IMAGE GRADIENTS AND EDGES

Class 4: Artificial Vision

Source of most slides: Kristen Grauman, UT-Austin

Last lecture

- Linear filters and convolution useful for
  - Image smoothing, removing noise
    - Box/mask filter
    - Gaussian filter
    - Impact of scale / width (sigma) of the smoothing filter
- Mean filter:
  - Smoothing pixel values by averaging them
  - Uniform average vs. weighted average
- Median filter:
  - a non-linear filter,
  - assuring no averaging and inventing grey-levels,
  - edge-preserving.

Review

original image

filtered

Filter f = 1/9 x [1 1 1 1 1 1 1 1 1]^T

What happens if we have a smoothing filter that is *unnormalized* (does not sum to one)?

Recall: image filtering

- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.

- Uses of filtering:
  - Enhance an image (denoise, etc)
  - Extract information (texture, edges, etc) – to see later
  - Detect patterns (template matching) – to see later

Adapted from Derek Hoiem

Edge detection

- **Goal**: map image from 2d array of pixels to a set of curves or line segments or contours.
- **Why?**

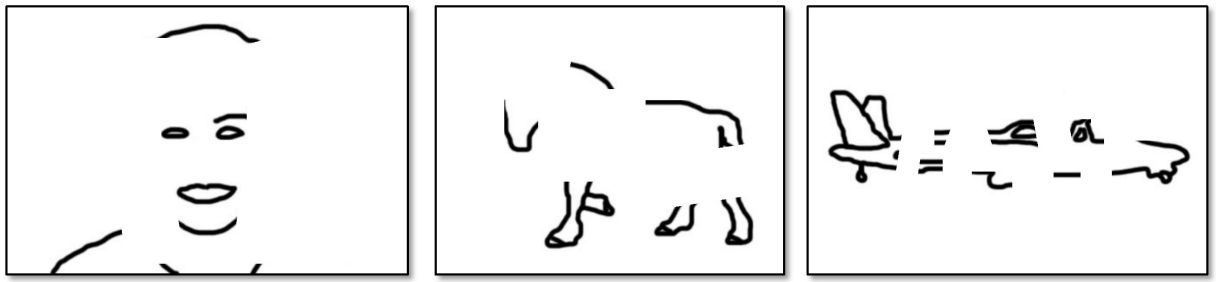
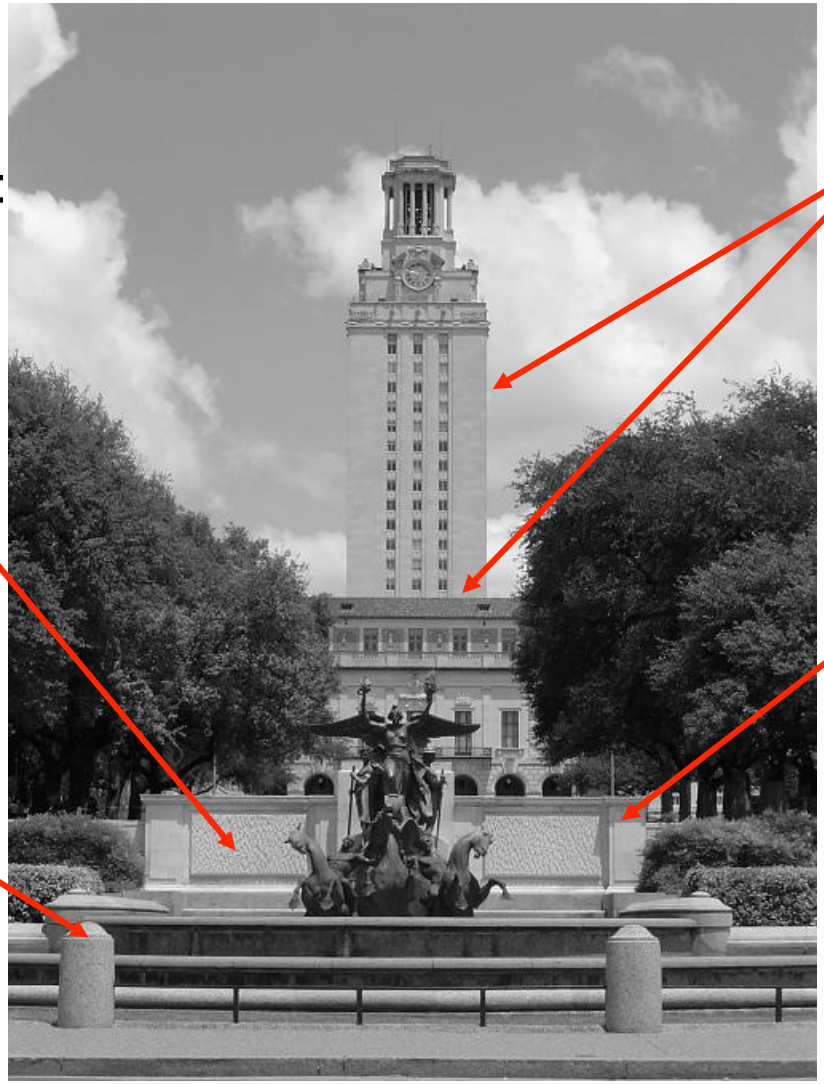


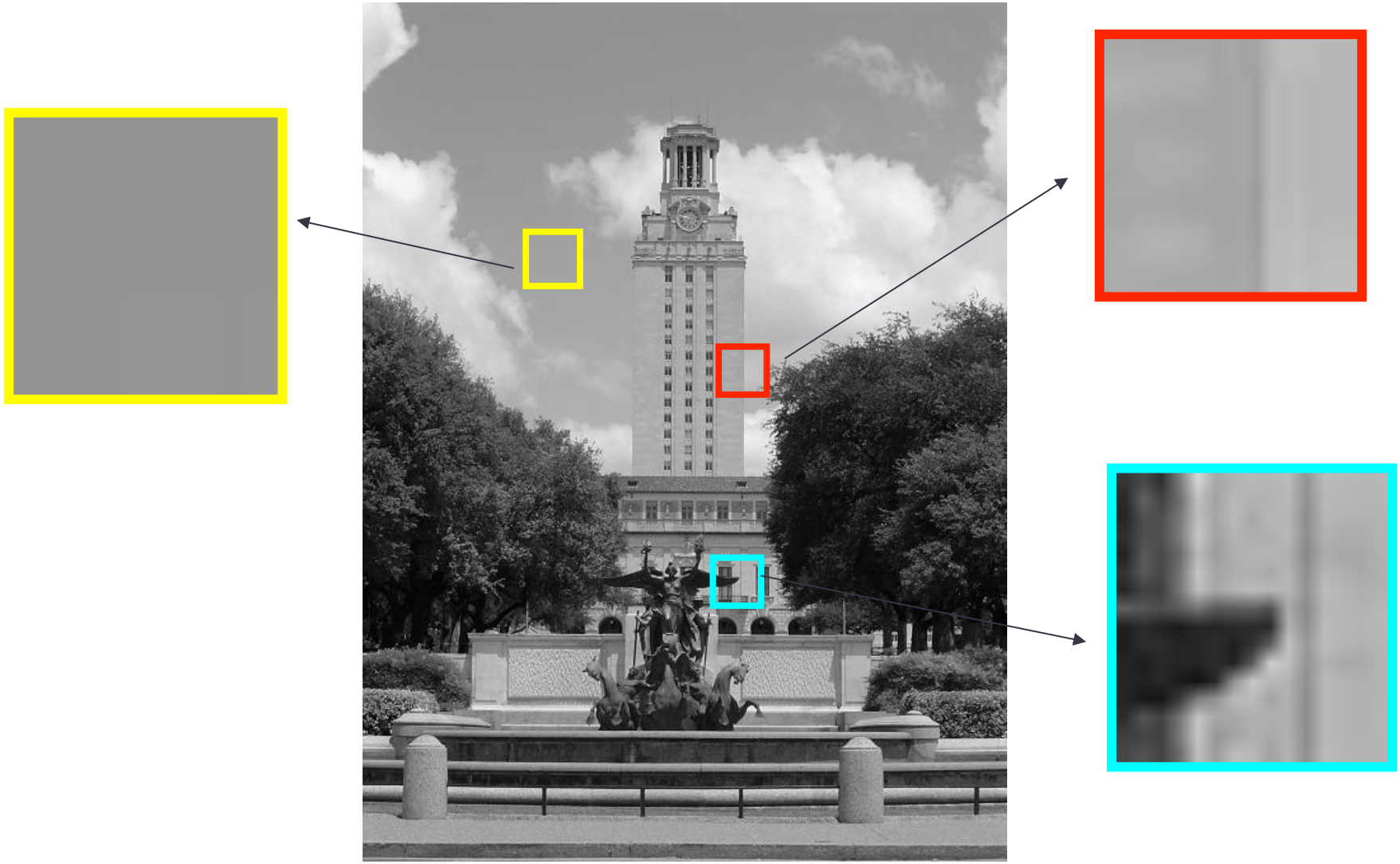
Figure from J. Shotton et al., PAMI 2007

- **Main idea**: look for strong gradients, post-process

What does cause an edge?

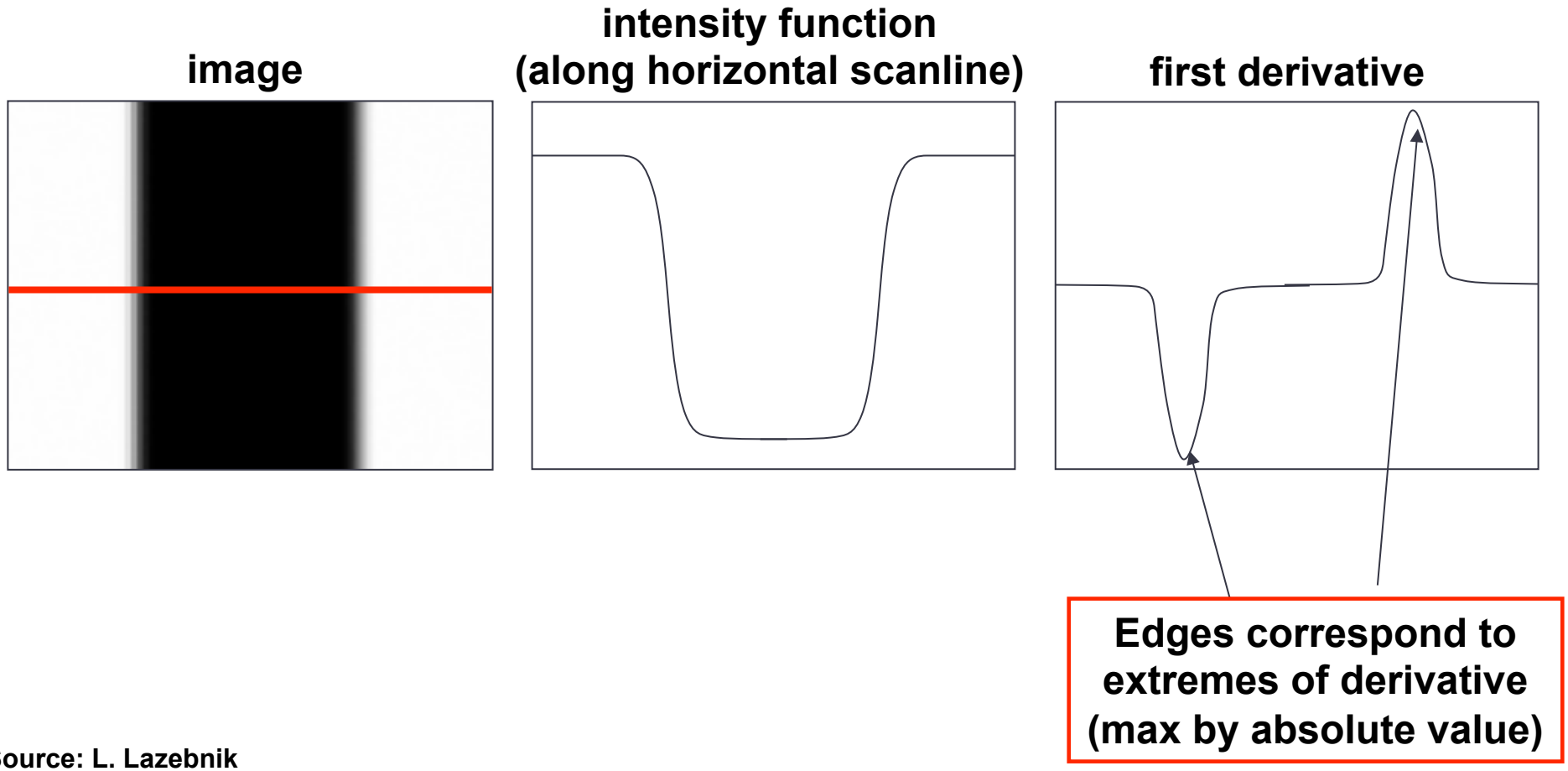


Edges/gradients and invariance



Derivatives and edges

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



Source: L. Lazebnik

Derivatives with convolution

For 2D function, f(x,y), the partial derivative in x is:

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

For discrete data, we can approximate it using finite differences:

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

To implement above as convolution, what would be the associated filter?

Partial derivatives of an image

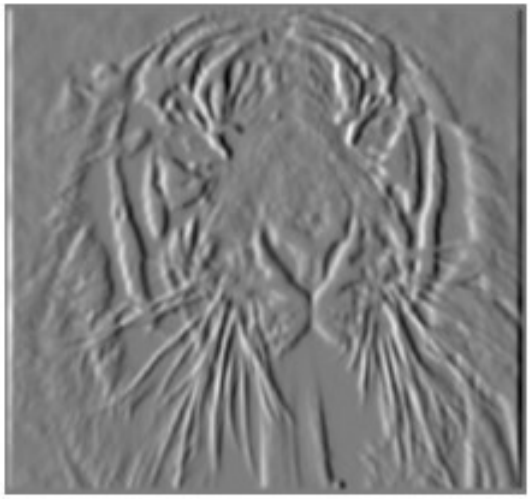
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y)-f(x,y)}{1}$$

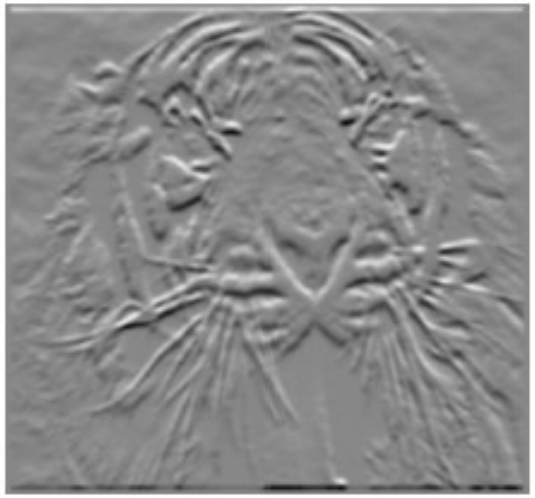
-1	1

Partial derivatives of an image

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

-1	1





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

-1
1

?

or

1
-1

Which one does show changes with respect to x?

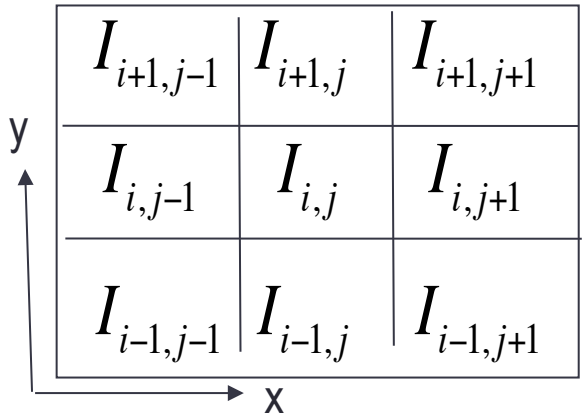
First derivatives: discrete operators.

Given that the function f to derive is our image I:

$$\frac{\partial f}{\partial y} = I_{i+1,j} - I_{i,j}$$

$$\frac{\partial f}{\partial x} = I_{i,j+1} - I_{i,j}$$

0	10	10
0	10	10
0	10	10



What happens when there is noise in the image?

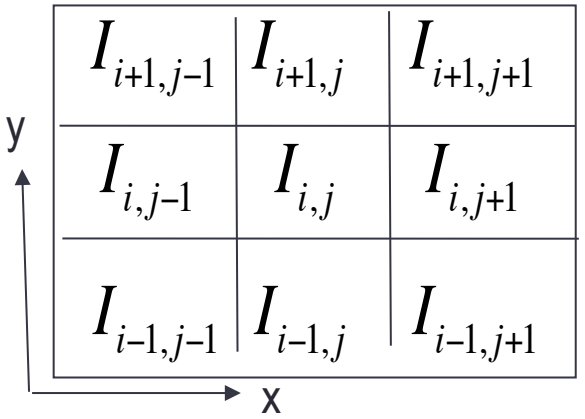
First derivatives: discrete operators.

Given that the function f to derive is our image I:

$$\frac{\partial f}{\partial y} = I_{i+1,j} - I_{i,j}$$

$$\frac{\partial f}{\partial x} = I_{i,j+1} - I_{i,j}$$

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 25 & 0 \\ 0 & 0 & 0 \end{matrix}$$



What happens when there is noise in the image?

How to be less sensitive to noise?

Alternative: getting the average of the 3 derivatives in order to be less sensitive to noise:

$$\frac{\partial I}{\partial y} = (I_{i+1,j+1} - I_{i,j+1}) + (I_{i+1,j} - I_{i,j}) + (I_{i+1,j-1} - I_{i,j-1})$$

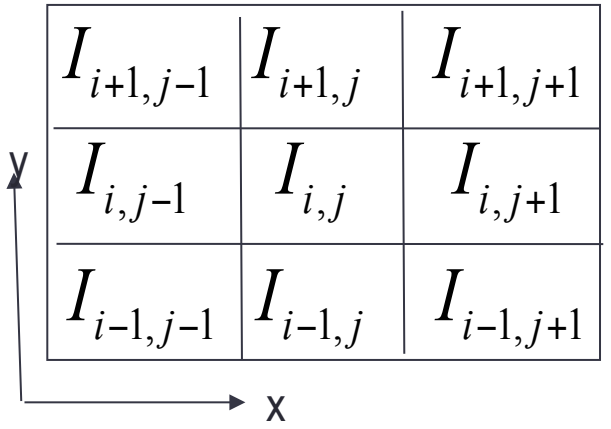
$$\frac{\partial I}{\partial x} = (I_{i+1,j+1} - I_{i+1,j}) + (I_{i,j+1} - I_{i,j}) + (I_{i-1,j+1} - I_{i-1,j})$$

First derivatives: discrete operators.

If we consider symetric finite differences:

$$\frac{\partial I}{\partial y} = (I_{i+1,j+1} - I_{i-1,j+1}) + (I_{i+1,j} - I_{i-1,j}) + (I_{i+1,j-1} - I_{i-1,j-1})$$

$$\frac{\partial I}{\partial x} = (I_{i+1,j+1} - I_{i+1,j-1}) + (I_{i,j+1} - I_{i,j-1}) + (I_{i-1,j+1} - I_{i-1,j-1})$$



What would be the conolution mask to compute the derivatives?

Mx=

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

Mx=

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

Prewitt masks

Assorted finite difference filters

Prewitt: $M_x =$

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

; $M_y =$

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

Sobel: $M_x =$

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

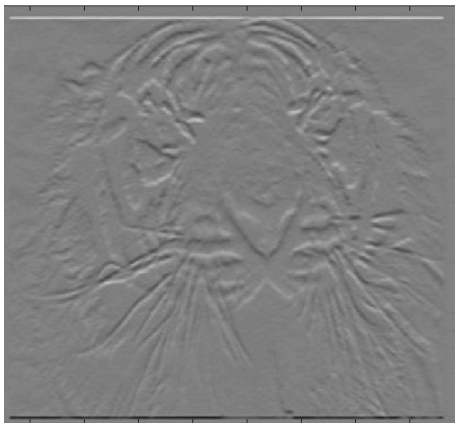
; $M_y =$

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

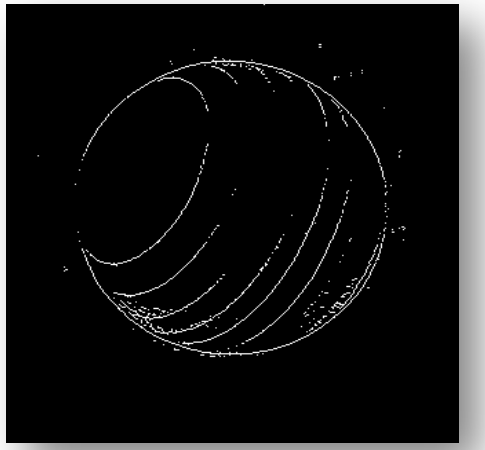
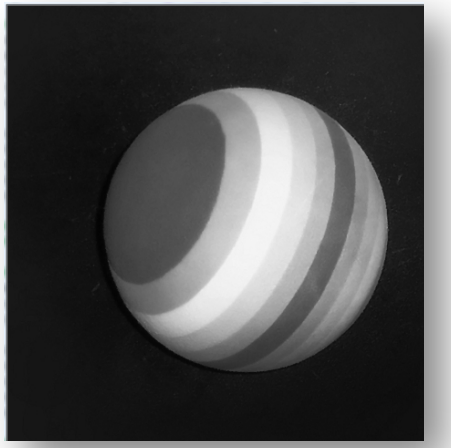
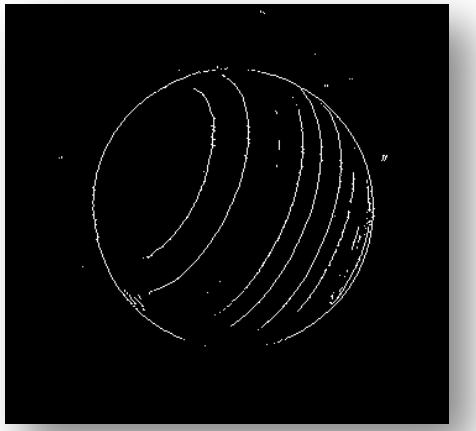
What is the difference between the Prewitt and Sobel operators?

- Sobel, by using non-uniform weights, gives more importance to the closest neighbors.

Which do you expect to be better?



Assorted finite difference filters

Determine to which Sobel masks (wrt x and y) do these images correspond to?!

How to apply Sobel in Matlab to detect the edges?

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

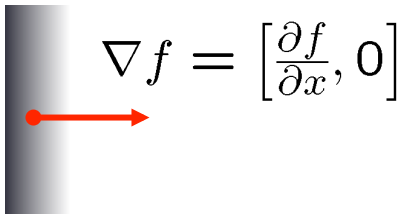
```
Alternative:
edge() -> Find edges in grayscale image

>> BW = edge(I)
>> BW = edge(I,'sobel')
>> BW = edge(I,'sobel',thresh,direction)
>> BW = edge(I,'prewitt')
```

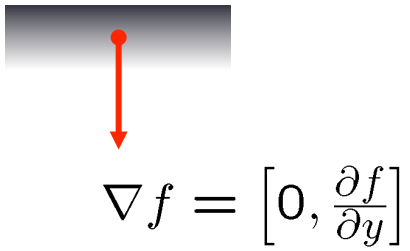
How to combine both derivatives (in x and y)?

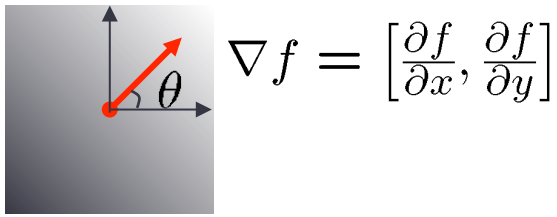
Image gradient

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 change in intensity







Slide credit Steve Seitz

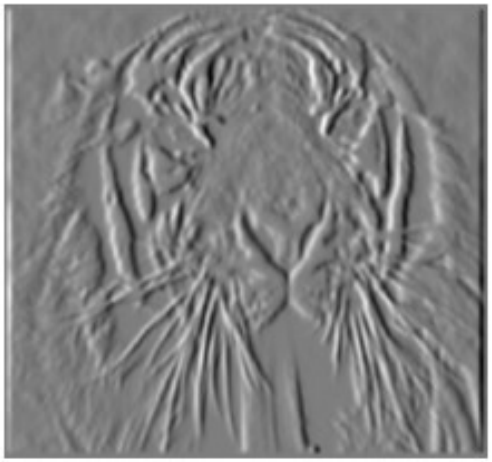
Image gradient

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 gradient magnitude

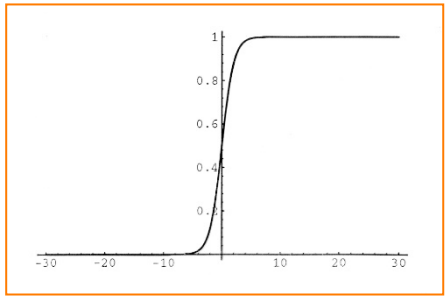
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

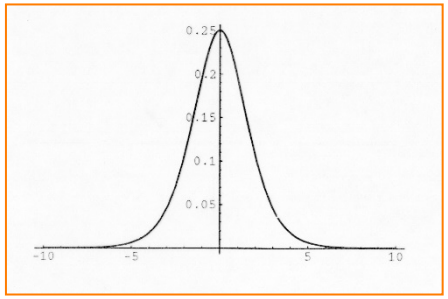


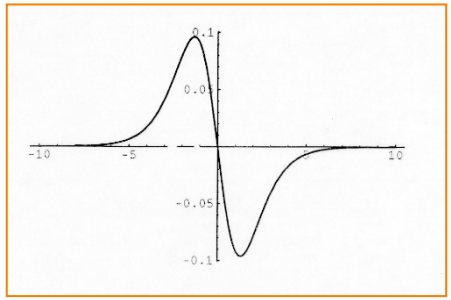
Slide credit Steve Seitz

Discrete operators: second derivatives

What do you expect about the second derivatives?



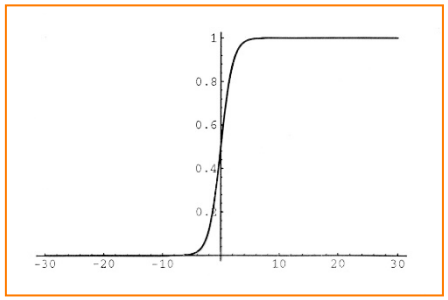
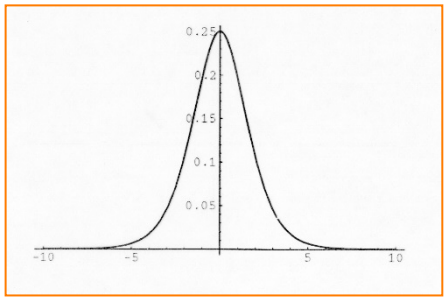
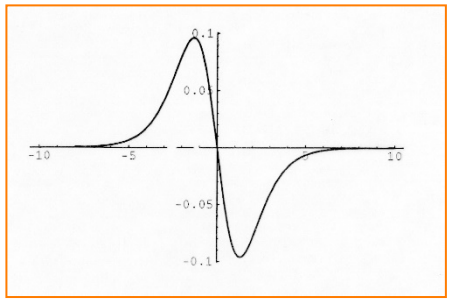




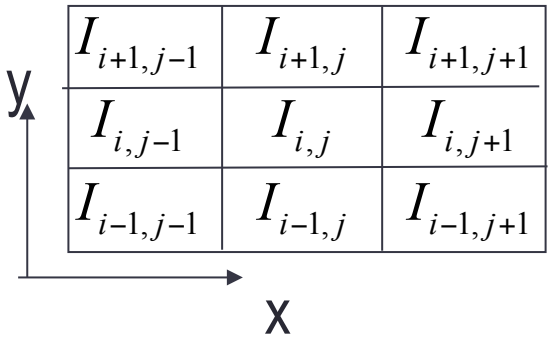
Given an edge, the first derivative has an extreme, and the second one has a zero-crossing in the edge and extremes before and after the edge.

Discrete operators: second derivatives

What do you expect about the second derivatives?

Given an edge, the first derivative has an extreme, and the second one has a zero-crossing in the edge and extremes before and after the edge.



The second derivative is the derivative of the first derivative:

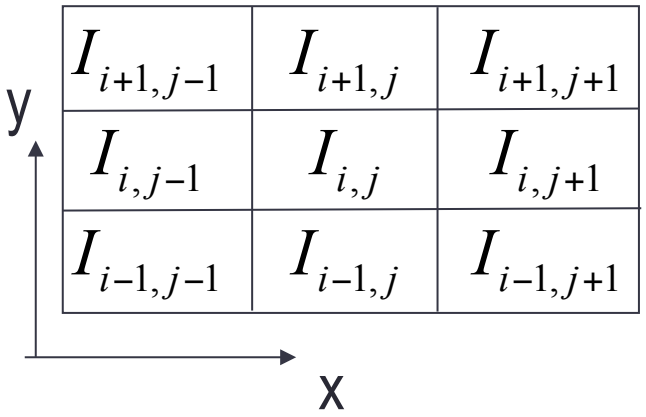
$$\frac{\partial^2 I}{\partial y^2} = (I_{i+1,j} - I_{i,j}) - (I_{i,j} - I_{i-1,j}) = (I_{i-1,j} - 2I_{i,j} + I_{i+1,j})$$

$$\frac{\partial^2 I}{\partial x^2} = (I_{i,j-1} - 2I_{i,j} + I_{i,j+1})$$

Discrete operators: Laplacian

Let's define:

$$\Delta I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = (I_{i-1,j} + I_{i,j-1} + I_{i+1,j} + I_{i,j+1}) - 4I_{i,j}$$



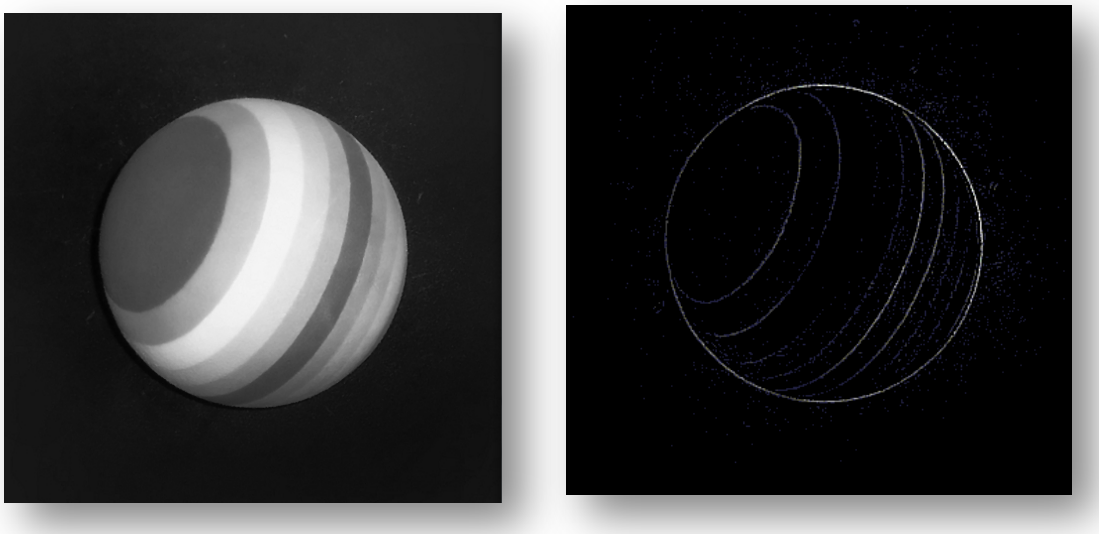
Laplacian mask:

$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Discrete operators: Laplacian

Laplacian mask:

$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$



Matlab:

h = fspecial('laplacian', alpha) returns a 3-by-3 filter approximating the shape of the two-dimensional Laplacian operator. The parameter alpha controls the shape of the Laplacian and must be in the range 0.0 to 1.0.

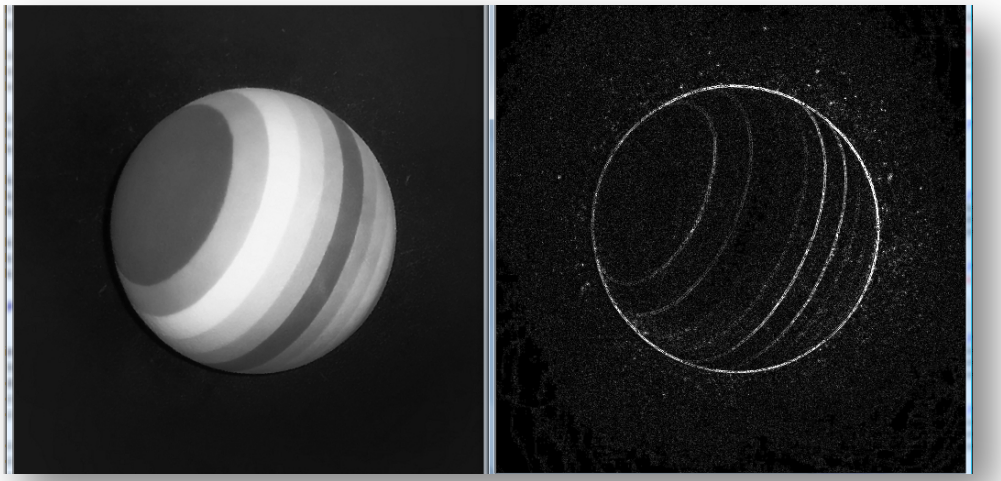
B = imfilter(im, h) filters the image im with the filter/mask h.

Discrete operators: Laplacian

Including the diagonal neighbours:

Laplacian:

$$\begin{matrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{matrix}$$

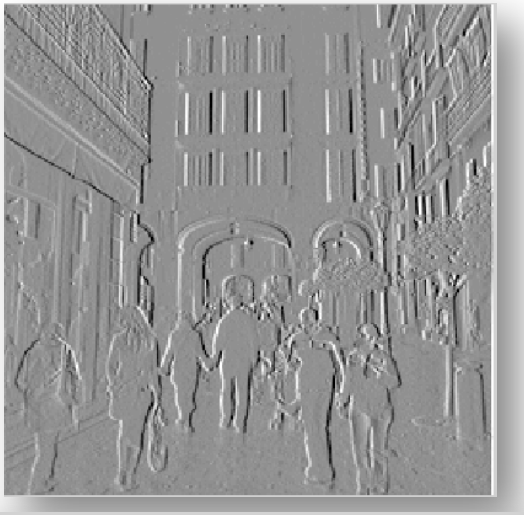


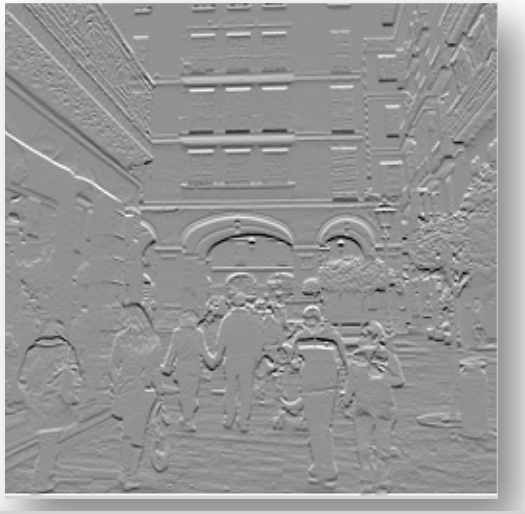
Mask properties

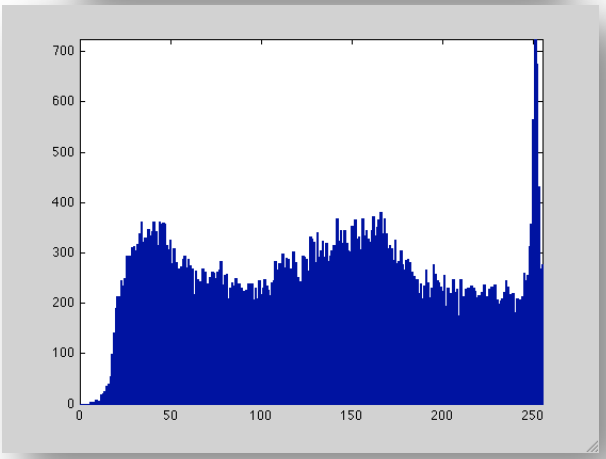
- Test on smoothing
  - Values should be _____
  - Sum to ___→ constant regions same as input
  - Amount of smoothing _____ to mask size
  - Remove "_____-frequency" components; "_____-pass" filter

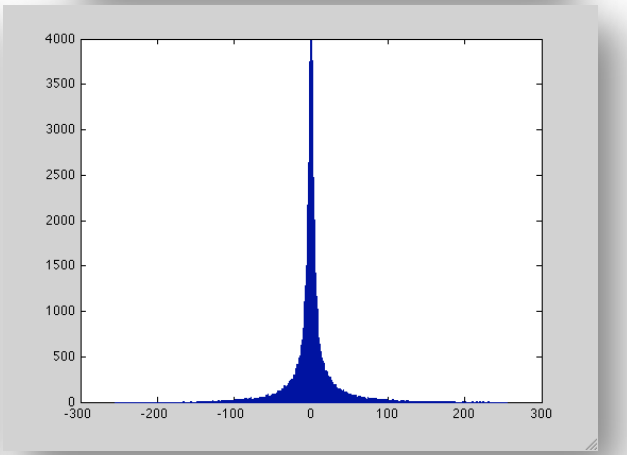
- Test on derivatives
  - _____ signs used to get high response in regions of high contrast
  - Sum to ___ → no response in constant regions
  - _____ value at points of high contrast

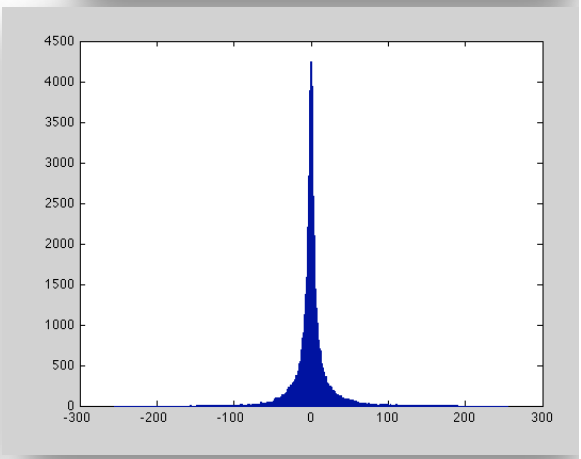
Explain the histogram of x and y edge maps









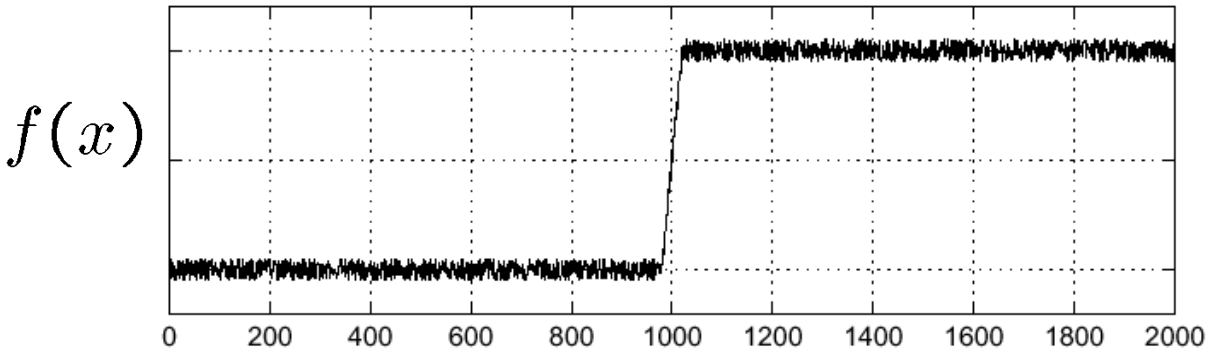


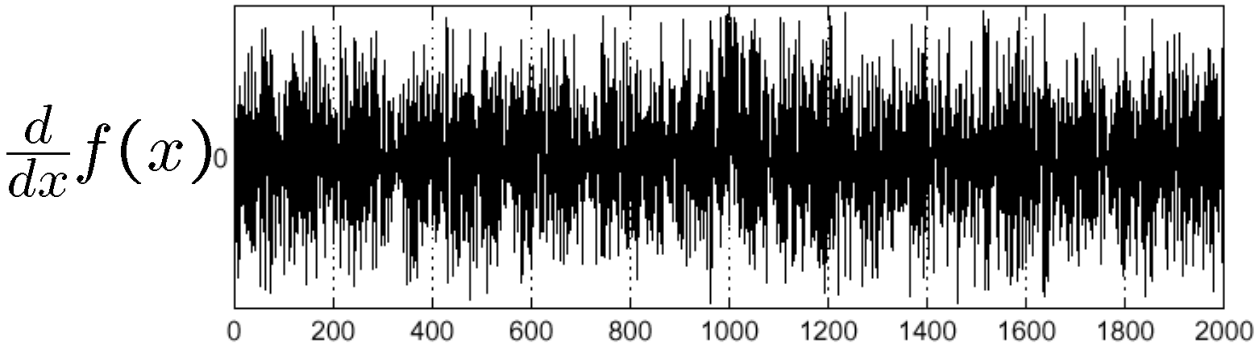
Horitzontal and vertical derivatives distribution: why are they positive and negative? Why the maxima is in 0? Can I store them in a uint8 type image?.

Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal





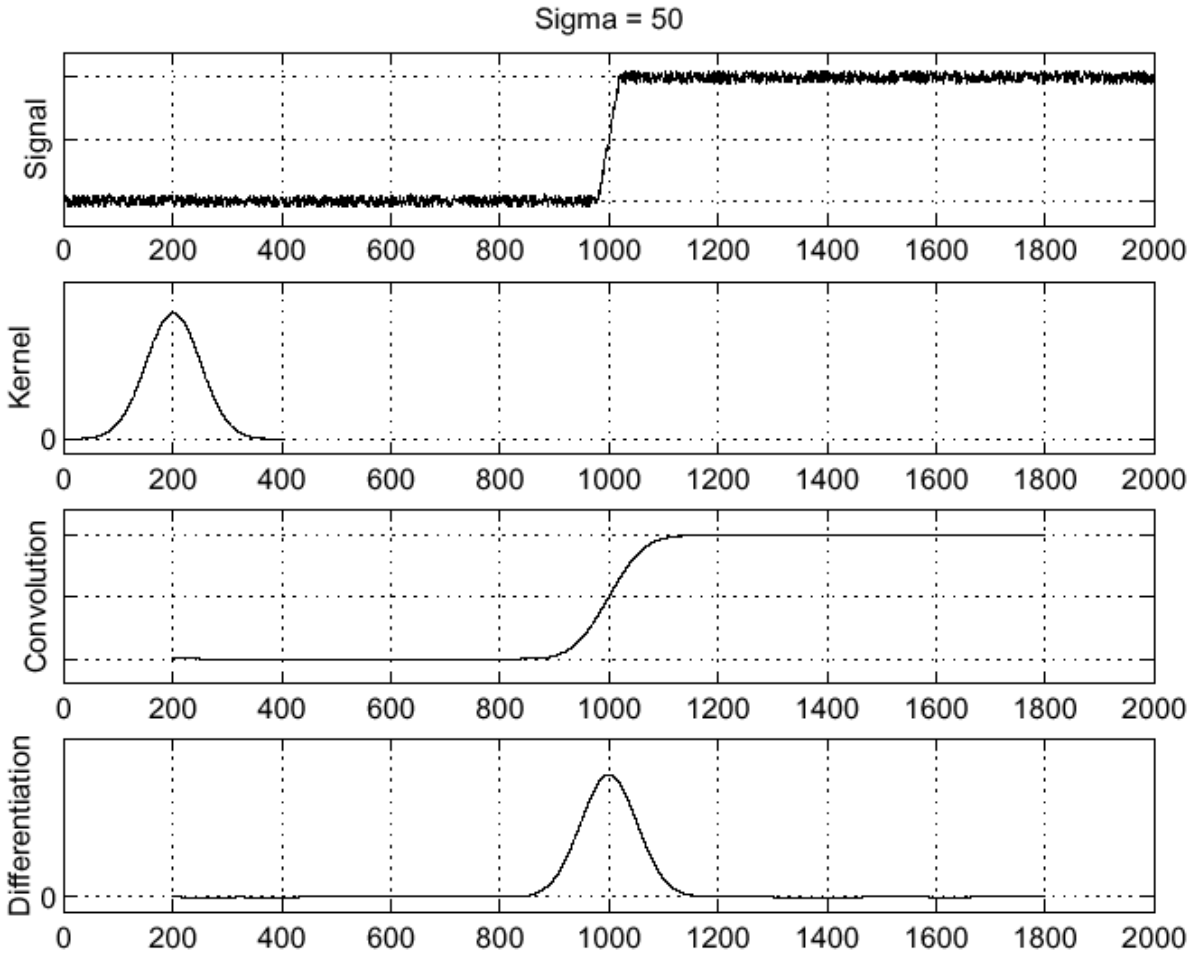
Where is the edge?

Slide credit Steve Seitz

Solution: smooth first

f

h



Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

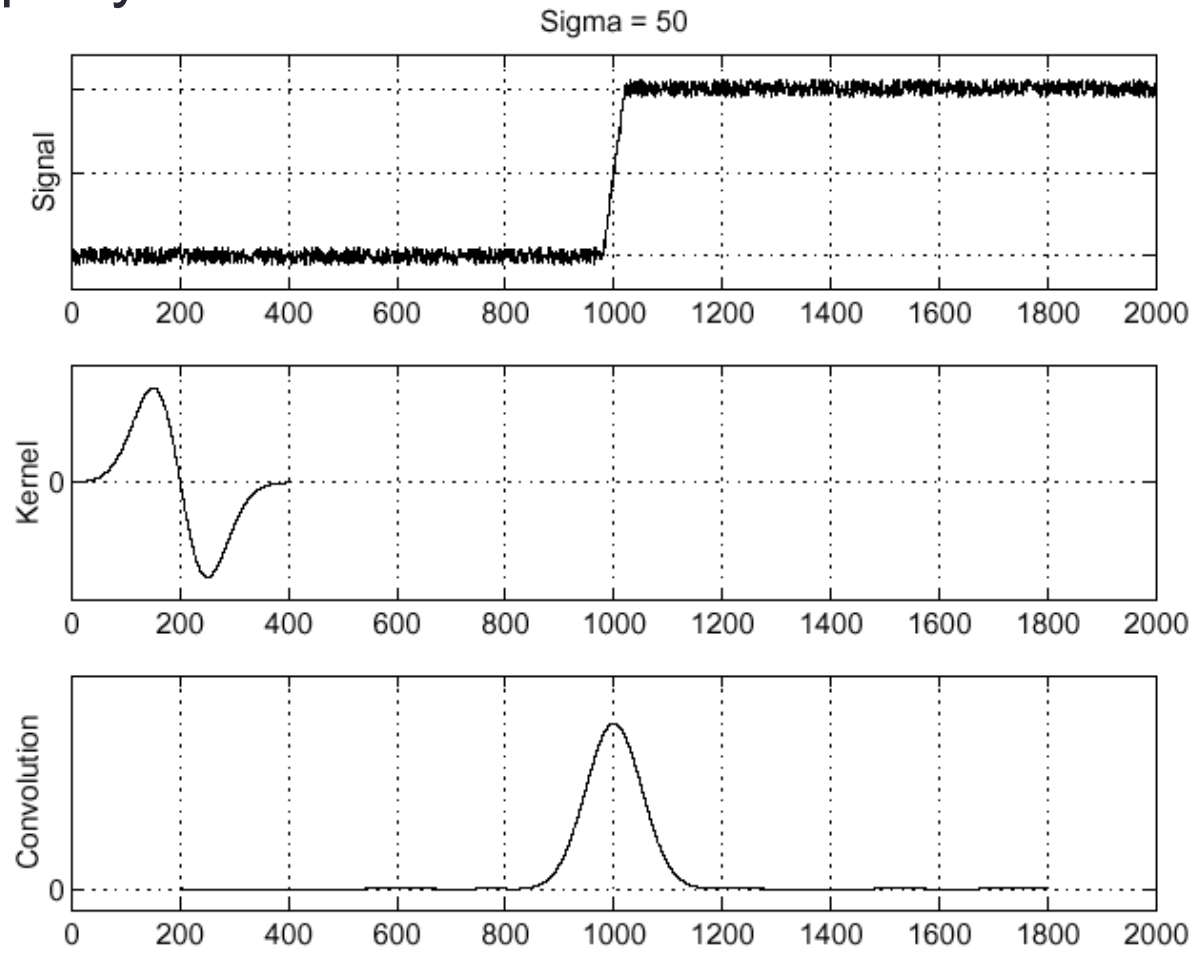
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

Differentiation property of convolution.

f

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

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

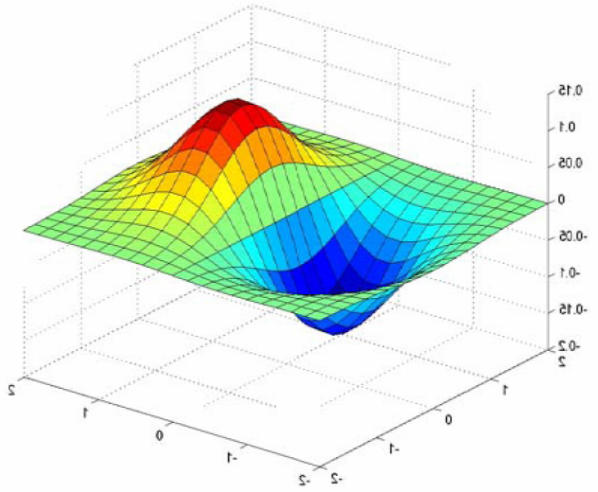


Slide credit Steve Seitz

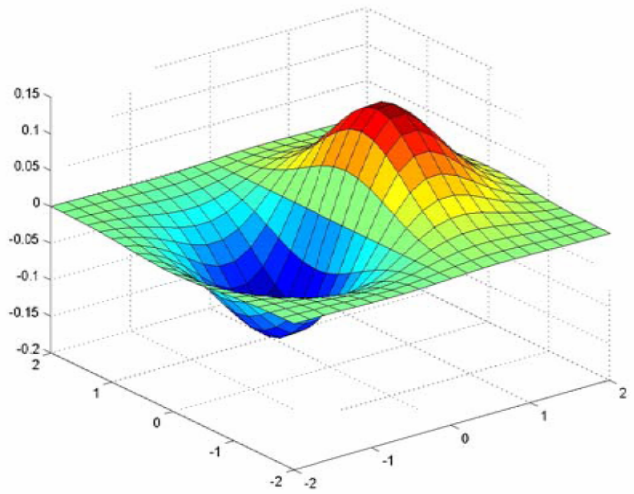
Derivative of Gaussian filters

$$(I \otimes g) \otimes h = I \otimes (g \otimes h)$$

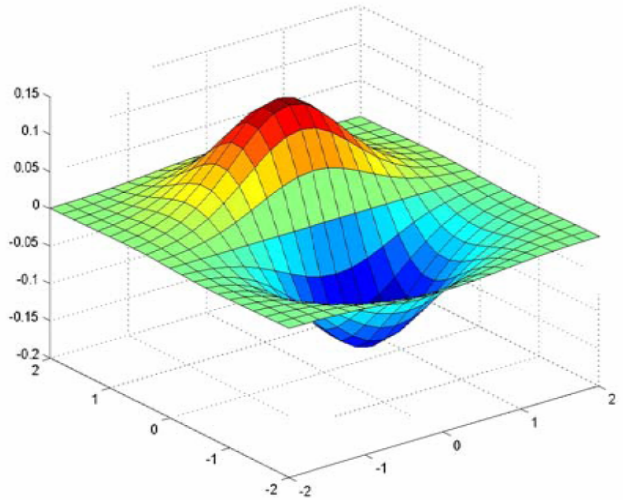
$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix}$$



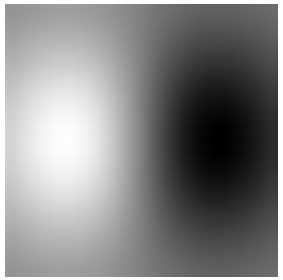
Derivative of Gaussian filters

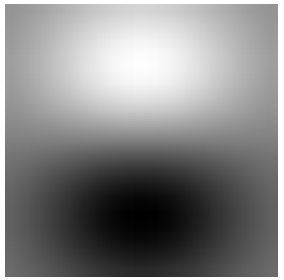


***x*-direction**



***y*-direction**





Gaussian

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

Derivative of Gaussian

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

Source: L. Lazebnik

Implementation in Matlab

```
function[gx,gy]=gaussianDerivatives(IM, sigma)
% Given: IM – image and sigma – parameter of the Gaussian:

    epsilon=1e-2;
    halfsize=ceil(sigma*sqrt(-2*log(sqrt(2*pi)*sigma*epsilon)));
    size=2*halfsize+1;
    IM=double(IM);

    %generate a 2-D Gaussian kernel along x direction
    for i=1:size
       for j=1:size
          u=[i-halfsize-1 j-halfsize-1];
          hx(i,j)=gauss(u(1),sigma)*dgauss(u(2),sigma);
       end
    end

    hx=hx/sqrt(sum(sum(abs(hx).*abs(hx)))); %normalization

    %generate a 2-D Gaussian kernel along y direction
    hy=hx';

    %2-D filtering
    gx=imfilter(IM,hx,'replicate','conv');
    gy=imfilter(IM,hy,'replicate','conv');
end
```

```
function y = gauss(x,sigma)
%Gaussian

            y = exp(-x^2/(2*sigma^2)) /  ...
                  (sigma*sqrt(2*pi));
end
%%%%%%%%%%%%%%%%%%%%%%

function y = dgauss(x,sigma)
%first order derivative of Gaussian

            y = -x * gauss(x,sigma) / sigma^2;
end
```

It can be shown that **convolving with a 2D Gaussian** is the same that **convolving with a 1D Gaussian** in x direction followed by convolving in y direction!

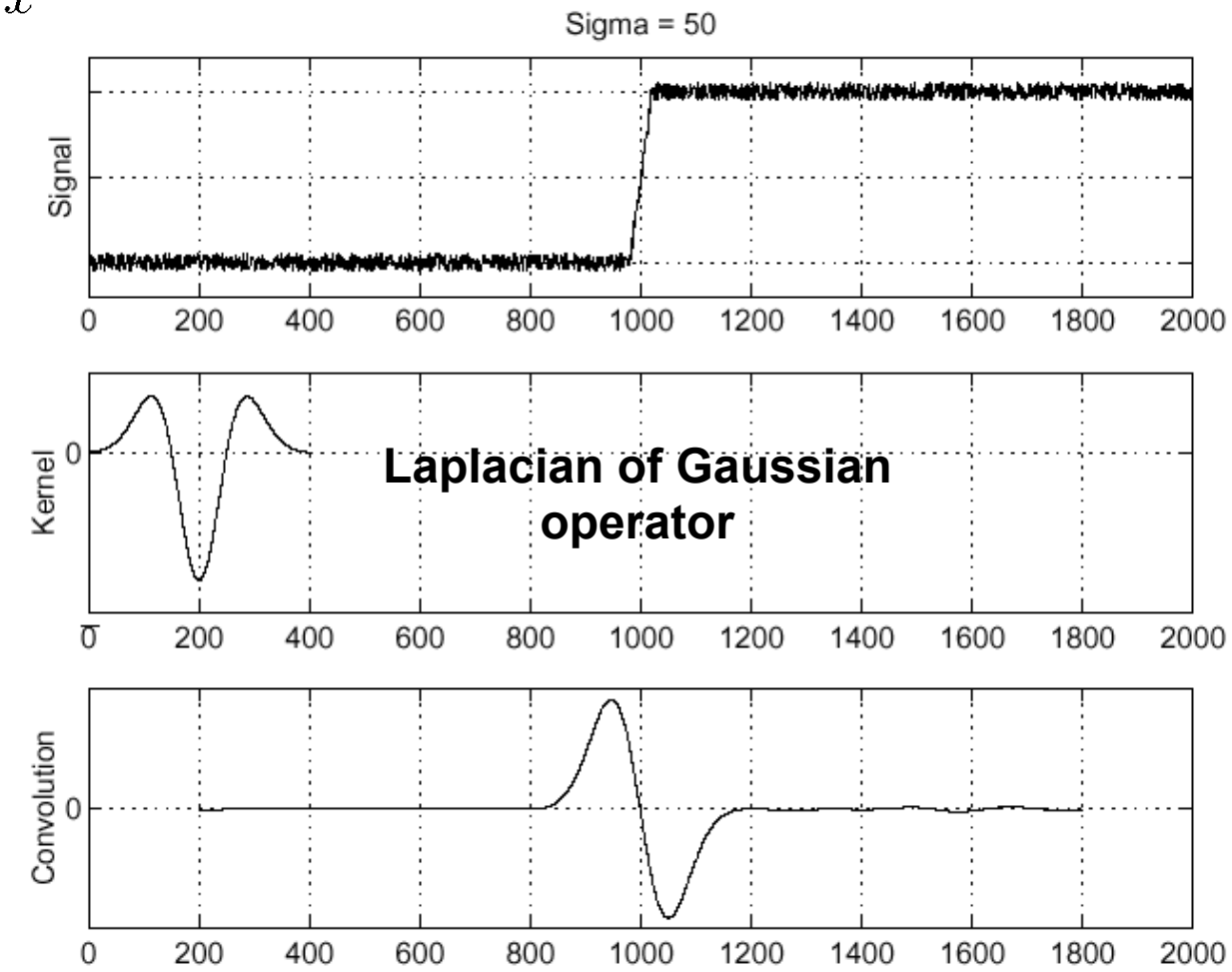
Laplacian of Gaussian

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

f

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

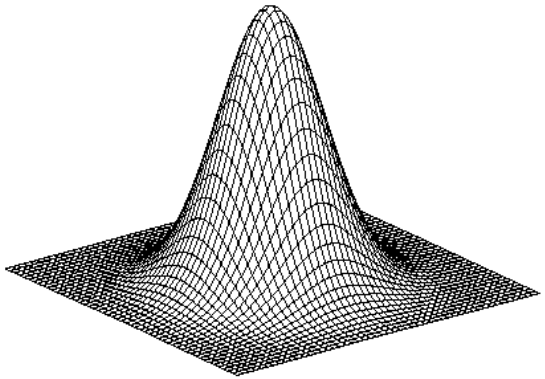
$(\frac{\partial^2}{\partial x^2}h) \star f$



Where is the edge? Zero-crossings of bottom graph

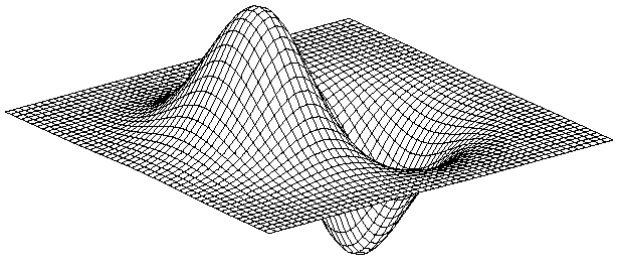
Slide credit: Steve Seitz

2D edge detection filters



Gaussian

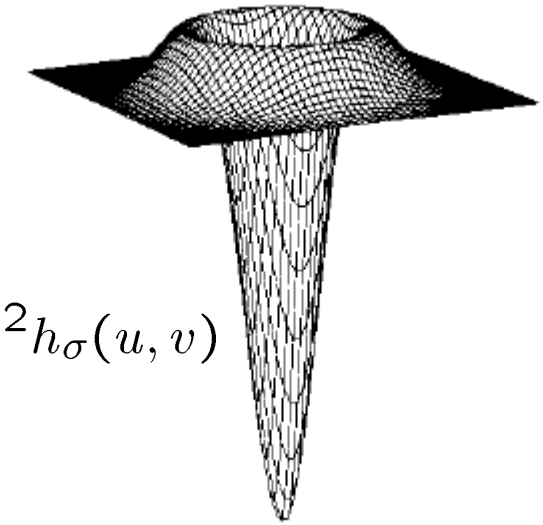
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Derivative of Gaussian

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

Laplacian of Gaussian



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

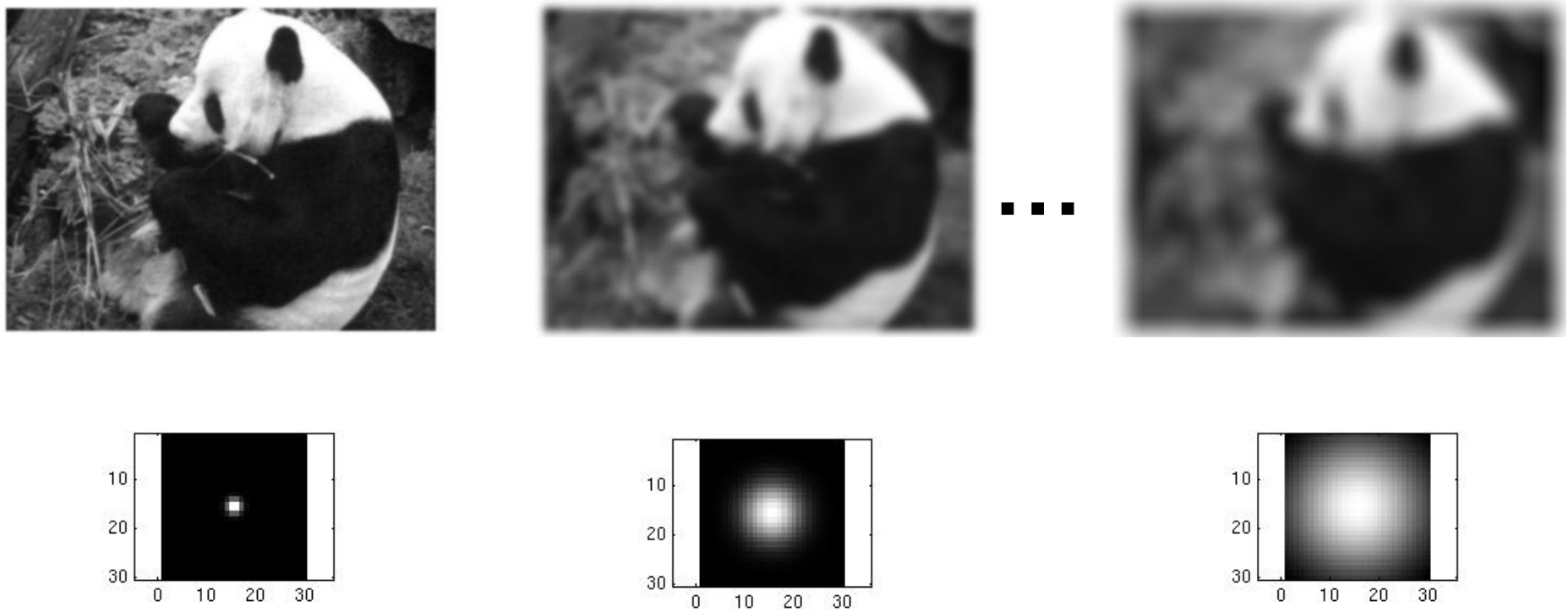
- ∇^2 is the Laplacian operator:

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

Slide credit: Steve Seitz

Smoothing with a Gaussian

Recall: parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.



Effect of σ on derivatives

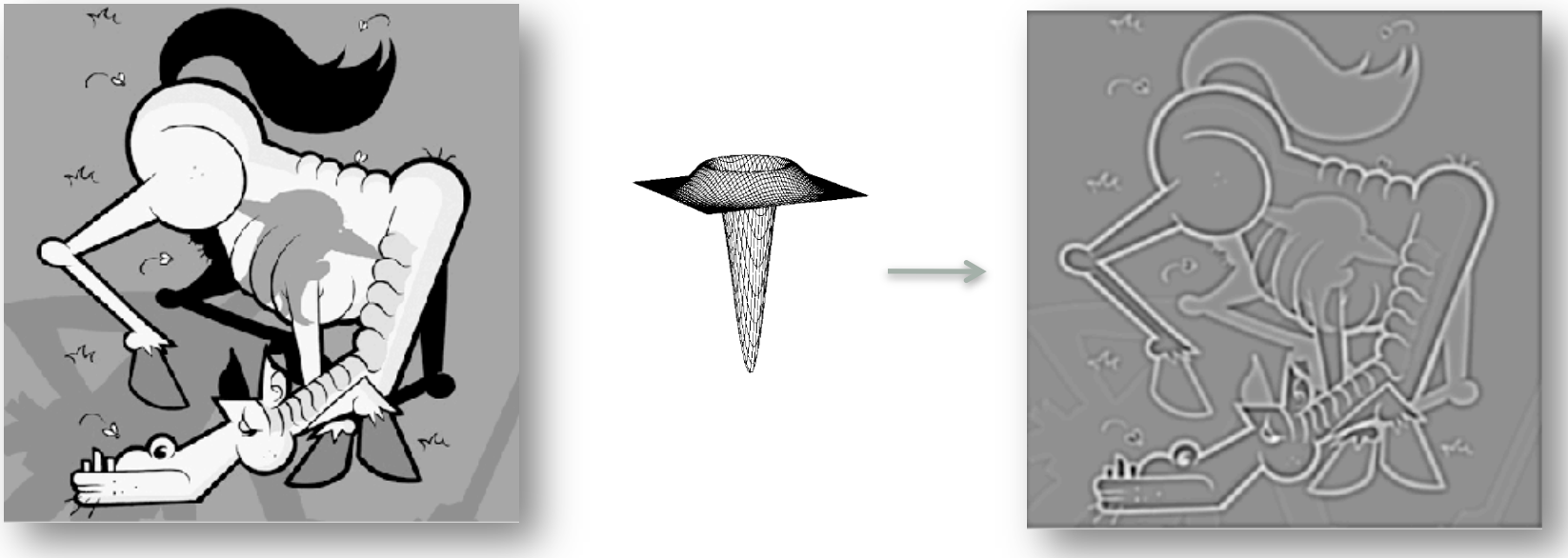
σ = 1 pixel

σ = 3 pixels

The apparent structures differ depending on Gaussian's scale parameter.

Larger values: larger scale edges detected.
Smaller values: finer features detected.

Laplacian



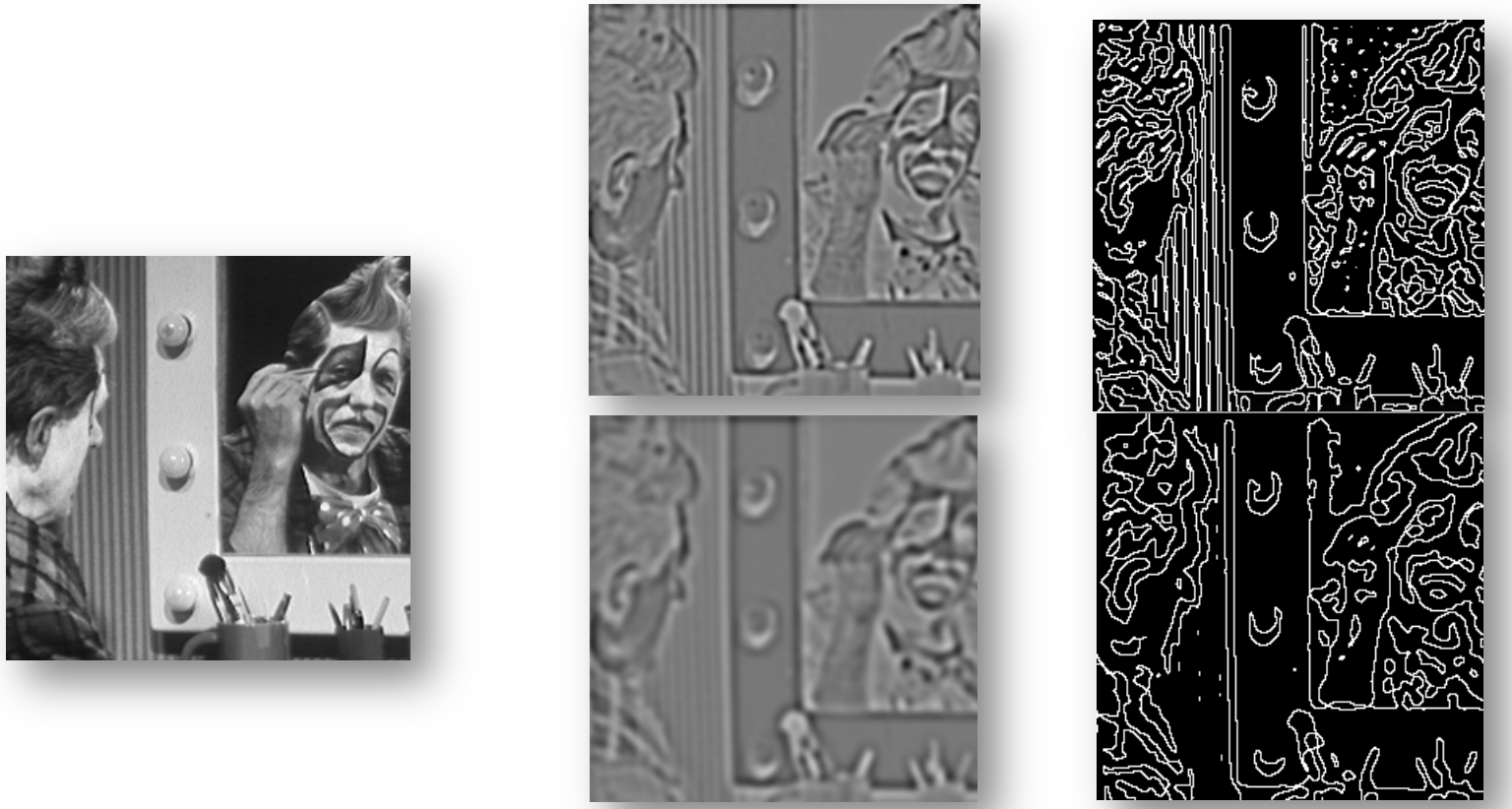
$$\nabla^2 h(x,y) = \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) h(x,y)$$

Matlab:

h=fspecial('laplacian',0.2);

imLaplacian=imfilter(im,h);

Laplacian and zero-crossings



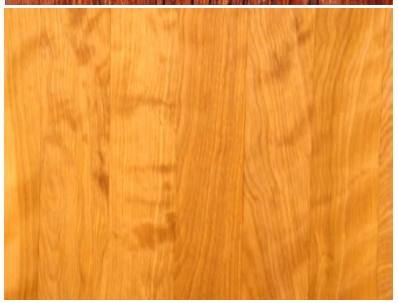
Using different sigma values.

Why edges obtained by the zero-crossing of the Laplacian map assure to be continuous?

Although the Laplacian operator convolved with the image leads to contours, they can be too much (depends on sigma).

So, what scale to choose?

It depends what we're looking for.

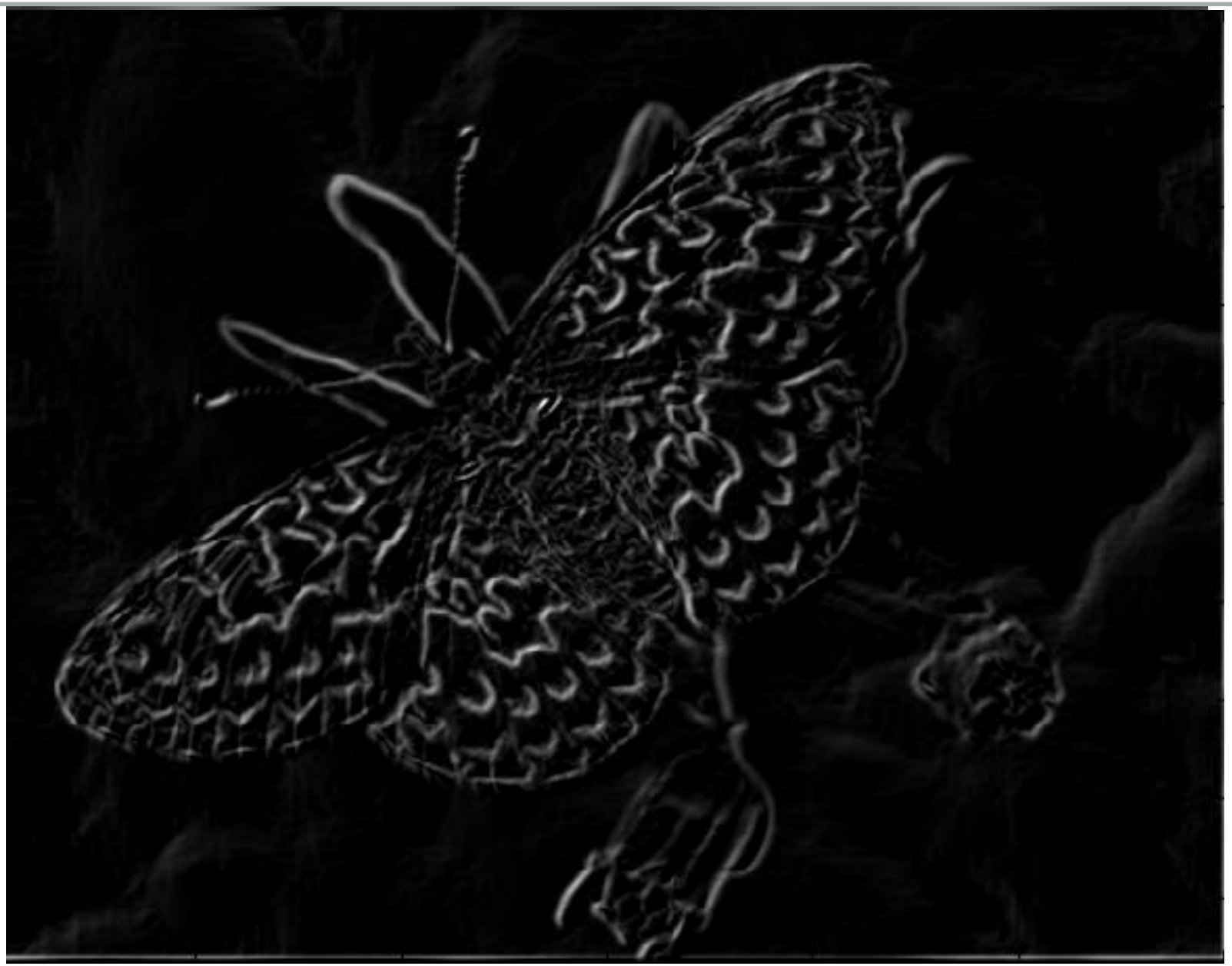


We need previous knowledge about the right scale.

Or manage different scales. How?

Original image

Gradient magnitude image



Thresholding gradient with a lower threshold

Thresholding gradient with a higher threshold

Canny edge detector

- Filter image with derivative of Gaussian
- Find magnitude and orientation of gradient

- **Non-maximum suppression**:
  - Thin wide “ridges” down to single pixel width

- **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

- MATLAB: `edge(image, 'canny');`
- `>>help edge`

Source: D. Lowe, L. Fei-Fei

The Canny edge detector

original image (Lena)

Slide credit: Steve Seitz

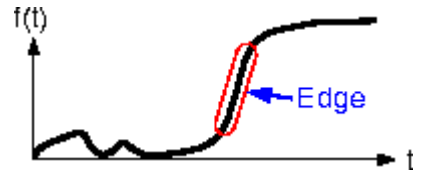
The Canny edge detector

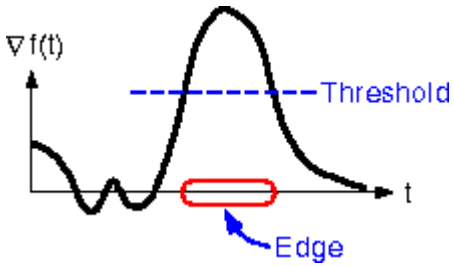
norm of the gradient

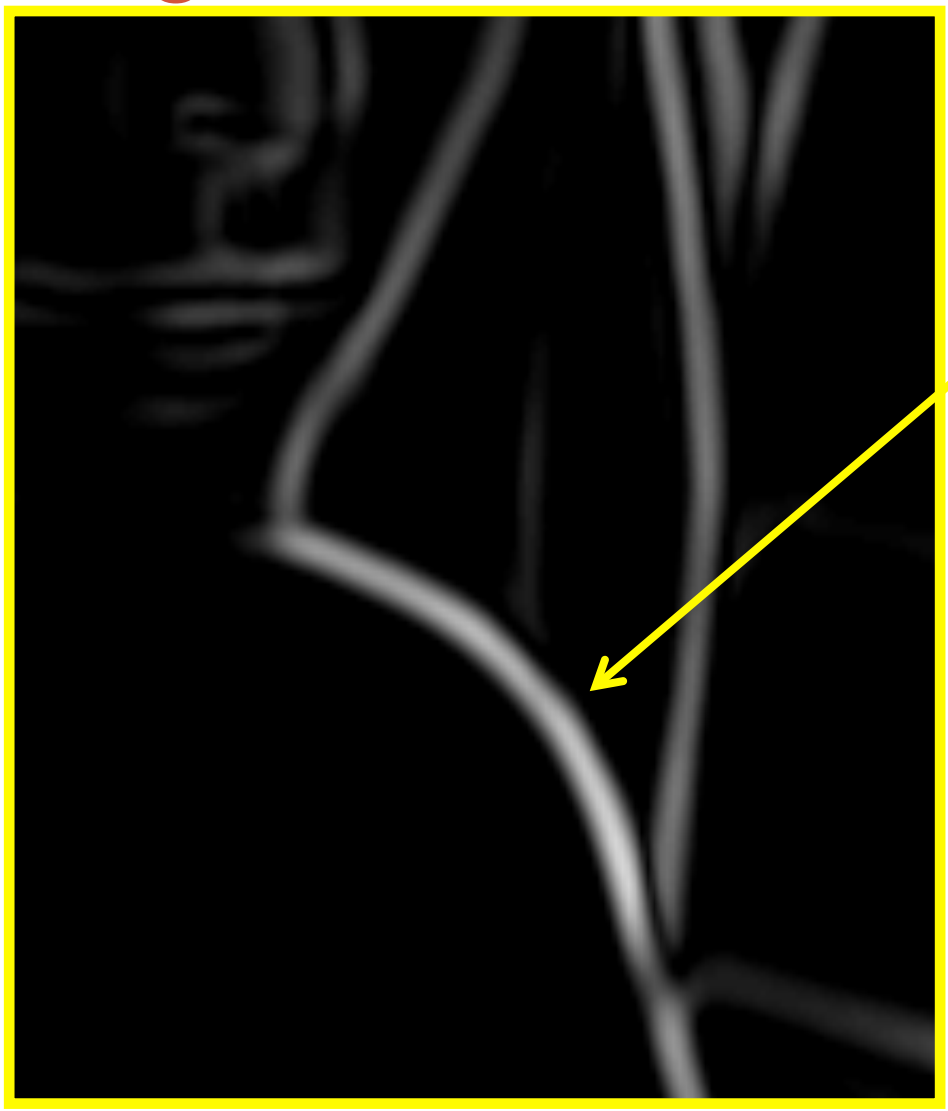
The Canny edge detector

thresholding

The Canny edge detector

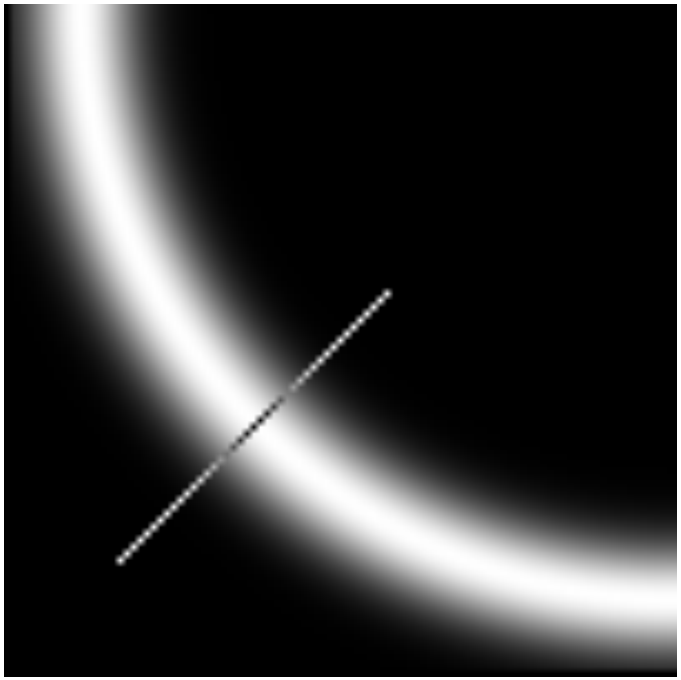






How to turn these thick regions of the gradient into curves?

Non-maximum suppression



Check if pixel is local maximum along gradient direction,
select single max across width of the edge

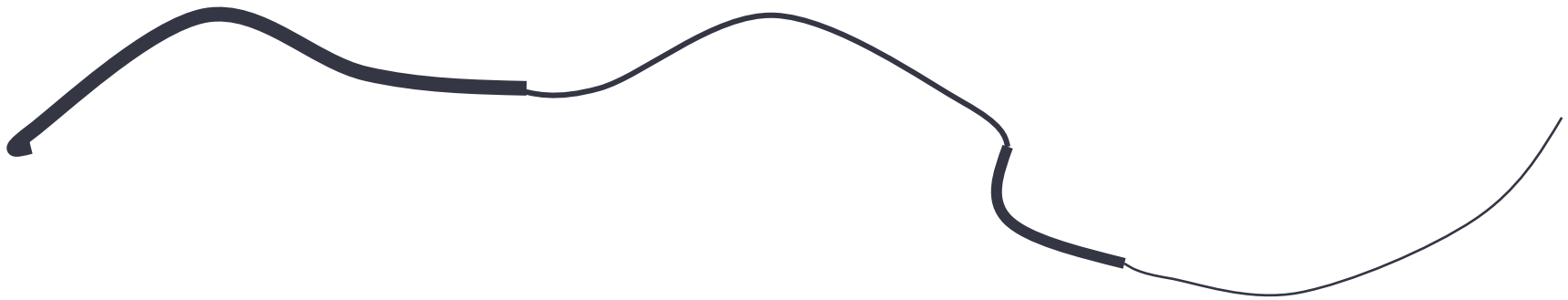
The Canny edge detector

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

thinning
(non-maximum suppression)

Hysteresis thresholding

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



Source: Steve Seitz

Hysteresis thresholding

original image

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Source: L. Fei-Fei

Hysteresis thresholding

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Source: L. Fei-Fei

Recap: Canny edge detector

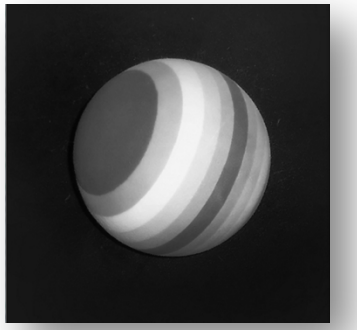
1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. **Non-maximum suppression**:
   - Thin wide “ridges” down to single pixel width
4. **Linking and thresholding** (**hysteresis**):
   - Define two thresholds: low and high
   - Use the high threshold to start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

- MATLAB: `edge(image, 'canny');`
- `>>help edge`

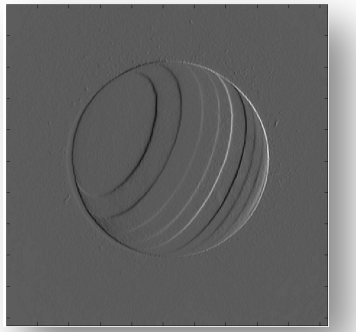
Source: D. Lowe, L. Fei-Fei

Exercices (comparison)

- Determine the operator applied:

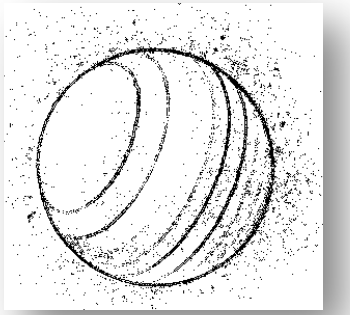


Original image



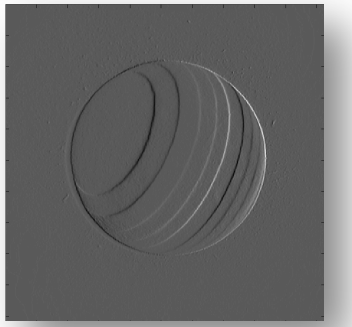
Prewit with a mask

```
mask=[[-1, 0, 1];[-1, 0, 1];[-1, 0,1]];
imP=conv2(double(im),mask);
imshow(imP,[]);
```



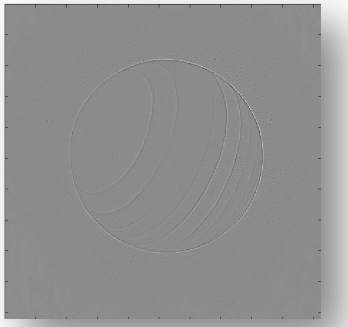
Laplacian with a mask

```
mask=[[0,-1,0];[-1,4,-1];[0,-1,0]];
imLM=conv2(double(im),mask);
imshow(abs(imLM)<10,[]);
```



Sobel with a mask

```
mask=[[-1, 0, 1];[-2, 0, 2];[-1, 0,1]];
im2=conv2(double(im),mask);
imshow(im2,[]);
```

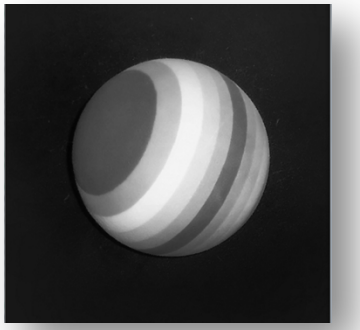


Laplacian with a mask

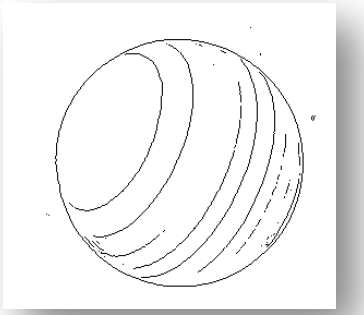
```
mask=[[0,-1,0];[-1,4,-1];[0,-1,0]];
imLM=conv2(double(im),mask);
```

Exercices (comparison)

- Determine the operator applied:

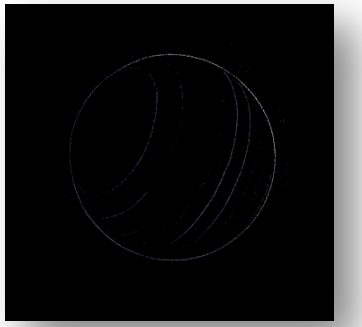


Original image



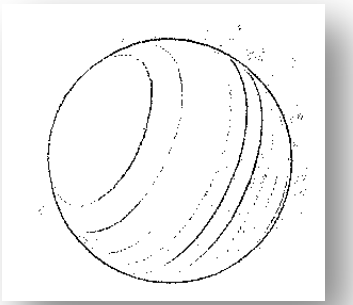
Sobel

ims=edge(im,'sobel');
imshow(1-ims,[]);



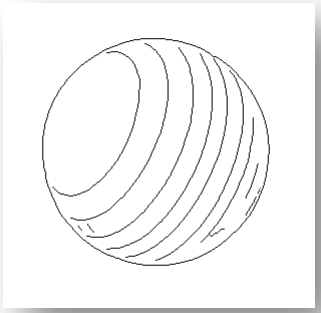
Laplacian of a Gaussian

Lapl=fspecial('log');
imL=imfilter(im,Lapl);
imshow(imL,[]);



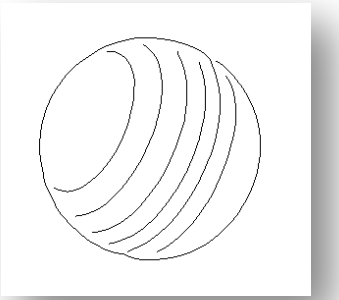
Laplacian of a Gaussian

Lapl=fspecial('log');
imL=imfilter(im,Lapl);
imshow(1-im2bw(imL,0.08));



Canny sigma=1, thr=0.2

imC=edge(im,'canny',0.2,1)
imshow(1-imC,[])



Canny sigma=2, thr=0.2

imC=edge(im,'canny',0.2,2)
imshow(1-imC,[])

Low-level edges vs. perceived contours

Background

Texture

Shadows

Is Canny edge detector distinguishing these contours?

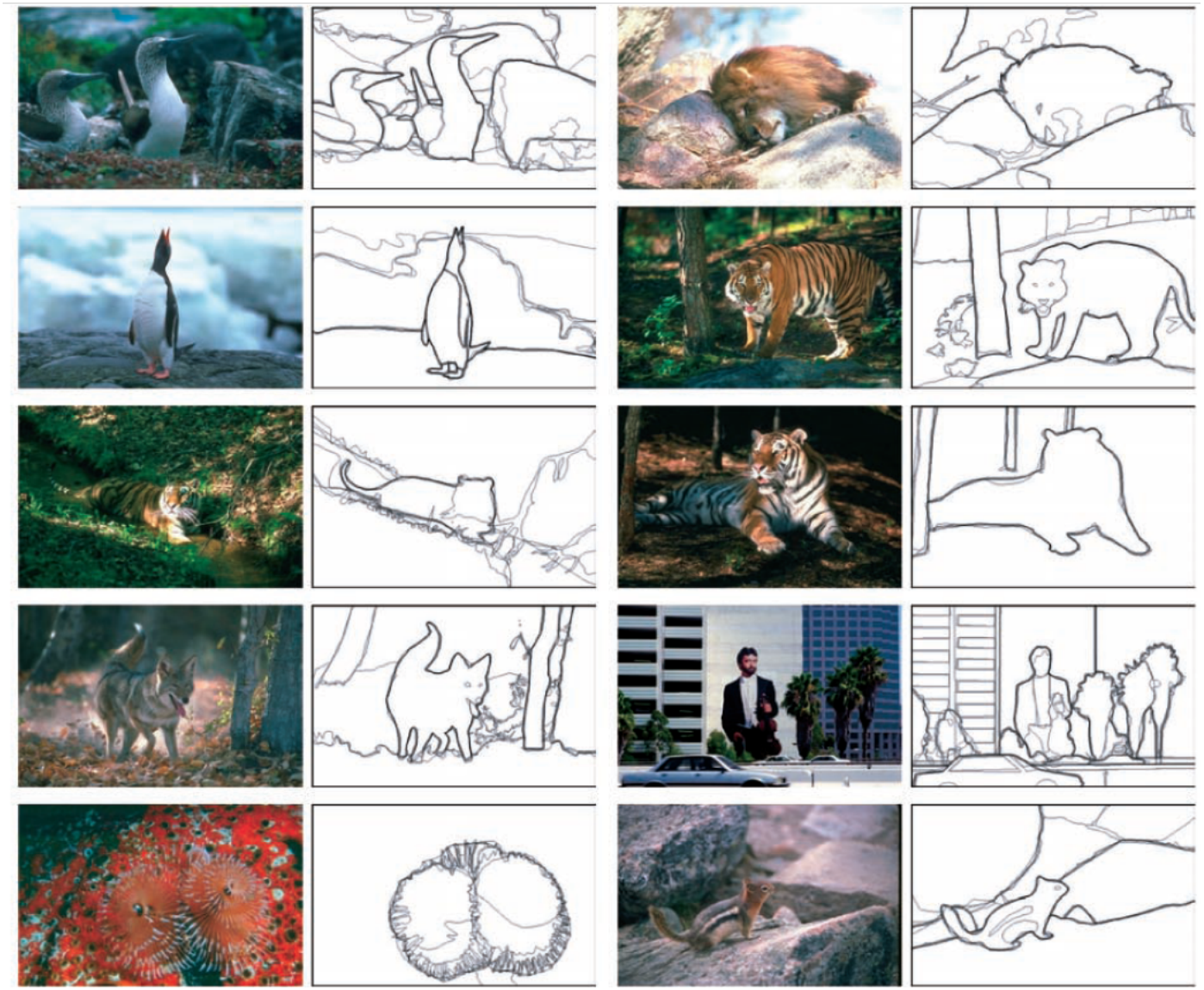
Low-level edges vs. perceived contours

image **human segmentation** **gradient magnitude**

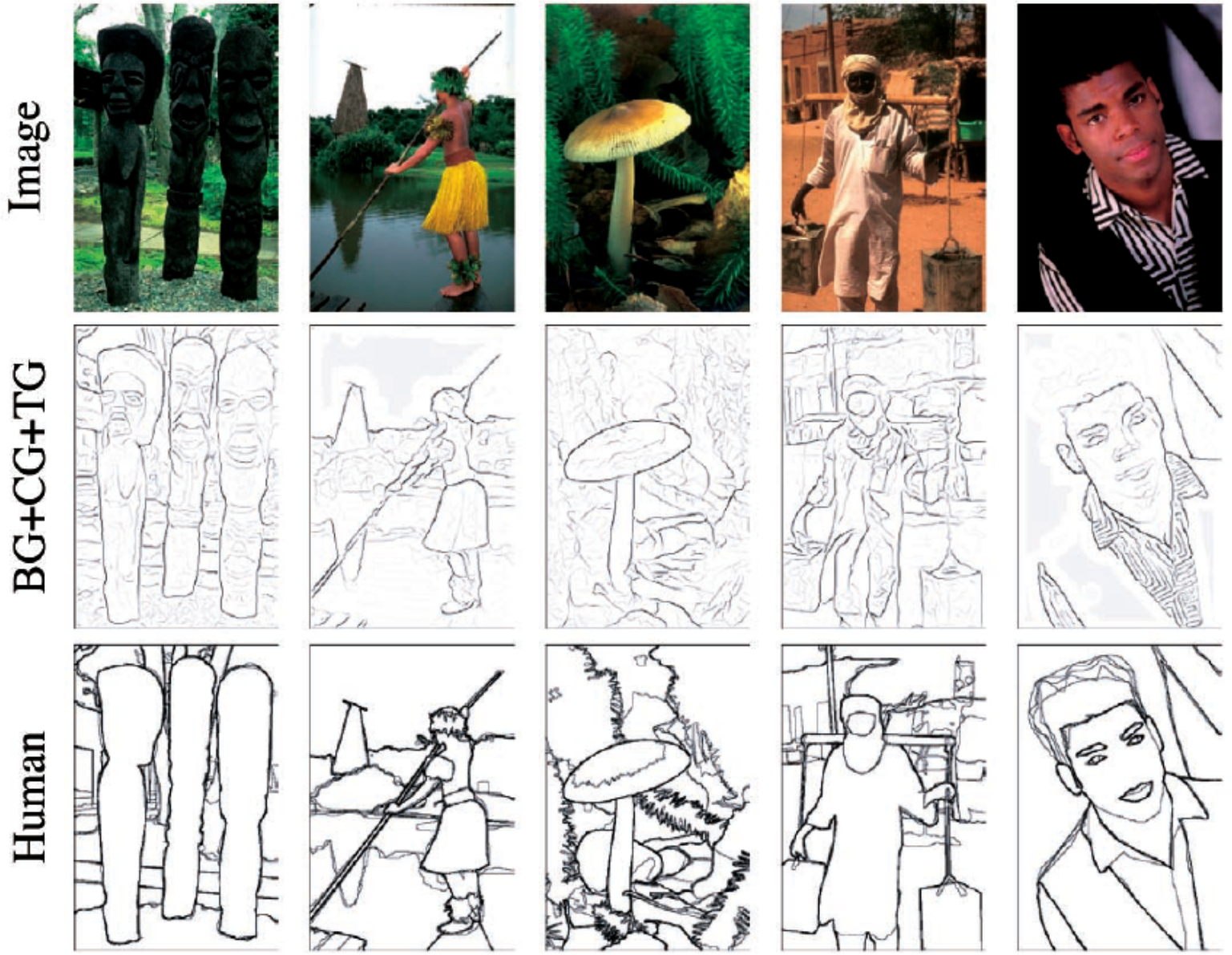
- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Source: L. Lazebnik

Learn from humans which combination of features is most indicative of a “good” contour



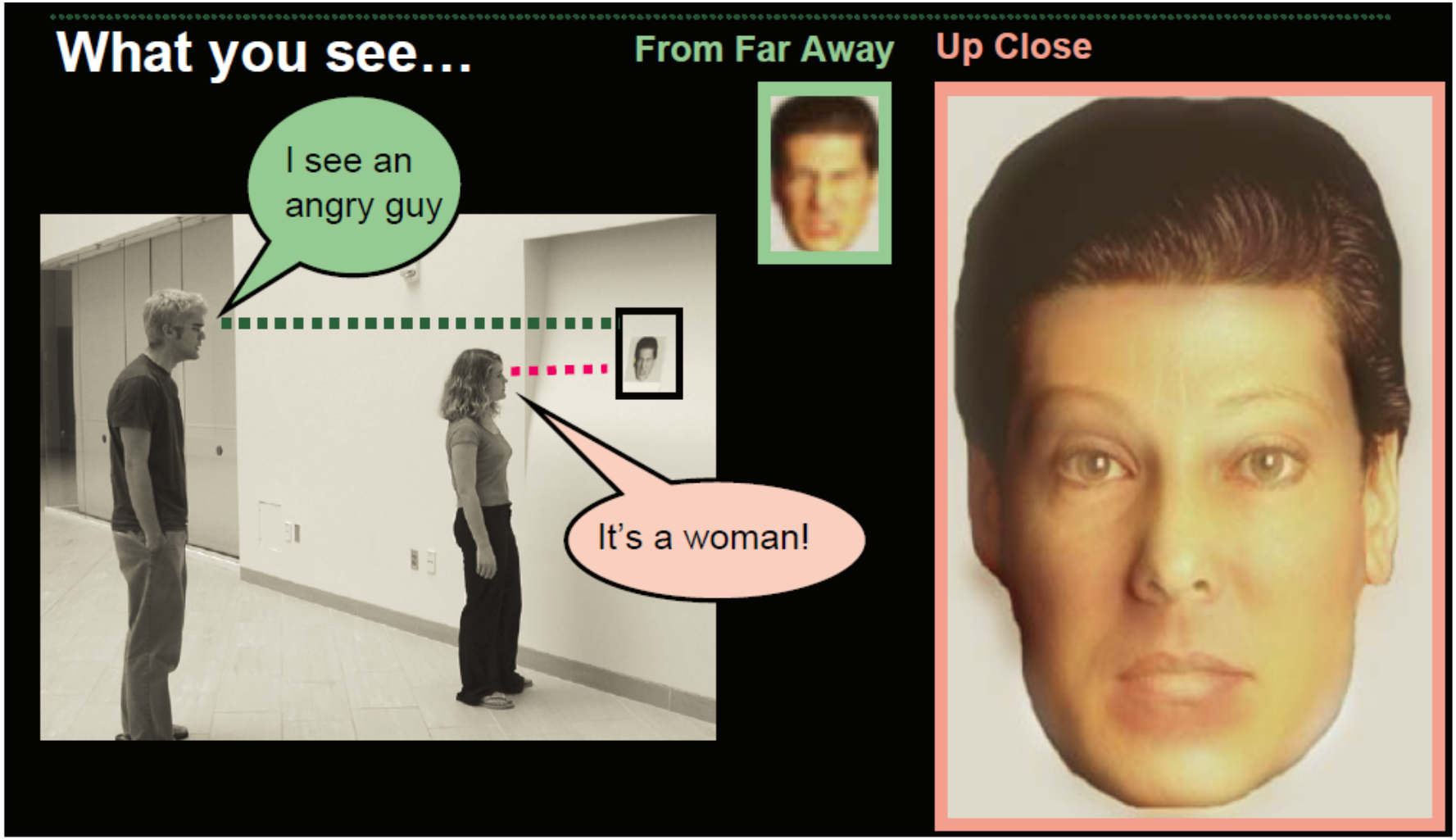
[D. Martin et al. PAMI 2004] Human-marked segment boundaries



Summing the brightness gradient magnitude with the color gradient and texture gradient approximates better the good contours.

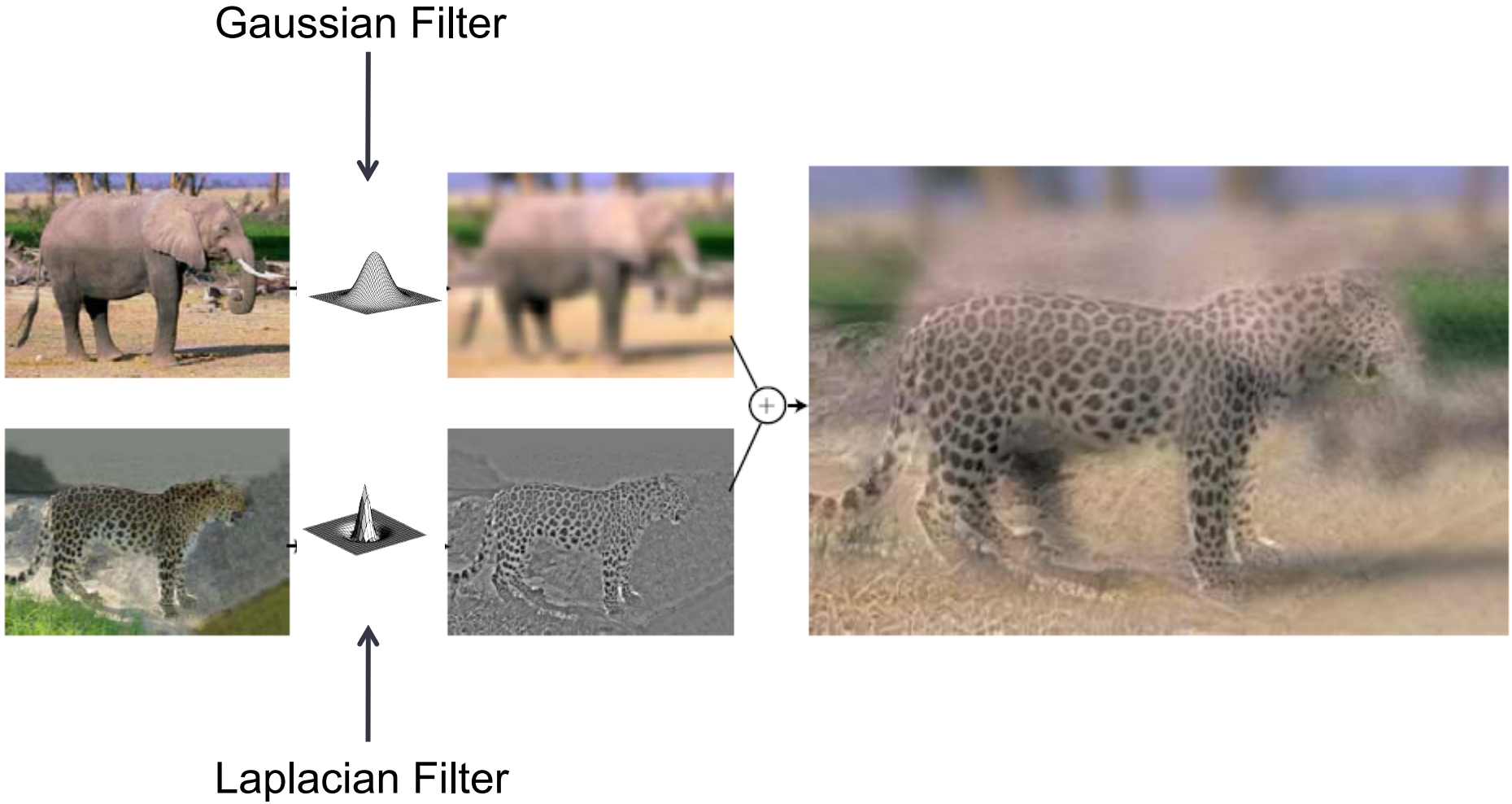
[D. Martin et al. PAMI 2004]

Application: Hybrid Images

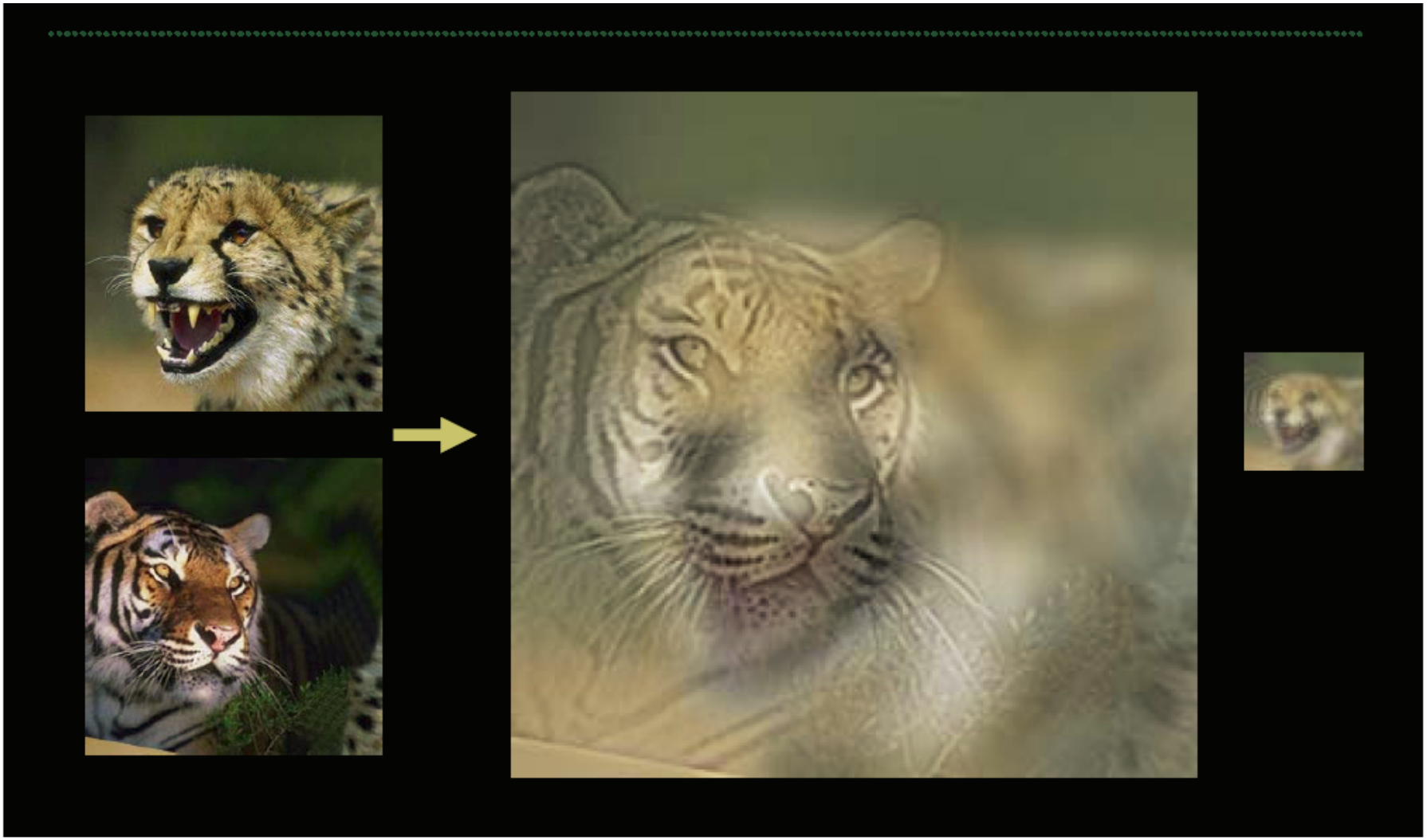


Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

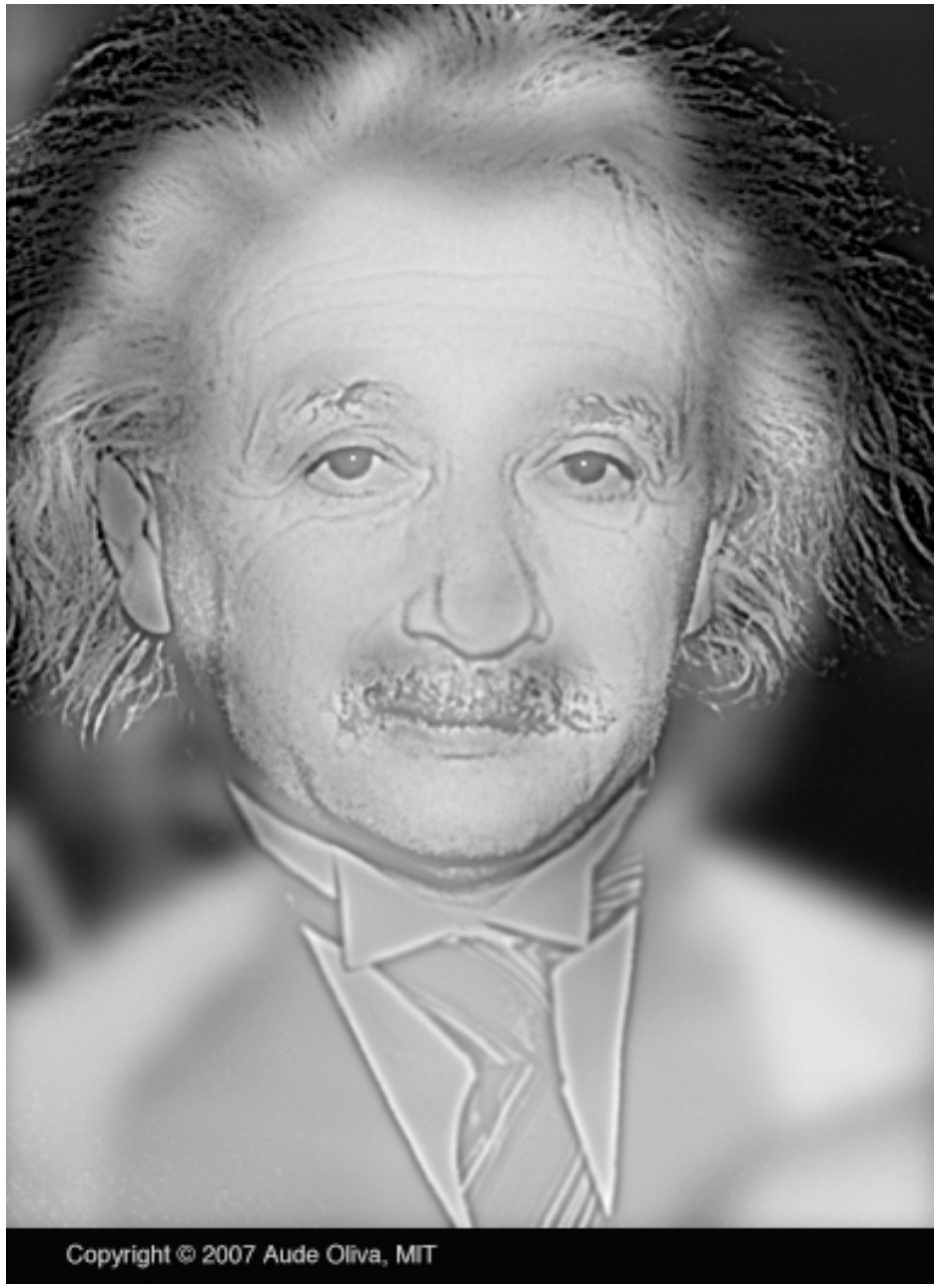
Application: Hybrid Images

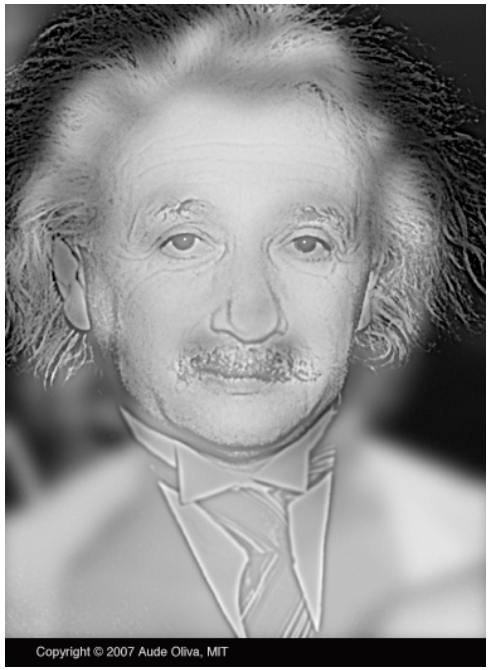


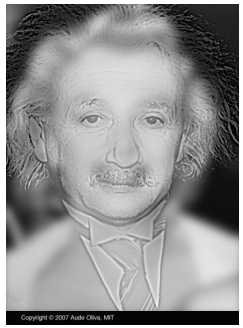
A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006



Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006







http://cvcl.mit.edu/hybrid_gallery/gallery.html

Changing expression

SIGGRAPH2006

Sad ⟷ Surprised

Summary

- Filters allow local image neighborhood to influence our description and features
  - Smoothing to reduce noise
  - Derivatives to locate contrast, high image gradient
- Different edge detectors:
  - Sobel & Prewitt – fast but sensitive to noise
  - Convolution with the Gradient of a Gaussian –
    - Less fast but more robust
    - Using different sigma allows to smooth more or less
    - Zero-crossing with a Laplacian – assures closed contours
  - Canny edge detector **– state of the art edge detector**
    - Assures continuous and thin contours due to the hysteresis and the thinning steps but needs parameters
    - Edges used to contain good contours but also many other pixels with high intensity change
- Still the question about the object contours is not solved!