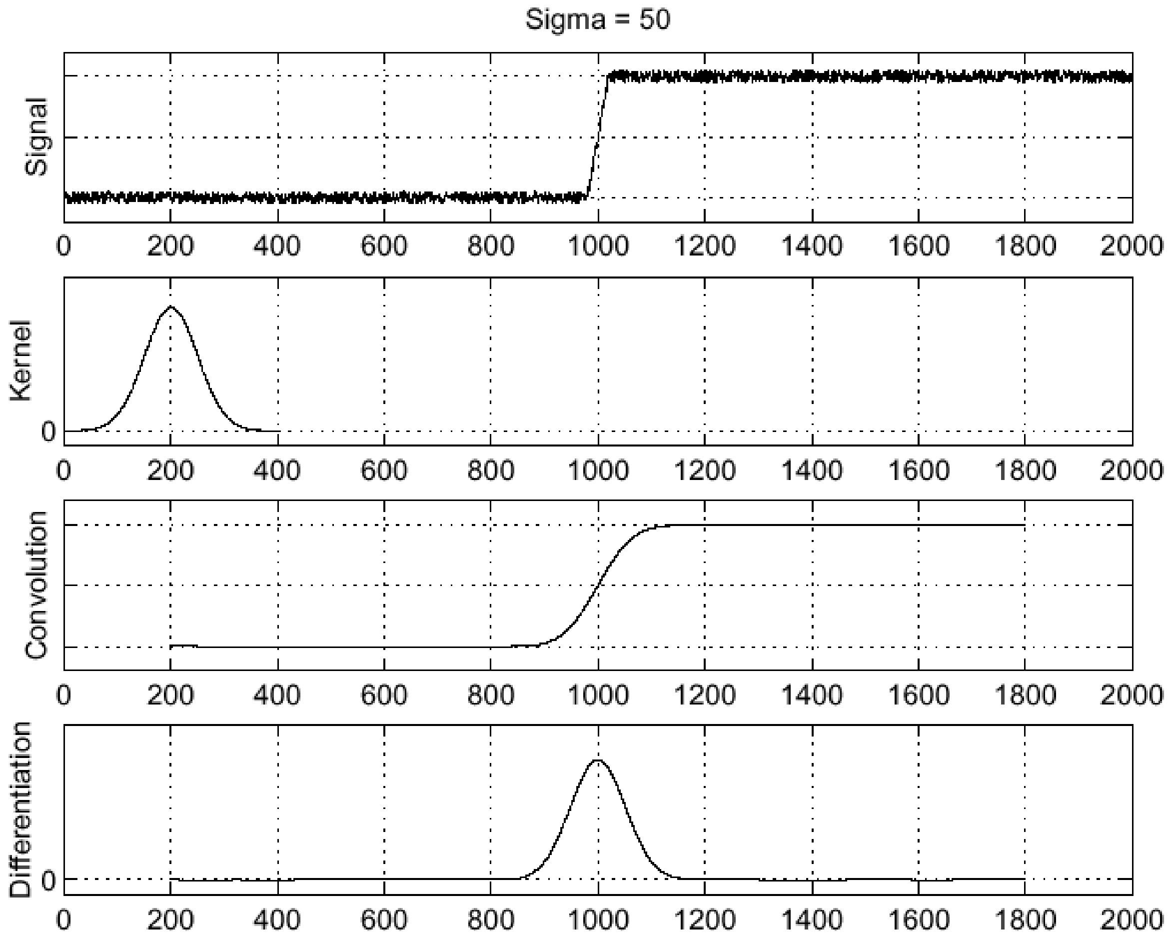
Solution: smooth first

f

h

$h \star f$

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

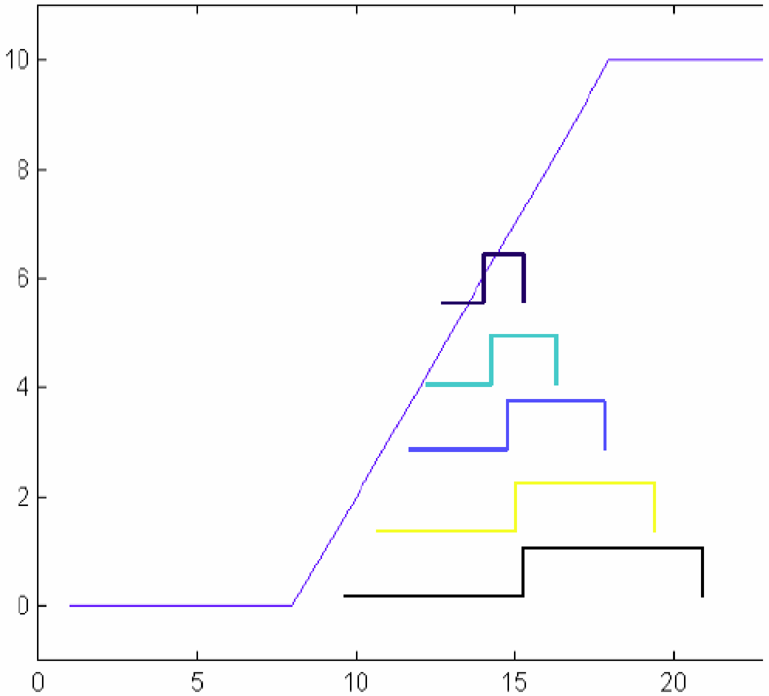


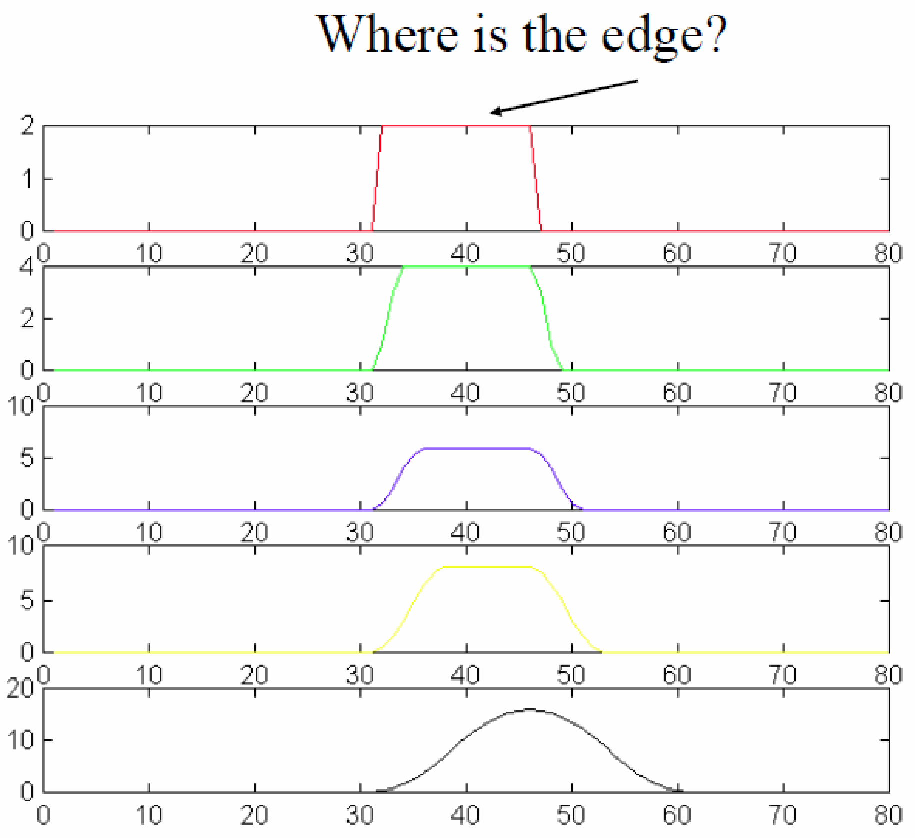
Where is the edge?

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

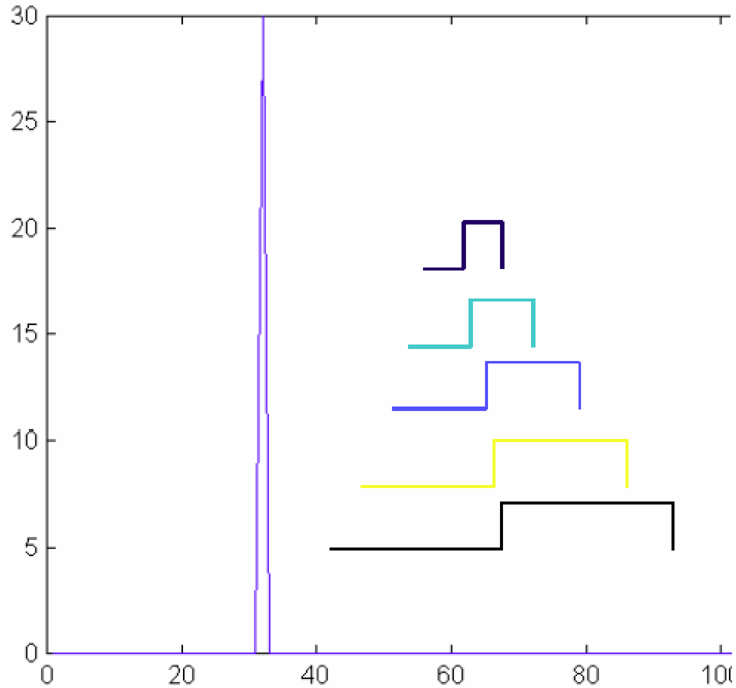
Multiple Scale

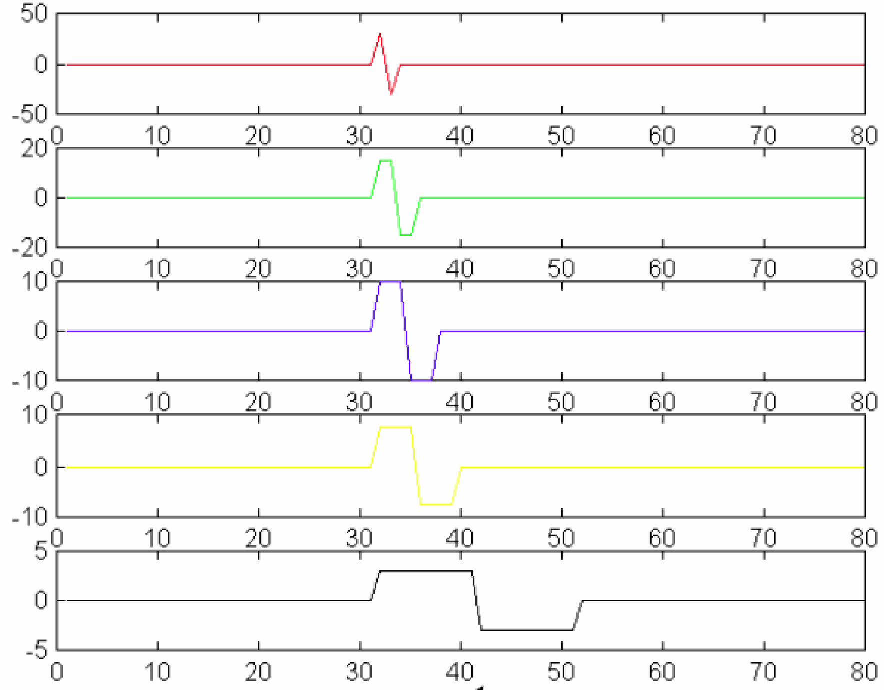
How to determine the size of smoothing windows?





Multiple Scale



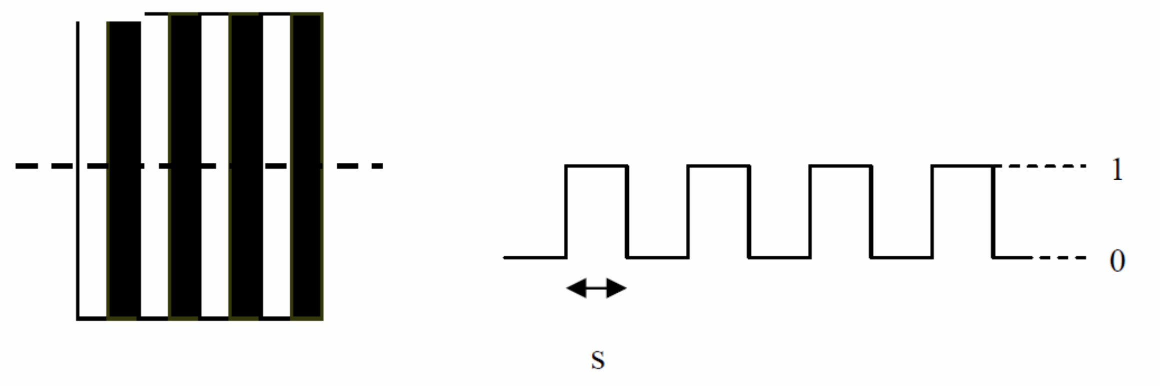


Bad localization

Where is the edge?

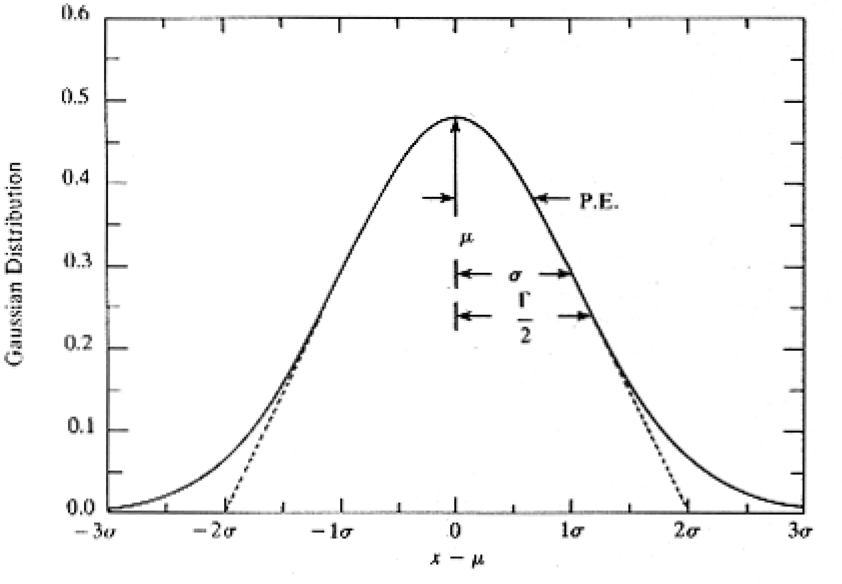
Assignment 3

- Q2. Consider the zebra stripe as noise. If we don’t smooth it. Edge detector [-1 1] will get strong response at the intersection of black and white.
- So, we want to smooth it with gaussian filter $e^{-\frac{x^2}{2\sigma^2}}$ to reduce the response to 10%.



Q2 -Gaussian Filter

- You need to get the sigma value by run a loop to try different sigma to get the qualified one.
  - Crate the stripe image and the program in matlab
  - No need to submit the code.
- The filter's window size can be set to 6*sigma to get most of the gaussian energy inside of the window.



Q2-Gaussian Filter

- To create the 2D gaussian filter, you can se matlab's fspecial function.
- Eg: with sigma = 1 and 6x6 size
  - G = fspecial('gaussian',[6 6],1);
- Filter it
  - New = imfilter(I,G,'same');
  - 'same': the output array is the same size as the input array
- To plot relationship between s and sigma, use plot(S,Sigma), where S and Sigma are vector of same size

Q2-Update

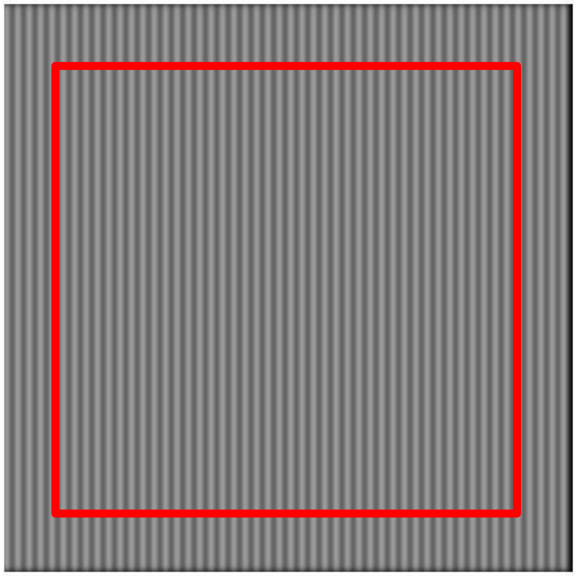
- Here’s codes to generate a 100x100s striped image, you can change the dimension to meet your needs.

```
function stripe(s)
stripe = [ ];
sub = ones(100,s); %one stripe
for i=1:100
    % if i is even number we add black stripe, otherwise, white
    color = mod(i,2);
    stripe = [stripe, color*sub];
end;
imshow(stripe);
end
```

Q2-Update

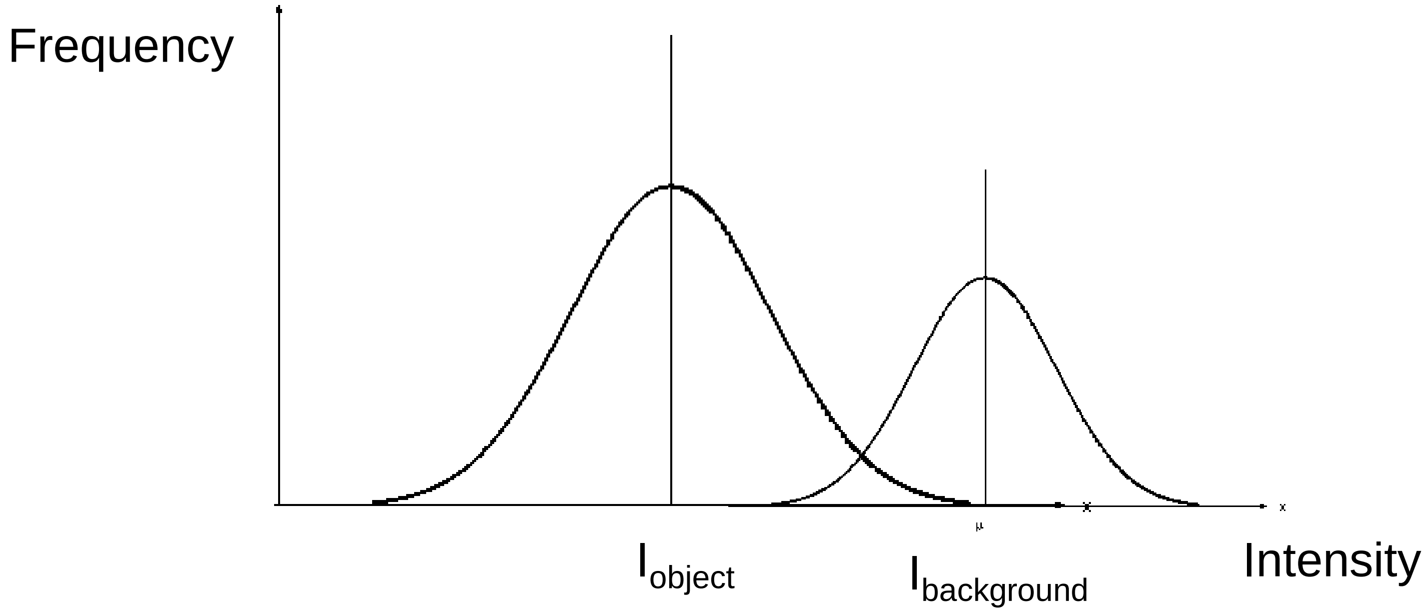
- When calculate the maximum response, you can ignore the boundary pixels, i.e. just focus on the pixels that have enough neighbors covered by filter kernel.

- Sigma should be precise to .1
- You can use ‘break’, to stop loop once you find proper sigma value

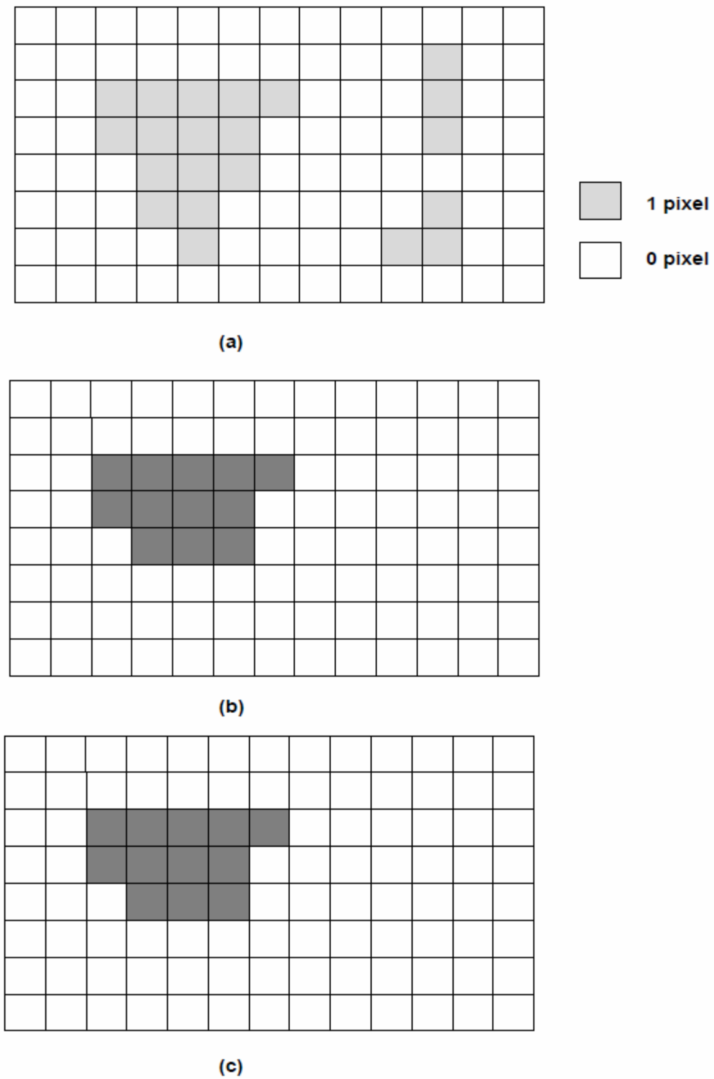


Q1 Segmentation

- Object and background intensity functions
- Where to cut to minimize wrong classification (i.e. classify background as object and vice versa)?



Midterm Review: Q1&Q2



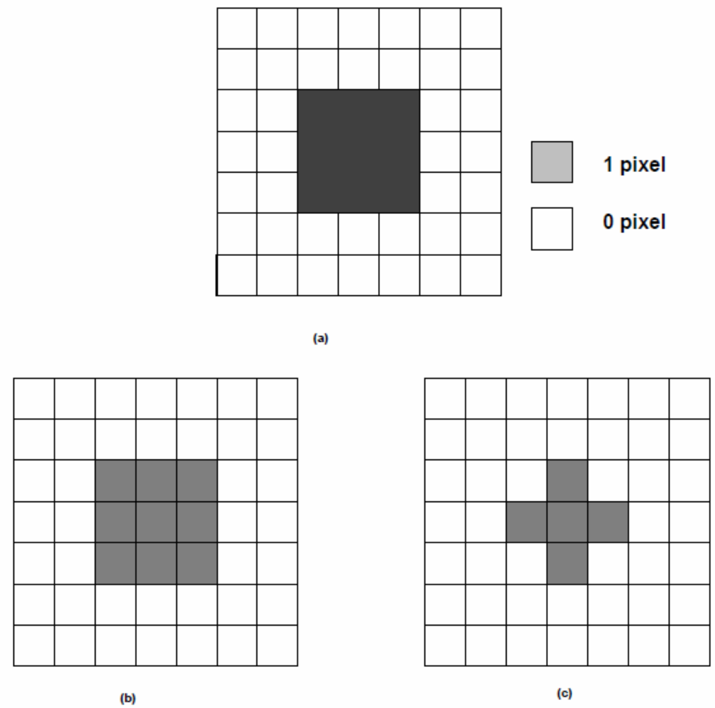


Figure 2:

Q3

- Because brightness =0.5I, that means the N dot L =0.5. i.e. The angle between surface normal and light direction is 60 degree.
  The shape of the iso-brightness points on the surface is a circle.
  The equation of the circle is
- $x^2 + y^2 = 7.9555$
  z = (30+5*sqrt(52))/16= 4.1285

Q4

- Left: camera as origin; Right: view point as origin
- $$\begin{pmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & -10 \\ 0 & 0 & 0 & 1 \end{pmatrix} or \begin{pmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$