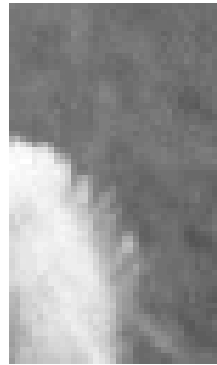
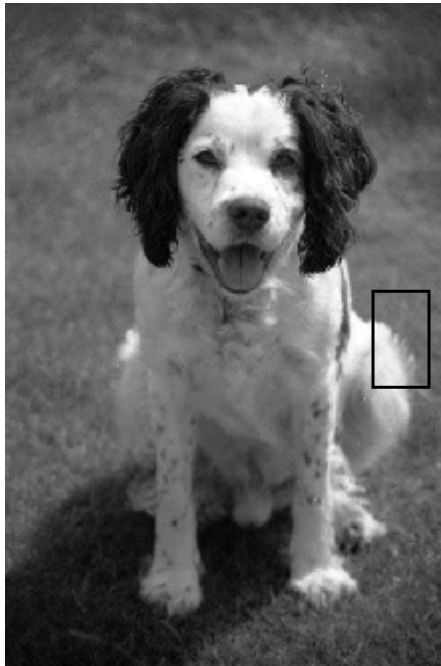
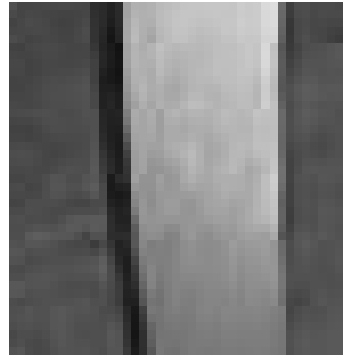


Boundary Detection - Edges

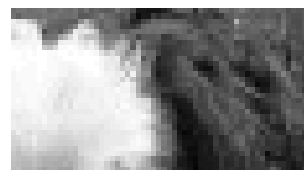
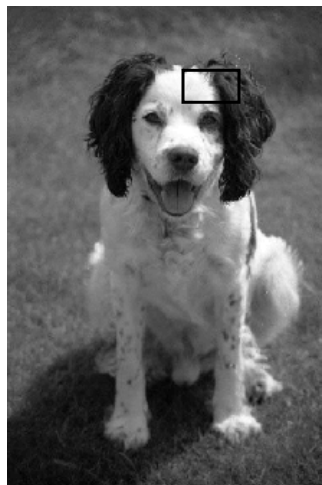
- Edge Detection is a local (for now) decision about whether there is a boundary in an image.
- Boundaries of objects
 - Usually different materials/orientations, intensity changes.



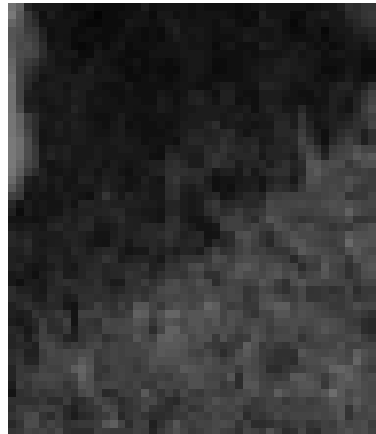
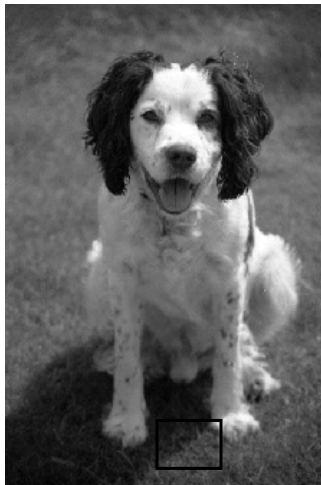
We also get:
Boundaries of surfaces



Boundaries of materials
properties

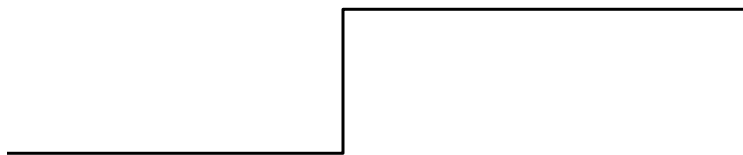


Boundaries of lighting



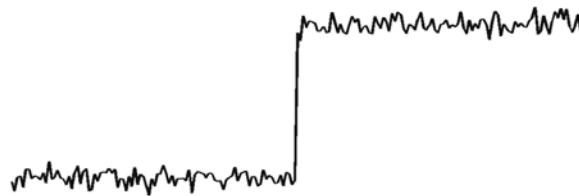
Edge is Where Change Occurs

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



Noisy Step Edge

- Derivative is high everywhere.
- Must smooth before taking derivative.



Implementing 1D Edge Detection

1. Filter out noise: correlate with Gaussian
 2. Take a derivative: correlate with $[-.5 \ 0 \ .5]$
- We can combine 1 and 2.

How do we take the first derivative with a convolution? Let $y = f(x)$. Recall that $y' = \lim_{dx \rightarrow 0} \frac{f(x+dx) - f(x)}{dx}$. In a discrete image, the smallest we can make dx is 1 pixel, so we can take $f(x+1) - f(x)$, which is correlation with a filter of $[-1 \ 1 \ 0]$. This is asymmetric, we could just as easily use a filter like $[0 \ -1 \ 1]$. And, it's also reasonable to say: $y' = \lim_{dx \rightarrow 0} \frac{f(x+dx) - f(x-dx)}{2dx}$, which leads to a filter of $[-.5 \ 0 \ .5]$. In the limit, these are the same, but before that they are different.

Implementing 1D Edge Detection

3. Find the peak: Two issues:
 - Should be a local maximum.
 - Should be sufficiently high.

Scale

- Smoothing more removes small scale structures.
- Varying smoothing varies the scale of edges we locate.
- *Matlab*

2D Edge Detection: Canny

1. Filter out noise
 - Use a 2D Gaussian Filter. $J = G \circ I$
2. Take a derivative

$\nabla J = (J_x, J_y) = \left(\frac{\partial J}{\partial x}, \frac{\partial J}{\partial y} \right)$ is the Gradient

$\|\nabla J\| = \sqrt{J_x^2 + J_y^2}$ tells how fast image changes

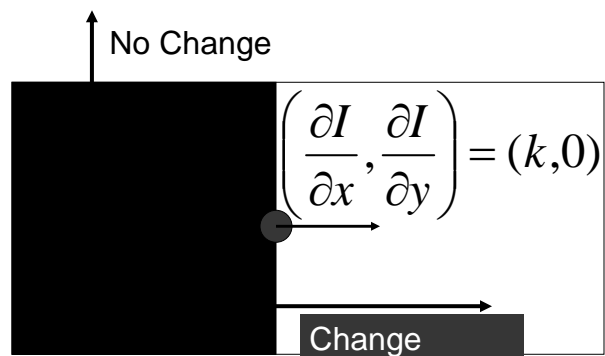
$\frac{\nabla J}{\|\nabla J\|}$ is the direction of fastest change.

Take a derivative - discretely

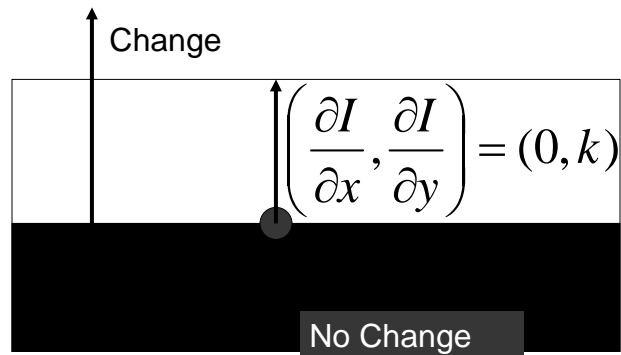
$$\frac{\partial I}{\partial x} \approx \begin{bmatrix} -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \otimes I \quad (\text{Just like in 1D})$$

$$\frac{\partial I}{\partial y} \approx \begin{bmatrix} -\frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix} \otimes I$$

What is the gradient?



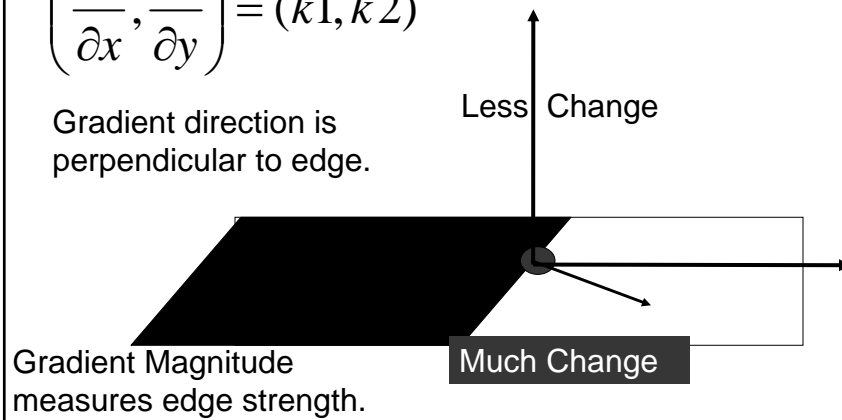
What is the gradient?



What is the gradient?

$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) = (k_1, k_2)$$

Gradient direction is perpendicular to edge.



Gradient – More formal

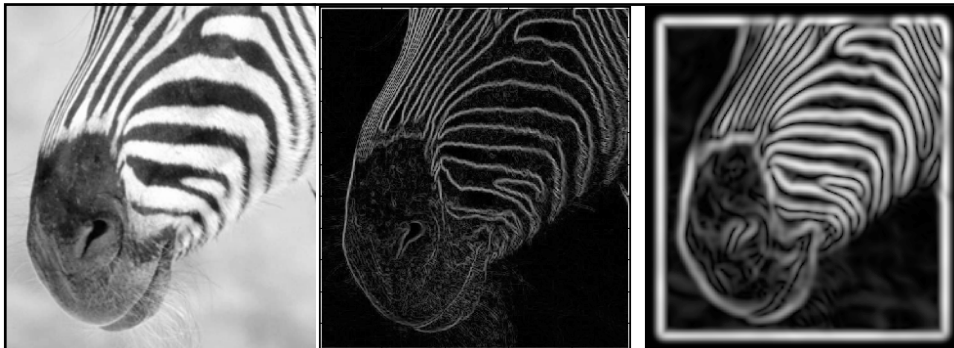
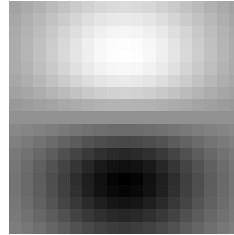
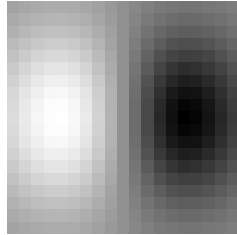
- Suppose we move Δ in direction θ .
- Motion: $v = (\Delta \cos \theta, \Delta \sin \theta)$
- Intensity change: $\langle v, \nabla I \rangle$
- Recall $\langle v, w \rangle = \|v\| \|w\| \cos \alpha$.
- So $\langle v, \nabla I \rangle$ is maximized when v is in the direction of ∇I .
- In that direction, rate of image change is: $\|\nabla I\|$.

Gradient and Convolution

- How do we compute gradients with convolution?
- $\frac{\partial I}{\partial x}$ = derivative in x direction
• Correlation with $[-.5, 0, .5]$
- Likewise, correlation with $[-.5; 0; .5]$ for derivative in y direction.

Smoothing and Differentiation

- Need two derivatives, in x and y direction.
- We can use a derivative of Gaussian filter
 - because differentiation is convolution, and convolution is associative



Scale

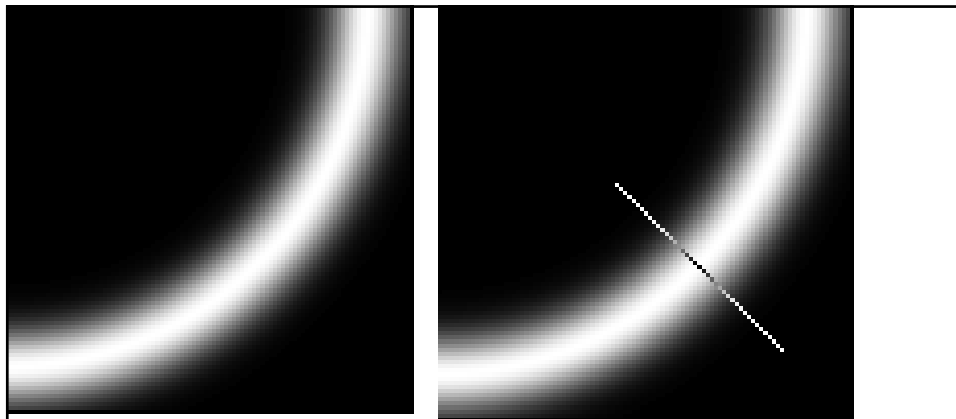
Smoothing

- Eliminates noise edges.
- Makes edges smoother.
- Removes fine detail.
- *Matlab*

(Forsyth & Ponce)

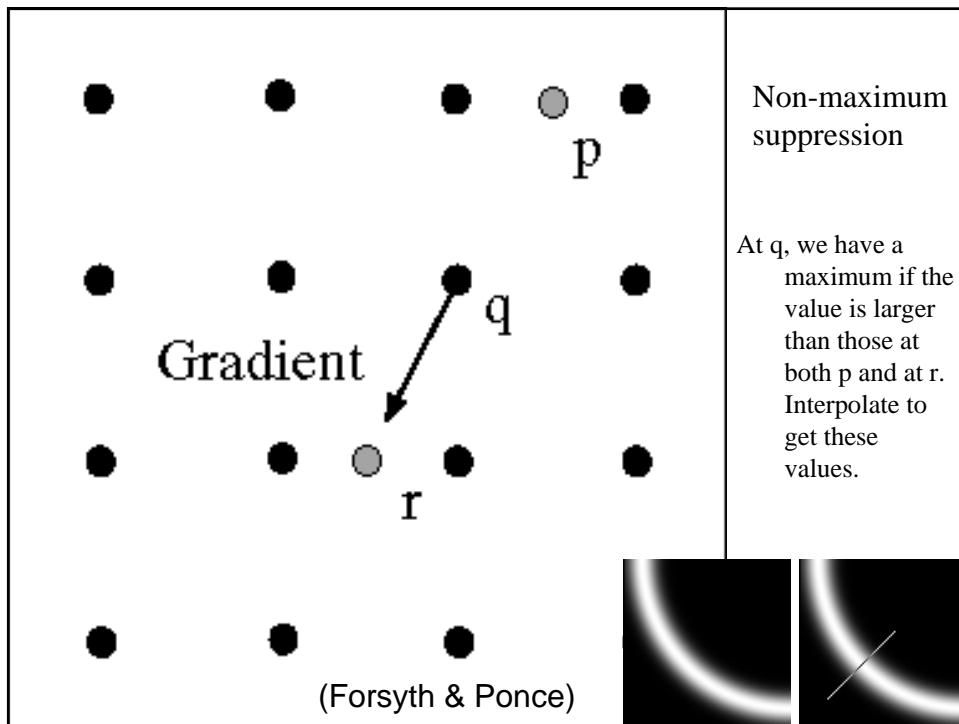
Finding the Peak

- 1) The gradient magnitude is large along thick trail; how do we identify the significant points?
- 2) How do we link the relevant points up into curves?



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.

(Forsyth & Ponce)



Linear Interpolation

- Given a function defined at two points, $f(0)$, $f(1)$, we want to find values for intermediate points, eg., $f(x)$, $0 < x < 1$.
- Can take weighted average:

$$f(x) = (1-x)*f(0) + x*f(1) = f(0) + x(f(1)-f(0))$$
- This is equation for line with slope $f(1)-f(0)$.

Bilinear Interpolation – 4 points

- Given values at $(0,0)$, $(1,0)$, $(0,1)$, $(1,1)$ find value at (x,y) .
- Linearly interpolate $(x,0)$, $(x,1)$, then interpolate (x,y) .
- Or, find $(0,y)$ and $(1,y)$ and interpolate.
- These produce same results.

If we interpolate to get $f(x,0) = (1-x)f(0,0) + xf(1,0)$, $f(x,1) = (1-x)f(0,1) + xf(1,1)$. Then $f(x,y) = ((1-x)f(0,0) + xf(1,0))(1-y) + (f(x,1) = (1-x)f(0,1) + xf(1,1))y$.

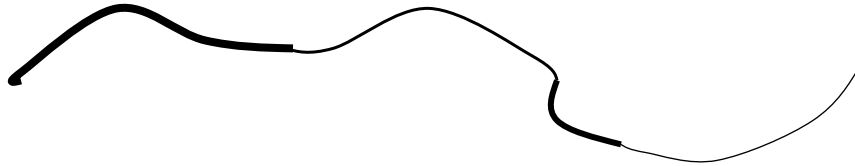
If we interpolate to get $f(0,y) = (1-y)f(0,0) + yf(0,1)$, $f(1,y) = (1-y)f(1,0) + yf(1,1)$. Then

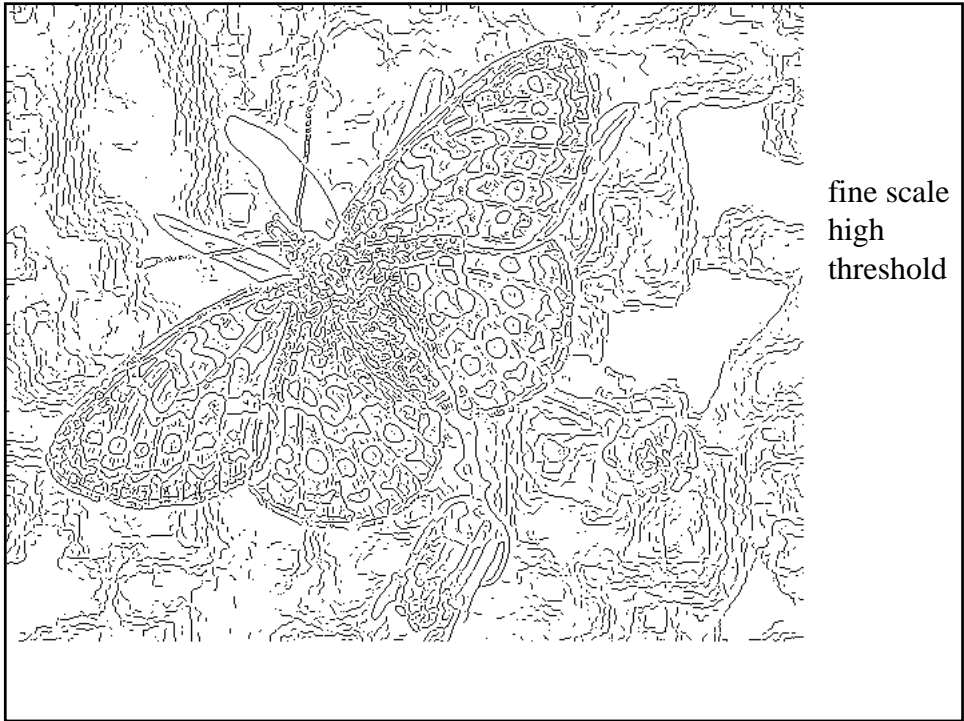
$$f(x,y) = ((1-y)f(0,0) + yf(0,1))(1-x) + ((1-y)f(1,0) + yf(1,1))x$$

These are the same.

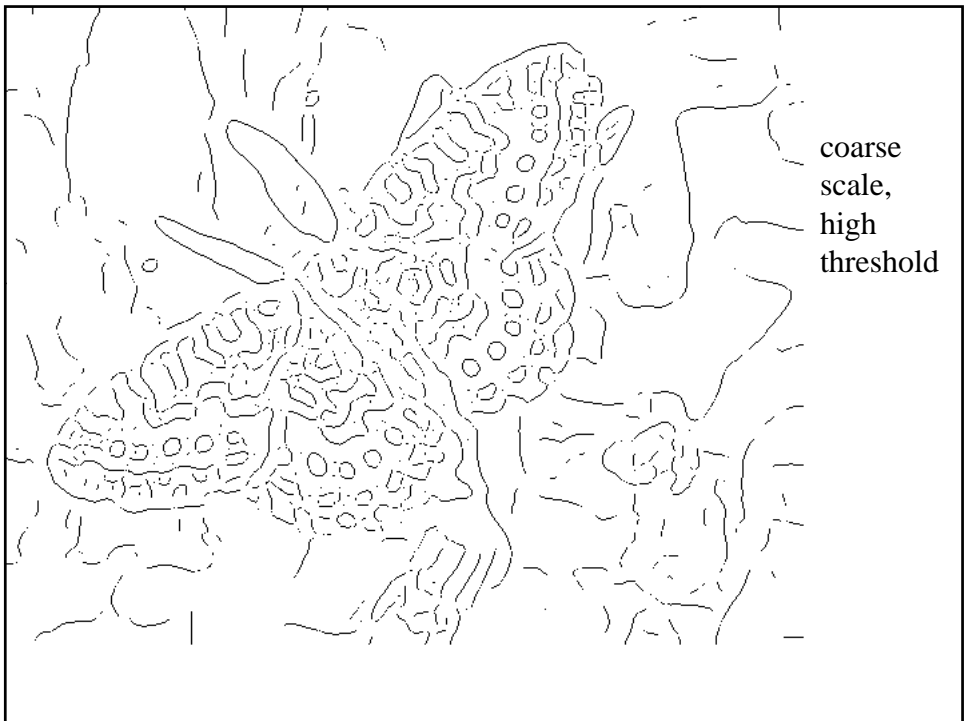
Hysteresis

- 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.

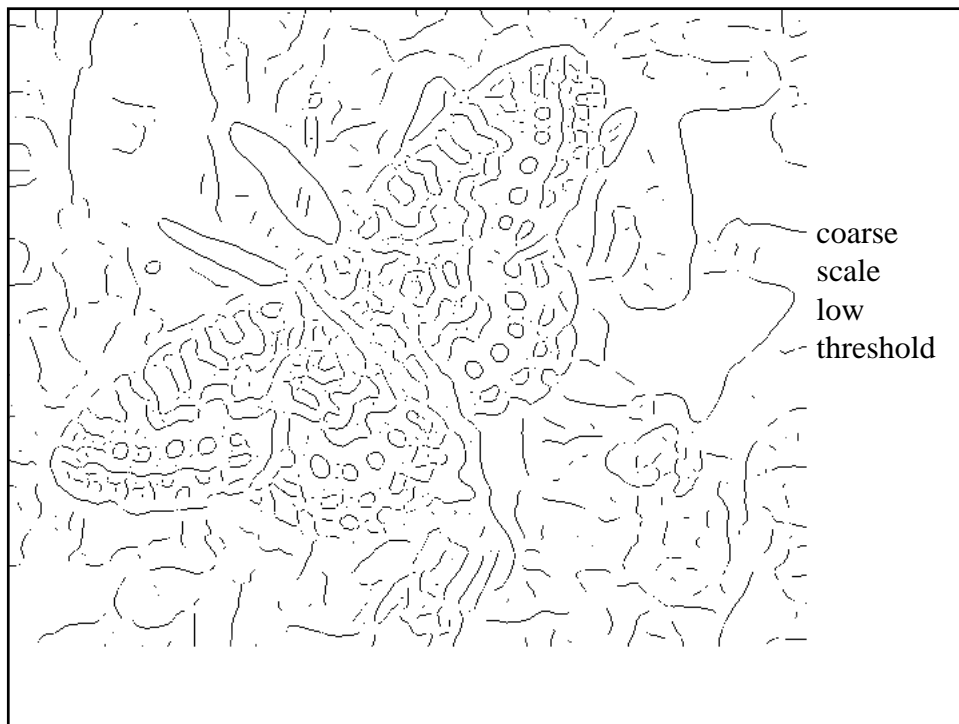




fine scale
high
threshold



coarse
scale,
high
threshold



Demo of Edge Detection

What is missing from Canny?

- Texture.
- Scale selection
- Learning for specific domains
- Edge classification