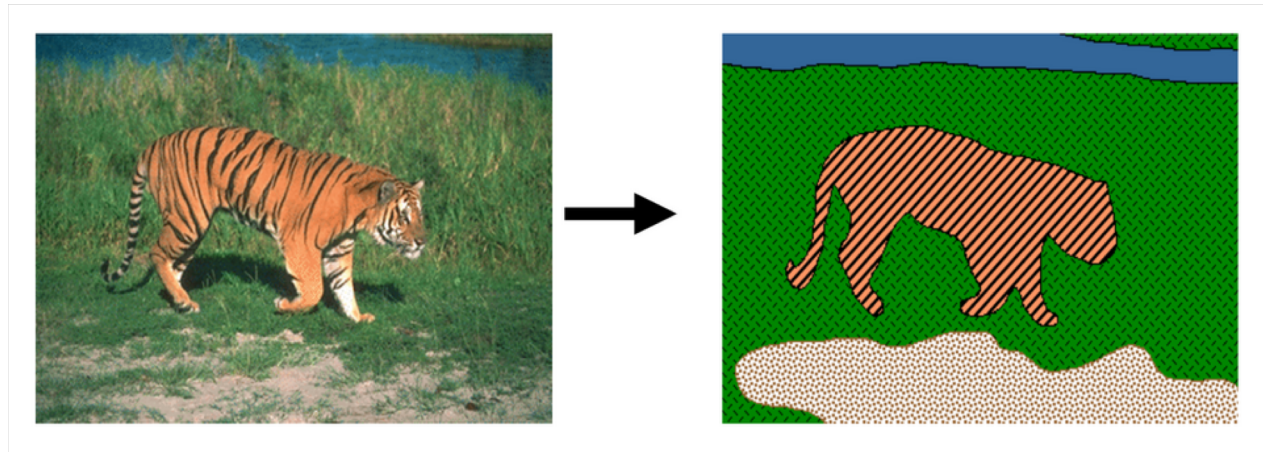


Grouping

Grouping

- Grouping pixels into objects (“perceptual organization”)

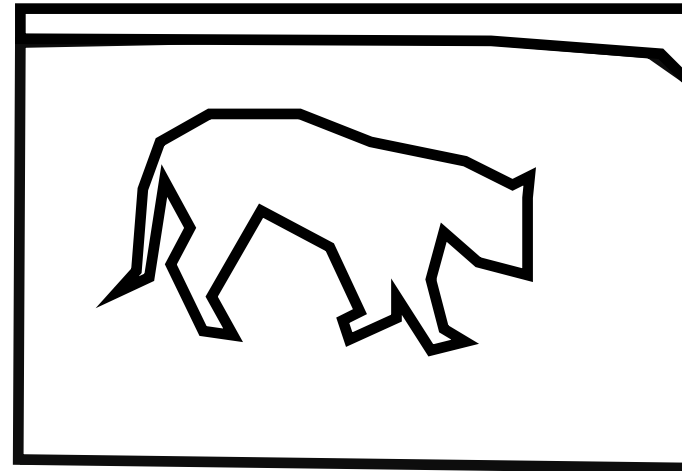
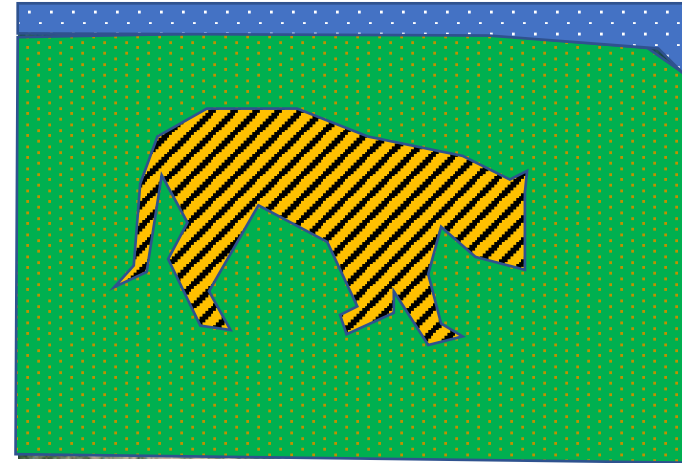


Why grouping?

- Pixels property of sensor, not world
- Reasoning at object level (might) make things easy:
 - objects at consistent depth
 - objects can be recognized
 - objects move as one

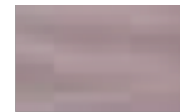
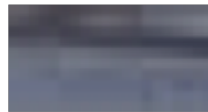
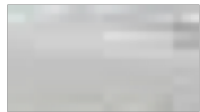
*"I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of colour. Do I have "327"? No. I have sky, house, and trees."
Max Wertheimer*

Regions \leftrightarrow Boundaries

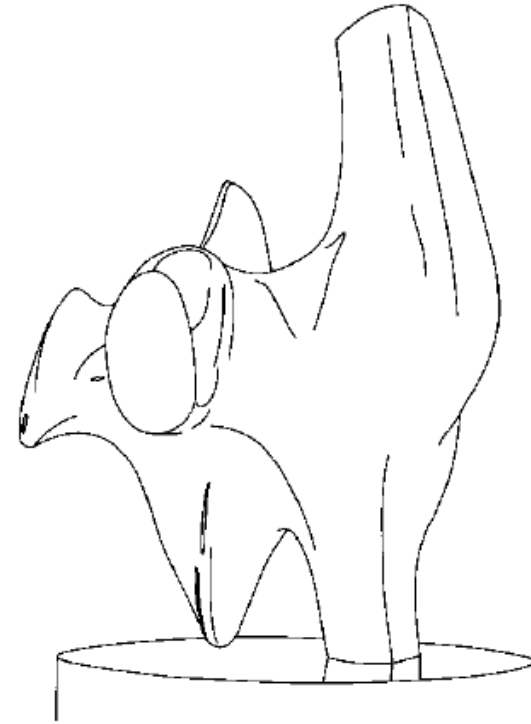
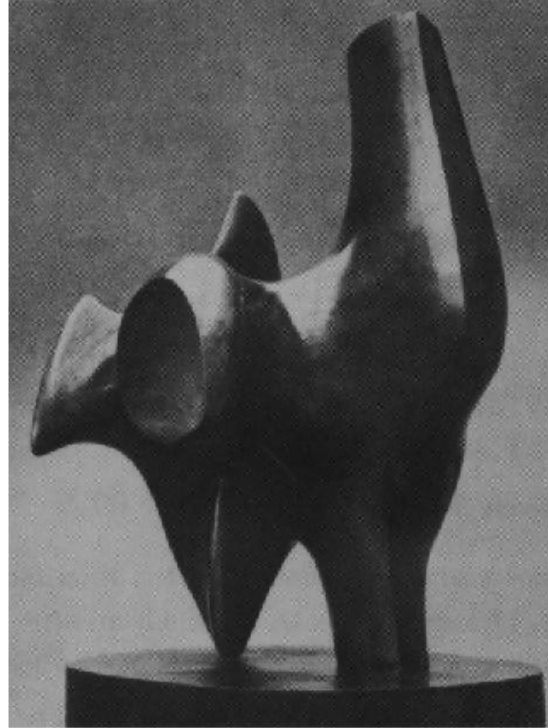


Why edges?

- Resilience to lighting and color
 - useful for recognition, matching patches across images



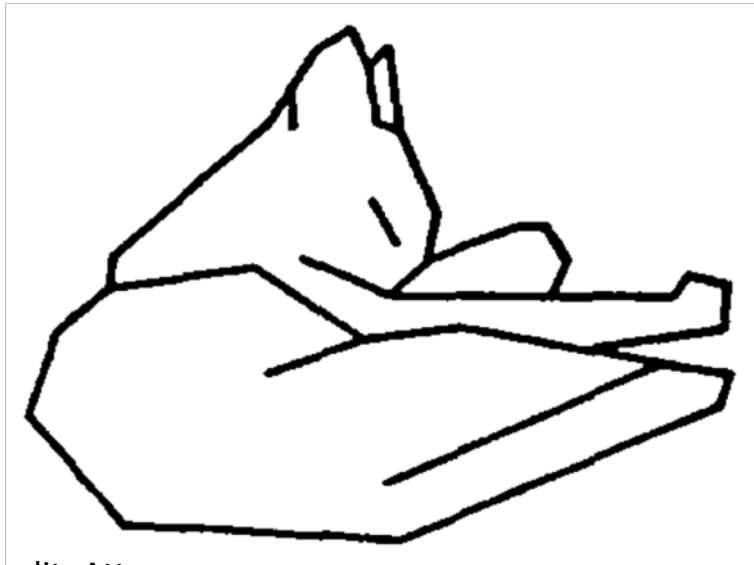
Why edges?



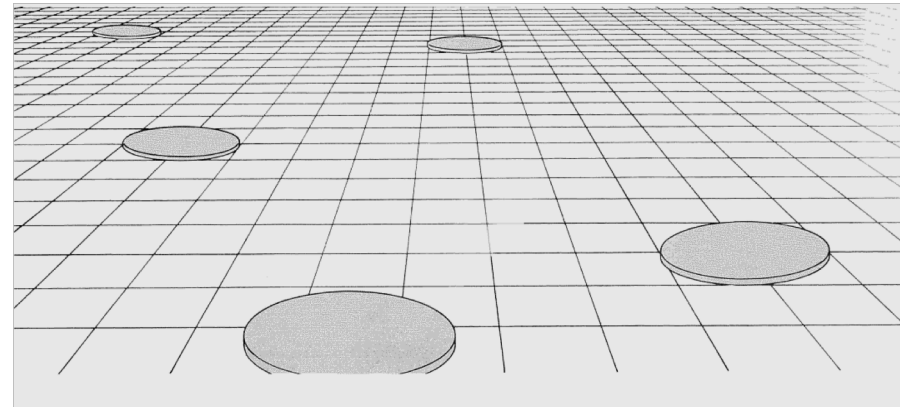
- Humans are sensitive to edges
- Convert a 2D image into a set of curves
 - Extracts salient features of the scene, more compact

Why edges?

- Cue to shape and geometry
 - useful for recognition, understanding 3D structure



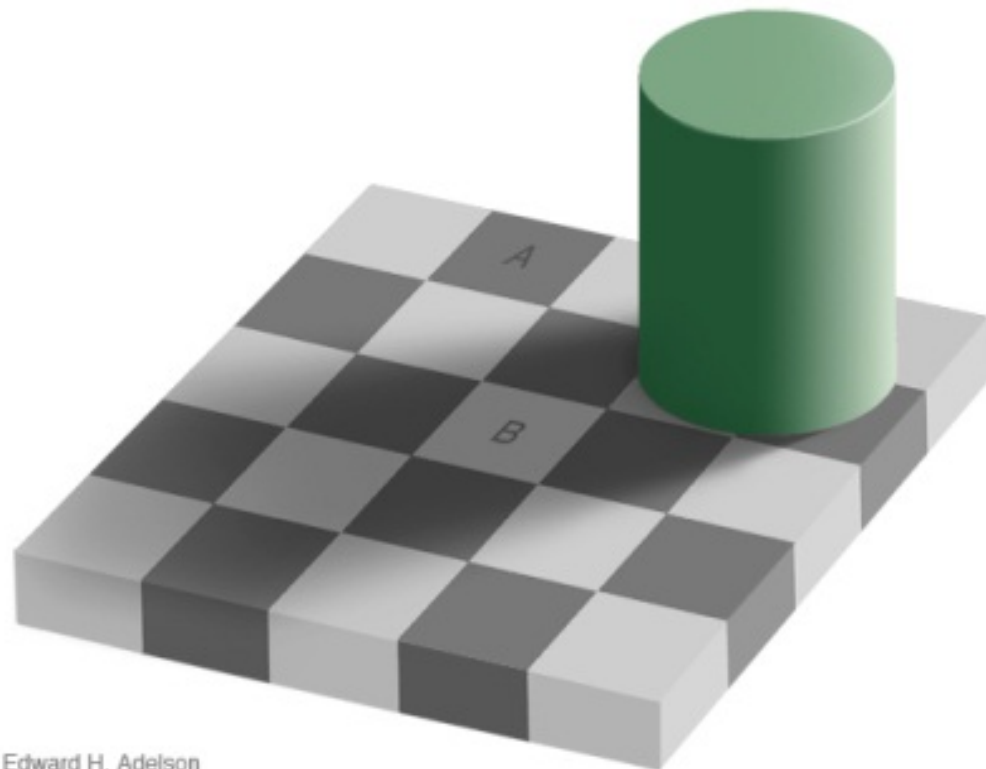
Credit: Attneave



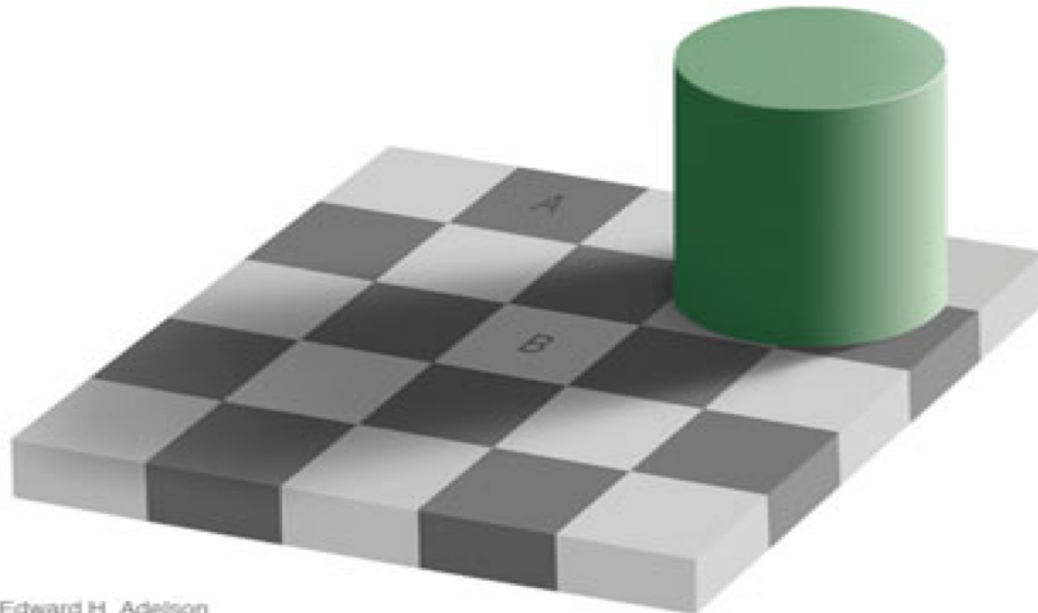
Credit: Jitendra Malik

Edges

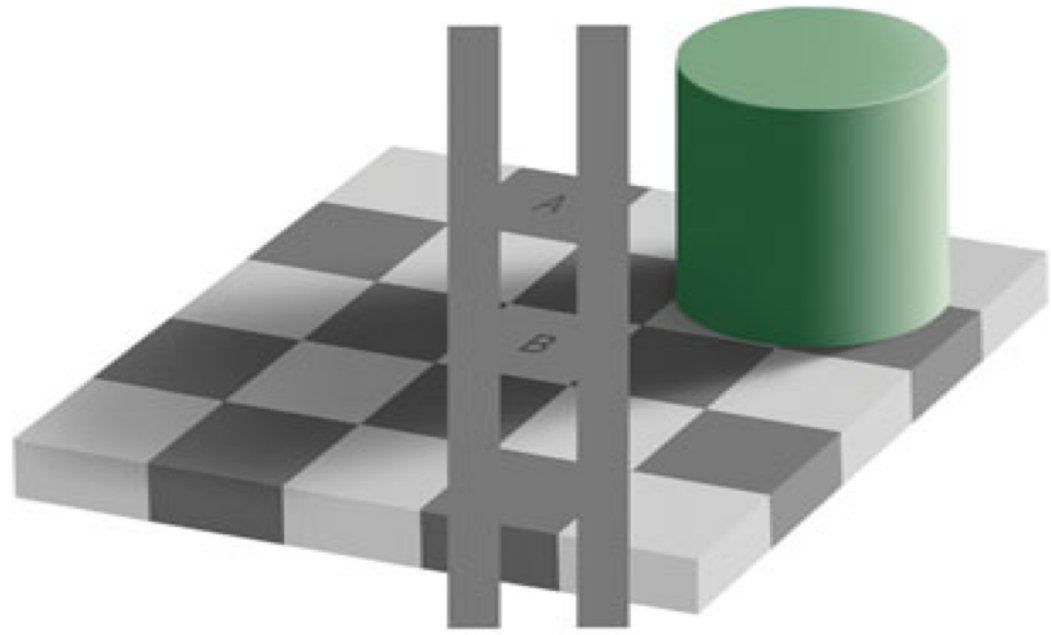
- Edges are curves in the image, across which the brightness changes “a lot”
- Corners/Junctions



Aside



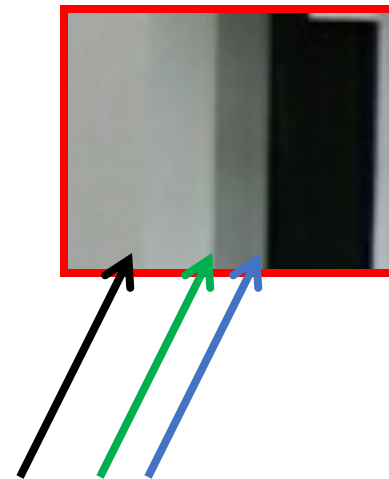
Edward H. Adelson



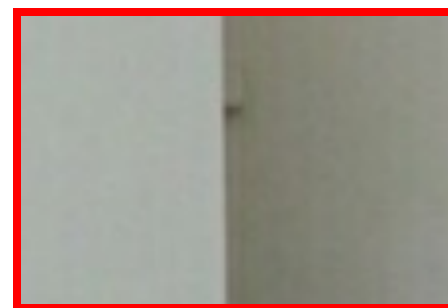
Closeup of edges



Closeup of edges

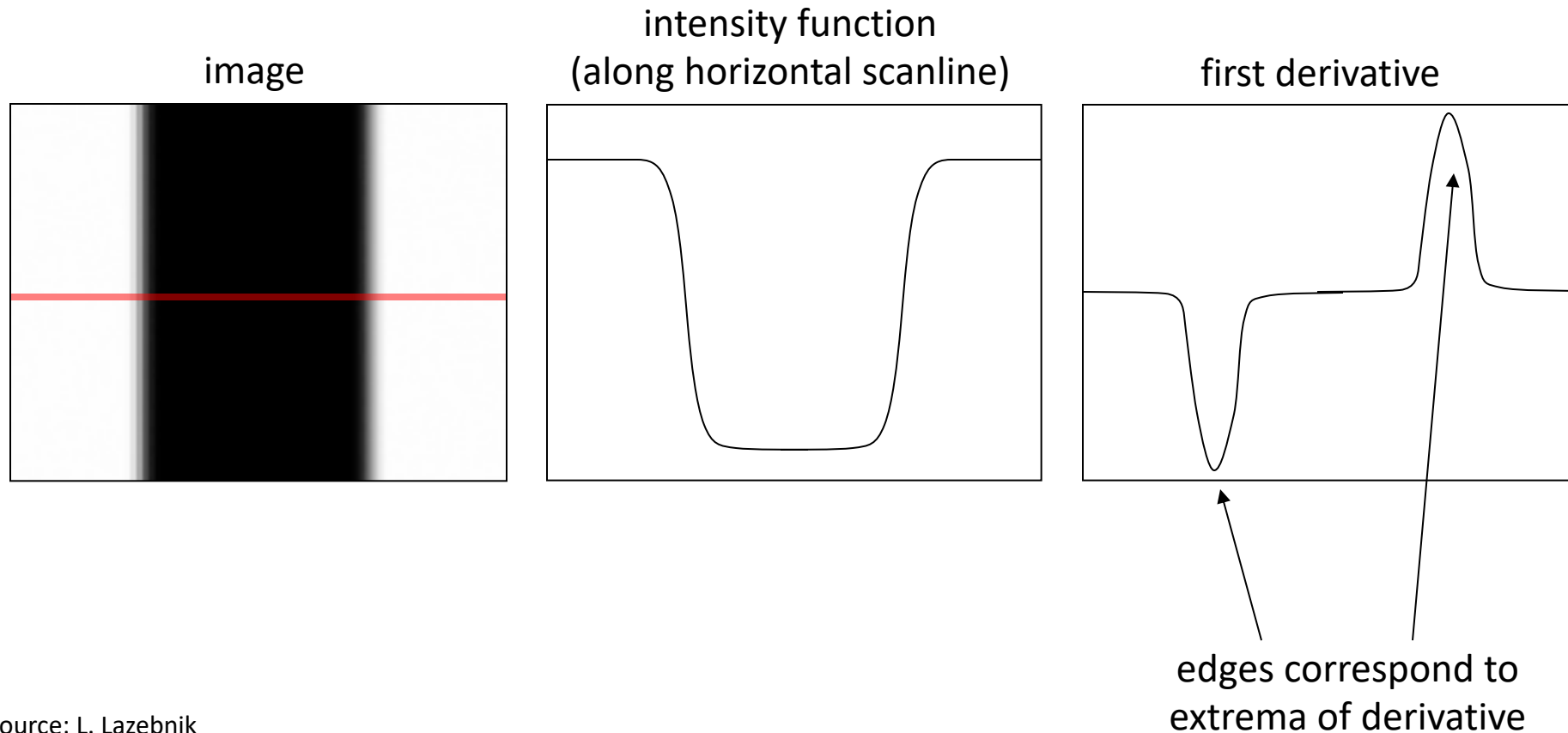


Closeup of edges



Characterizing edges

- An edge is a place of *rapid change* in the image intensity function



Derivatives and convolution

- Differentiation is *linear*

$$\frac{\partial(a f(x) + b g(x))}{\partial x} = a \frac{\partial f(x)}{\partial x} + b \frac{\partial g(x)}{\partial x}$$

- Differentiation is *shift-invariant*
 - Derivative of shifted signal is shifted derivative
- Hence, differentiation can be represented as convolution!

Image derivatives

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

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

How would you implement this as a linear filter?

$$\frac{\partial f}{\partial x} : \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array}$$

H_x

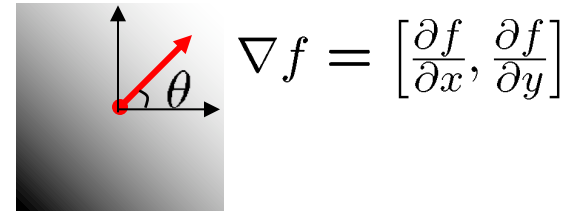
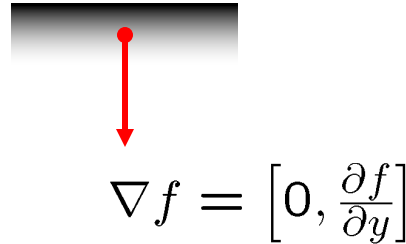
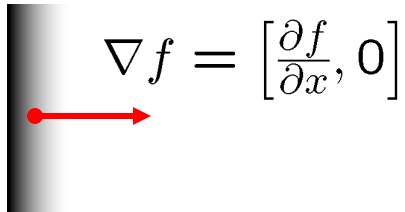
$$\frac{\partial f}{\partial y} : \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array}$$

H_y

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 increase in intensity



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 gradient direction is given by:

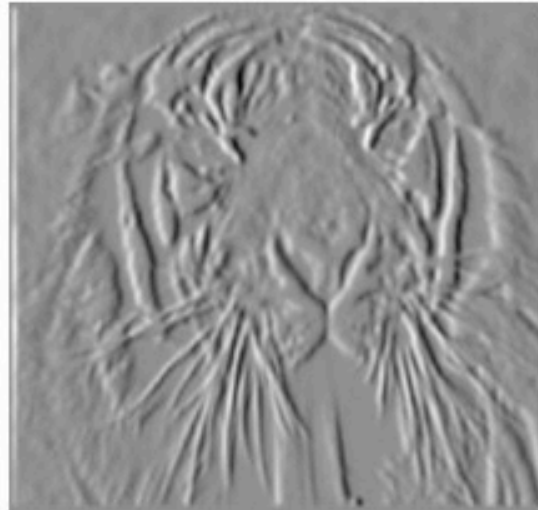
$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$

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

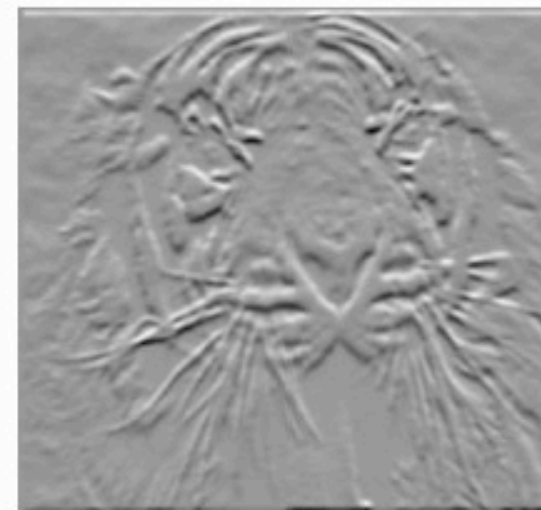
Image gradient



Gradient magnitude

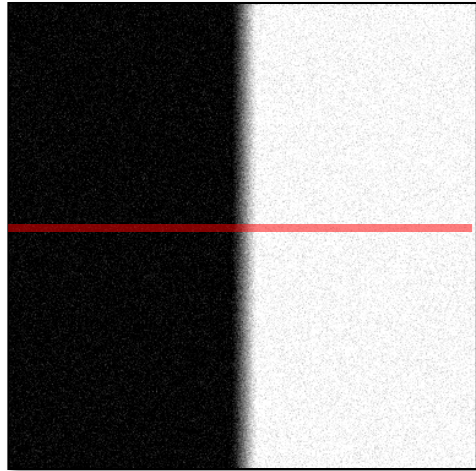


Gradient in x



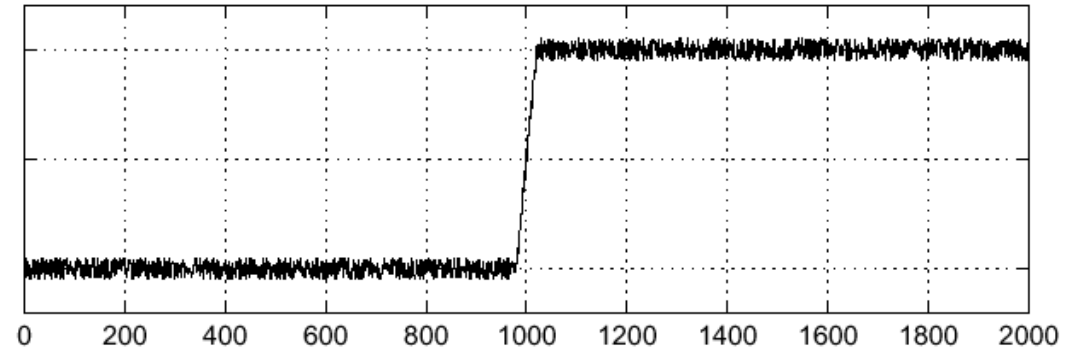
Gradient in y

Effects of noise

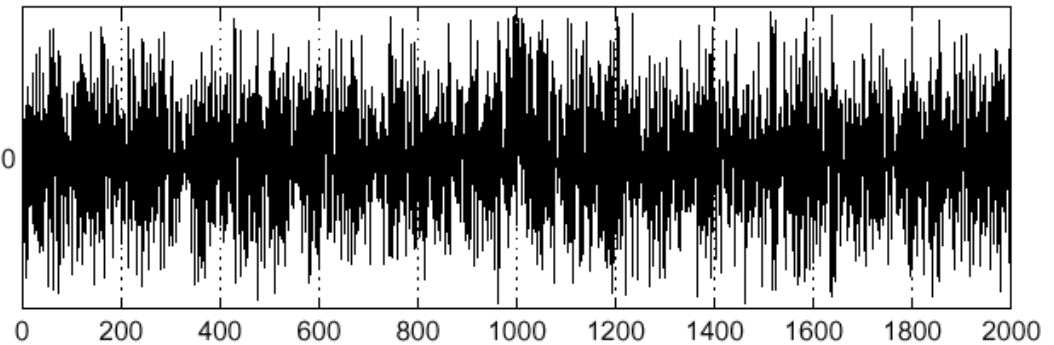


Noisy input image

$$f(x)$$



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



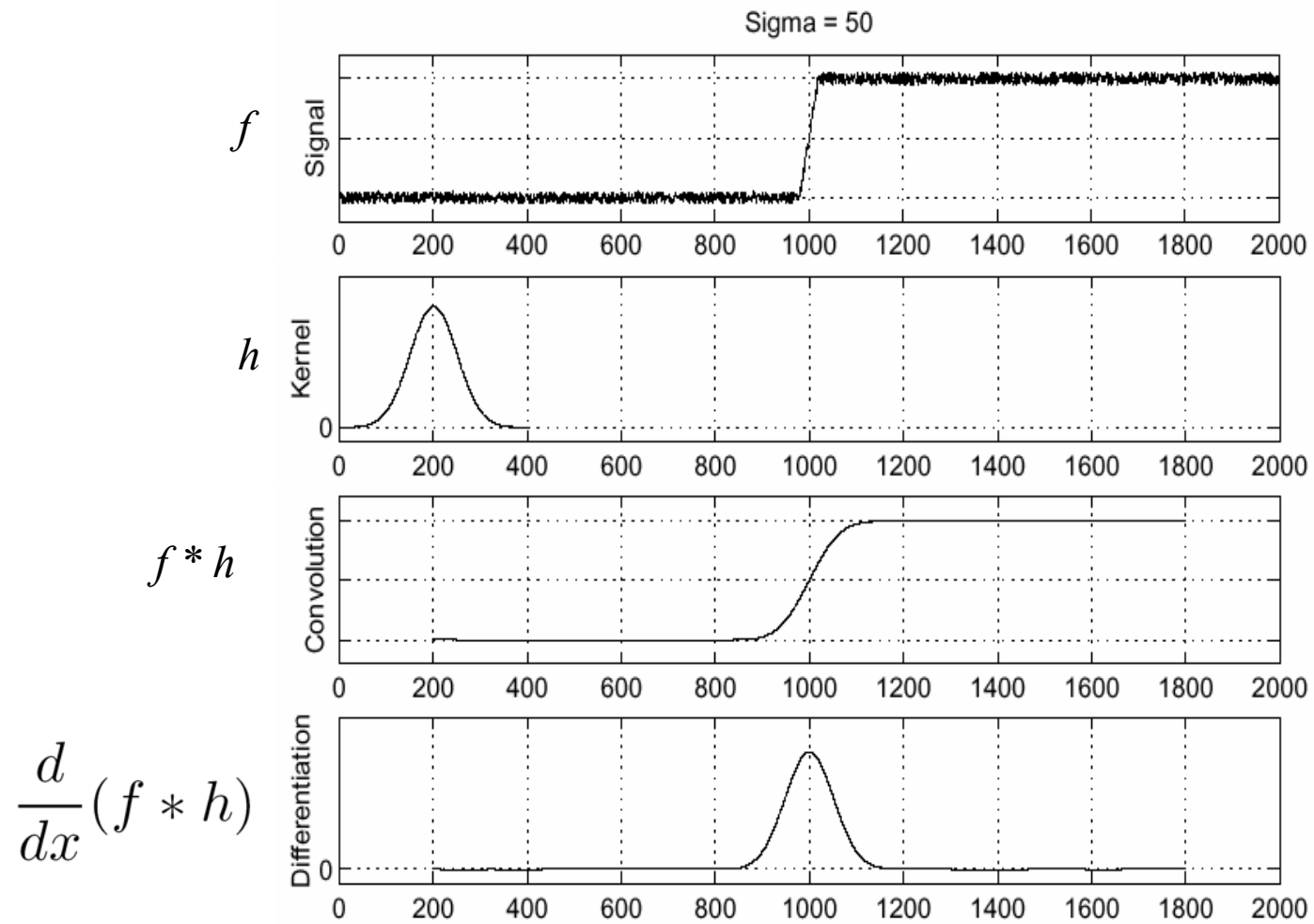
Where is the edge?

Effects of noise

- Noise is high frequency
- Differentiation accentuates noise

$$\frac{d \sin \omega x}{dx} = \omega \cos \omega x$$

Solution: smooth first

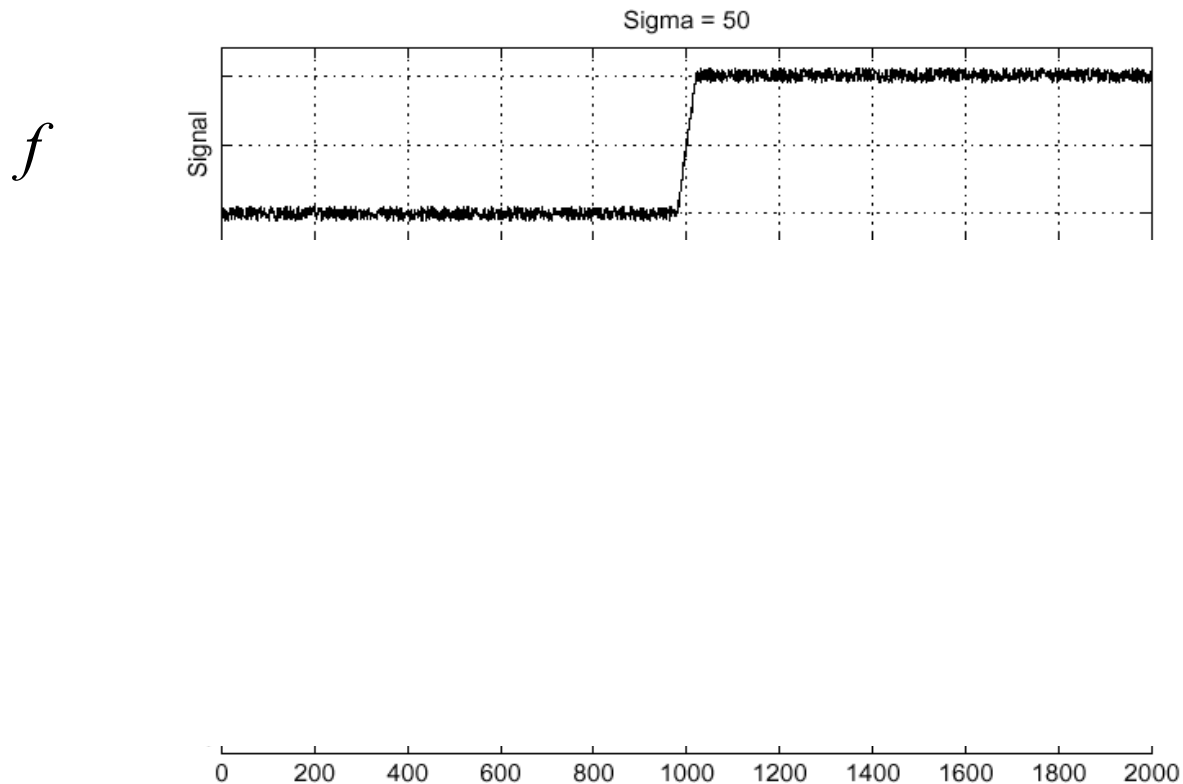


To find edges, look for peaks in $\frac{d}{dx}(f * h)$

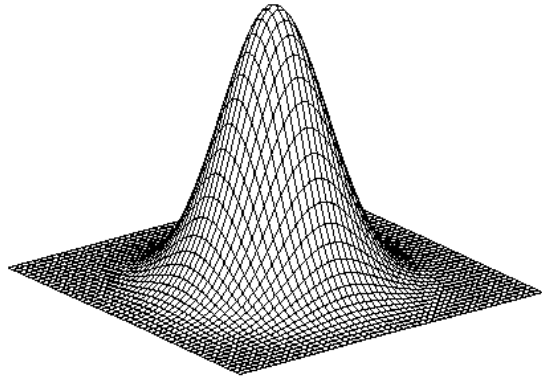
Associative property of convolution

- Differentiation is a convolution
- Convolution is associative:
- This saves us one operation:

$$\frac{d}{dx}(f * h) = f * \frac{d}{dx}h$$

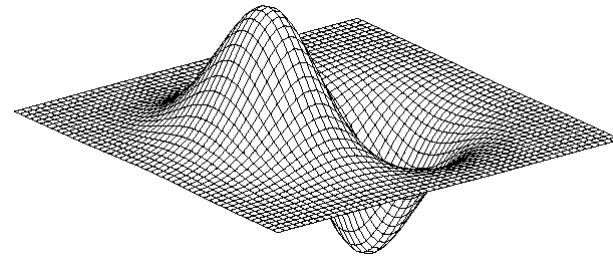


2D edge detection filters



Gaussian

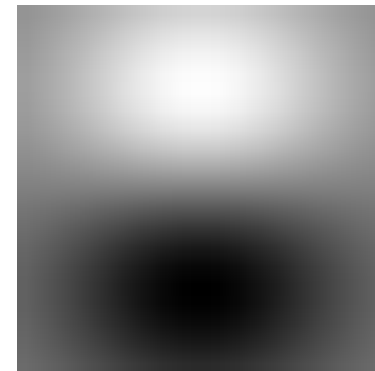
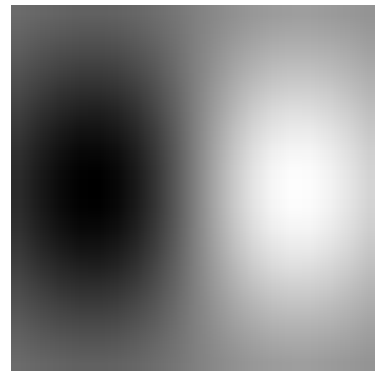
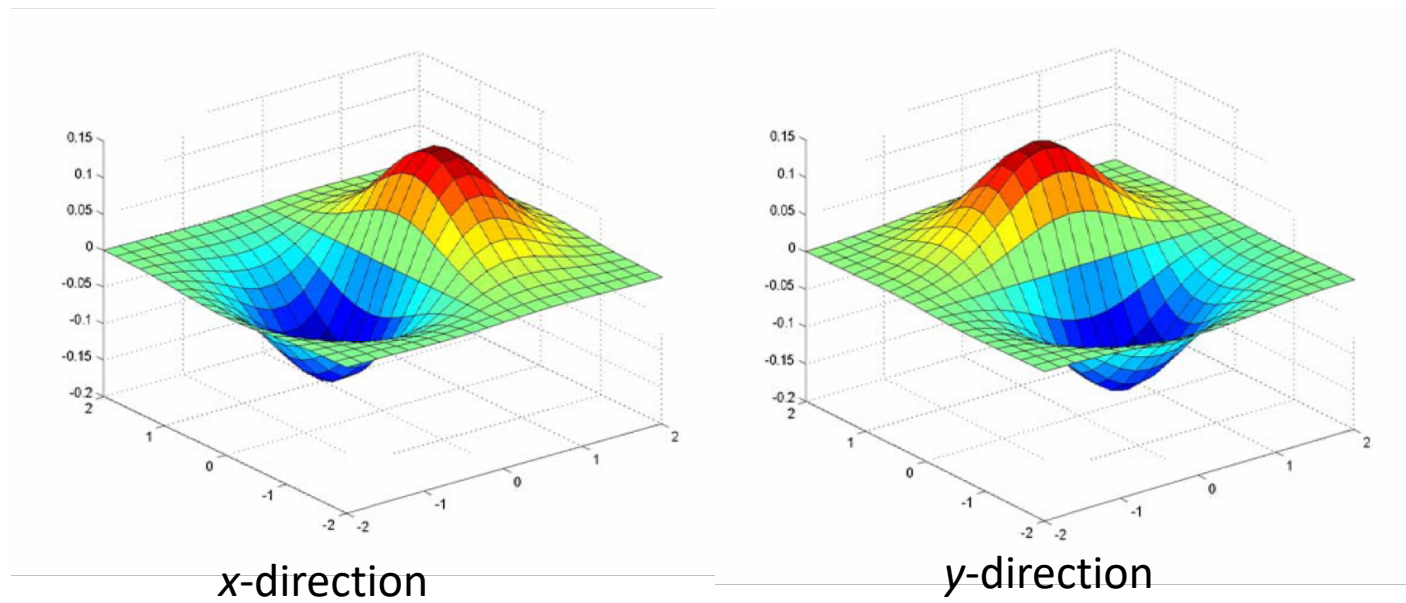
$$h_{\sigma}(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian (x)

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

Derivative of Gaussian filter



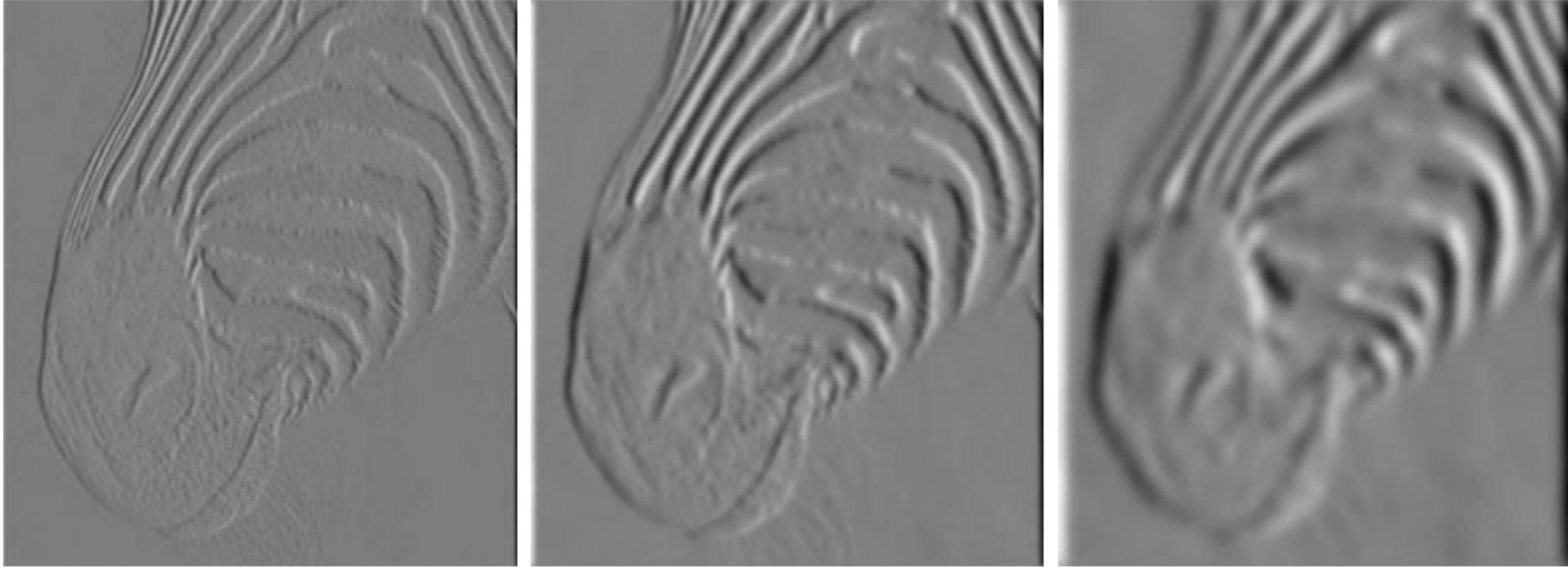
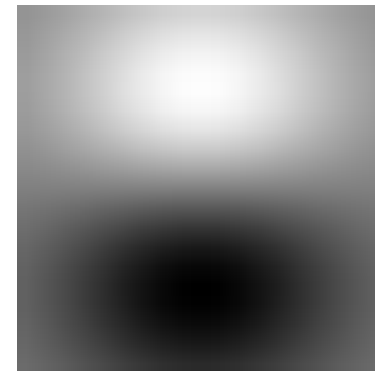
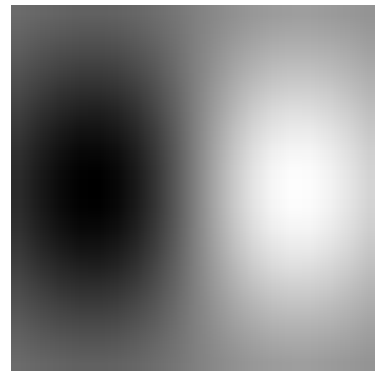
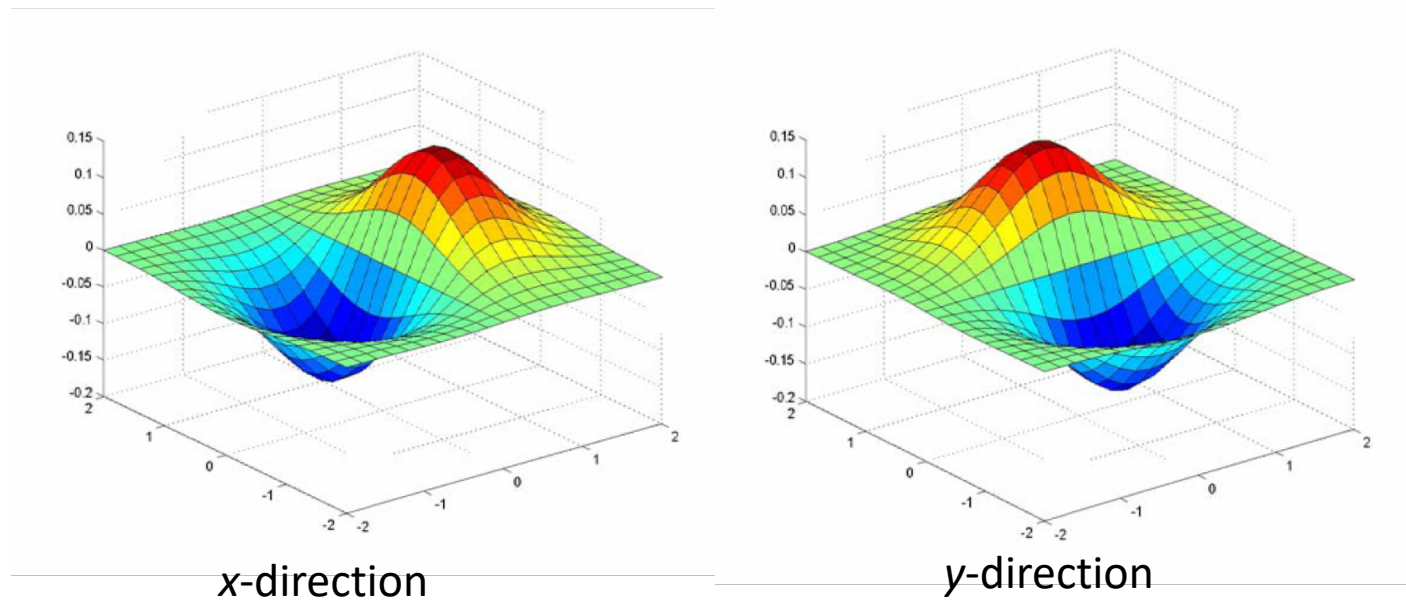


FIGURE 5.3: The scale (i.e., σ) of the Gaussian used in a derivative of Gaussian filter has significant effects on the results. The three images show estimates of the derivative in the x direction of an image of the head of a zebra obtained using a derivative of Gaussian filter with σ one pixel, three pixels, and seven pixels (**left to right**). Note how images at a finer scale show some hair, the animal's whiskers disappear at a medium scale, and the fine stripes at the top of the muzzle disappear at the coarser scale.

Derivative of Gaussian filter

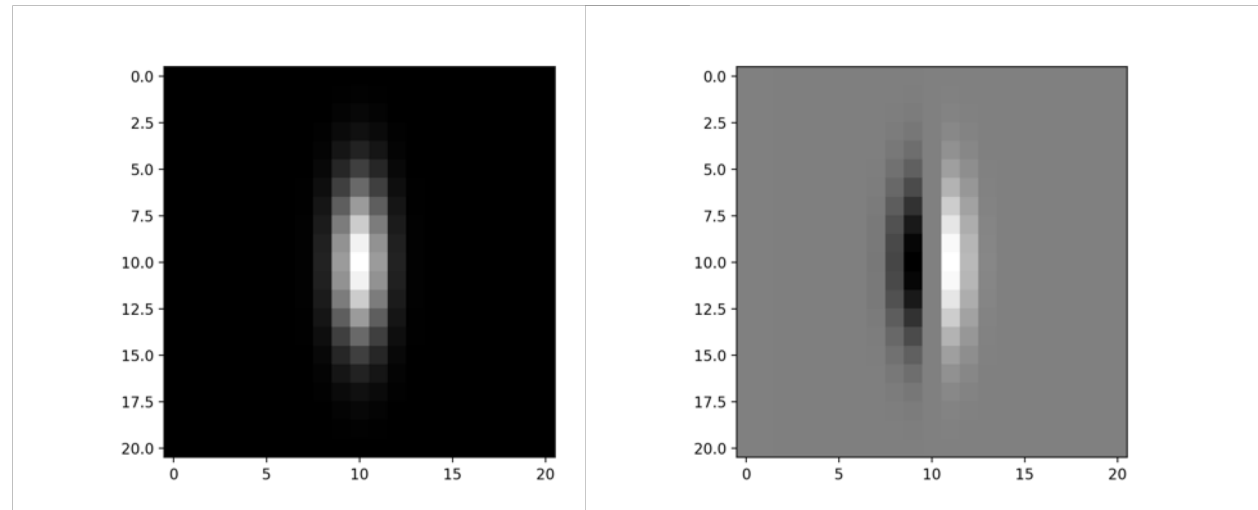


Anisotropic Gaussian

- Should smoothing along the edge be the same as smoothing across it?
- Smooth more along the edge than across it
- Use a lower standard deviation along one direction

- $G(x, y) = e^{-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}}$

- Can also do for derivative



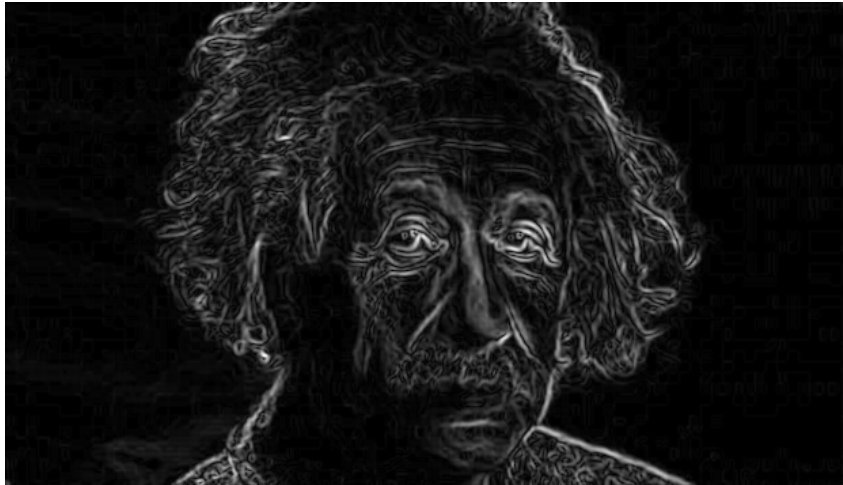
Oriented Gaussian Derivatives

- Instead of just derivative along x and y, can also rotate filter to get derivatives along different directions



Gradient magnitude and orientation

- Orientation is undefined at pixels with 0 gradient



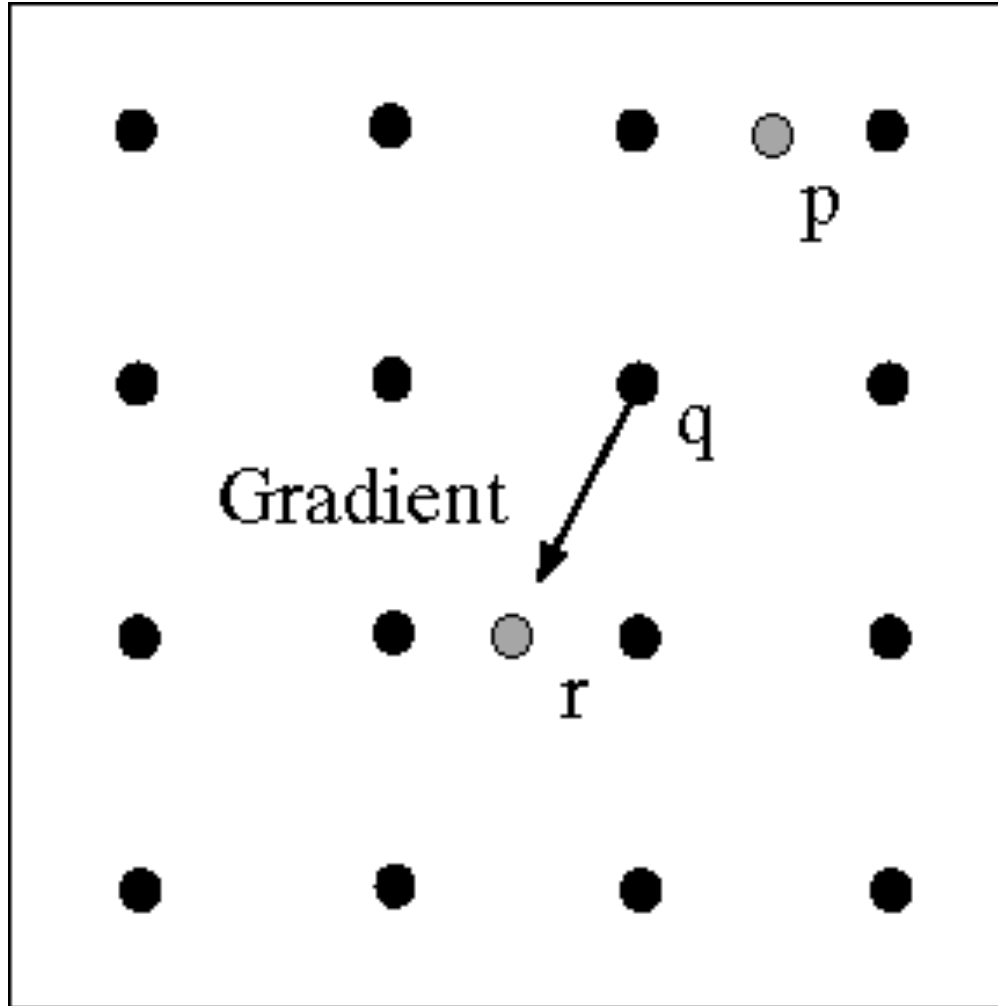
Magnitude



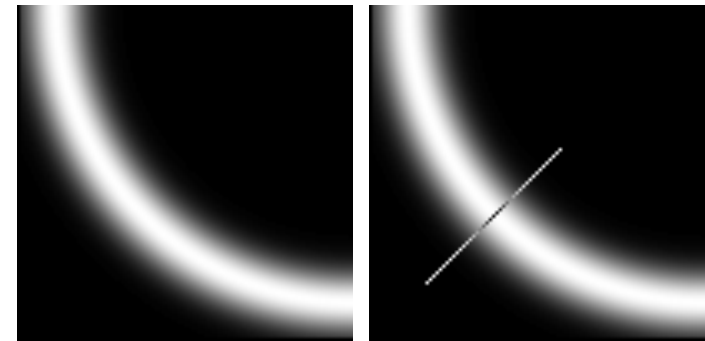
Orientation

$\text{theta} = \text{numpy.arctan2}(\text{gy}, \text{gx})$

Non-maximum suppression for each orientation



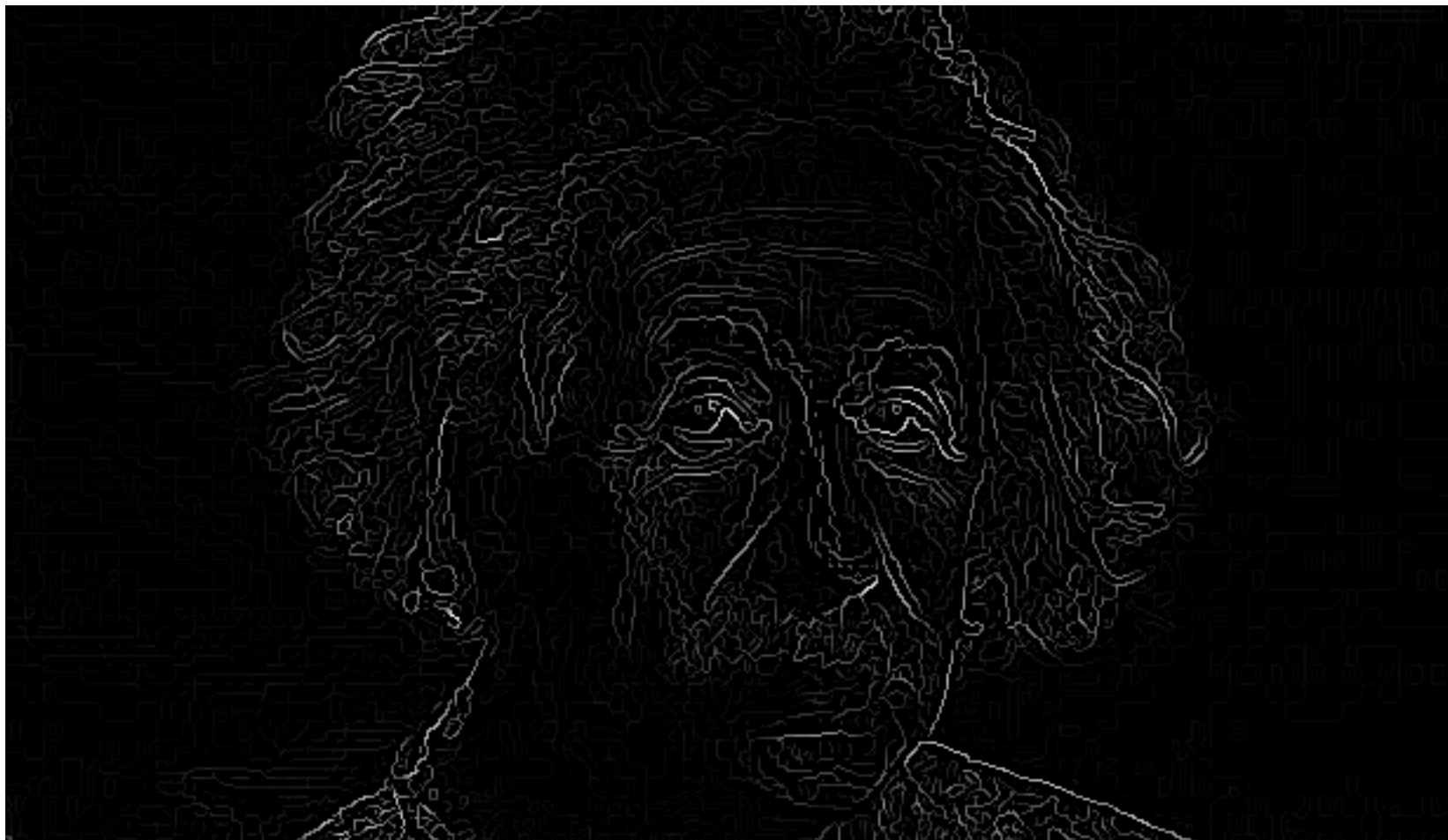
At q , we have a maximum if the value is larger than those at both p and at r . Interpolate to get these values.



Before Non-max Suppression



After Non-max Suppression



Hysteresis thresholding

- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels



Final Canny Edges



Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
 - Thin multi-pixel wide “ridges” down to single pixel width
4. Thresholding and linking (hysteresis):
 - Define two thresholds: low and high
 - Use the high threshold to start edge curves and the low threshold to continue them