

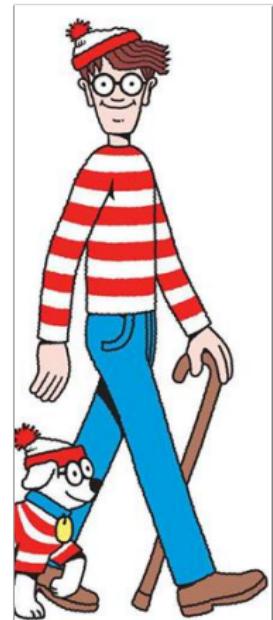
# Edge Detection

# Finding Waldo

- Let's revisit the problem of finding Waldo
- And let's take a simple example



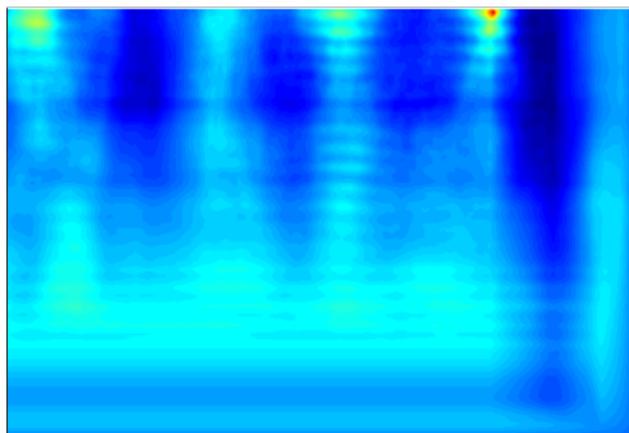
image



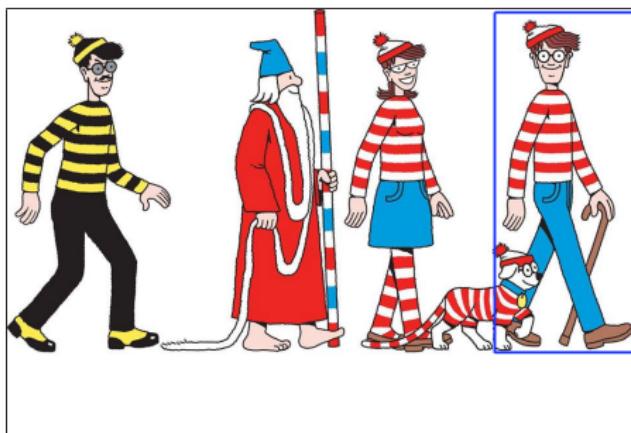
template (filter)

# Finding Waldo

- Let's revisit the problem of finding Waldo
- And let's take a simple example



normalized cross-correlation



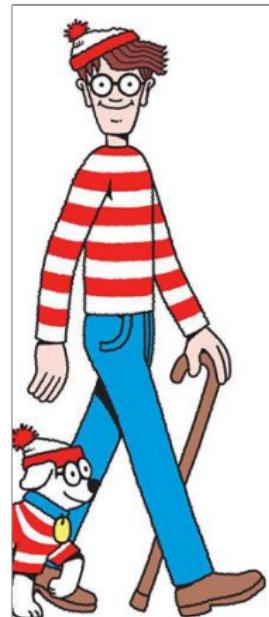
Waldo detection  
(putting box around max response)

# Finding Waldo

- Now imagine Waldo goes shopping
- ... but our filter **doesn't know that**



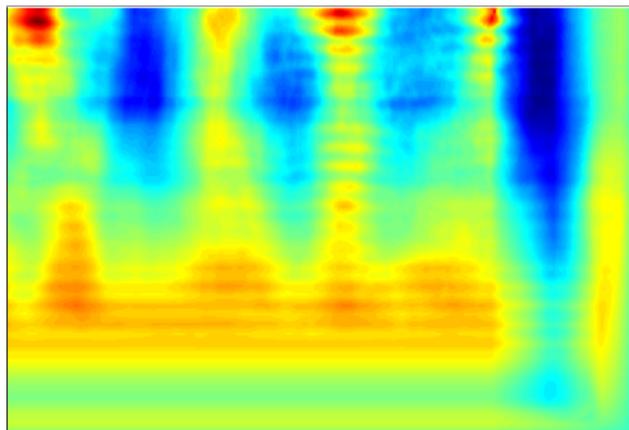
image



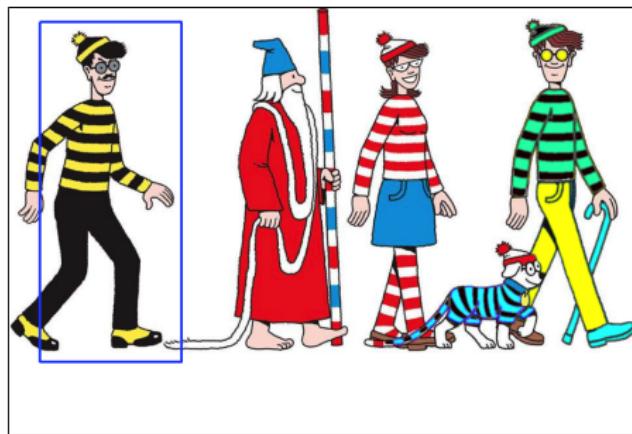
template (filter)

# Finding Waldo

- Now imagine Waldo goes shopping (and the dog too)
- ... but our filter **doesn't know that**



normalized cross-correlation



Waldo detection  
(putting box around max response)

# Finding Waldo (again)

- What can we do to find Waldo again?

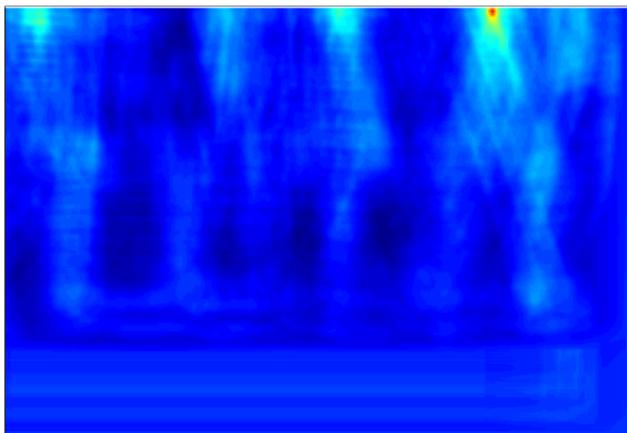
# Finding Waldo (again)

- What can we do to find Waldo again?
- **Edges!!!**

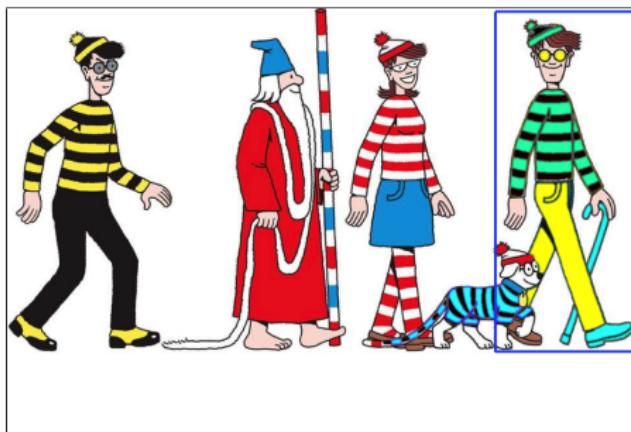


# Finding Waldo (again)

- What can we do to find Waldo again?
- **Edges!!!**



normalized cross-correlation  
(using the edge maps)



Waldo detection  
(putting box around max response)

# Waldo and Edges



# Edge detection

- Map image to a set of **curves** or **line segments** or **contours**.
- More compact than pixels.
- Edges are invariant to changes in illumination
- Important for recognition

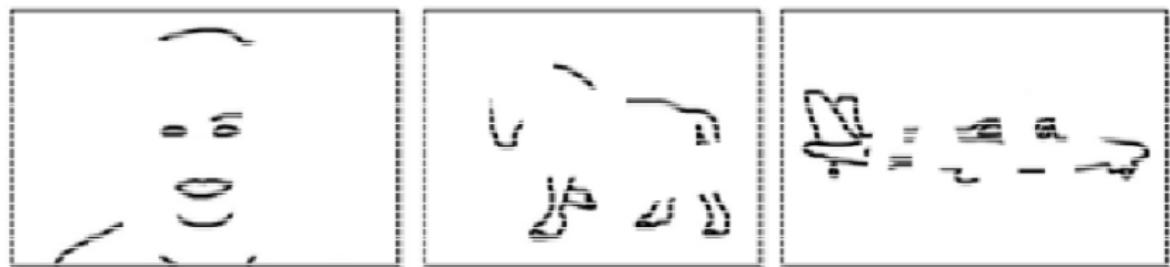


Figure: [Shotton et al. PAMI, 07]

[Source: K. Grauman]

# Edge detection

- Map image to a set of **curves** or **line segments** or **contours**.
- More compact than pixels.
- Edges are invariant to changes in illumination
- Important for recognition
- Important for various applications

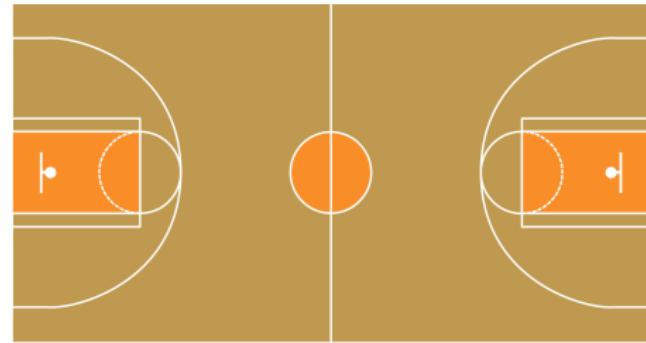


Figure: Parse basketball court (left) to figure out how far the guy is from net

# Edge detection

- Map image to a set of **curves** or **line segments** or **contours**.
- More compact than pixels.
- Edges are invariant to changes in illumination
- Important for recognition
- Important for various applications



Figure: How can a robot pick up or grasp objects?

# Edge detection

- Map image to a set of **curves** or **line segments** or **contours**.
- More compact than pixels.
- Edges are invariant to changes in illumination
- Important for recognition
- Important for various applications

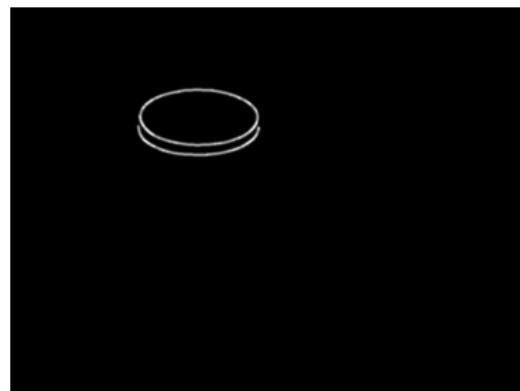
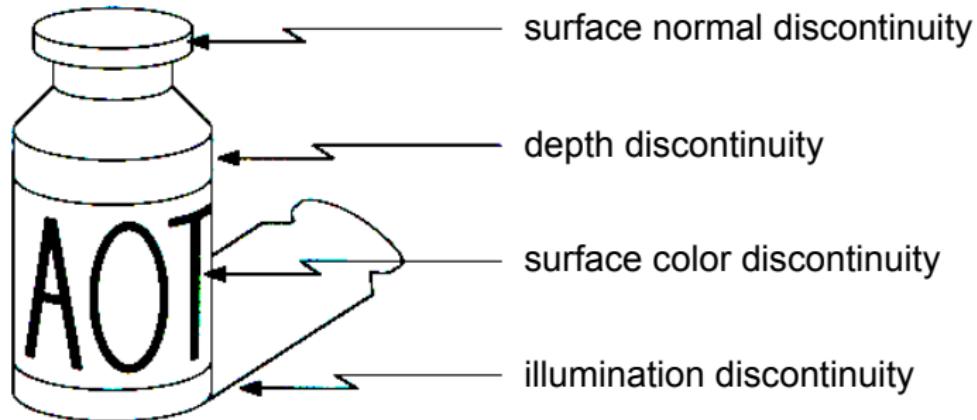


Figure: How can a robot pick up or grasp objects?

# Origin of Edges

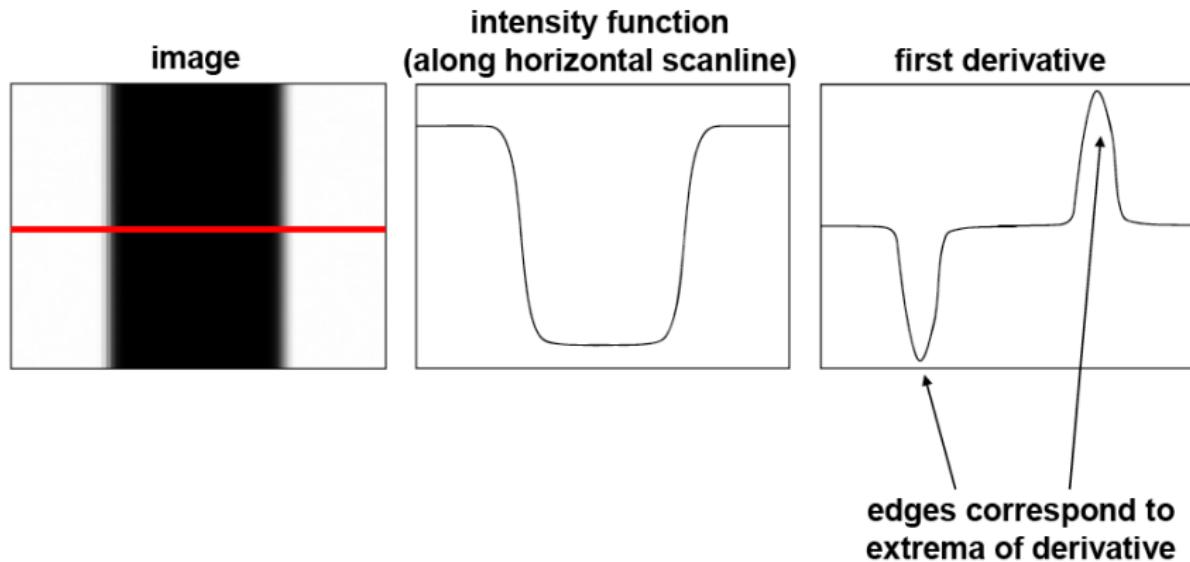
- Edges are caused by a variety of factors



[Source: N. Snavely]

# Characterizing Edges

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



[Source: S. Lazebnik]

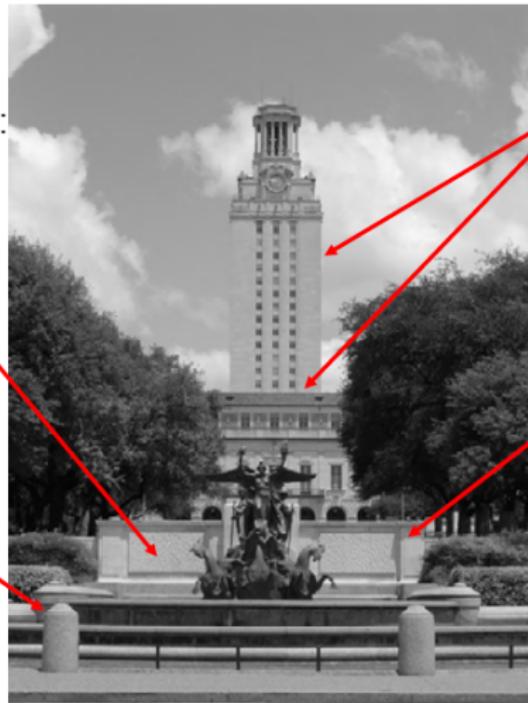
# What Causes an Edge?

Reflectance change:  
appearance  
information, texture

Depth discontinuity:  
object boundary

Change in surface  
orientation: shape

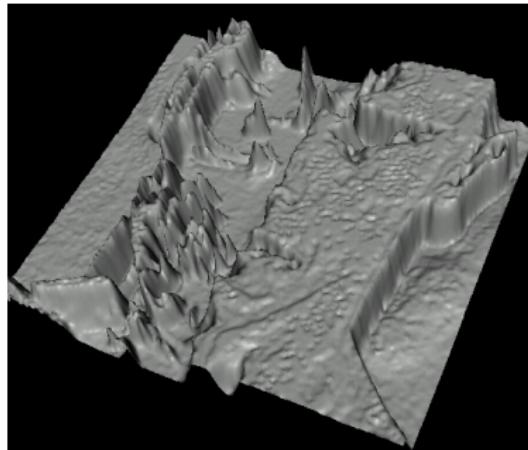
Cast shadows



[Source: K. Grauman]

# Images as Functions

- Edges look like steep cliffs



[Source: N. Snavely]

# How to Implement Derivatives with Convolution

How can we differentiate a digital image  $f[x, y]$ ?

- If image  $f$  was continuous, then compute the partial derivative as

$$\frac{\partial f(x, y)}{\partial x} = \lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon}$$

# How to Implement Derivatives with Convolution

How can we differentiate a digital image  $f[x, y]$ ?

- If image  $f$  was continuous, then compute the partial derivative as

$$\frac{\partial f(x, y)}{\partial x} = \lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon}$$

- Since it's discrete, take first-order forward discrete derivative (finite difference)

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

# How to Implement Derivatives with Convolution

How can we differentiate a digital image  $f[x, y]$ ?

- If image  $f$  was continuous, then compute the partial derivative as

$$\frac{\partial f(x, y)}{\partial x} = \lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon}$$

- Since it's discrete, take first-order forward discrete derivative (finite difference)

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

- What would be the filter to implement this using correlation/convolution?

# How to Implement Derivatives with Convolution

How can we differentiate a digital image  $f[x, y]$ ?

- If image  $f$  was continuous, then compute the partial derivative as

$$\frac{\partial f(x, y)}{\partial x} = \lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon}$$

- Since it's discrete, take first-order forward discrete derivative (finite difference)

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

- What would be the filter to implement this using correlation/convolution?

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

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

[Source: S. Seitz]

## Examples: Partial Derivatives of an Image

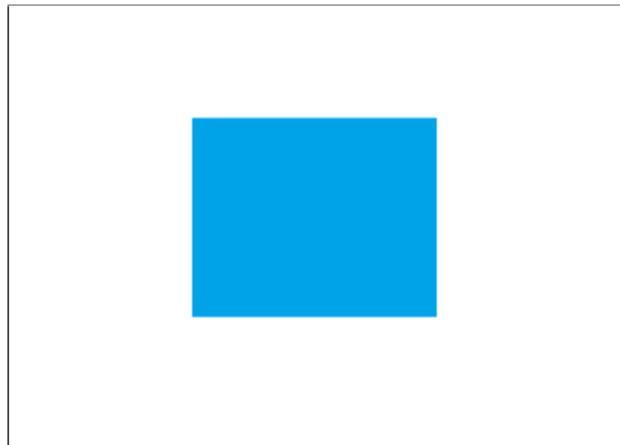
- How does the horizontal derivative using the filter  $[-1, 1]$  look like?



Image

## Examples: Partial Derivatives of an Image

- How does the horizontal derivative using the filter  $[-1, 1]$  look like?



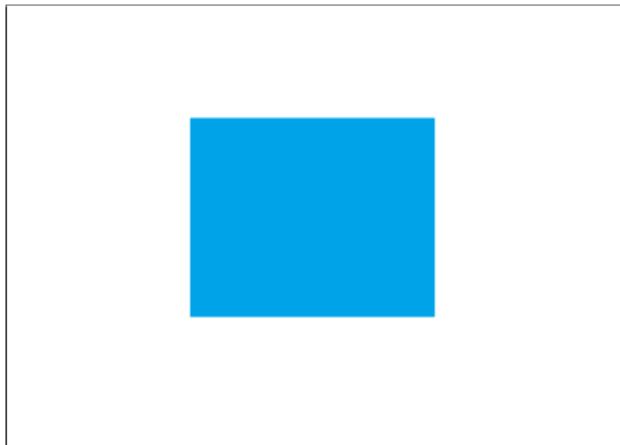
Image



$\frac{\partial f(x,y)}{\partial x}$  with  $[-1, 1]$  and correlation

## Examples: Partial Derivatives of an Image

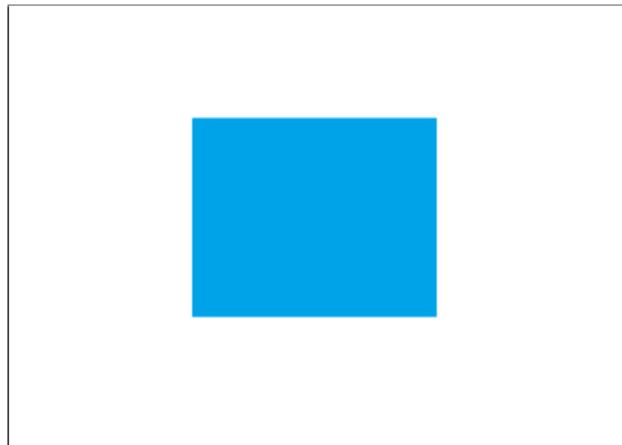
- How about the vertical derivative using filter  $[-1, 1]^T$ ?



Image

## Examples: Partial Derivatives of an Image

- How about the vertical derivative using filter  $[-1, 1]^T$ ?



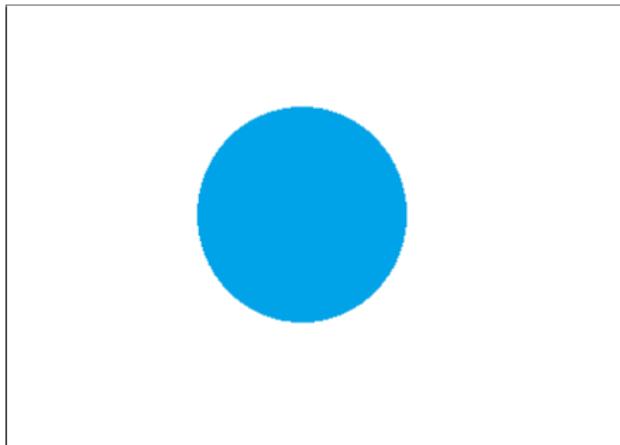
Image



$\frac{\partial f(x,y)}{\partial y}$  with  $[-1, 1]^T$  and correlation

## Examples: Partial Derivatives of an Image

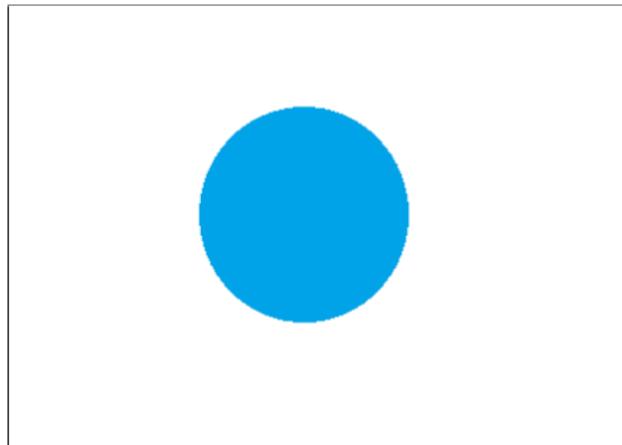
- How does the horizontal derivative using the filter  $[-1, 1]$  look like?



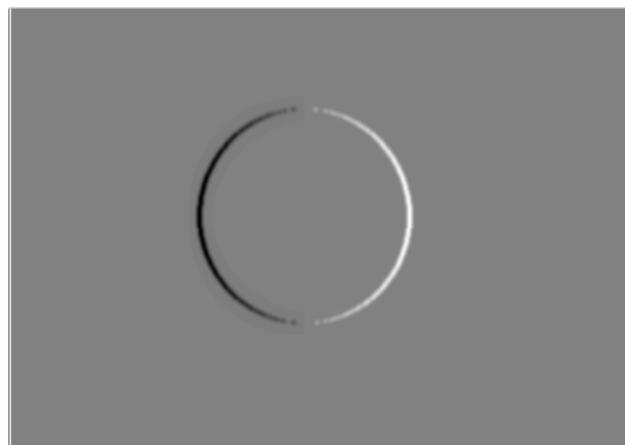
Image

## Examples: Partial Derivatives of an Image

- How does the horizontal derivative using the filter  $[-1, 1]$  look like?



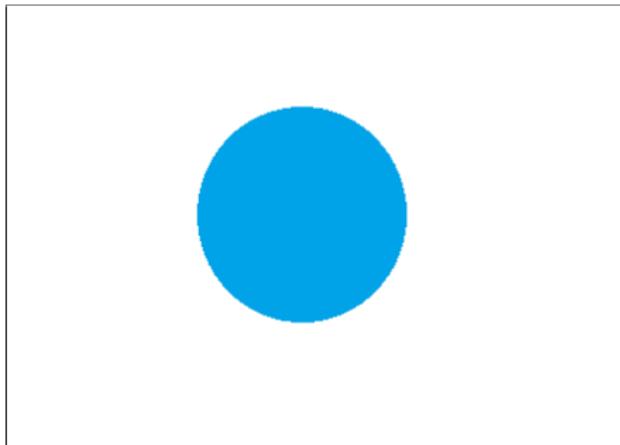
Image



$\frac{\partial f(x,y)}{\partial x}$  with  $[-1, 1]$  and correlation

## Examples: Partial Derivatives of an Image

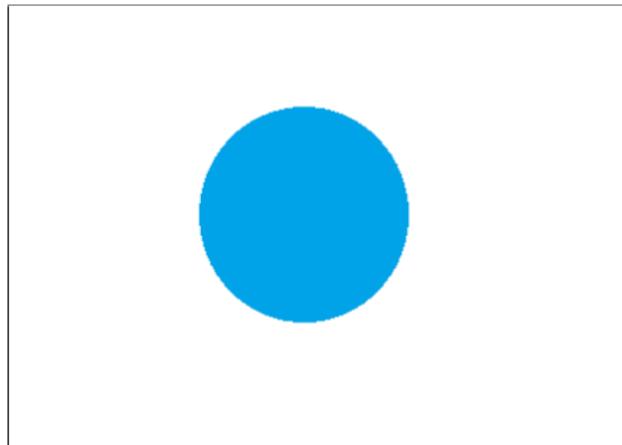
- How about the vertical derivative using filter  $[-1, 1]^T$ ?



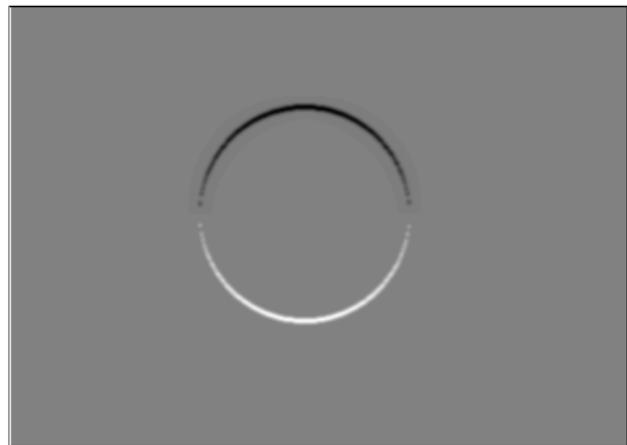
Image

## Examples: Partial Derivatives of an Image

- How about the vertical derivative using filter  $[-1, 1]^T$ ?



Image

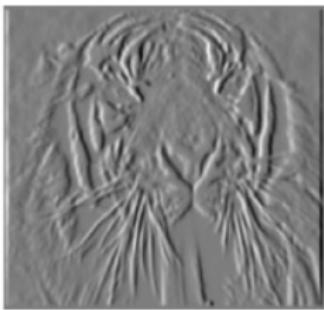


$\frac{\partial f(x,y)}{\partial y}$  with  $[-1, 1]^T$  and correlation

## Examples: Partial Derivatives of an Image

$$\frac{\partial f(x, y)}{\partial x}$$

-1	1
----	---



$$\frac{\partial f(x, y)}{\partial y}$$

-1	?	1
1	or	-1

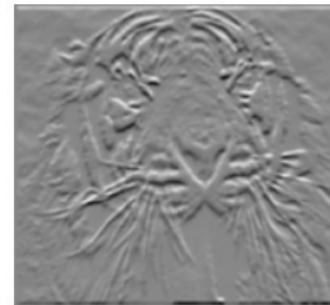


Figure: Using correlation filters

[Source: K. Grauman]

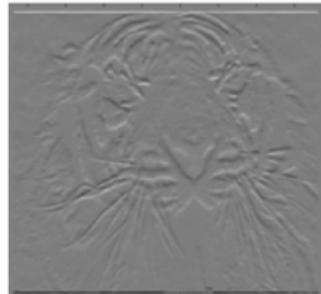
# Finite Difference Filters

Prewitt:  $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$  ;  $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel:  $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$  ;  $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts:  $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$  ;  $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

```
>> My = fspecial('sobel');  
>> outim = imfilter(double(im), My);  
>> imagesc(outim);  
>> colormap gray;
```



[Source: K. Grauman]

# Image Gradient

- The gradient of an image  $\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$

# 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

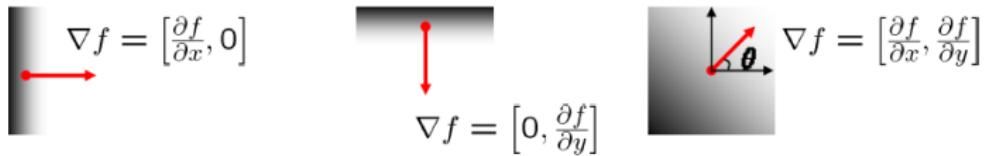

$$\nabla f = \left[ \frac{\partial f}{\partial x}, 0 \right]$$


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


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

# 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

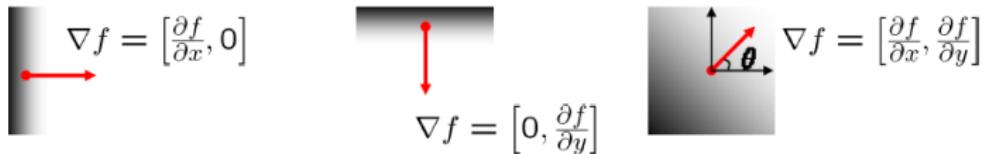


- The **gradient direction** (orientation of edge normal) is given by:

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

# 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



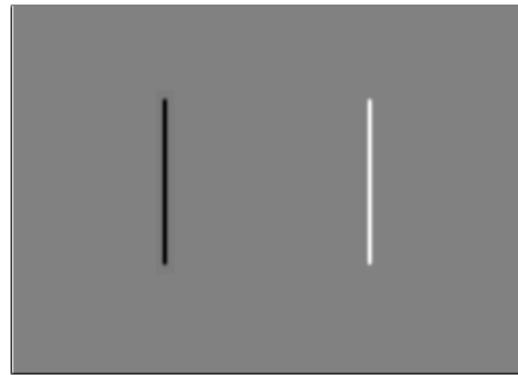
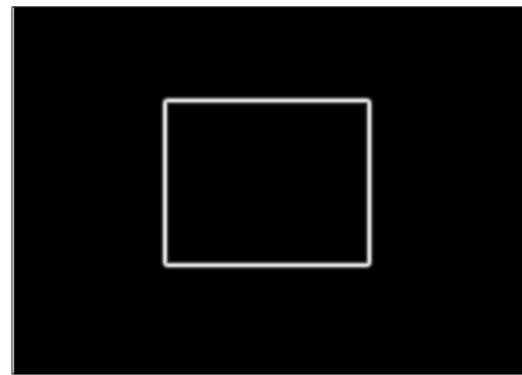
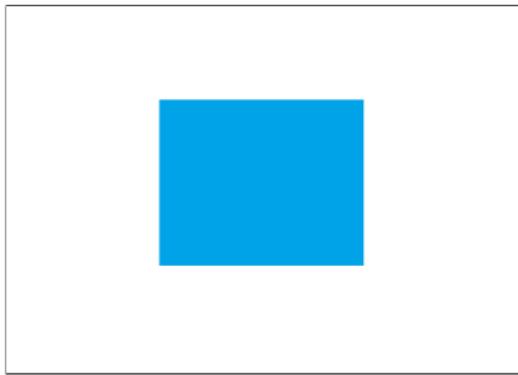
- The **gradient direction** (orientation of edge normal) is given by:

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

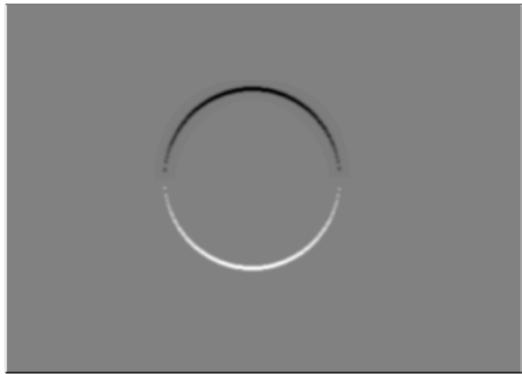
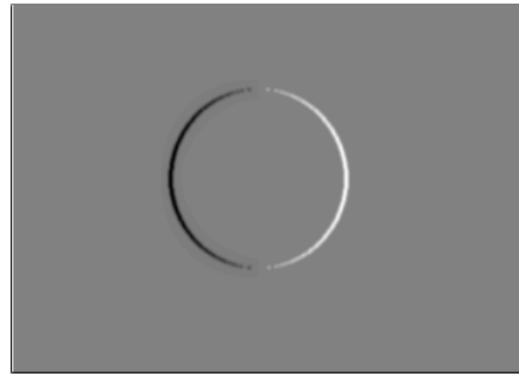
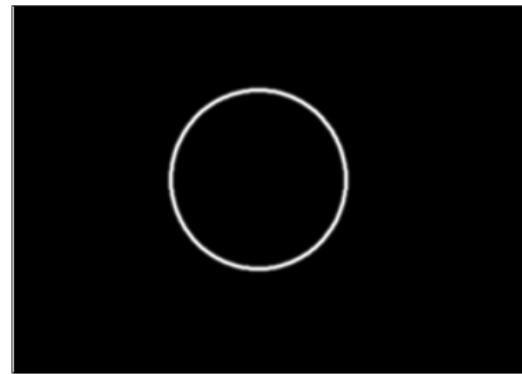
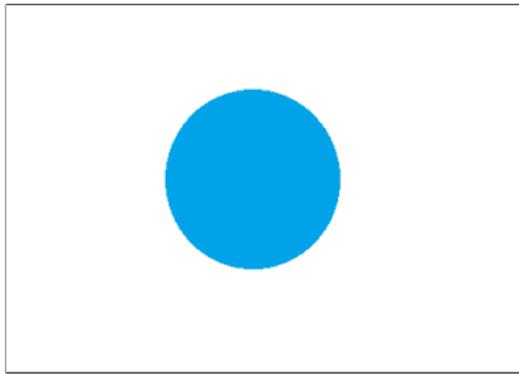
- The **edge strength** is given by the magnitude  $||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

[Source: S. Seitz]

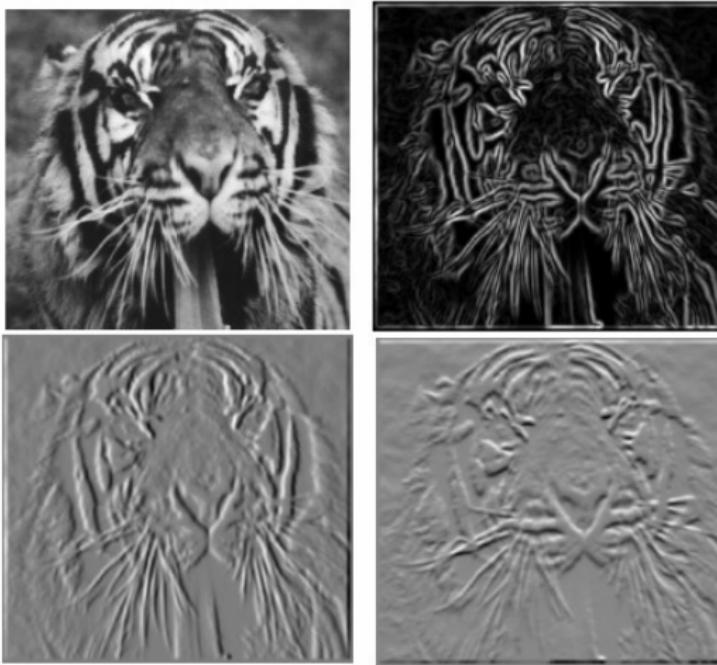
## Example: Image Gradient



## Example: Image Gradient



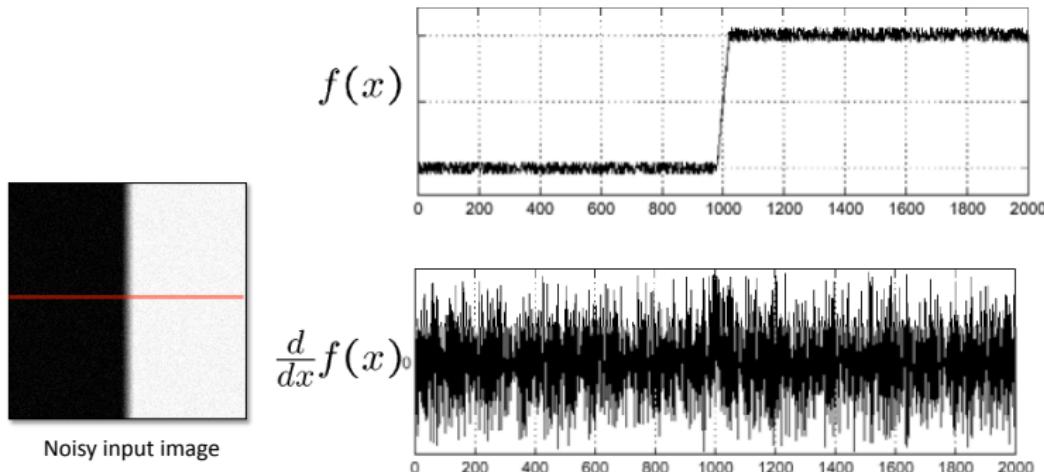
## Example: Image Gradient



[Source: S. Lazebnik]

# Effects of noise

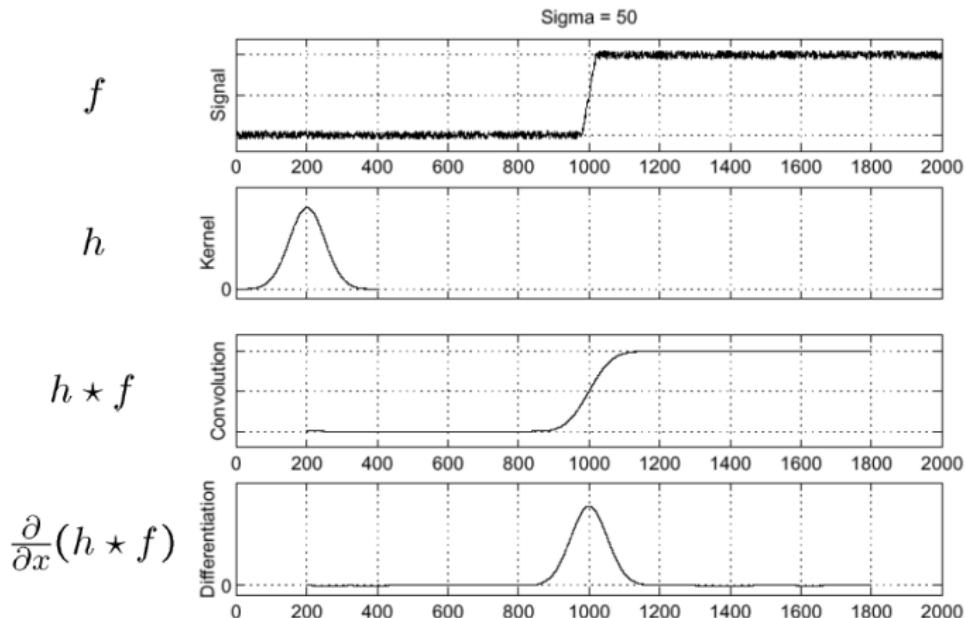
- What if our image is noisy? What can we do?
- Consider a single row or column of the image.
- Plotting intensity as a function of position gives a signal.



[Source: S. Seitz]

# Effects of noise

- Smooth first with  $h$  (e.g. Gaussian), and look for peaks in  $\frac{\partial}{\partial x}(h * f)$ .



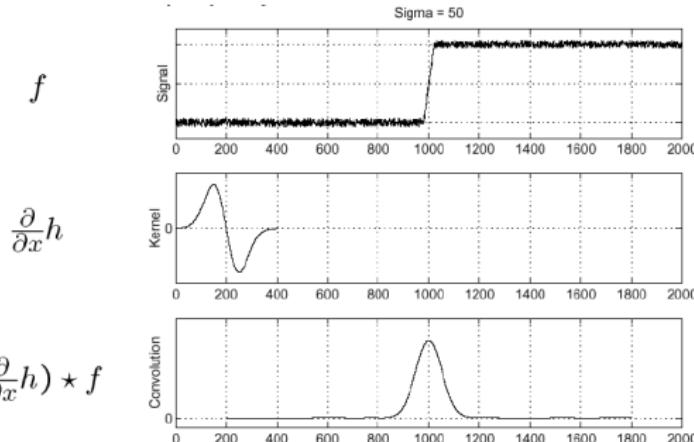
[Source: S. Seitz]

# Derivative theorem of convolution

- Differentiation property of convolution

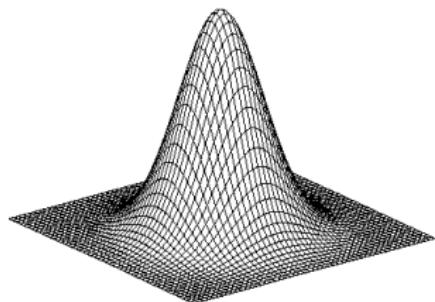
$$\frac{\partial}{\partial x}(h * f) = \left(\frac{\partial h}{\partial x}\right) * f = h * \left(\frac{\partial f}{\partial x}\right)$$

- From last time, why does this work?
- It saves one operation



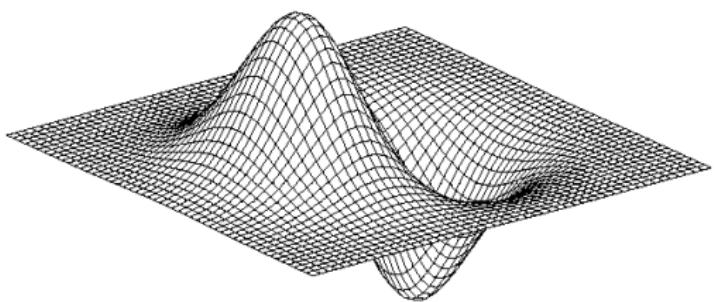
[Source: S. Seitz]

# 2D Edge Detection Filters



Gaussian

$$h_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

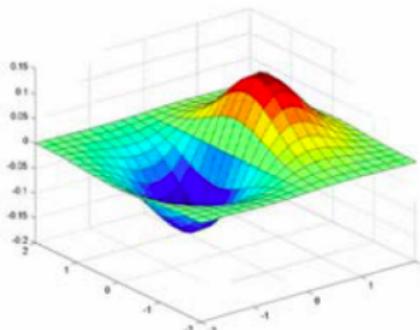


Derivative of Gaussian (x)

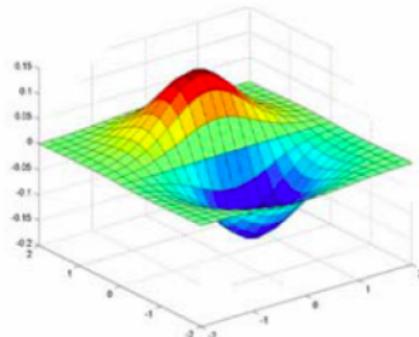
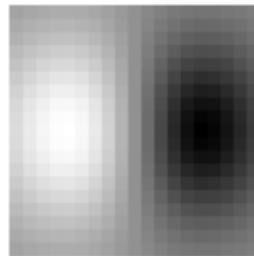
$$\frac{\partial}{\partial x} h_{\sigma}(x, y)$$

[Source: N. Snavely]

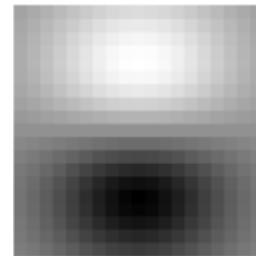
# Derivative of Gaussians



x-direction



y-direction



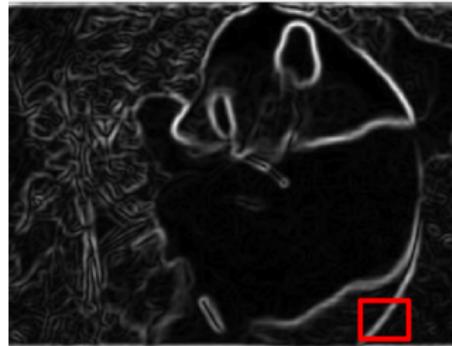
[Source: K. Grauman]

## Example



- Applying the Gaussian derivatives to image

# Example

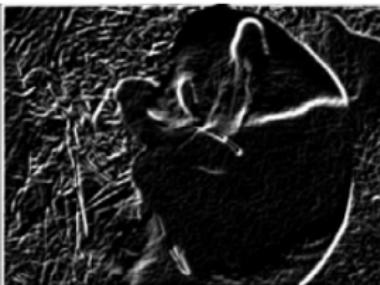


- Applying the Gaussian derivatives to image

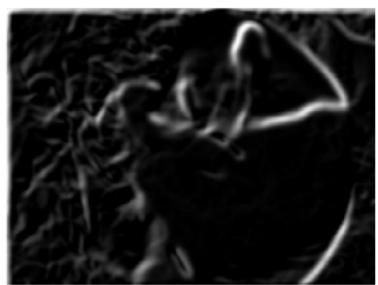
# Effect of $\sigma$ on derivatives

The detected structures differ depending on the **Gaussian's scale parameter**:

- Larger values: detects edges of larger scale
- Smaller values: detects finer structures



$\sigma = 1$  pixel



$\sigma = 3$  pixels

[Source: K. Grauman]

# Canny Edge Detector

Matlab: `edge(image, 'canny')`

- ① Filter image with derivative of Gaussian (horizontal and vertical directions)
- ② Find magnitude and orientation of gradient
- ③ Non-maximum suppression
- ④ Linking and thresholding (hysteresis):
  - Define two thresholds: low and high
  - Use the high threshold to start edge curves and the low threshold to continue them

[Source: D. Lowe and L. Fei-Fei]

# Locating Edges – Canny's Edge Detector

Let's take the most popular picture in computer vision: Lena (appeared in November 1972 issue of Playboy magazine)



[Source: N. Snavely]

# Locating Edges – Canny's Edge Detector



**Figure:** Canny's approach takes gradient magnitude

[Source: N. Snavely]

# Locating Edges – Canny's Edge Detector



Figure: Thresholding

[Source: N. Snavely]

# Locating Edges – Canny's Edge Detector



Figure: Gradient magnitude

[Source: N. Snavely]

# Non-Maxima Suppression

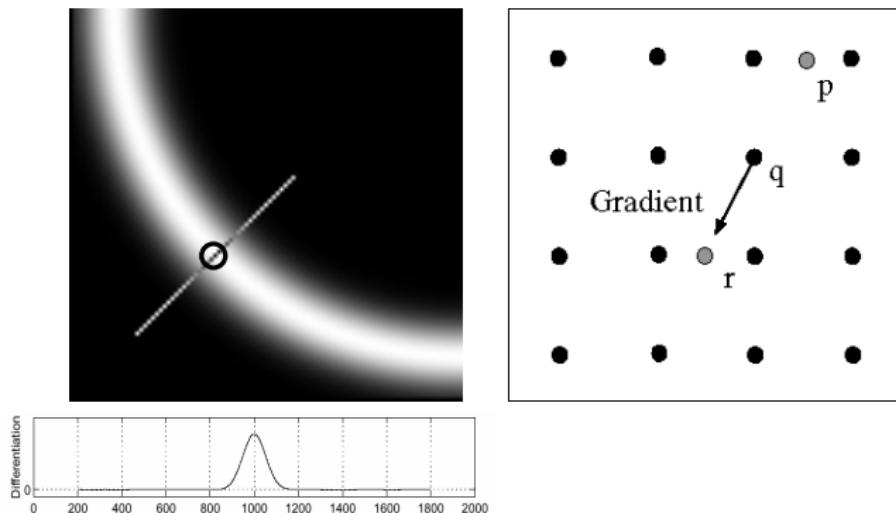


Figure: Gradient magnitude

- Check if pixel is local maximum along gradient direction
- If yes, take it

[Source: N. Snavely]

# Finding Edges



Problem:  
pixels along  
this edge  
didn't  
survive the  
thresholding

Figure: Problem with thresholding

[Source: K. Grauman]

# Hysteresis thresholding

- Use a high threshold to start edge curves, and a low threshold to continue them



[Source: K. Grauman]

# Hysteresis thresholding



original image



high threshold  
(strong edges)



low threshold  
(weak edges)



hysteresis threshold

[Source: L. Fei Fei]

# Located Edges!



**Figure:** Thinning: Non-maxima suppression

[Source: N. Snavely]

# Canny Edge Detector

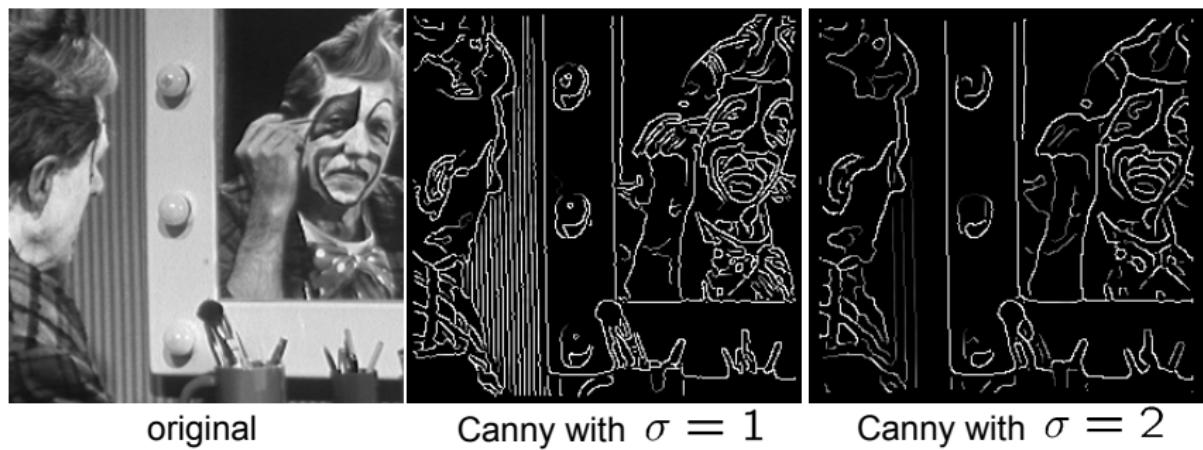
Matlab: `edge(image, 'canny')`

- ① Filter image with derivative of Gaussian (horizontal and vertical directions)
- ② Find magnitude and orientation of gradient
- ③ Non-maximum suppression
- ④ Linking and thresholding (hysteresis):
  - Define two thresholds: low and high
  - Use the high threshold to start edge curves and the low threshold to continue them

[Source: D. Lowe and L. Fei-Fei]

# Canny Edge Detector (again)

- large  $\sigma$  (in step 1) detects “large-scale” edges
- small  $\sigma$  detects fine edges



[Source: S. Seitz]

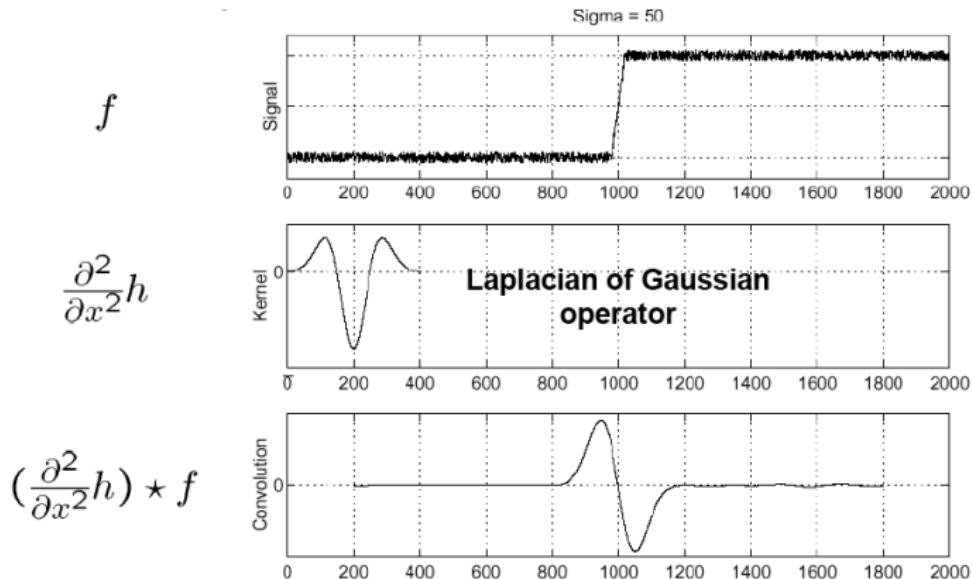
# Canny edge detector

- Still one of the most widely used edge detectors in computer vision
- J. Canny, A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.
- Depends on several parameters:  $\sigma$  of the **blur** and the **thresholds**

[Adopted by: R. Urtasun]

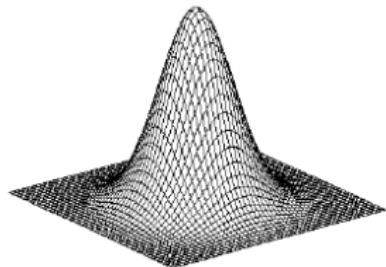
# Another Way of Finding Edges: Laplacian of Gaussians

- Edge by detecting **zero-crossings** of bottom graph



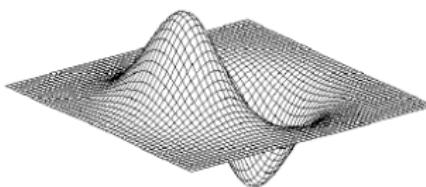
[Source: S. Seitz]

# 2D Edge Filtering



Gaussian

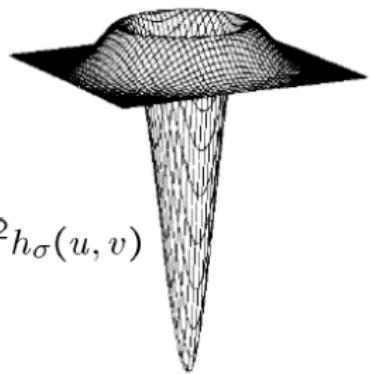
$$h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian



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

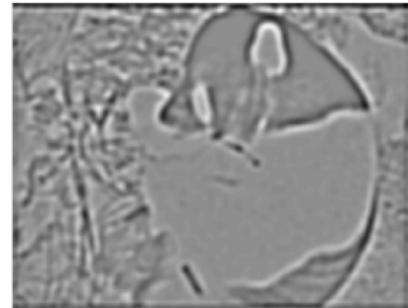
with  $\nabla^2$  the Laplacian operator  $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

[Source: S. Seitz]

## Example



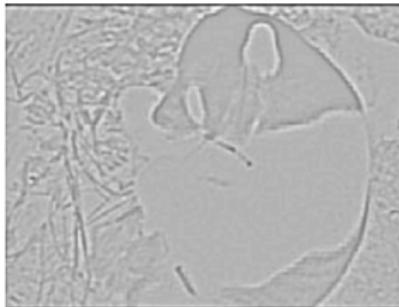
$\sigma = 1$  pixels



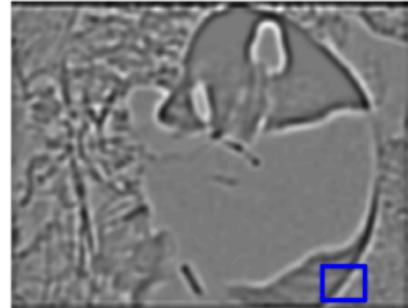
$\sigma = 3$  pixels

- Applying the Laplacian operator to image

# Example

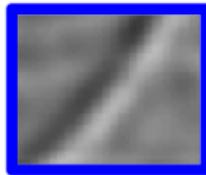


$\sigma = 1$  pixels



$\sigma = 3$  pixels

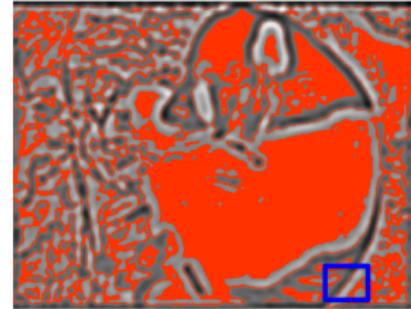
- Applying the Laplacian operator to image



# Example

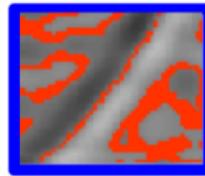


$\sigma = 1$  pixels



$\sigma = 3$  pixels

- Applying the Laplacian operator to image



# A More ‘Modern’ Approach

This is “old-style” Computer Vision. We are now in the era of successful Machine Learning techniques.

**Question:** Can we use ML to do a better job at finding edges?

# A More ‘Modern’ Approach

This is “old-style” Computer Vision. We are now in the era of successful Machine Learning techniques.

**Question:** Can we use ML to do a better job at finding edges?

We will see later.

# A More ‘Modern’ Approach

This is “old-style” Computer Vision. We are now in the era of successful Machine Learning techniques.

**Question:** Can we use ML to do a better job at finding edges?

OR Should we see right now?

# Summary – Stuff You Should Know

Not so good:

- **Horizontal image gradient:** Subtract intensity of left neighbor from pixel's intensity (filtering with  $[-1, 1]$ )
- **Vertical image gradient:** Subtract intensity of bottom neighbor from pixel's intensity (filtering with  $[-1, 1]^T$ )

Much better (more robust to noise):

- **Horizontal image gradient:** Apply derivative of Gaussian with respect to  $x$  to image (filtering!)
- **Vertical image gradient:** Apply derivative of Gaussian with respect to  $y$  to image
- **Magnitude of gradient:** compute the horizontal and vertical image gradients, square them, sum them, and  $\sqrt{}$  the sum
- **Edges:** Locations in image where magnitude of gradient is high
- Phenomena that **causes** edges: rapid change in surface's normals, depth discontinuity, rapid changes in color, change in illumination

# Summary – Stuff You Should Know

- **Properties of gradient's magnitude:**

- Zero far away from edge
- Positive on both sides of the edge
- Highest value directly on the edge
- Higher  $\sigma$  emphasizes larger structures

- **Canny's edge detector:**

- Compute gradient's direction and magnitude
- Non-maxima suppression
- Thresholding at two levels and linking

## Matlab functions:

- FSPECIAL: gives a few gradients filters (PREWITT, SOBEL, ROBERTS)
- SMOOTHGRADIENT: function to compute gradients with derivatives of Gaussians. Find it in Lecture's 3 code (Will be posted on class webpage)
- EDGE: use `EDGE(I, 'CANNY')` to detect edges with Canny's method, and `EDGE(I, 'LOG')` for Laplacian method