COMPUTATIONAL VISION:

EDGES

Class 3: Computational Vision

Master in Artificial Intelligence

2021-2022

Universitat de Barcelona

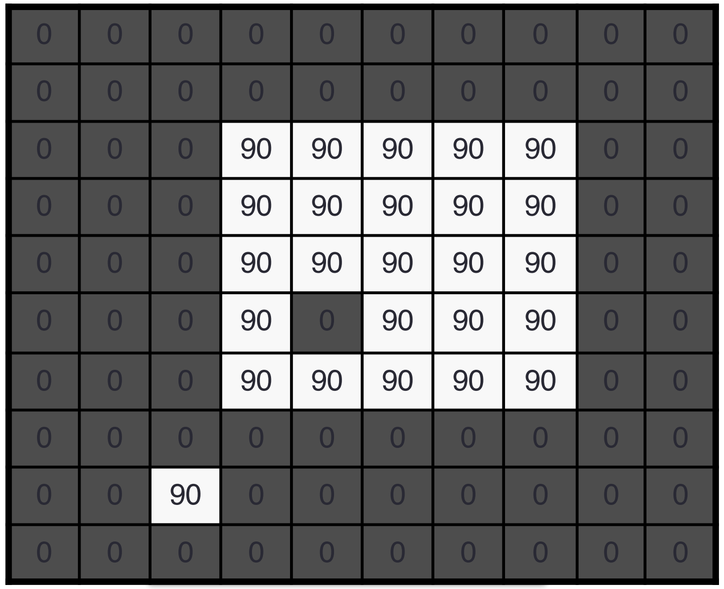
Recall: image filtering

- 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 (edges, etc) – to see later
  - Detect patterns (template matching) – to see later

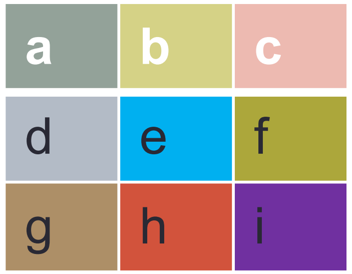
Adapted from Derek Hoiem

Linear filter

$F[x, y]$ $\otimes$ $H[u, v]$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0



a	b	c
d	e	f
g	h	i

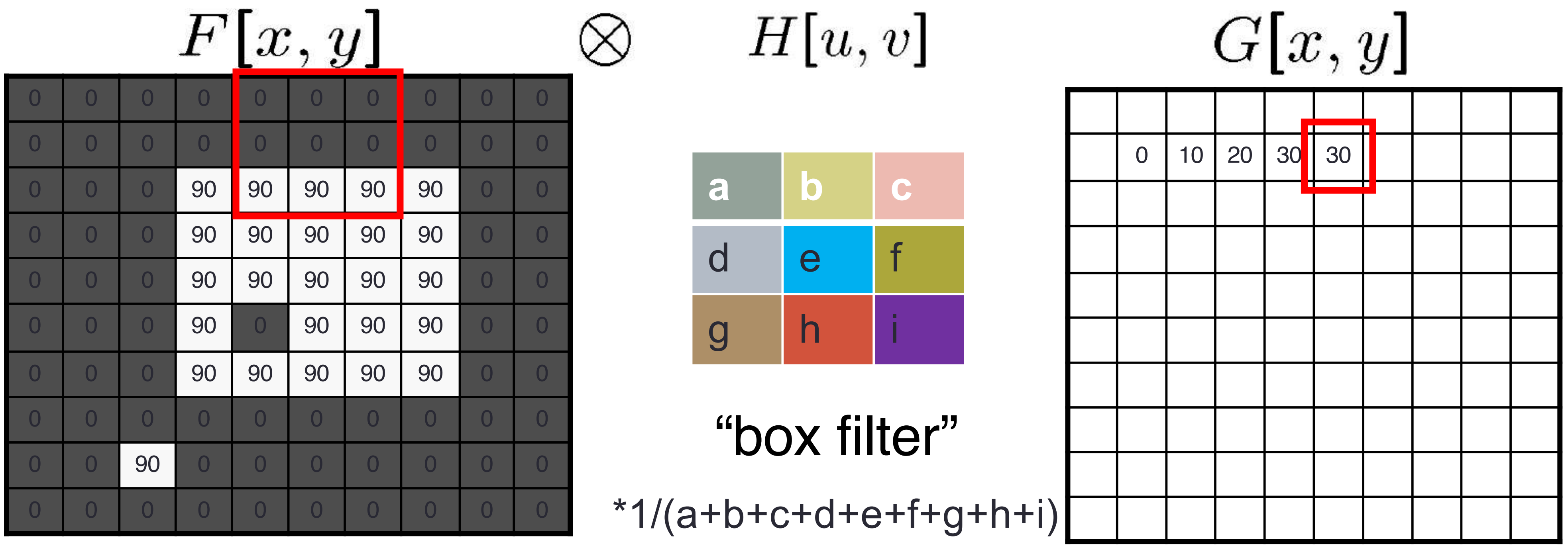
"box filter"

$$G = H \otimes F$$

$$\begin{aligned} df[i,j] = {} & a * f[i-1 j-1] + b * f[i-1, j] + c * f[i-1, j+1] + \\ & d * f[i, j-1] + e * f[i, j] + f * f[i, j+1] + \\ & g * f[i+1, j-1] + h * f[i+1, j] + i * f[i+1, j+1] \end{aligned}$$

Linear filter

- Normalization: divide the mask by (a+b+c+d+e+f+g+h+i)



$F[x,y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$\otimes$

$H[u,v]$

a	b	c
d	e	f
g	h	i

“box filter”

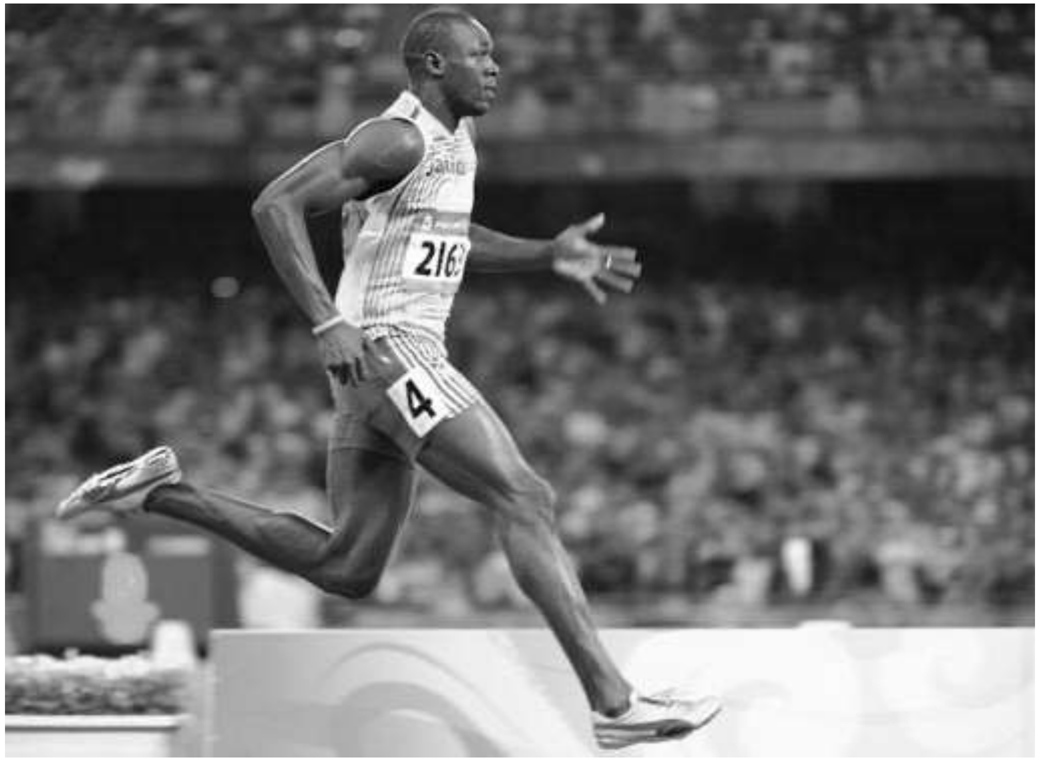
*1/(a+b+c+d+e+f+g+h+i)

$G[x,y]$

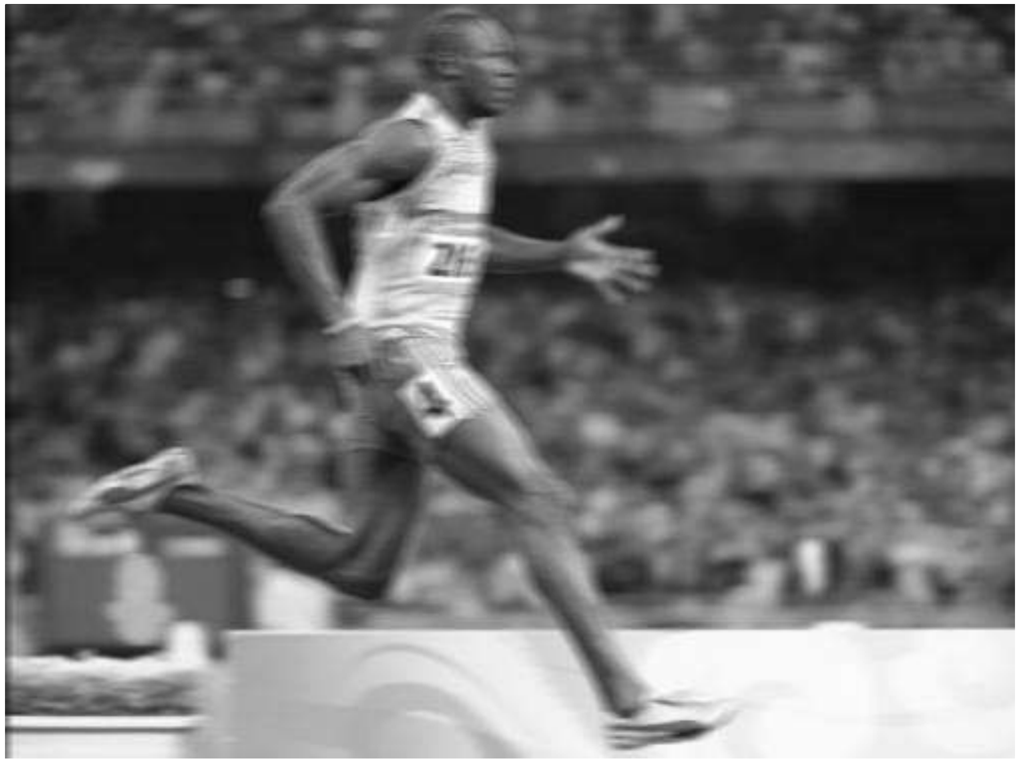
	0	10	20	30	30				

$$G = H \otimes F$$

Review



original image



filtered

Filter f = $1/9 \times [\,1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\,]^T$
o f = $1/9 \times [\,1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\,]$?!

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

Today

- What is an image derivative and its relation to the edges
  - How to implement different edge detectors and what is their difference?
  - First and second derivatives, image gradient and Laplacian
- How to get image derivatives with a Gaussian function
- Canny edge detector
- Application: hybrid images

Edge detection

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

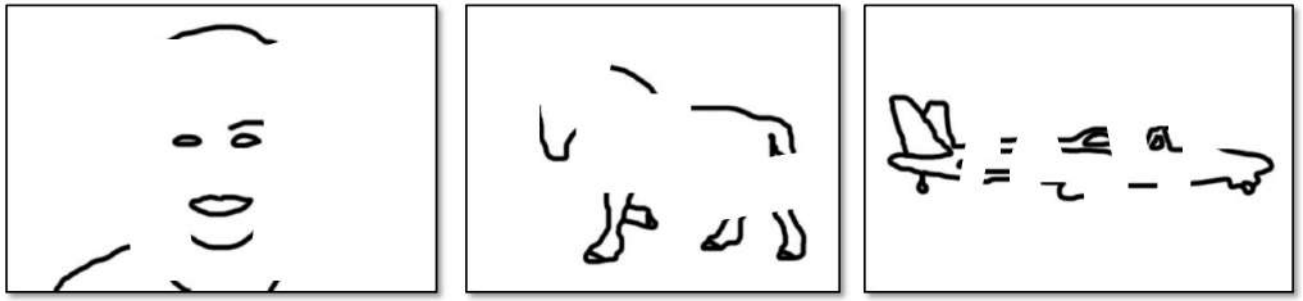
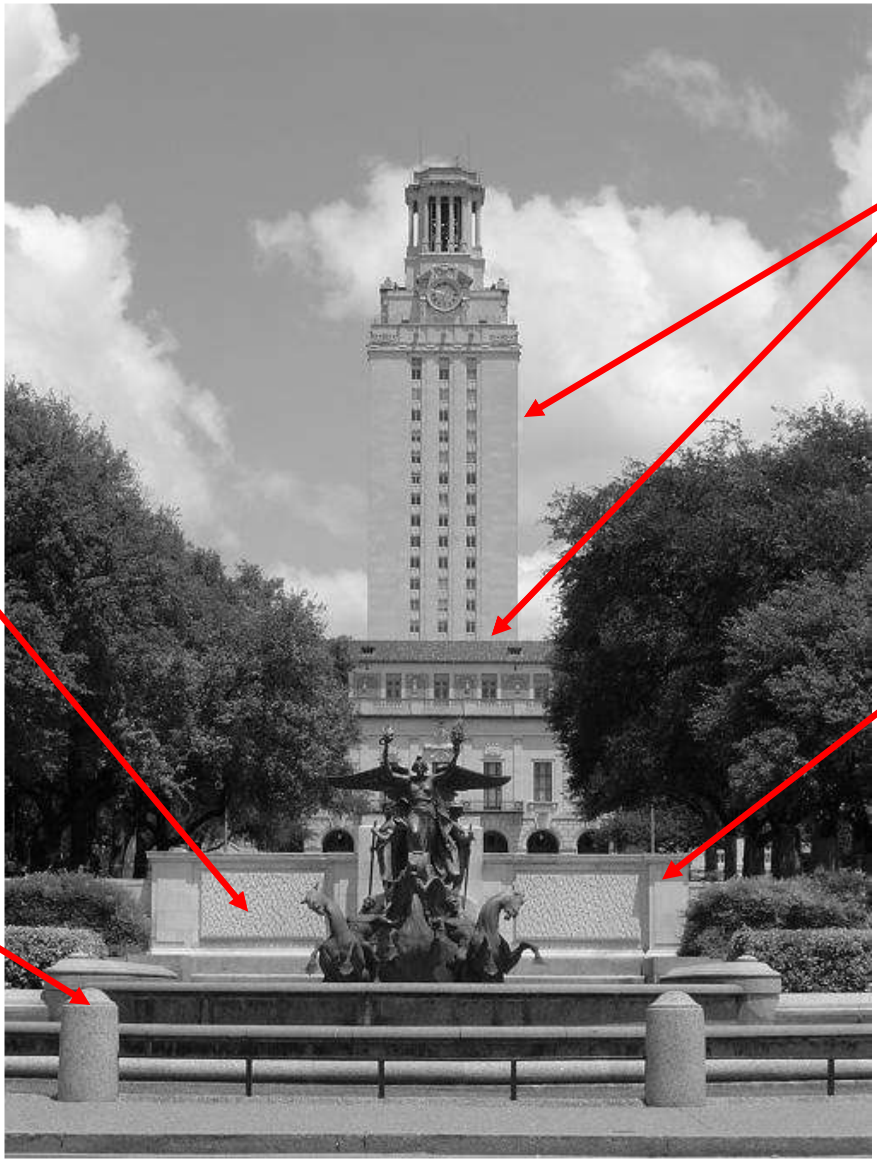


Figure from J. Shotton et al., PAMI 2007

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

What does cause an edge?



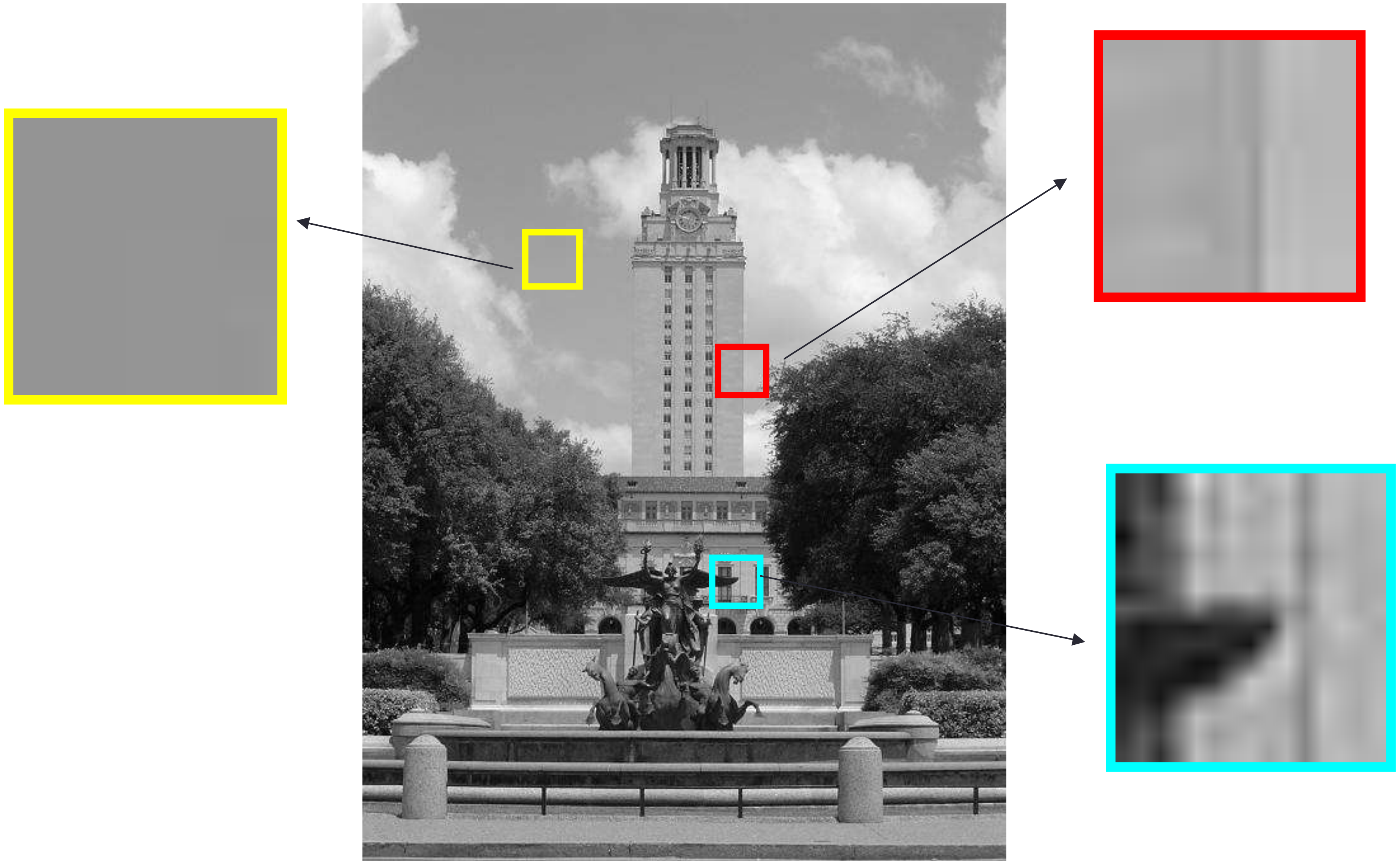
Reflectance change: appearance information, texture

Depth discontinuity: object boundary

Cast shadows

Change in surface orientation: shape

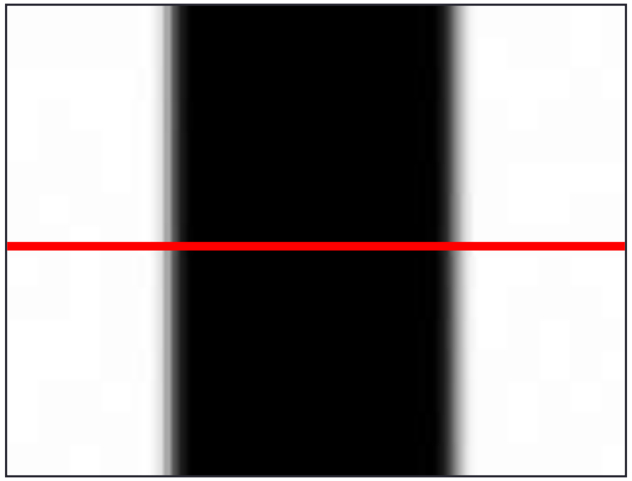
Edges/gradients and invariance



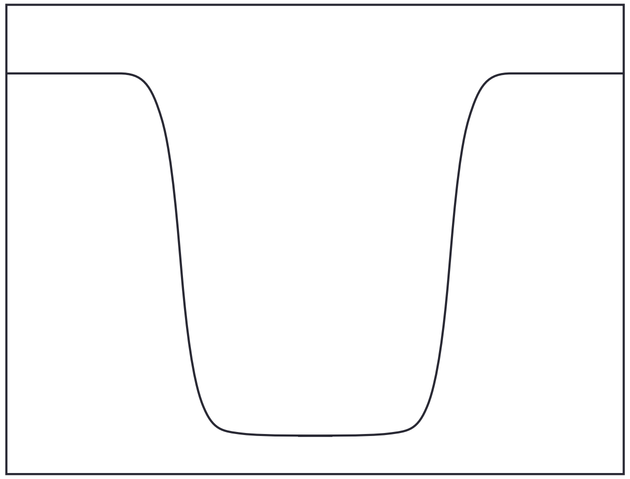
Derivatives and edges

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

image



intensity function (along horizontal scanline)



How to detect the points (pixels) of maximal change?

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

Source: L. Lazebnik

Derivatives and edges

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

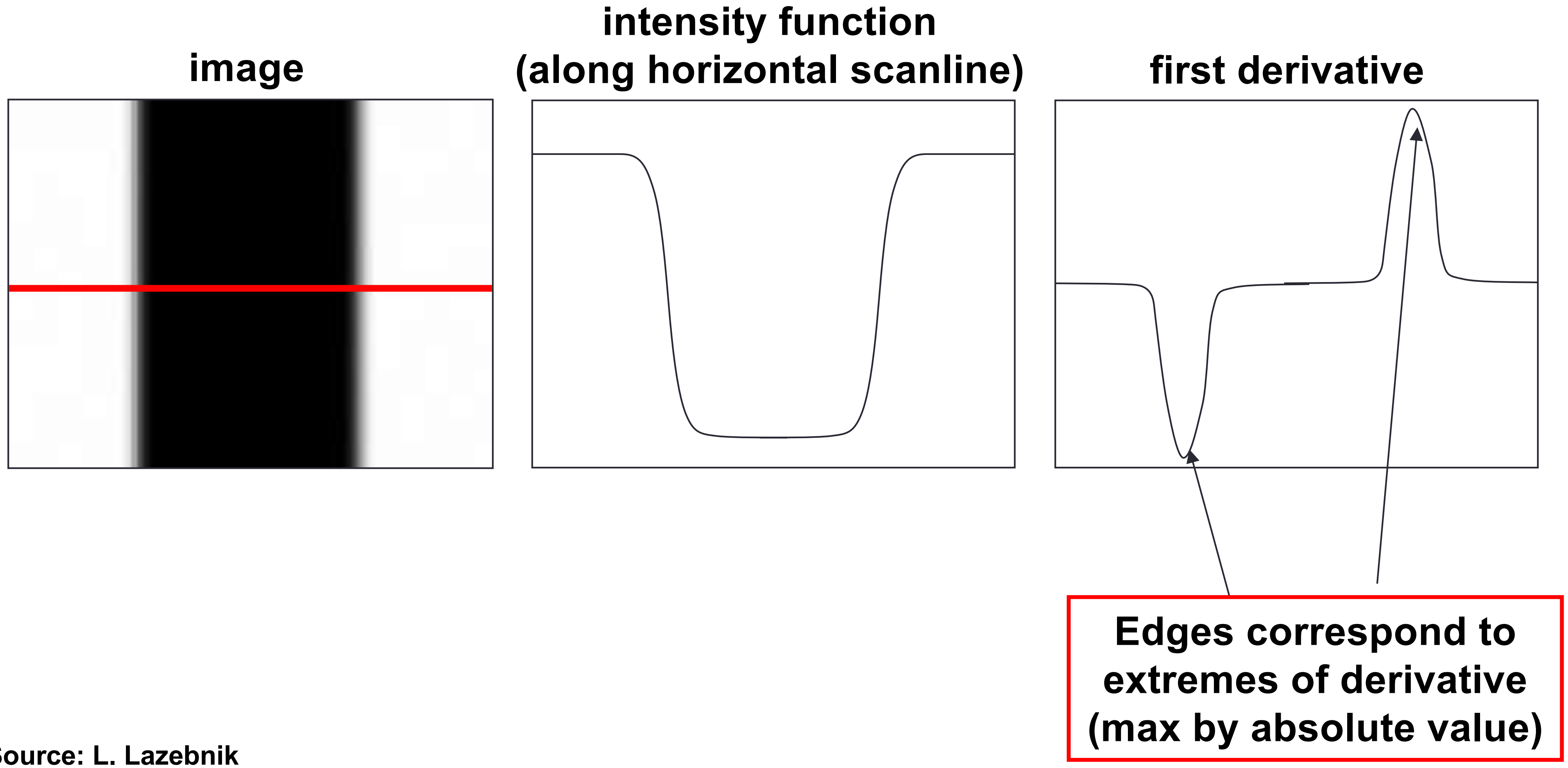
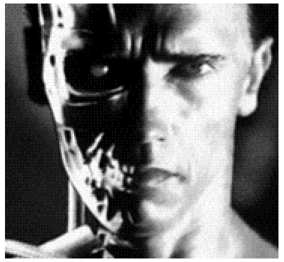


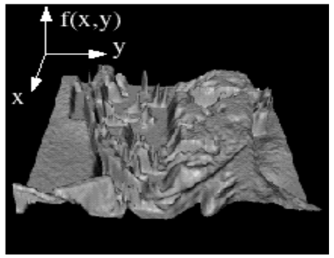
image
intensity function
(along horizontal scanline)
first derivative
Edges correspond to extremes of derivative (max by absolute value)

Source: L. Lazebnik

Derivatives with convolution

But… the image is 2D!





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

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

Does it make sense to get the derivative wrt 3rd dimension / channel?

Derivatives with convolution

But… the image is a discrete matrix!

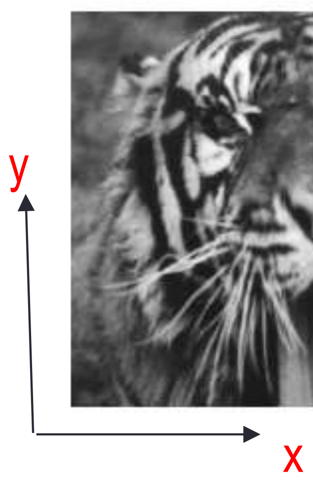
$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

We can approximate it using finite differences:

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

How to compute it on the image?

Partial derivatives of an image



What would be the associated convolution filter?

a	b	c
d	e	f
g	h	i

0	b	0
0	-1	1
0	0	0

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

Recall:

$$df[x,y] = a * f[x-1,y+1] + b * f[x,y+1] + c * f[x+1,y+1] +$$
$$d * f[x-1,y] + e * f[x,y] + f * f[x+1,y] +$$
$$g * f[x-1,y-1] + h * f[x,y-1] + i * f[x+1,y-1]$$

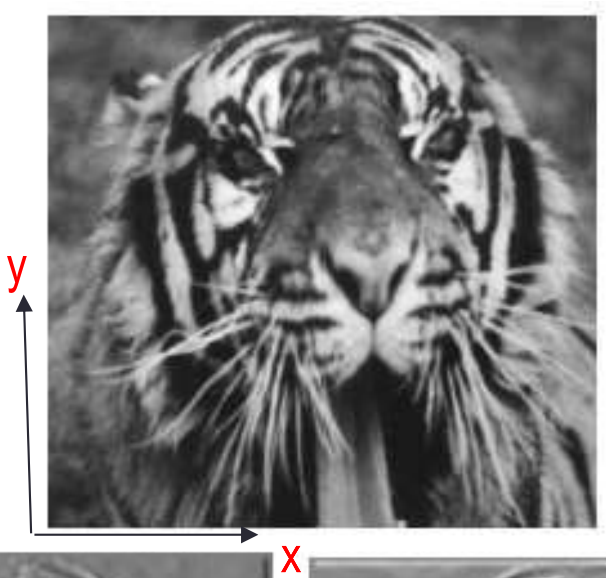
Partial derivatives of an image

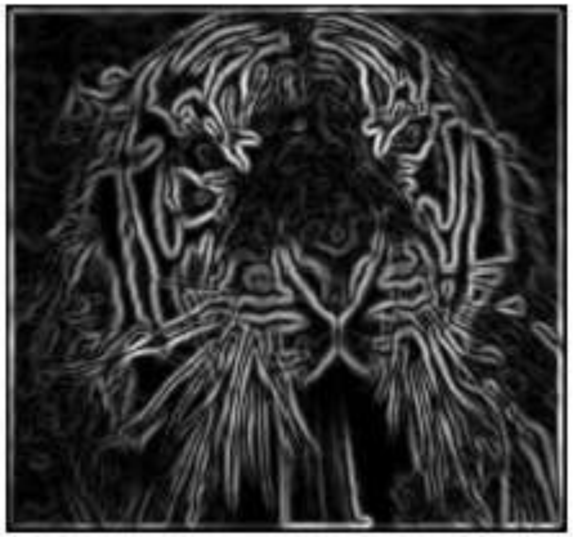
a	b	c
d	e	f
g	h	i

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

-1	1

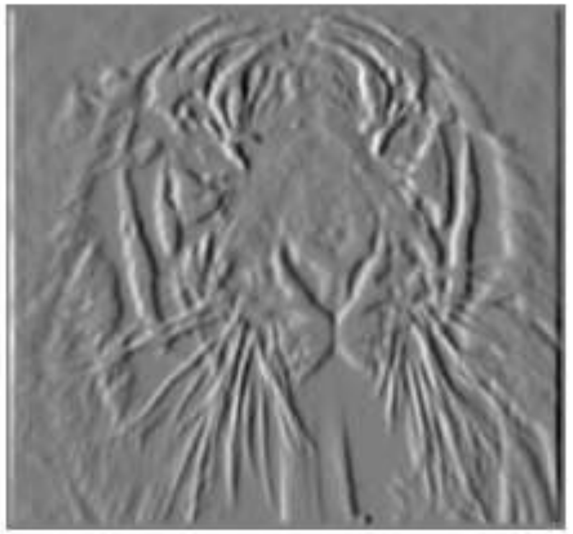
Partial derivatives of an image

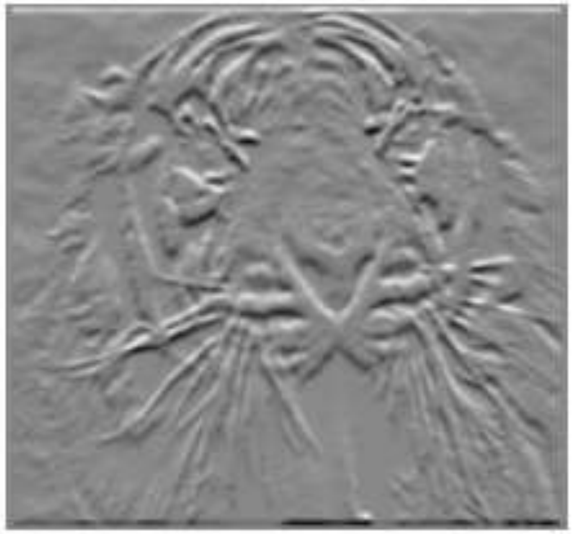




$$\frac{\partial f(x,y)}{\partial x}$$

-1	1





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

-1
1

? or

1
-1

Which derivative does show changes with respect to x?

First derivatives: discrete operators.

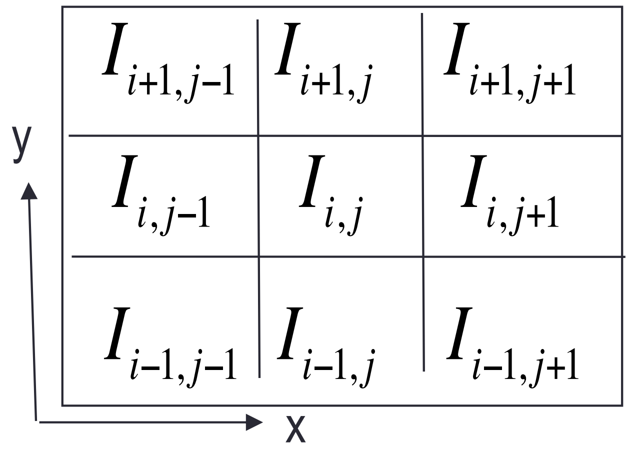
Given that the function f to derive our image I is:

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

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

$$\longrightarrow \quad \boxed{\begin{array}{c} \mathbf{1} \\ \hline \mathbf{-1} \end{array}} \qquad \boxed{\begin{array}{c|c} \mathbf{-1} & \mathbf{1} \end{array}}$$

Masks



Exercise: Which derivative will give maximal values on the image?

$$\begin{array}{ccc} 0 & 10 & 10 \\ 0 & 10 & 10 \\ 0 & 10 & 10 \end{array}$$

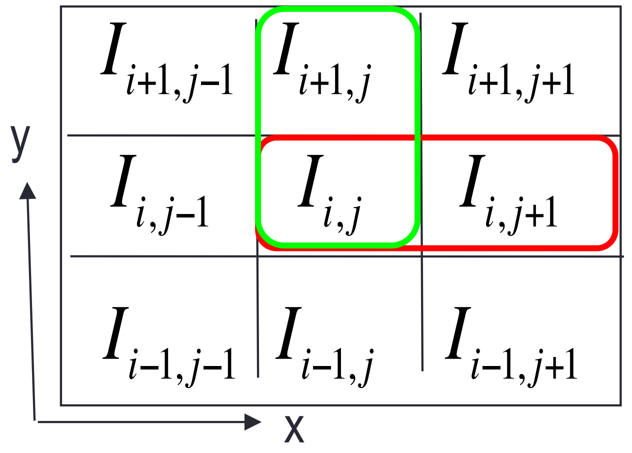
First derivatives: discrete operators.

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

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

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

$$\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 25 & 0 \\ 0 & 0 & 0 \end{array}$$

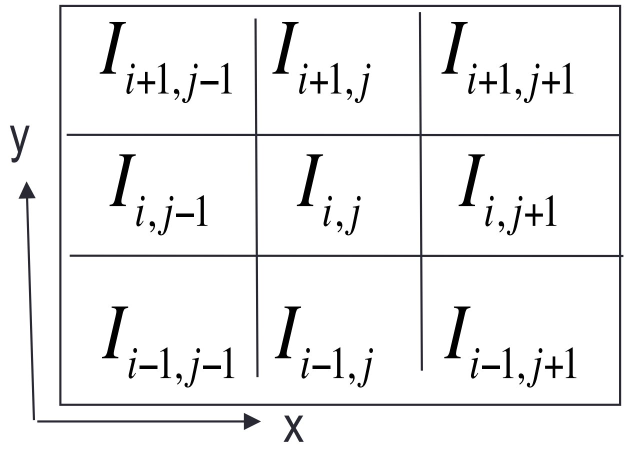


What happens when there is noise in the image?

First derivatives: discrete operators.

How to be less sensitive to noise?

$$\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 25 & 0 \\ 0 & 0 & 0 \end{array}$$



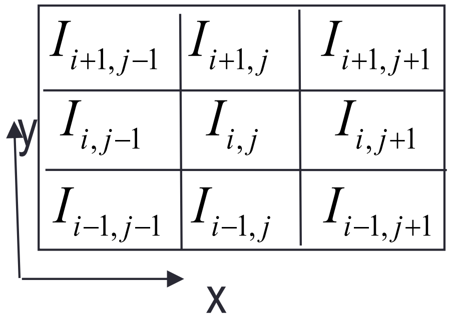
Alternative: getting the average of the 3 derivatives in order to be less sensitive to noise:

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

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

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

My=

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

Prewitt masks

Assorted finite difference filters

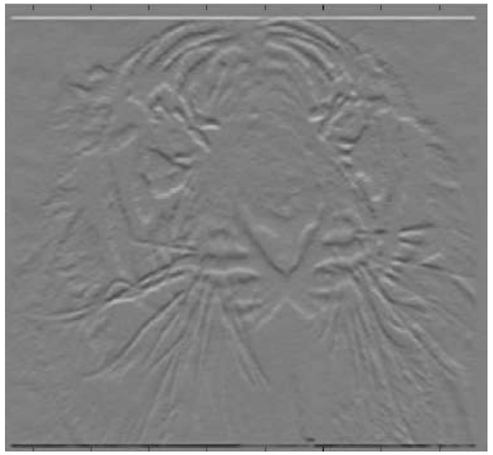
$$\textbf{Prewitt:}\quad M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \; ; \quad M_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

$$\textbf{Sobel:}\quad M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \; ; \quad \boxed{M_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}}$$

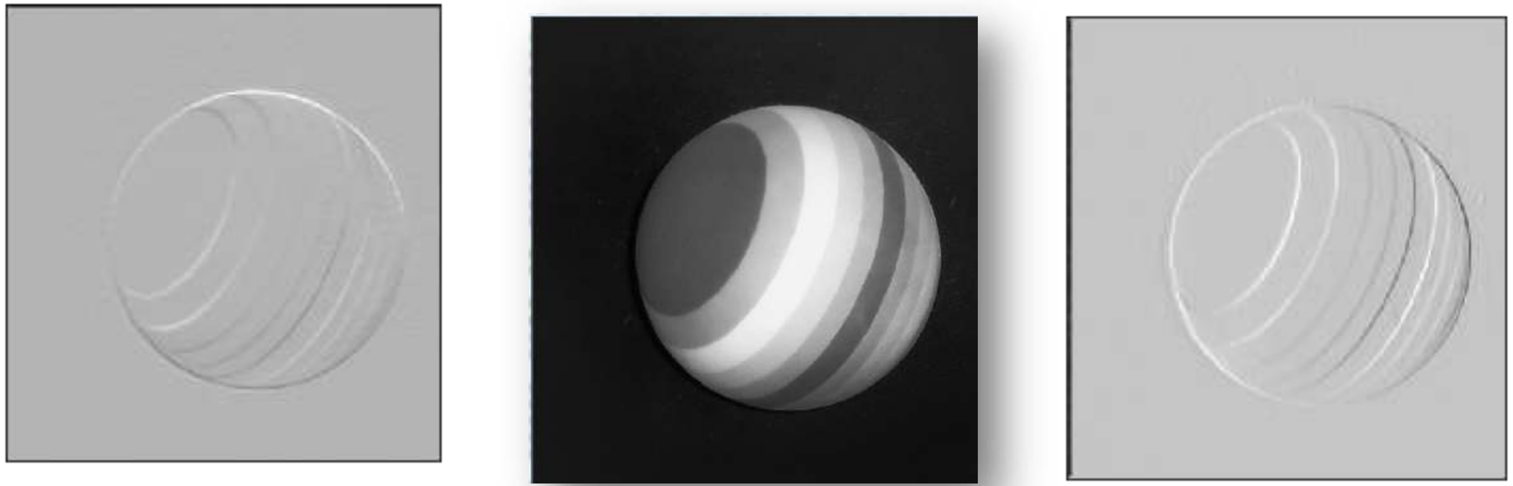
What is the difference between the Prewitt and Sobel operators?

- 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 skimage to detect the image edges?

```
Skimage:

>>from skimage import filters

>>im_vh=filters.sobel_v(im)
# Image should be float and gray value!
```

Assorted finite difference filters

```
Alternative:

>>from scipy.ndimage import
>>convolveim_sv=convolve(im,mask)
```

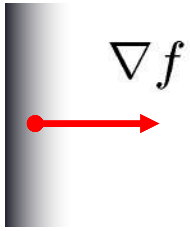
How to combine both derivatives (in x and y)?

```
>>im_s=filters.sobel(im)
```

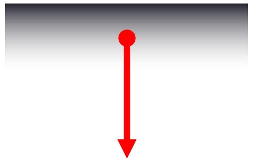
Image gradient

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

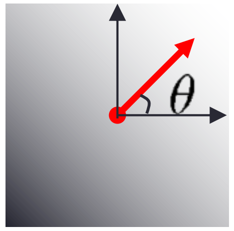
The gradient points in the direction of most rapid change in intensity



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



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



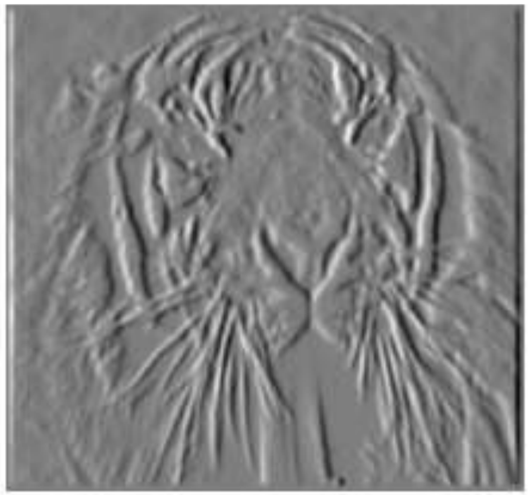
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

Slide credit Steve Seitz

Image gradient

The **edge strength** is given by the gradient magnitude

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



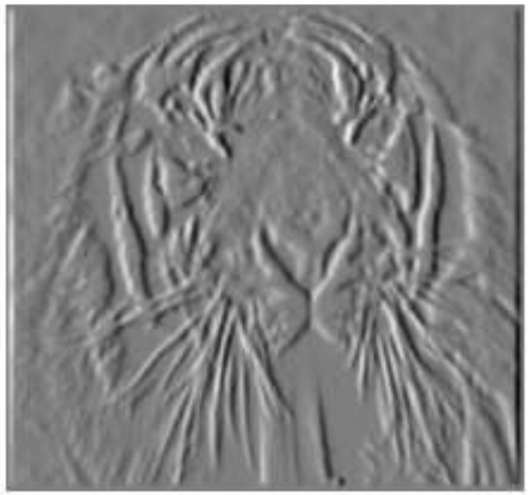
Slide credit Steve Seitz

Image gradient

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

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

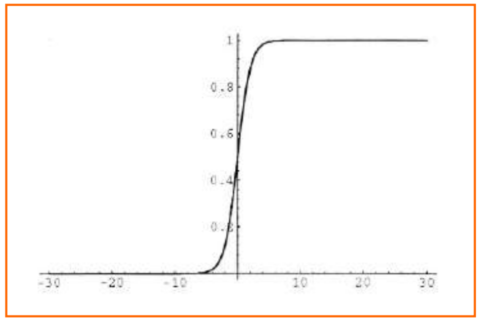
Which aspect (magnitude or direction) is invariant to image contrast?

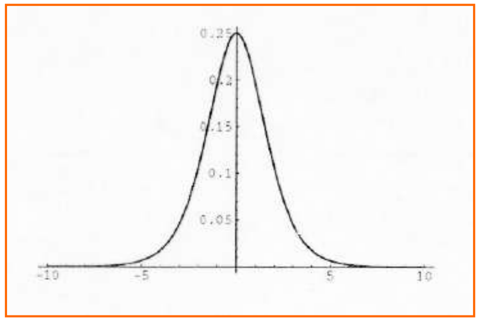


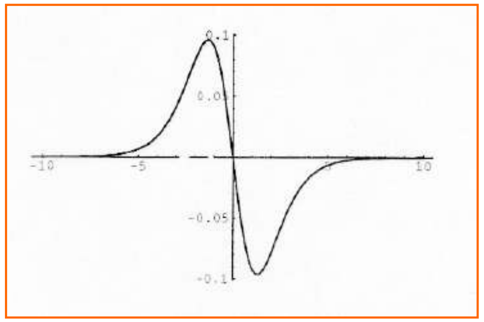
Slide credit Steve Seitz

Discrete operators: second derivatives

What do you expect about the second derivatives?





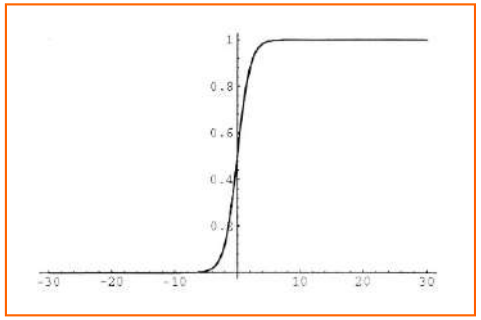


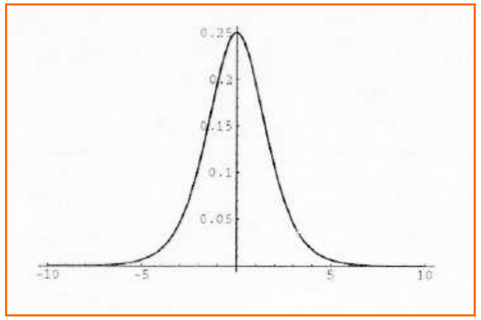
$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

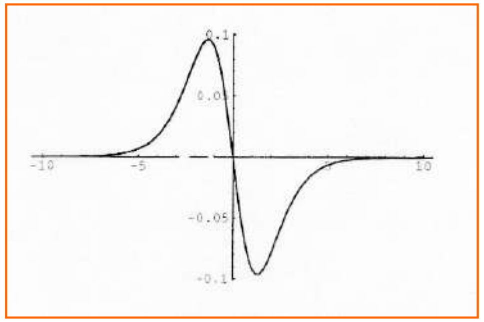
Given an edge, the first derivative has an extreme, and the second one has a zero-crossing in the edge and extremes before and after the edge.

Discrete operators: second derivatives

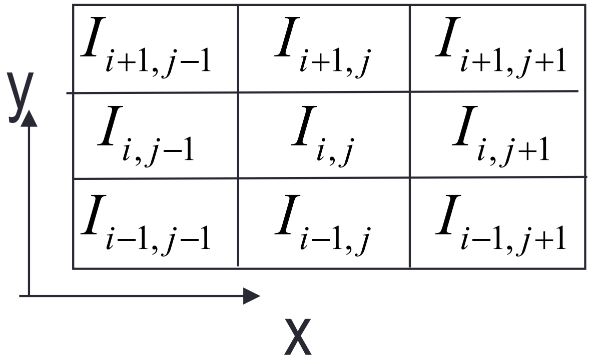
What do you expect about the second derivatives?







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.



$I_{i+1,j-1}$	$I_{i+1,j}$	$I_{i+1,j+1}$
$I_{i,j-1}$	$I_{i,j}$	$I_{i,j+1}$
$I_{i-1,j-1}$	$I_{i-1,j}$	$I_{i-1,j+1}$

The second derivative is the derivative of the first derivative:

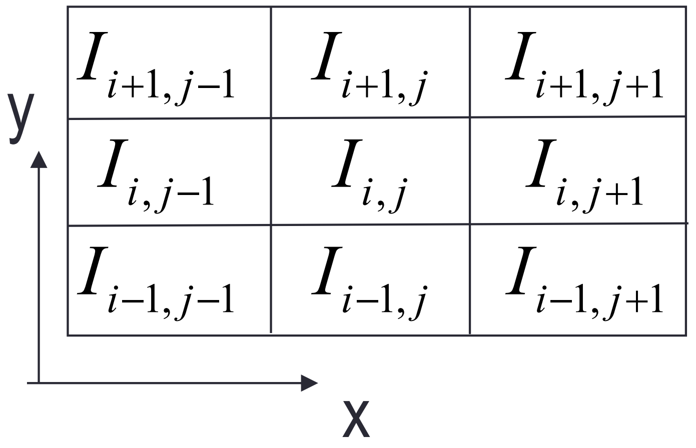
$$\frac{\partial^2 I}{\partial y^2} = (I_{i+1,j} - I_{i,j}) - (I_{i,j} - I_{i-1,j}) = (I_{i-1,j} - 2I_{i,j} + I_{i+1,j})$$

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

Discrete operators: Laplacian

Let’s define:

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



$I_{i+1,j-1}$	$I_{i+1,j}$	$I_{i+1,j+1}$
$I_{i,j-1}$	$I_{i,j}$	$I_{i,j+1}$
$I_{i-1,j-1}$	$I_{i-1,j}$	$I_{i-1,j+1}$

a	b	c
d	e	f
g	h	i

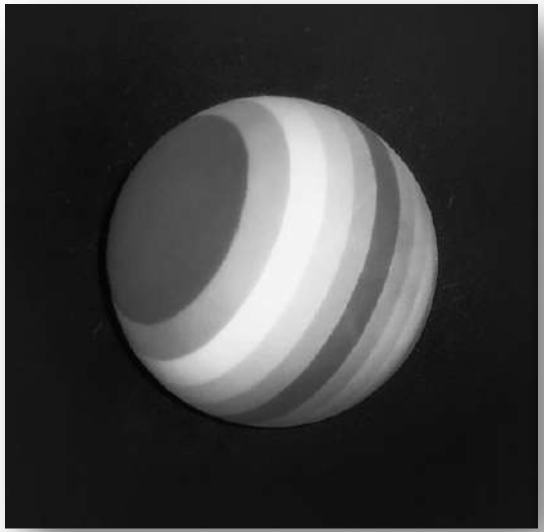
Laplacian mask:

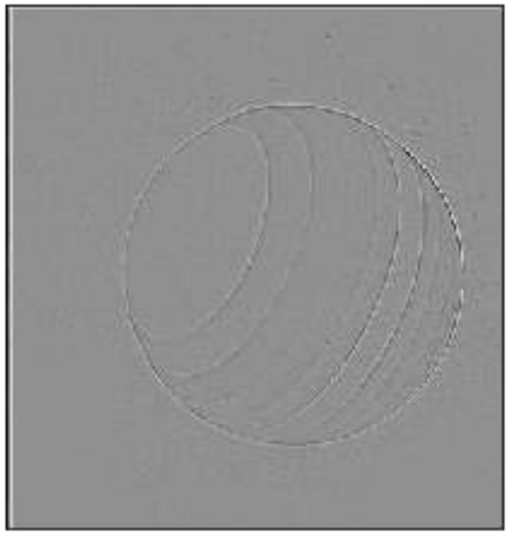
$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Discrete operators: Laplacian

Laplacian mask:

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





Skimage:

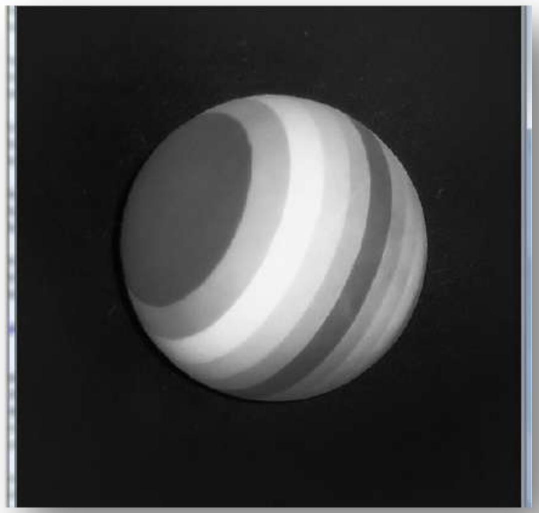
```
>> mask=numpy.array([[0,1,0],[1,-4,1],[0,1,0]], dtype='float')

>> im_l1= scipy.ndimage.convolve(skimage.img_as_float(im),mask)
```

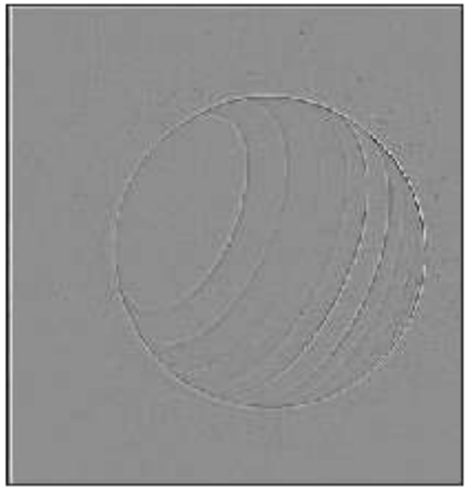
Discrete operators: Laplacian

Including the diagonal neighbours:

Laplacian:

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



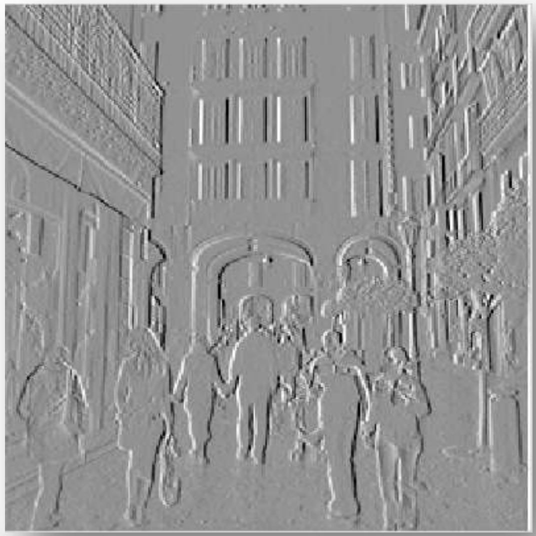


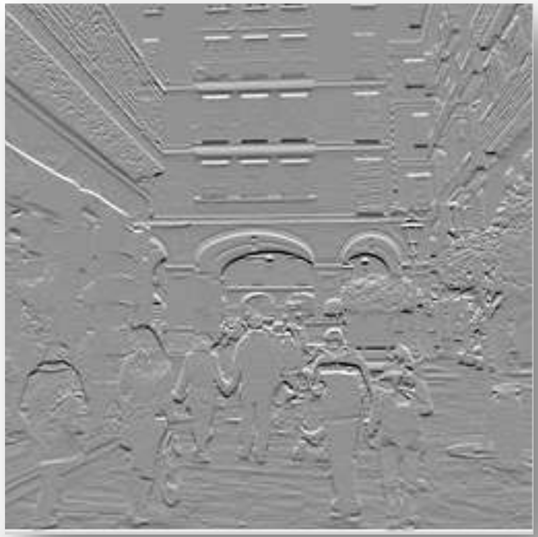
Mask properties

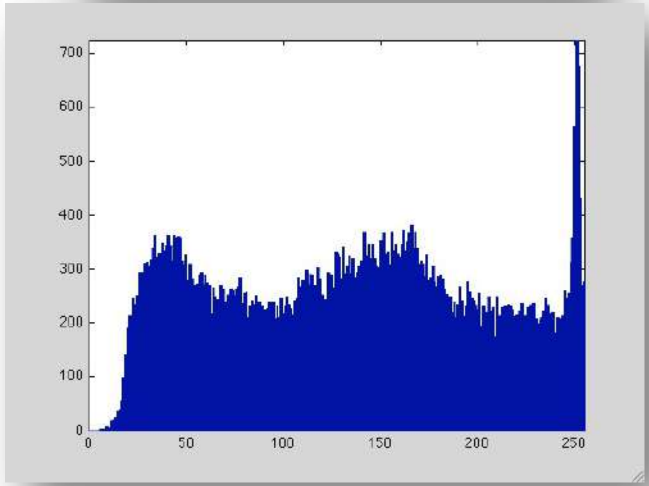
- Test on smoothing
  - Values should be __________
  - Sum to ___→ constant regions same as input
  - Amount of smoothing _______________ to mask size
  - Remove "_______-frequency" components; "_______-pass" filter

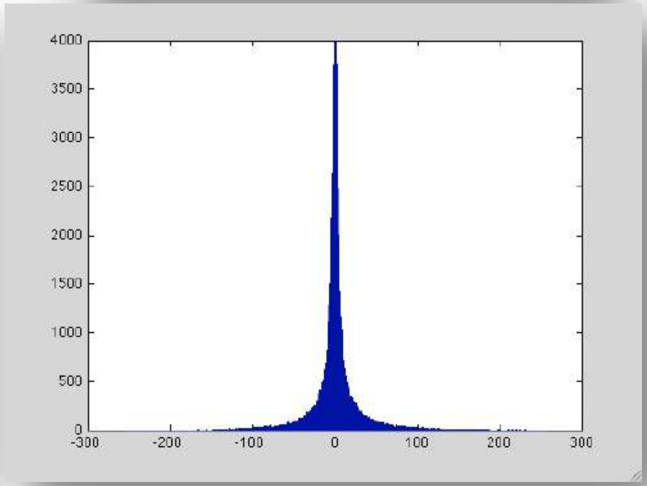
- Test on derivatives
  - _____________ signs used to get high response in regions of high contrast
  - Sum to ___ → no response in constant regions
  - _________________ value at points of high contrast

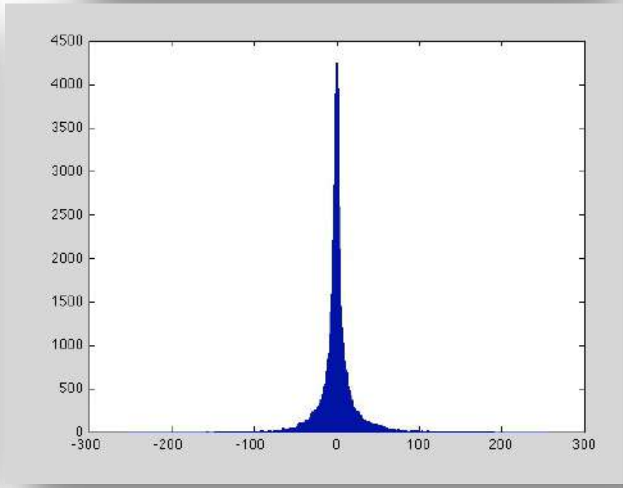
Explain the histogram of x and y edge maps











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?.

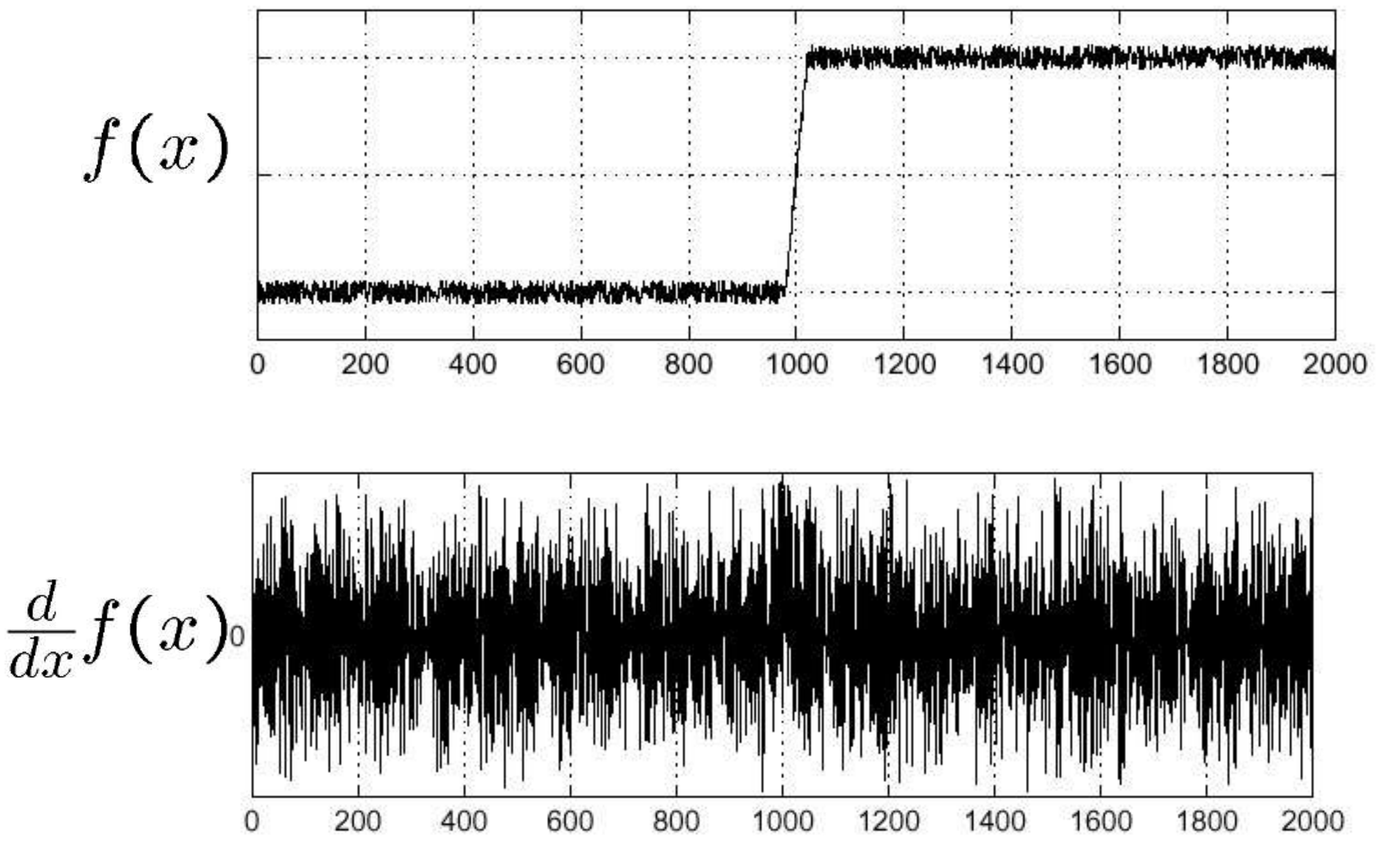
Today

- What is an image derivative and its relation to the edges
  - How to implement different edge detectors and what is their difference?
  - First and second derivatives, image gradient and Laplacian
- How to get image derivatives with a Gaussian
- Canny edge detector
- Application: hybrid images

Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal



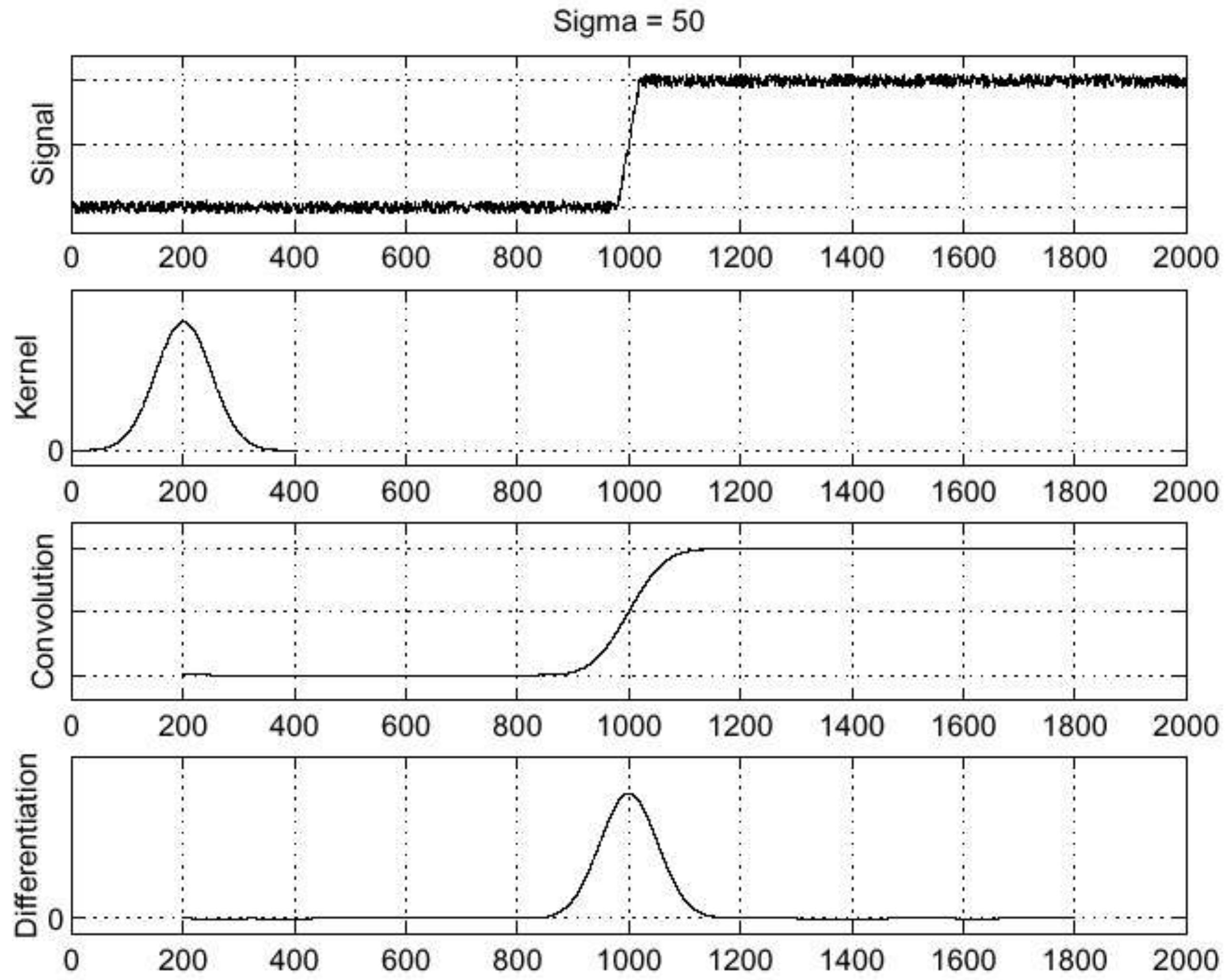
Where is the edge?

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

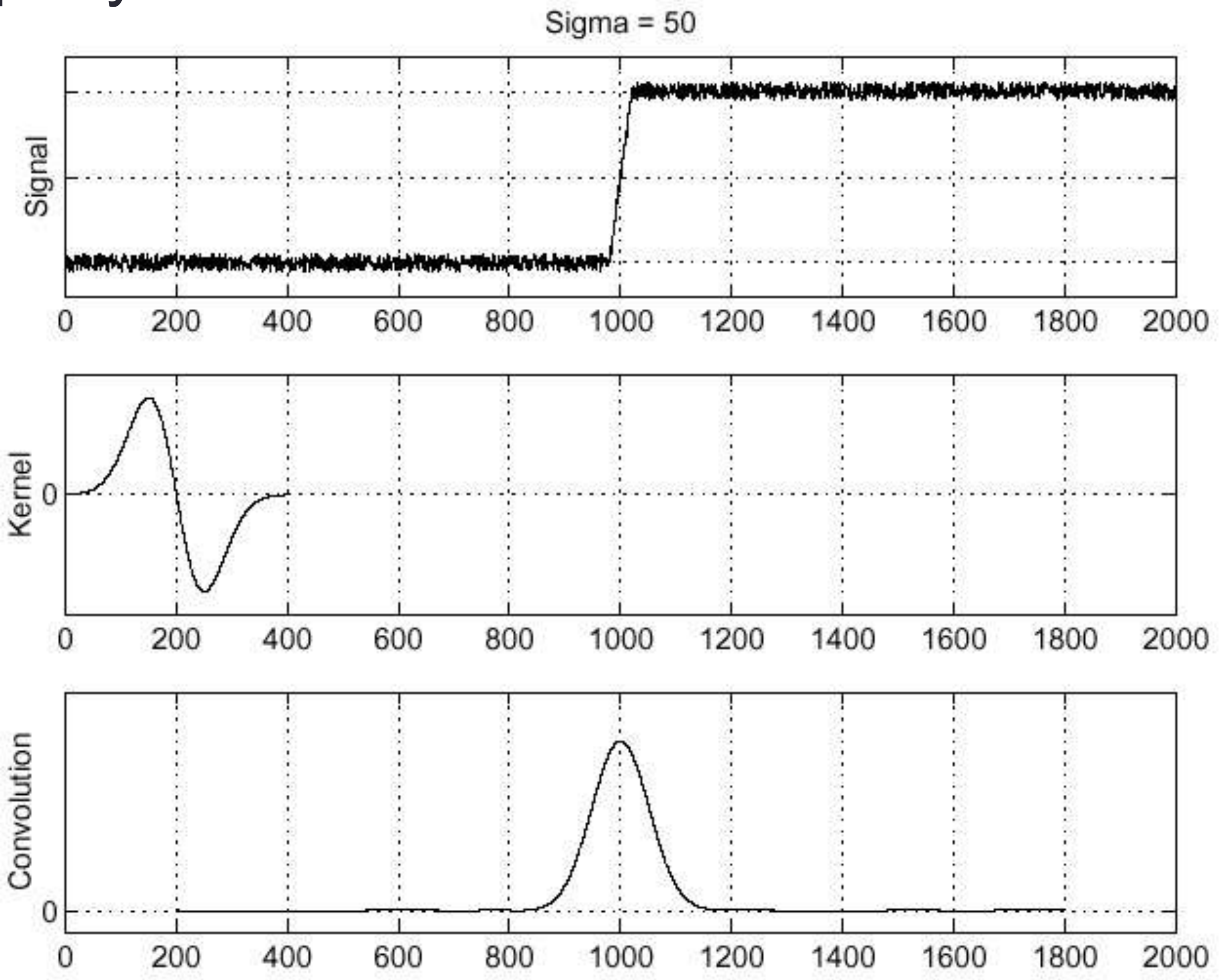
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

Differentiation property of convolution.

f

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

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

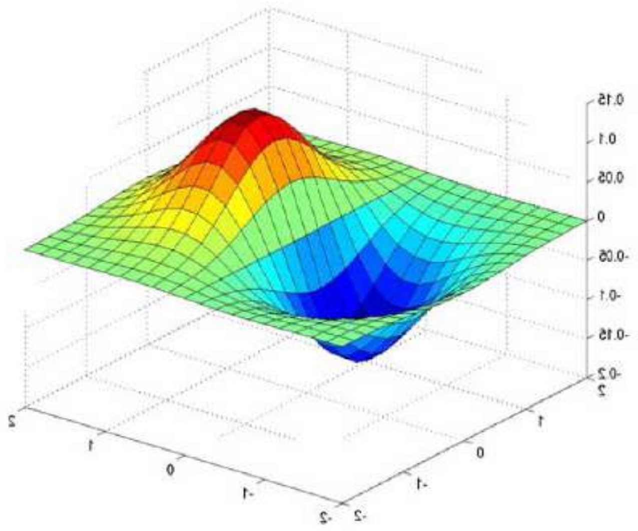


Slide credit Steve Seitz

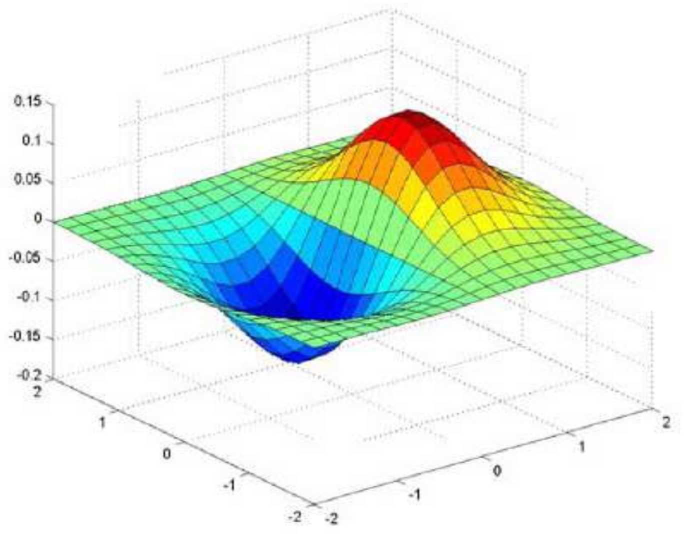
Derivative of Gaussian filters

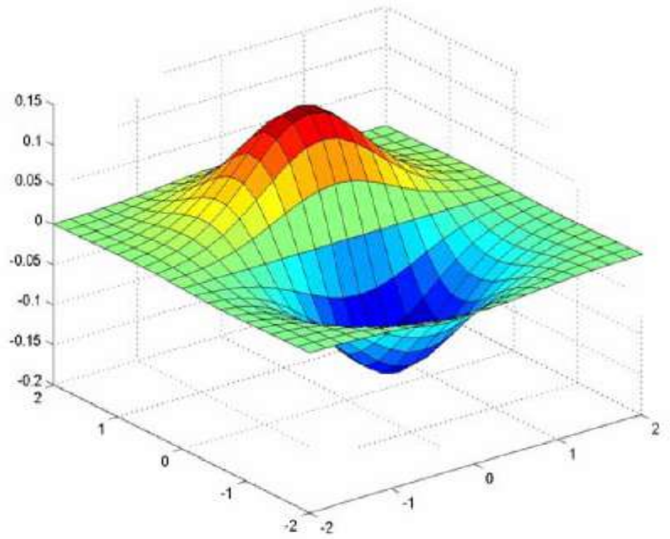
$$(I \otimes g) \otimes h = I \otimes (g \otimes h)$$

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

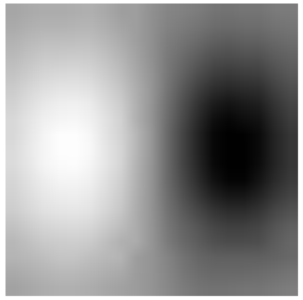


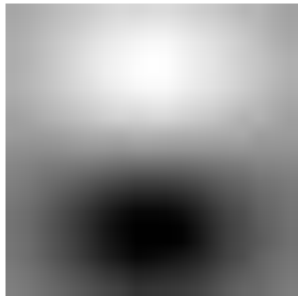
Derivative of Gaussian filters





***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 Skimage

Gaussian

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

Derivative of Gaussian

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

In skimage:

```
>>im_gaus_der11=scipy.ndimage.filters.gaussian_filter(im, 1, order=[1,1])

>>im_gaus_der10=scipy.ndimage.filters.gaussian_filter(im, 1, order=[1,0])

>>im_gaus_der01=scipy.ndimage.filters.gaussian_filter(im, 1, order=[0,1])
```

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!

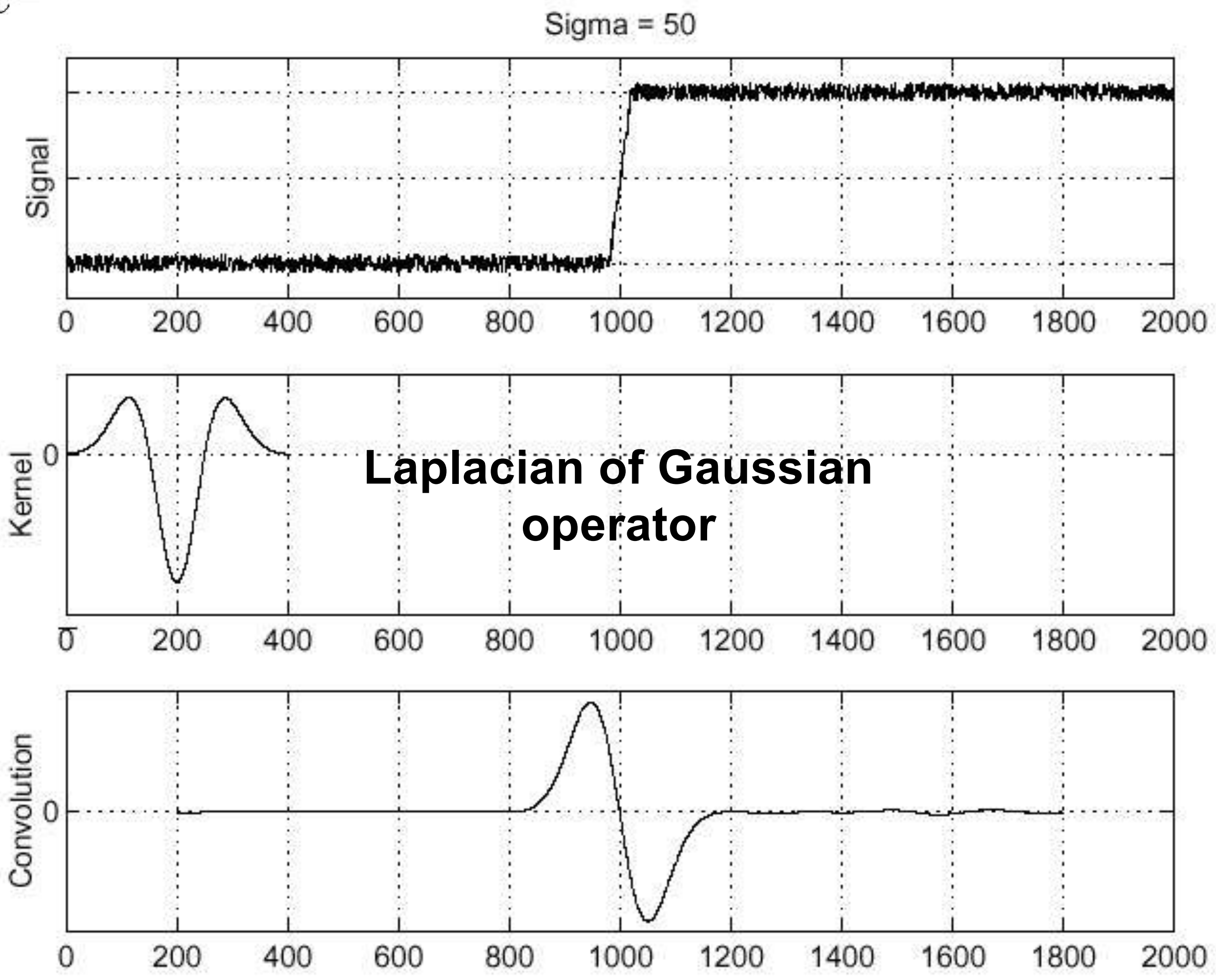
Laplacian of Gaussian

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

f

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

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

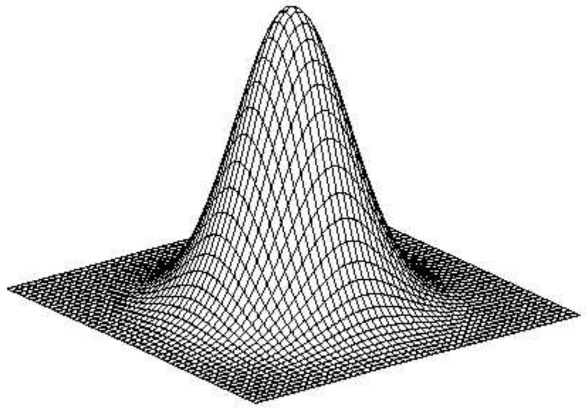


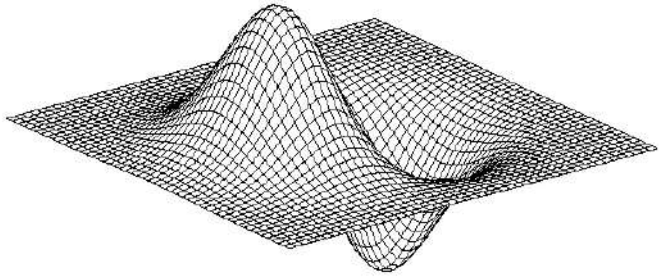
Where is the edge?

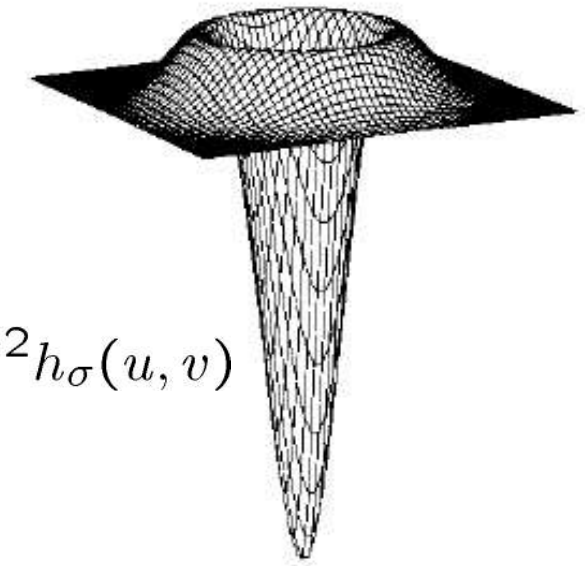
Zero-crossings of bottom graph

Slide credit: Steve Seitz

2D edge detection filters







Laplacian of Gaussian

Gaussian

Derivative of Gaussian

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

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

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

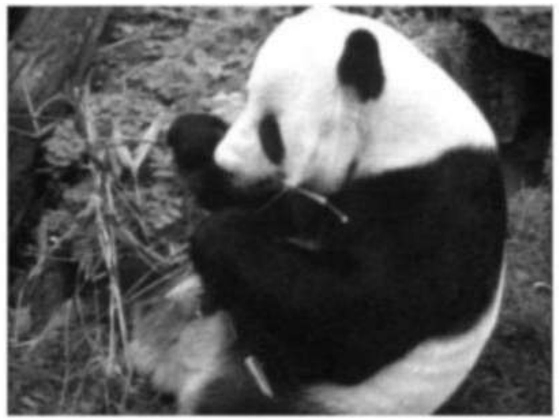
- ∇^2 is the Laplacian operator:

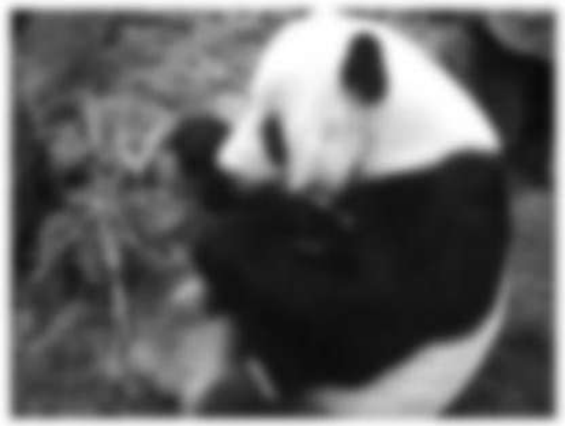
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Slide credit: Steve Seitz

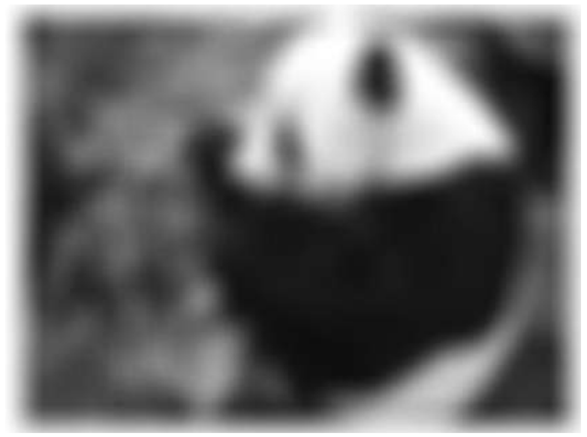
Smoothing with a Gaussian

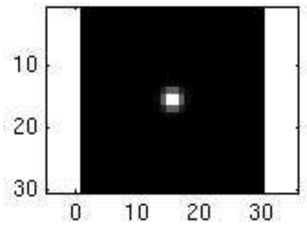
Recall: parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.

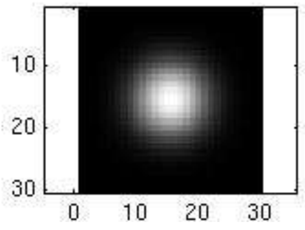


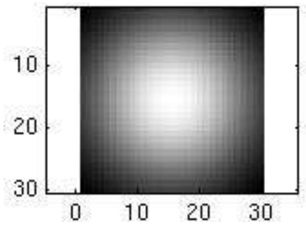


…

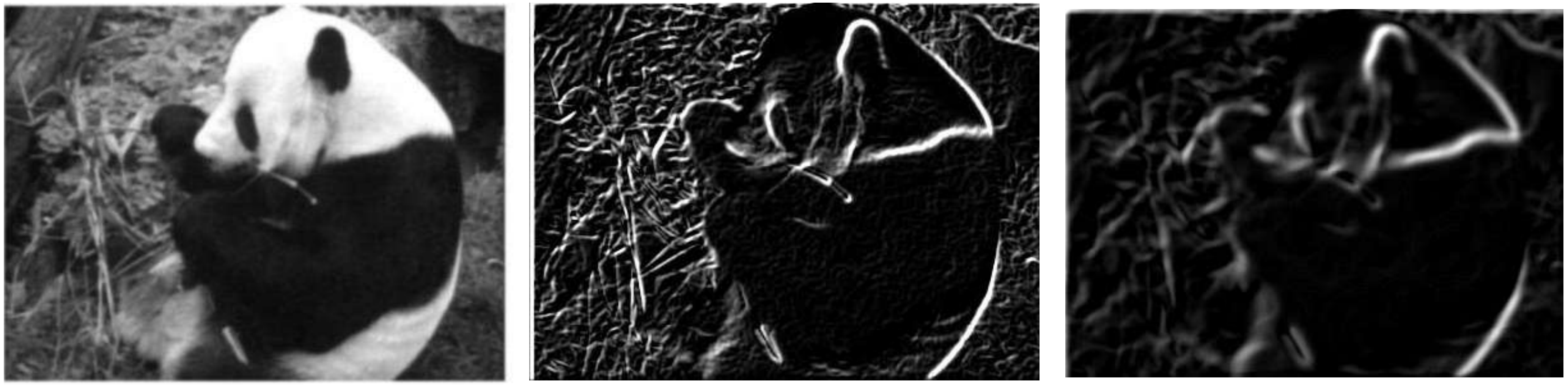








Effect of σ on derivatives

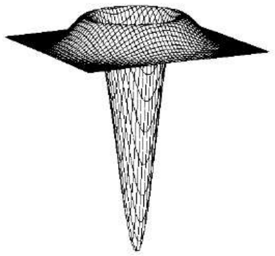
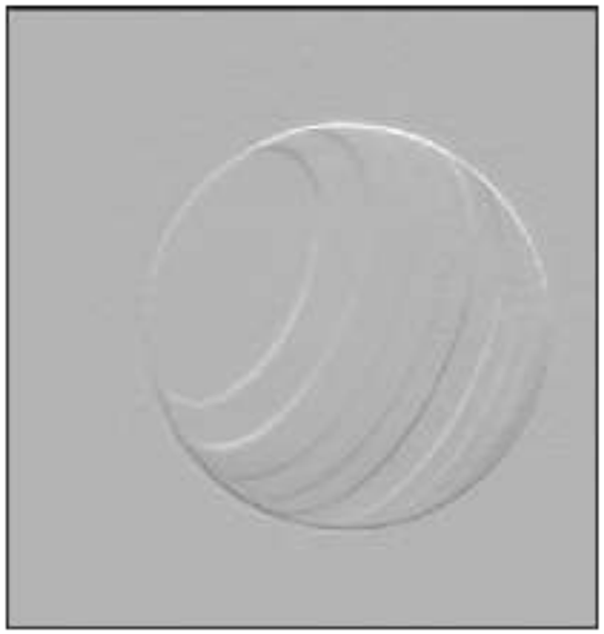


σ = 1 pixel σ = 3 pixels

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

Larger values: ……… scale edges detected.
Smaller values: ………features detected.

Laplacian

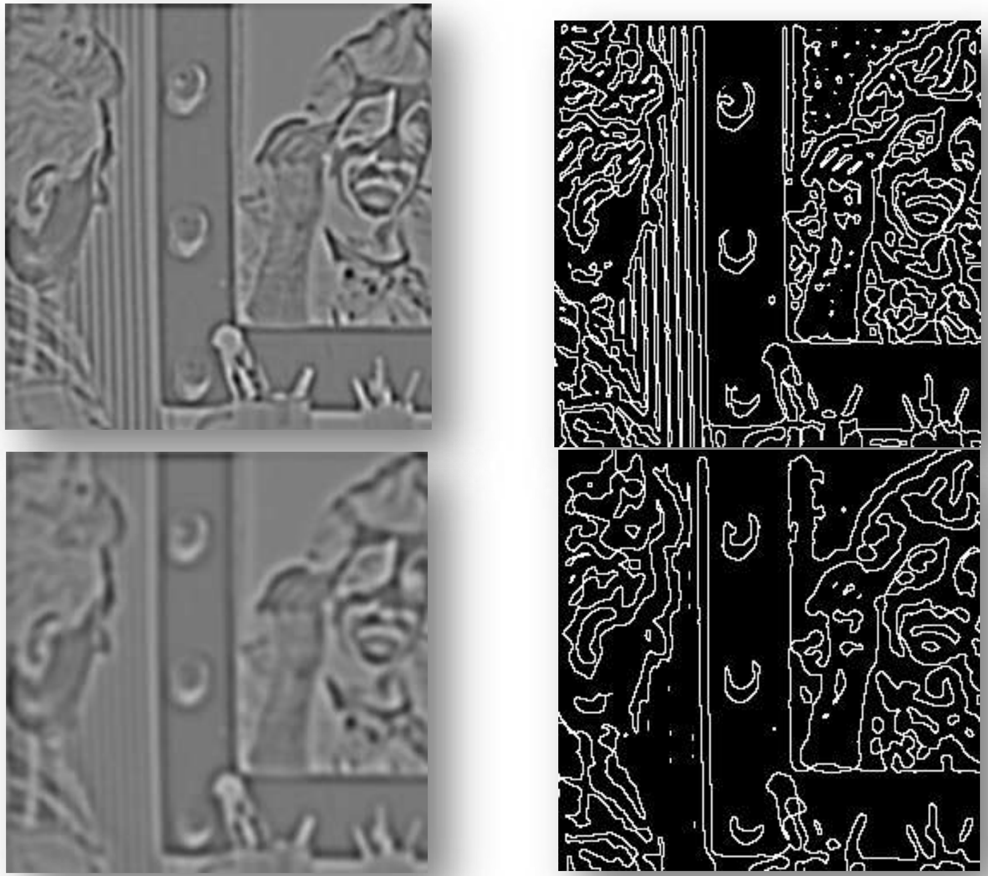



$$\nabla^2 h(x,y) = \left(\frac{x^2+y^2}{\sigma^4} - \frac{2}{\sigma^2}\right) h(x,y)$$

Skimage:

```
im_l=filters.laplace(im)
```

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).

Today

- What is an image derivative and its relation to the edges
  - How to implement different edge detectors and what is their difference?
  - First and second derivatives, image gradient and Laplacian
- How to get image derivatives with a Gaussian
- Canny edge detector
- Application: hybrid images

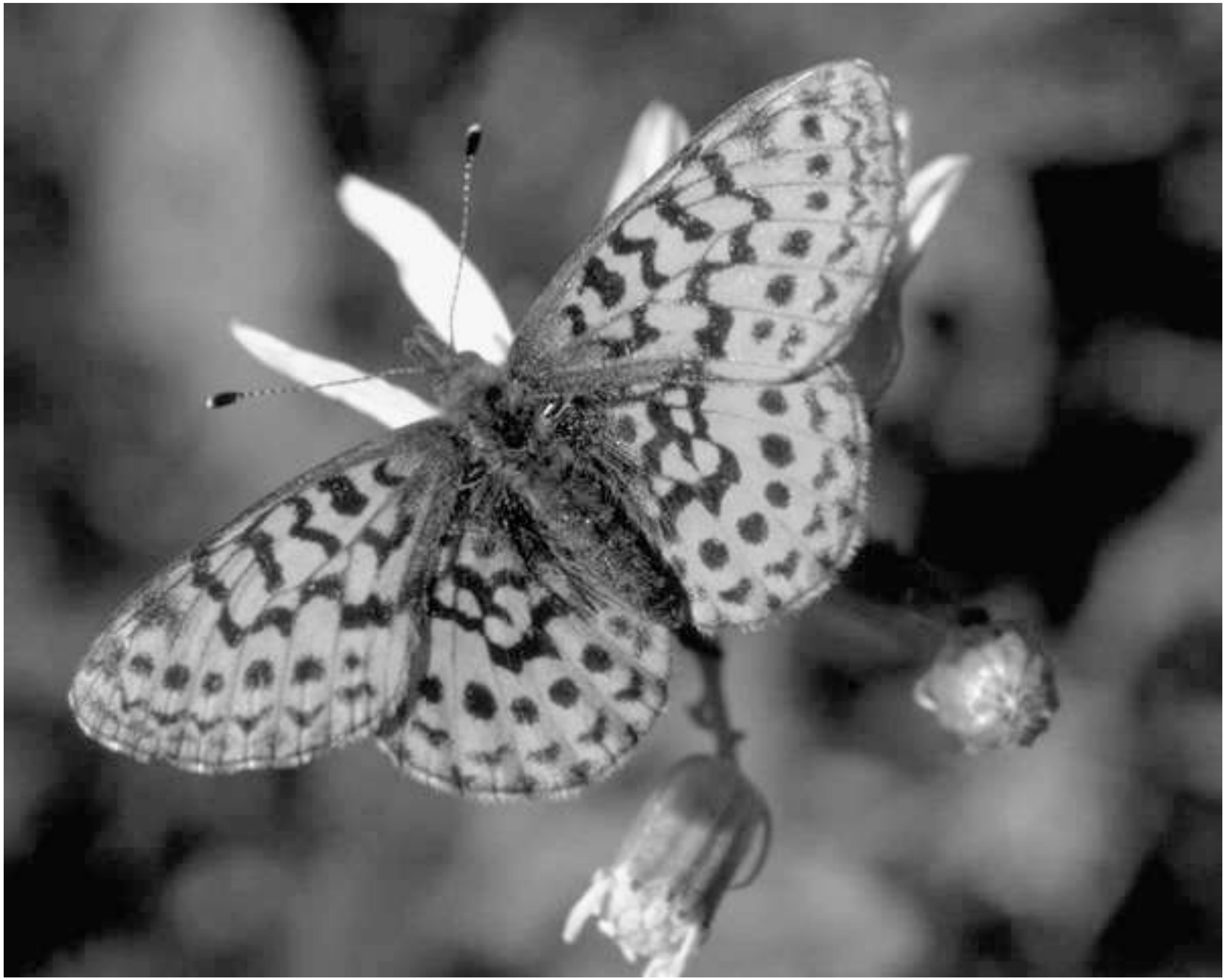
So, what scale to choose?

It depends what we're looking for.

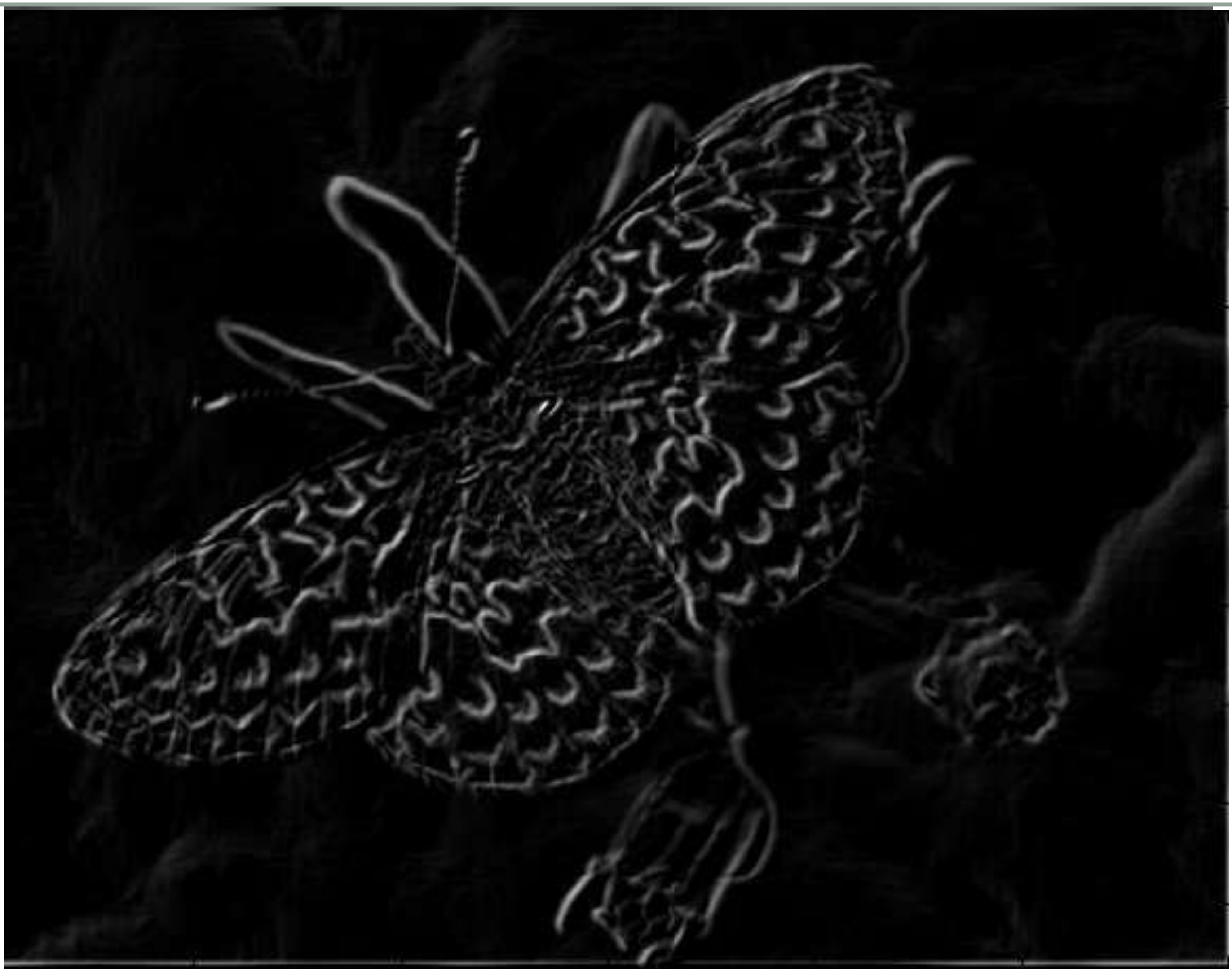
We need previous knowledge about the right scale.

Or manage different scales. How?

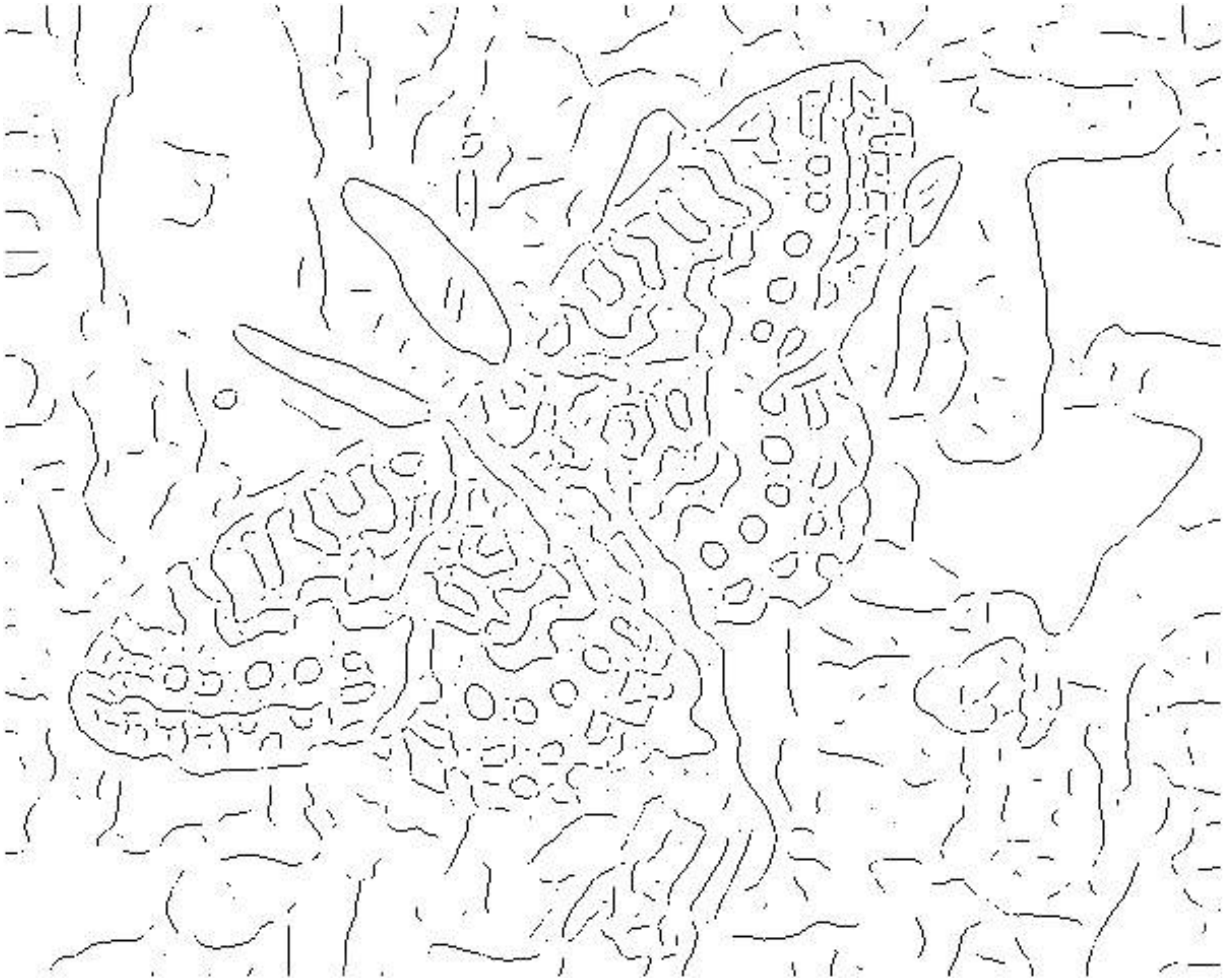
Original image



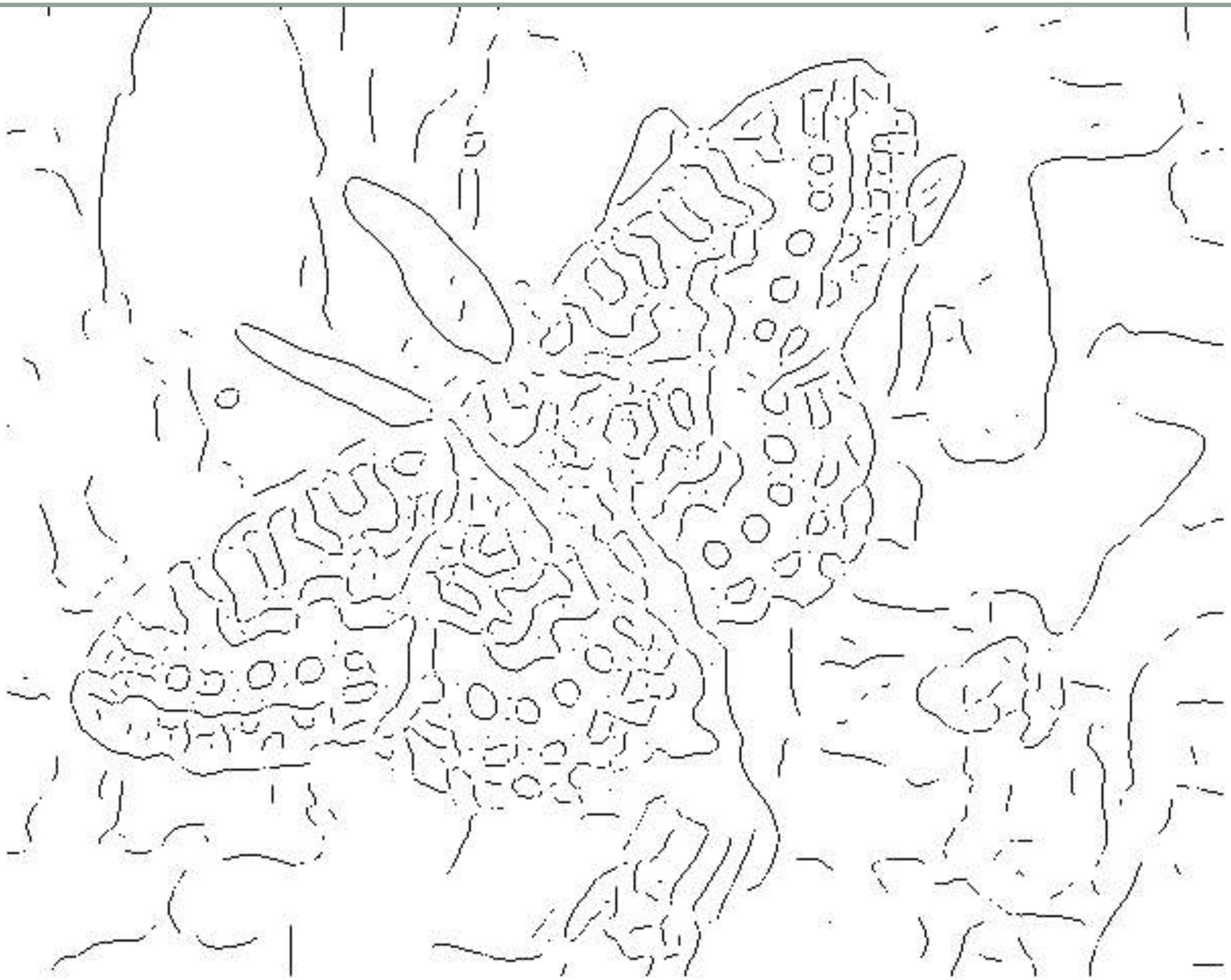
Gradient magnitude image



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

- Skimage:
- >>im_canny=skimage.feature.canny(im, sigma=5)

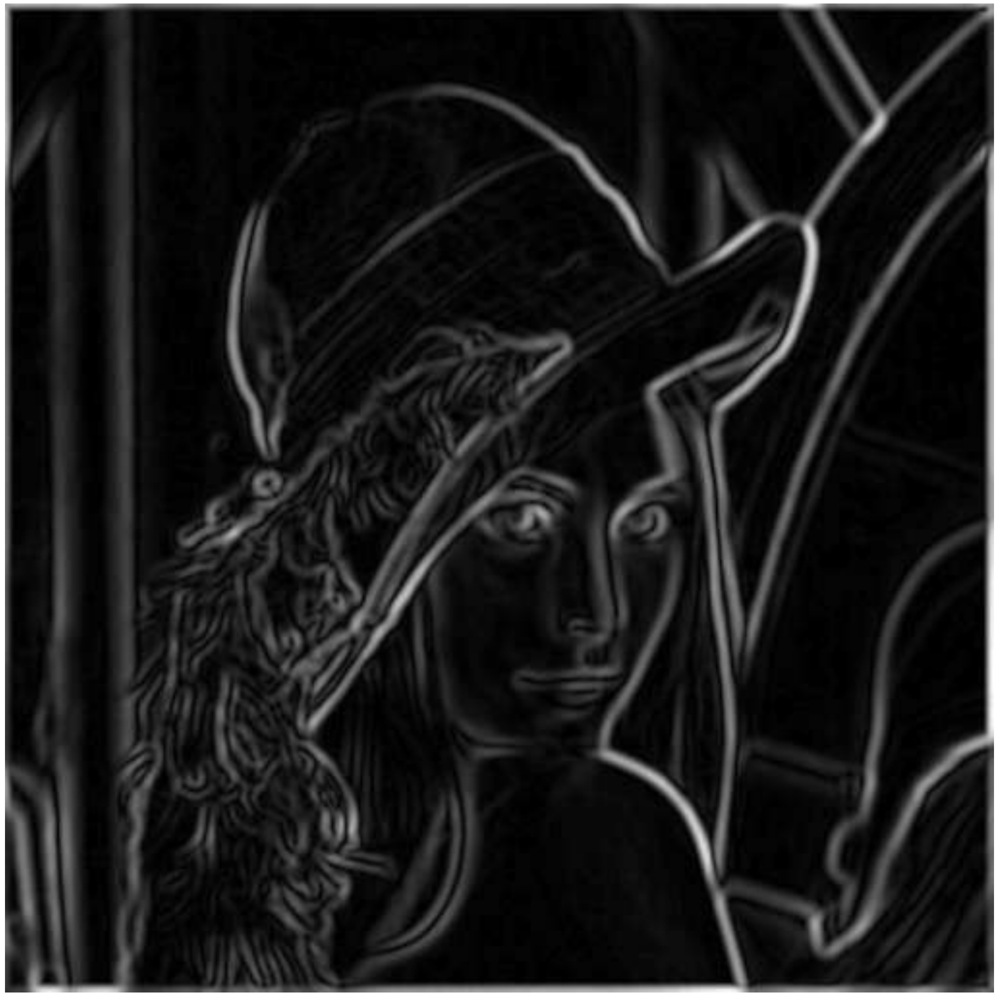
Source: D. Lowe, L. Fei-Fei

The Canny edge detector

original image (Lena)

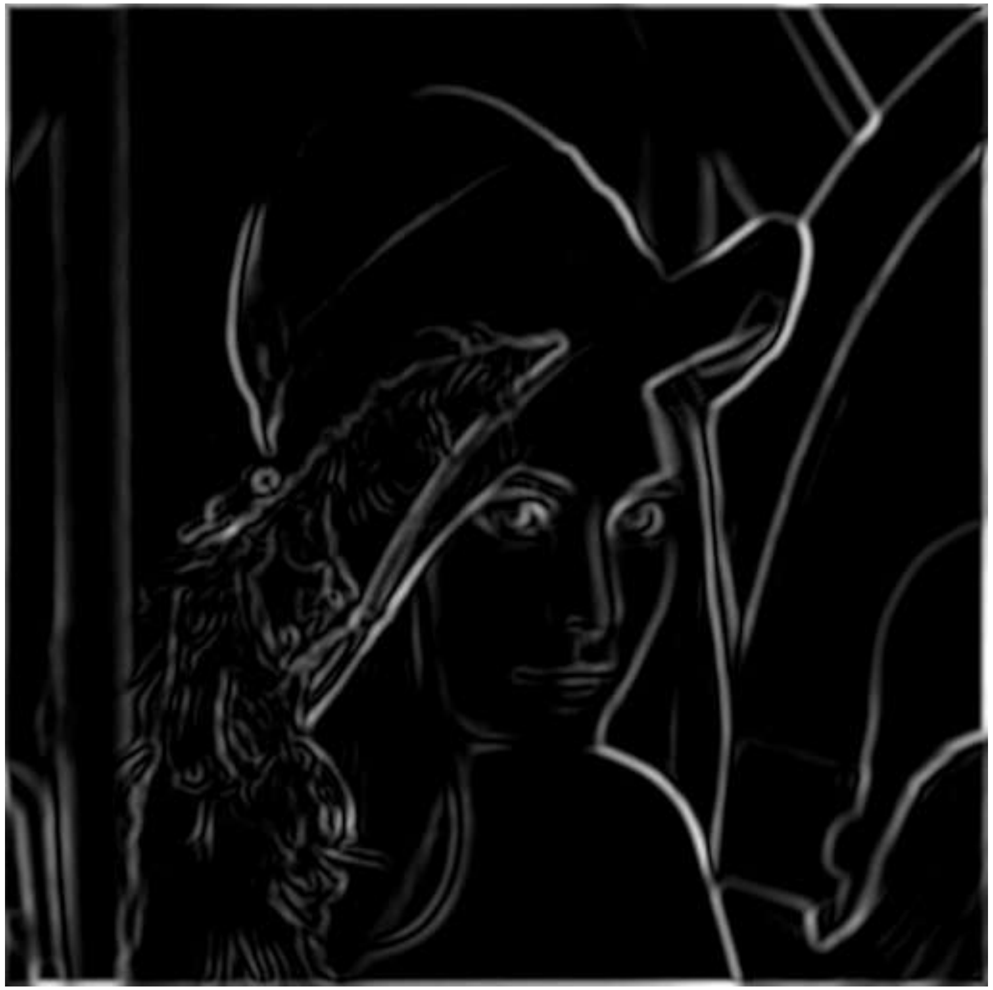
Slide credit: Steve Seitz

The Canny edge detector



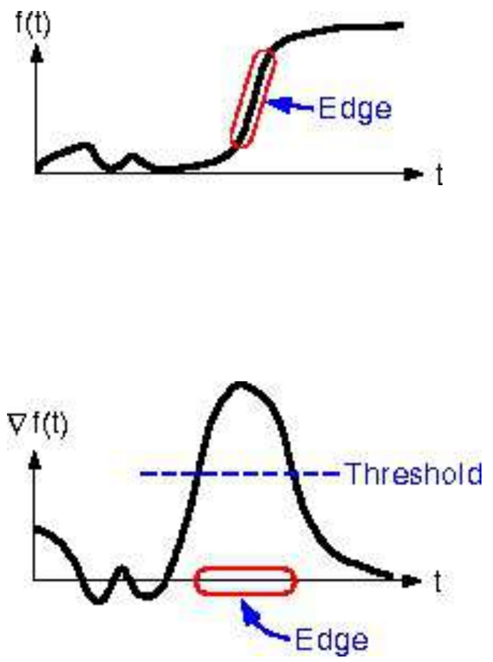
norm of the gradient

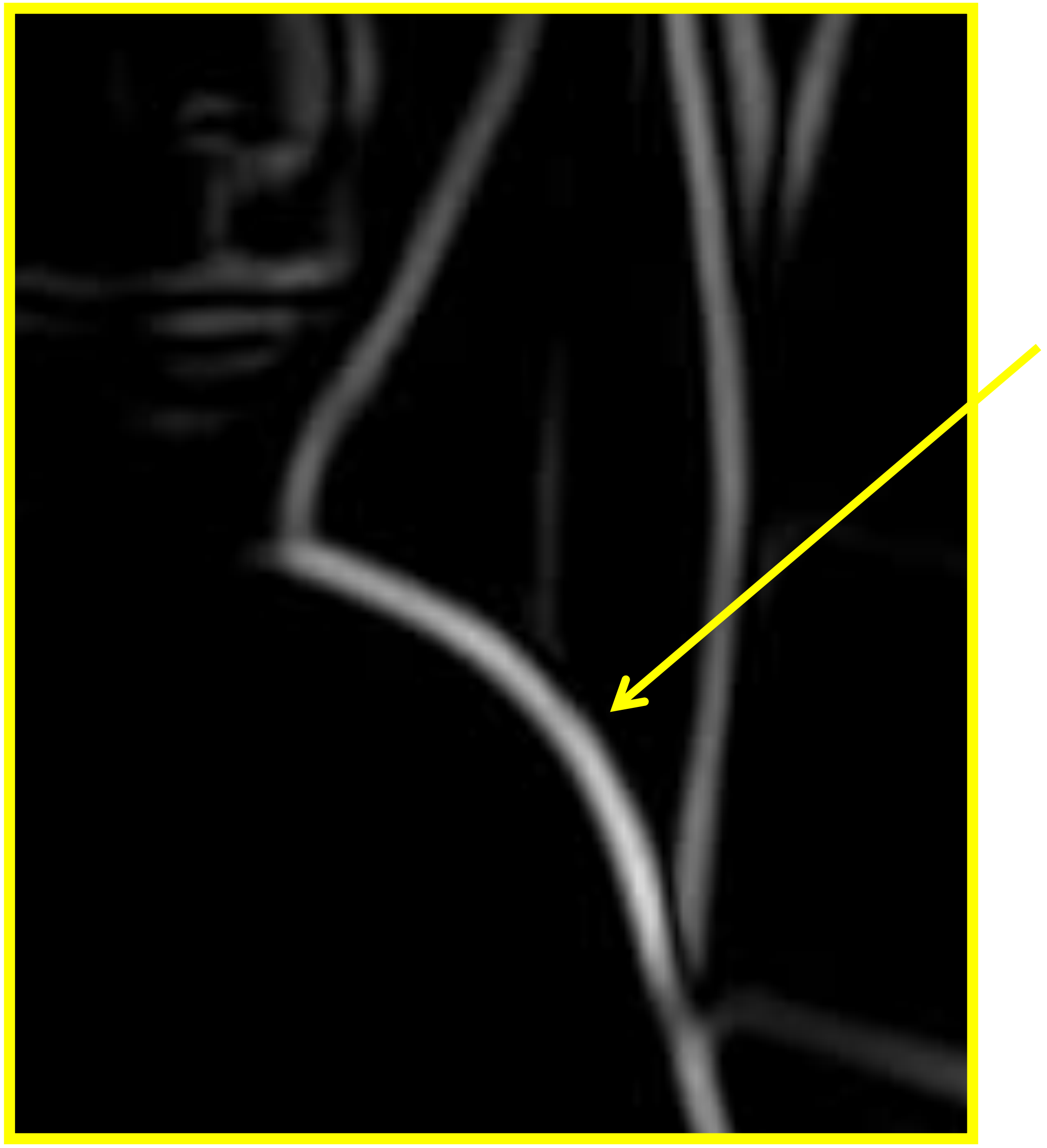
The Canny edge detector



thresholding

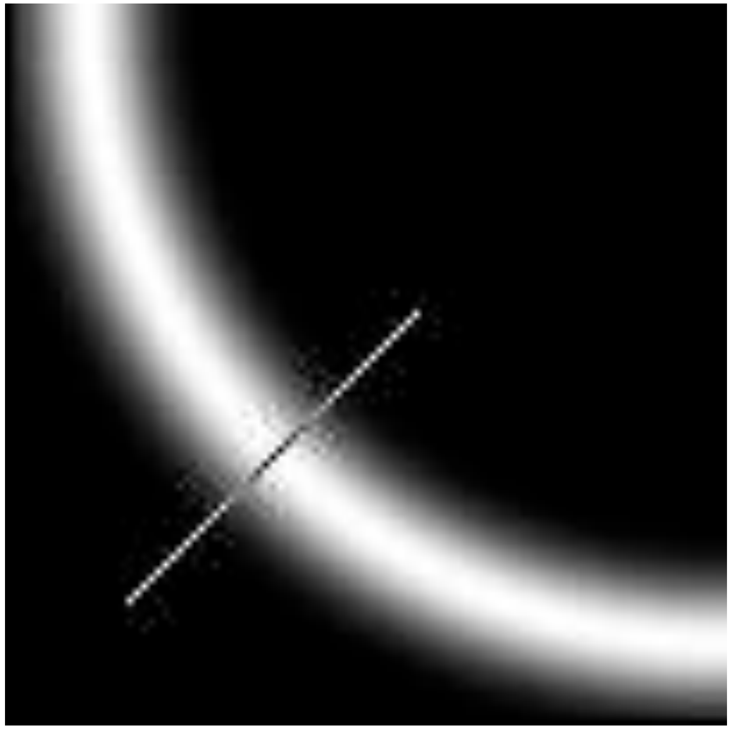
The Canny edge detector





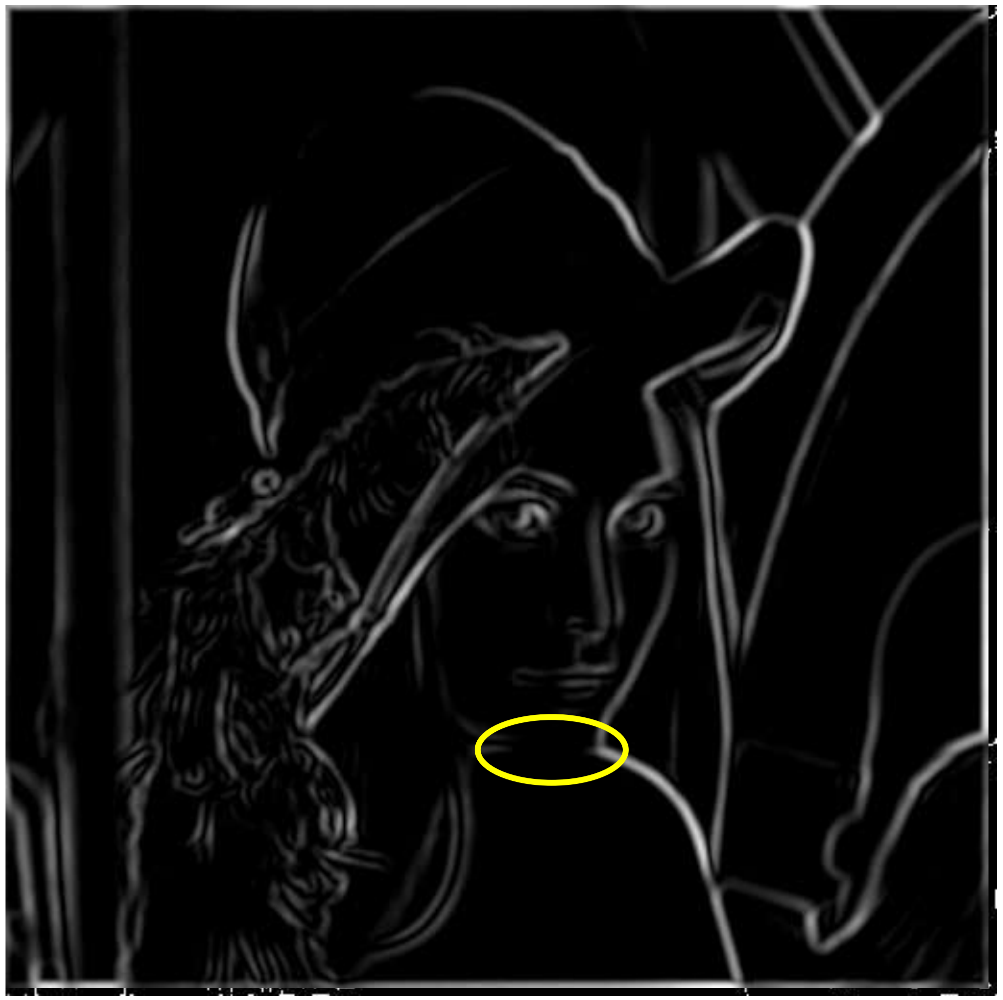
How to turn these thick regions of the gradient into curves?

Non-maximum suppression



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

The Canny edge detector

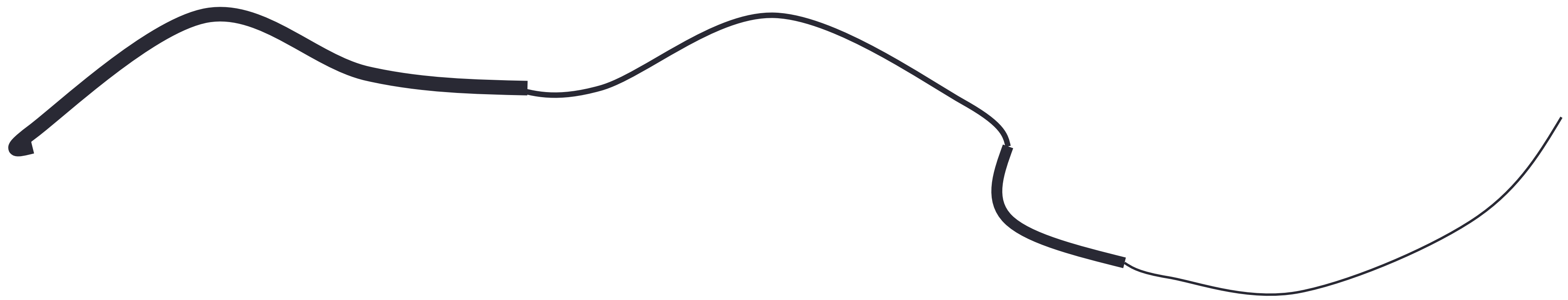


Problem: pixels along this edge didn’t survive the thresholding

thinning
(non-maximum suppression)

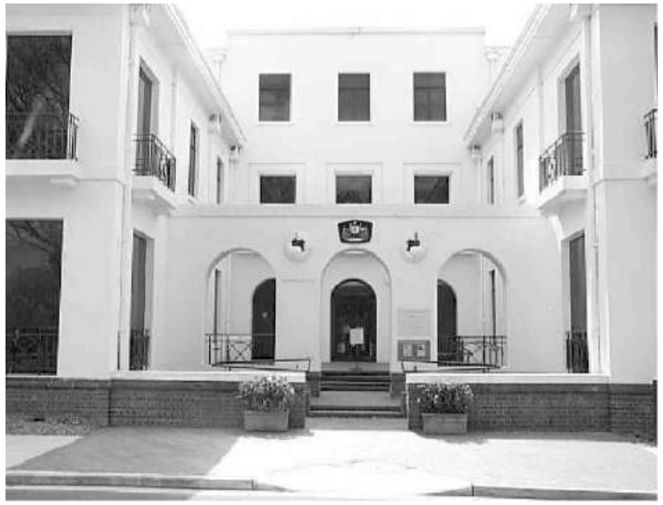
Hysteresis thresholding

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

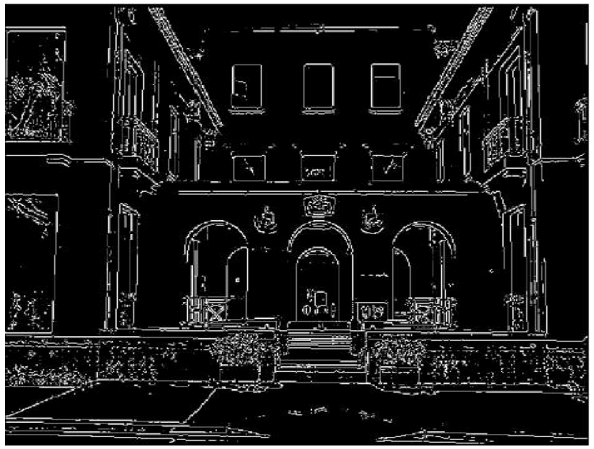


Source: Steve Seitz

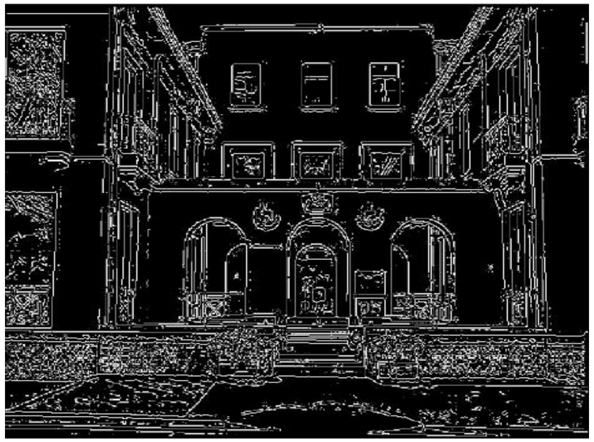
Hysteresis thresholding



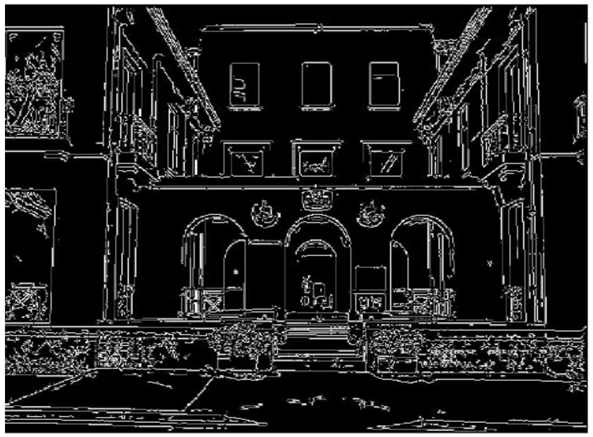
original image



high threshold
(strong edges)



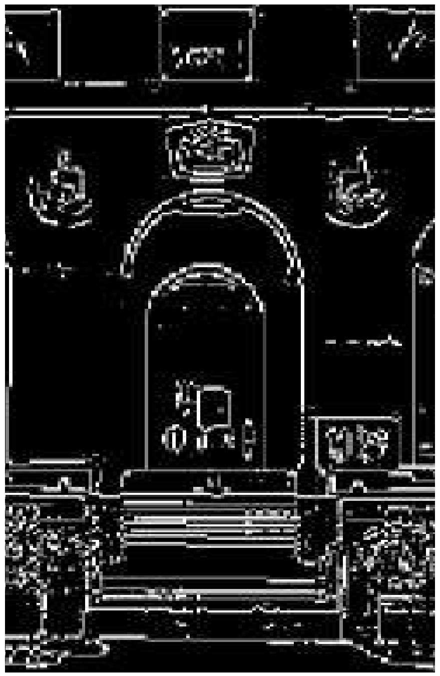
low threshold
(weak edges)

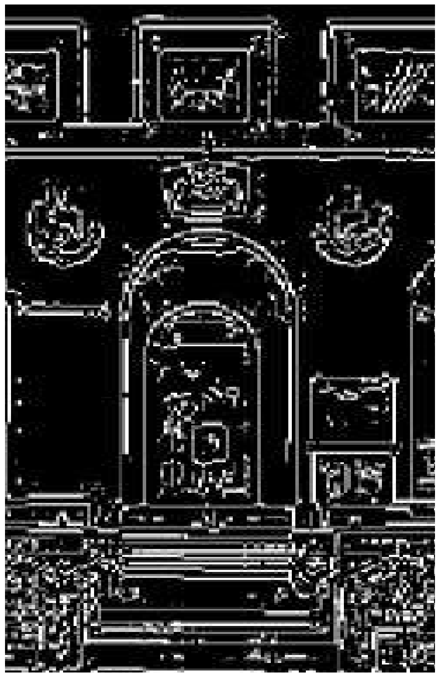


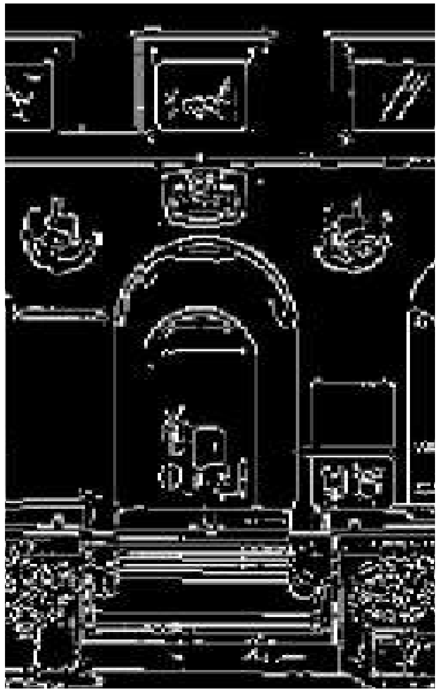
hysteresis threshold

Source: L. Fei-Fei

Hysteresis thresholding


high threshold (strong edges)


low threshold (weak edges)


hysteresis threshold

Source: L. Fei-Fei

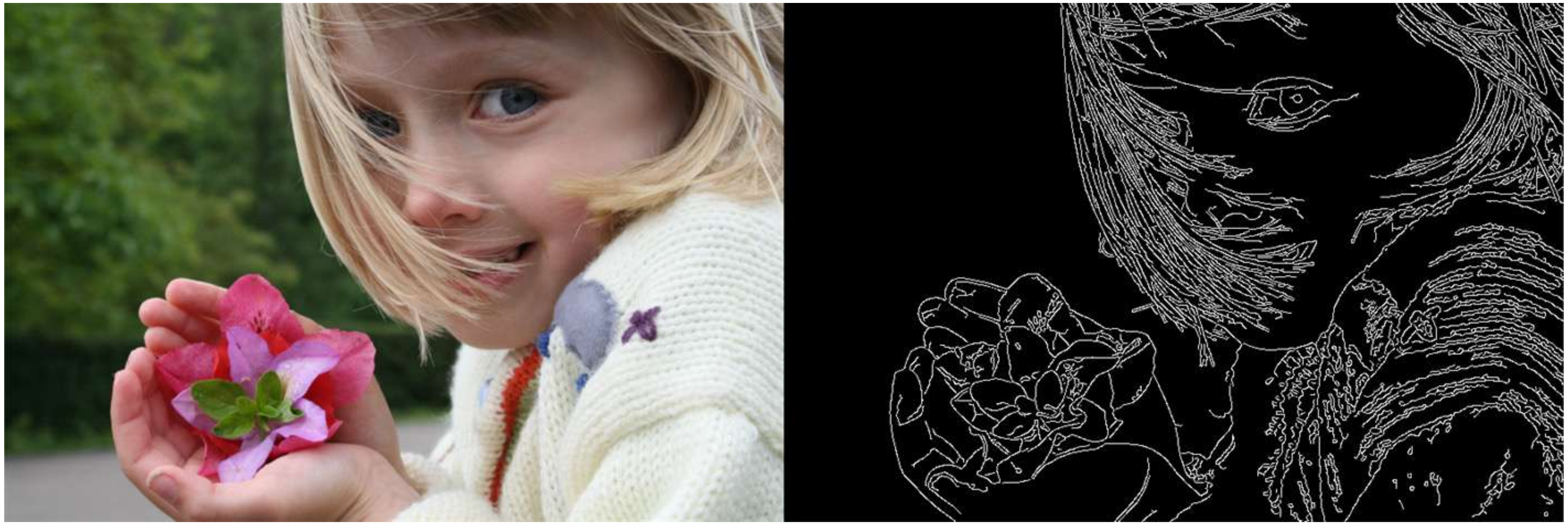
Recap: Canny edge detector

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

- In skimage:
- `>>im_canny=skimage.feature.canny(im, sigma=5)`

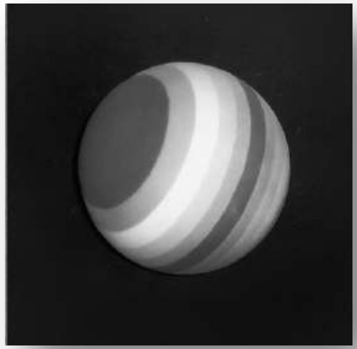
Source: D. Lowe, L. Fei-Fei

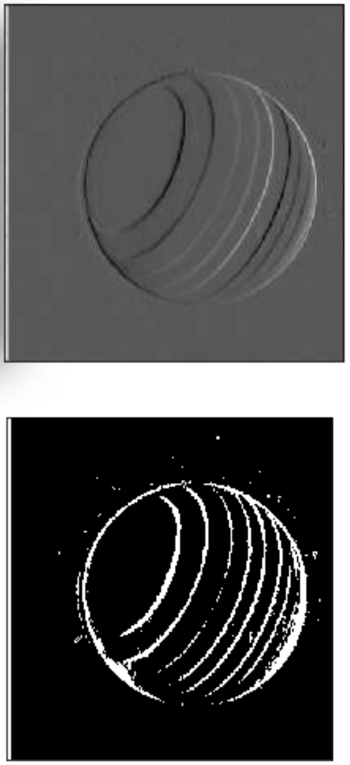
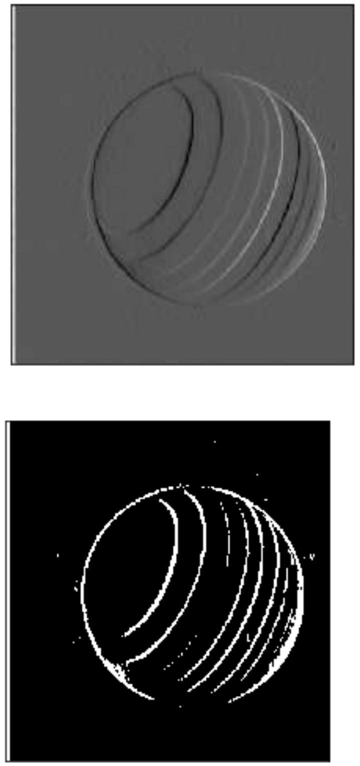
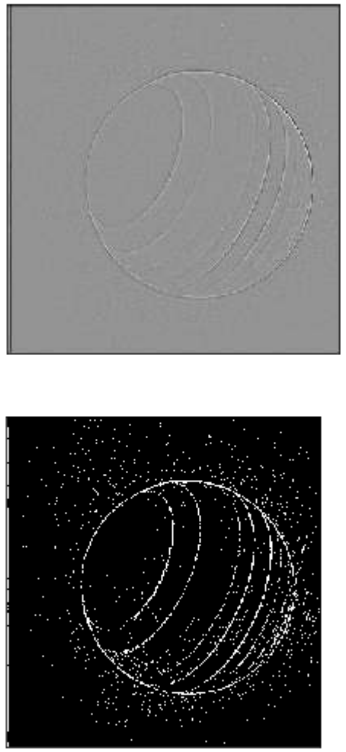
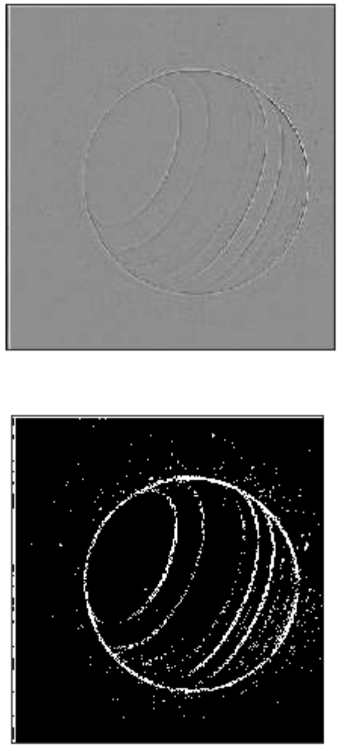
Canny edge detector



Exercices (comparison)

- Determine the operator applied:


Original image

Sobel with a mask

```
mask=np.array([[-1,0,1],[-
2,0,2],[-1,0,1]], dtype='float')

im_sobel=convolve(im,mask)
```

Prewitt with a mask

```
mask=np.array([[-1,0,1],[-
1,0,1],[-1,0,1]], dtype='float')

im_prewitt=convolve(im,mask)
```

Laplacian with a mask

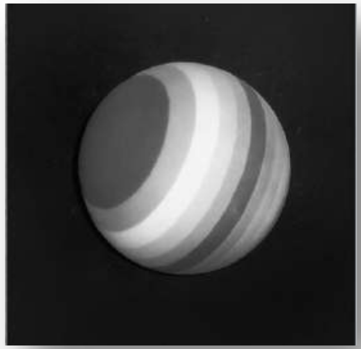
```
mask=np.array([[0,-1,0],[-1,4,-
1],[0,-1,0]], dtype='float')

im2=convolve(im,mask)

fig=plt.imshow(im2<7.5,
cmap='gray')
```

Laplacian with a mask

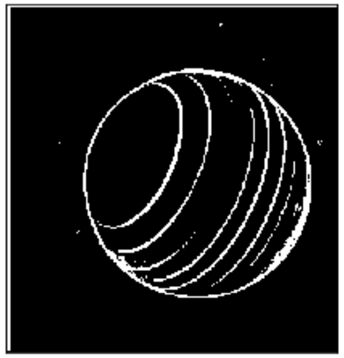
```
mask=np.array([[1,4,1],[4,-
20,4],[1,4,1]], dtype='float')

im_Lapl=convolve(im,mask)
```

Exercices (comparison)

- Determine the operator applied:



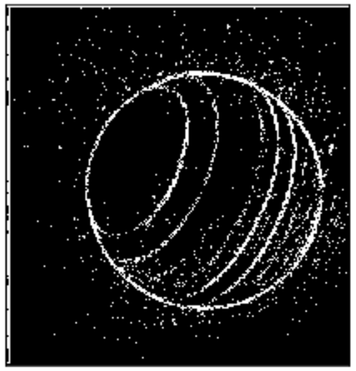
Original image



Sobel

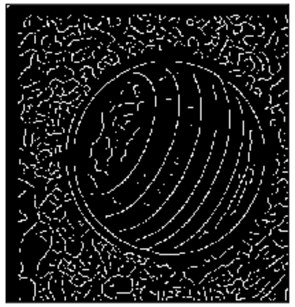
im2=filters.sobel(im)

fig=plt.imshow(abs(im2)>10.5)



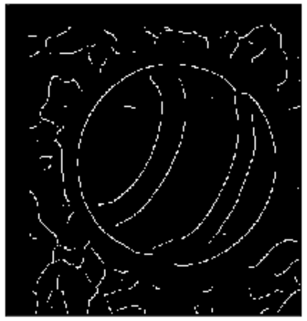
Laplacian of a Gaussian

im2=filters.laplace(im)



Canny sigma=4

im2=feature.canny(im, sigma=4)

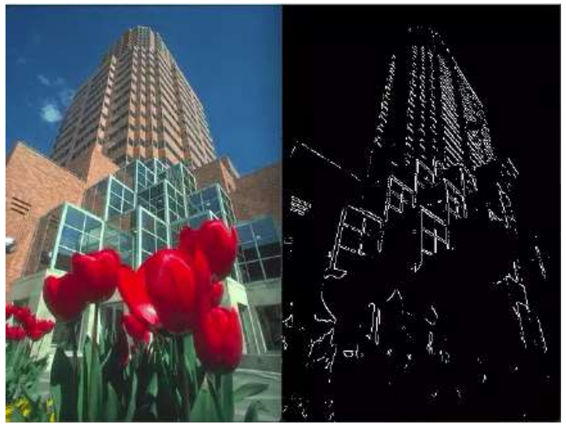


Canny sigma=6.5

im2=feature.canny(im, sigma=6.5)

Canny edge detector Limitations

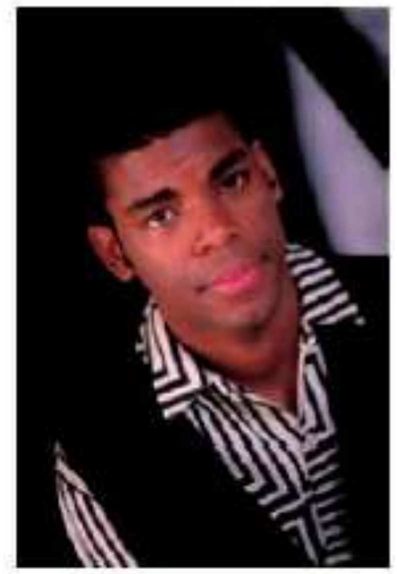
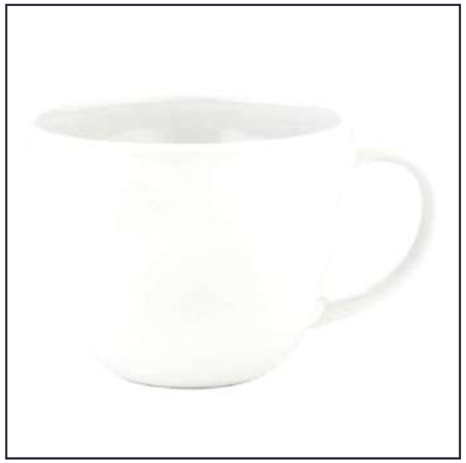
- Canny detector only focuses on local changes and it has no semantic understanding.



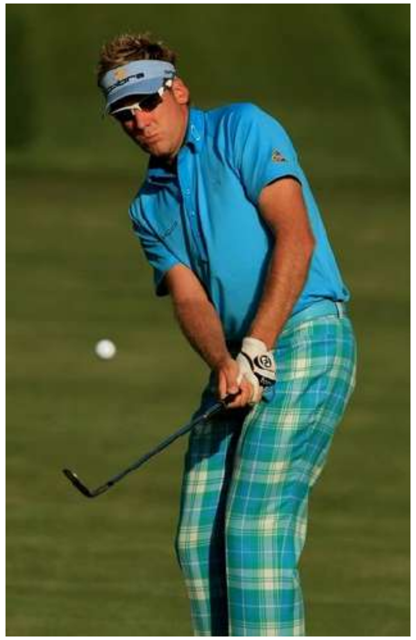
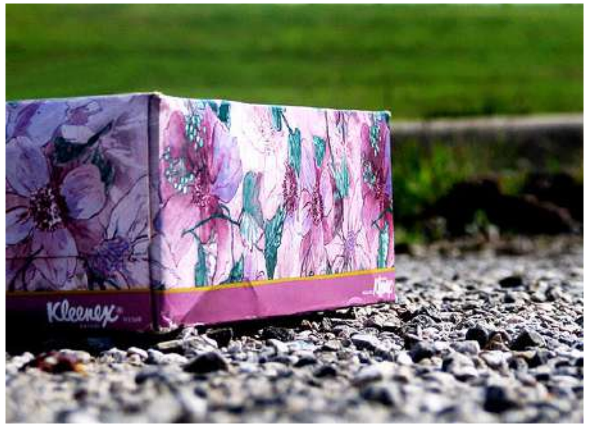
Canny Edge detector fails in this case as it has no understanding of the content of the image.

There are alternatives as the one in OpenCV: Holistically-Nested Edge Detection (HED), an end-to-end deep neural network. [Saining Xie, Zhuowen Tu "HED" 2015]

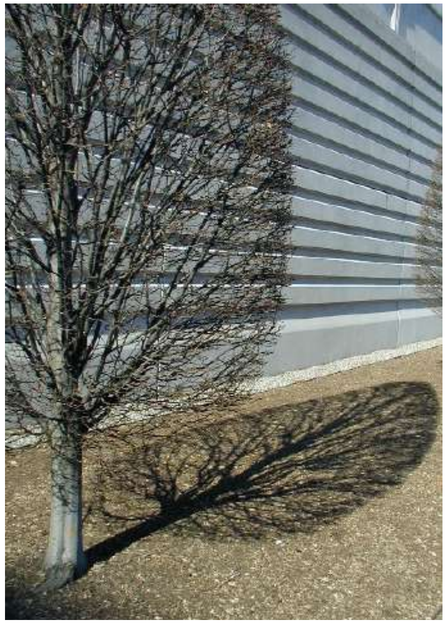
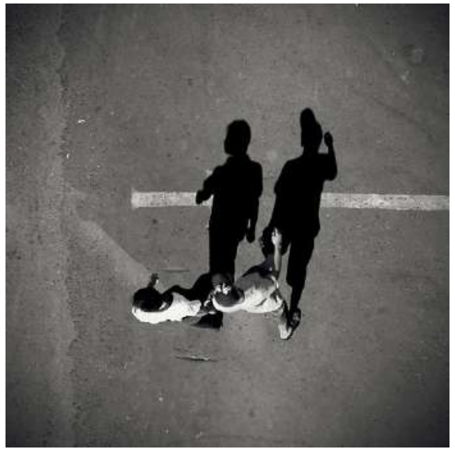
Low-level edges vs. perceived contours

Background

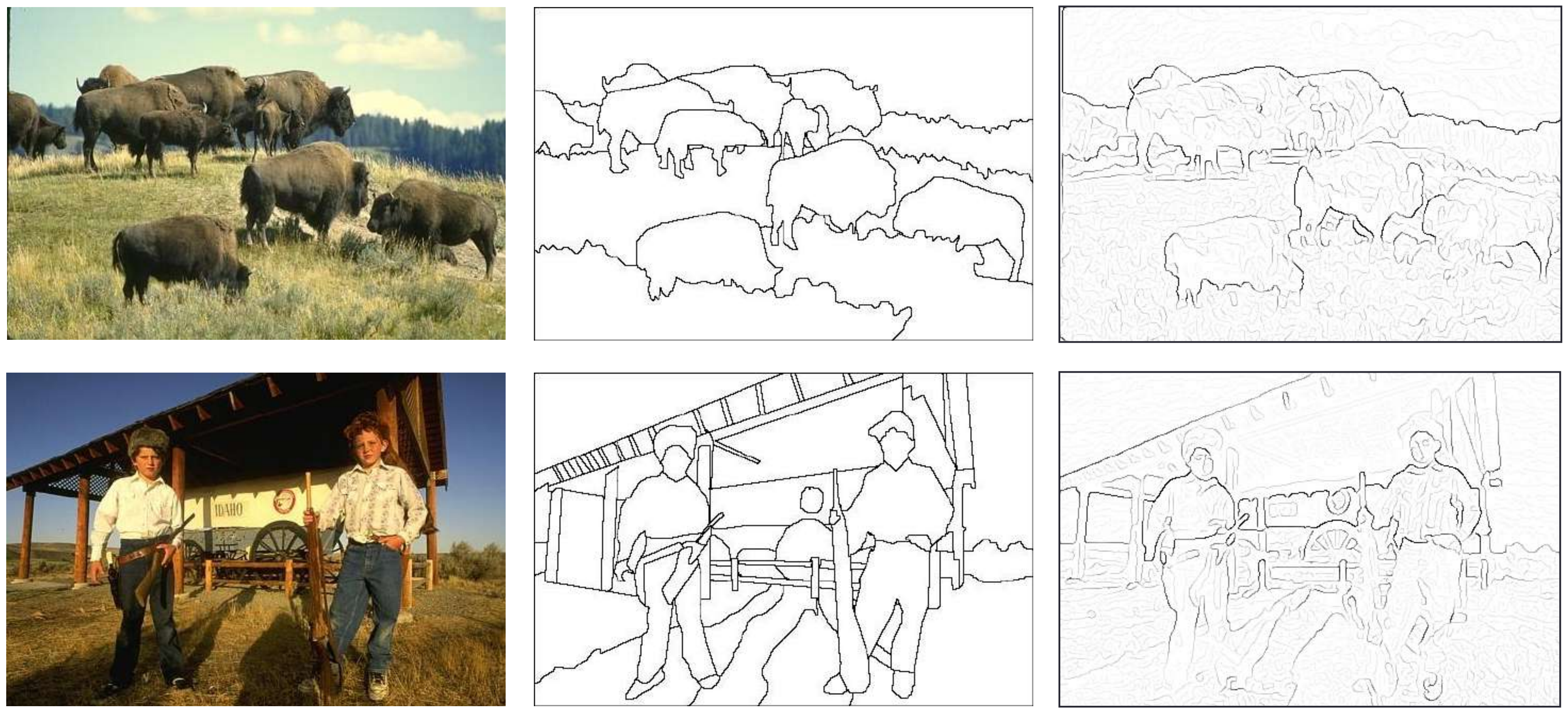
Texture

Shadows

Is Canny edge detector distinguishing these contours?

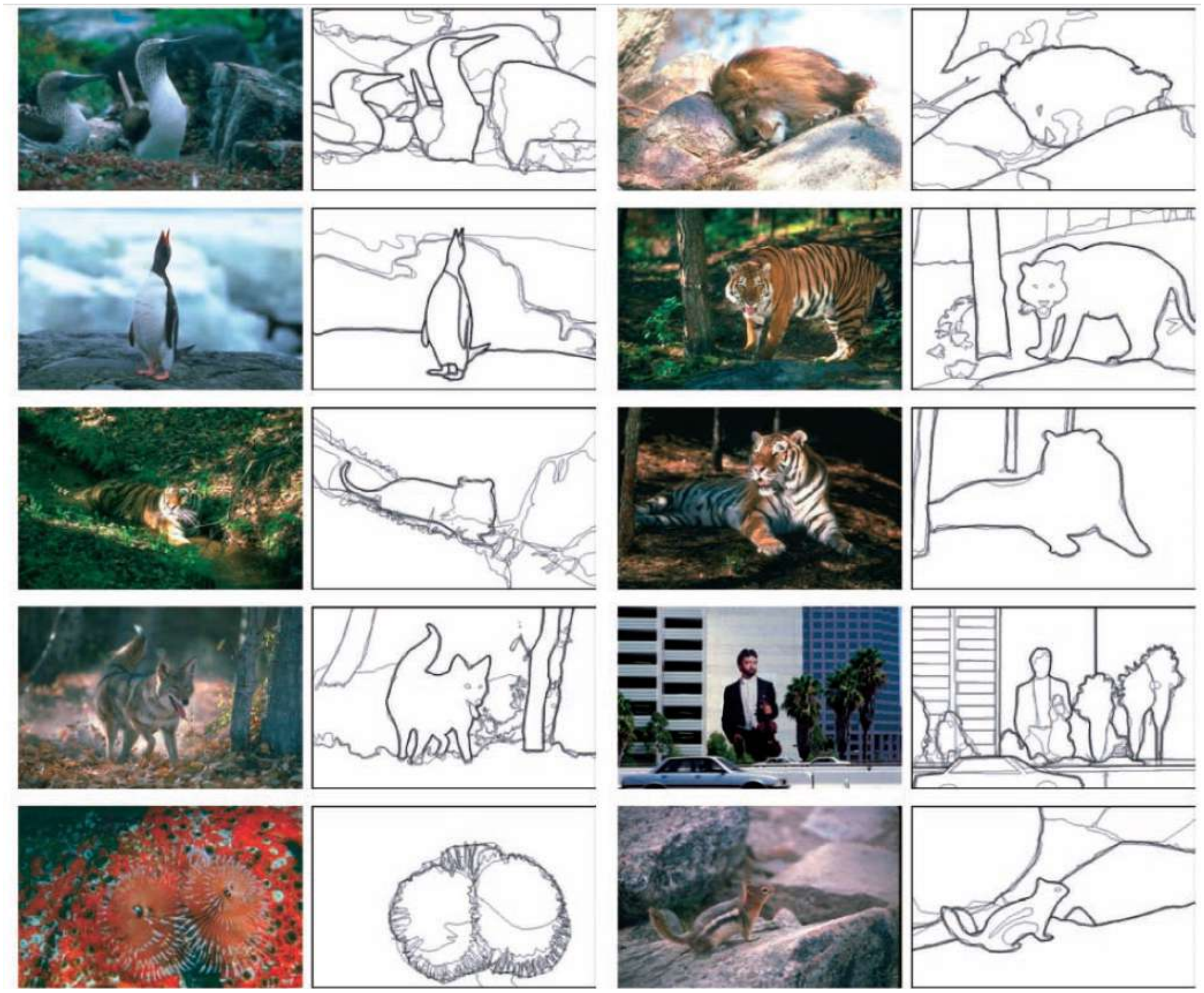
Low-level edges vs. perceived contours



- 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‘

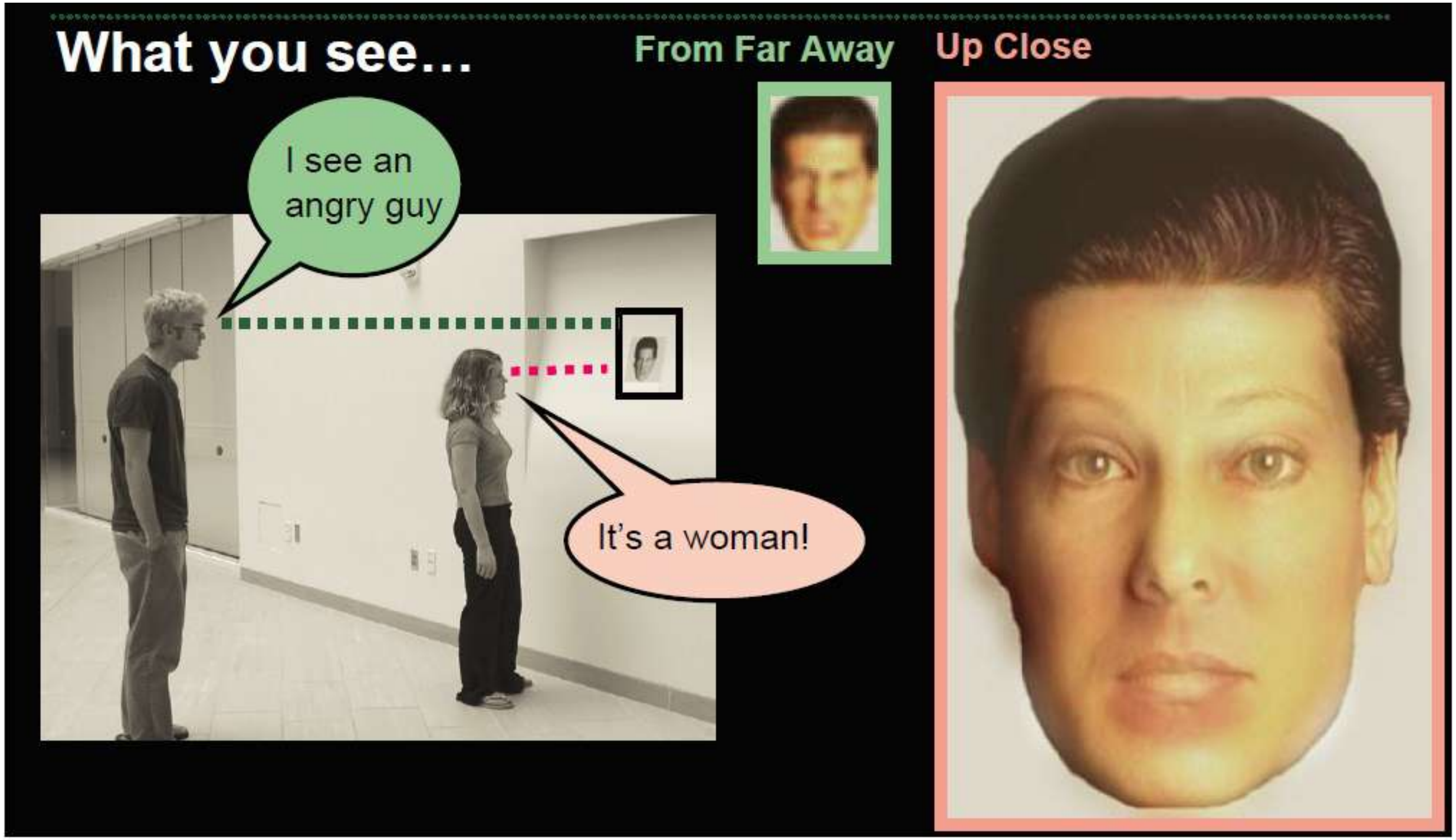


[D. Martin et al. PAMI 2004] Human-marked segment boundaries

Today

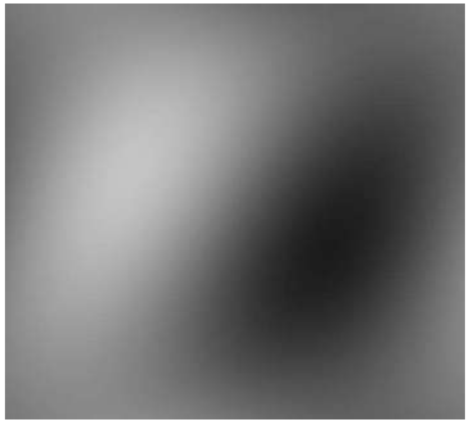
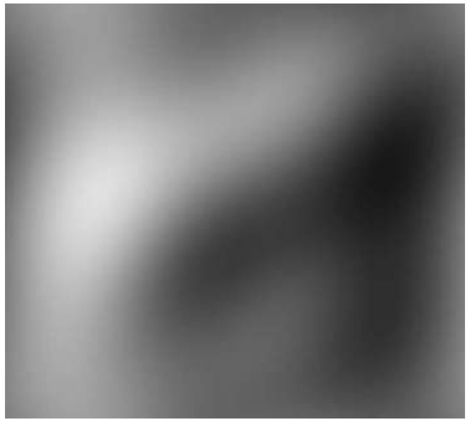
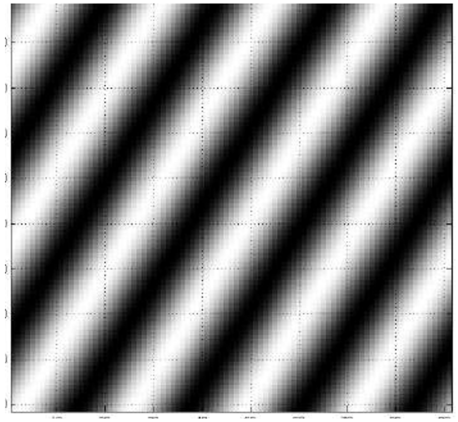
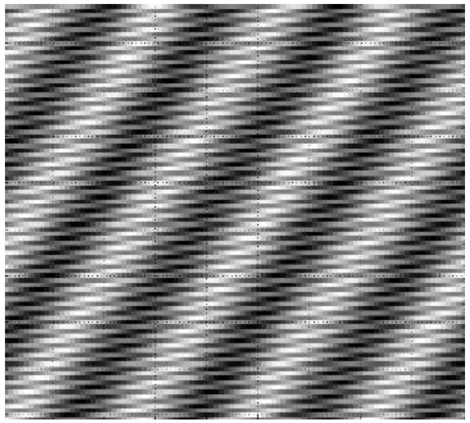
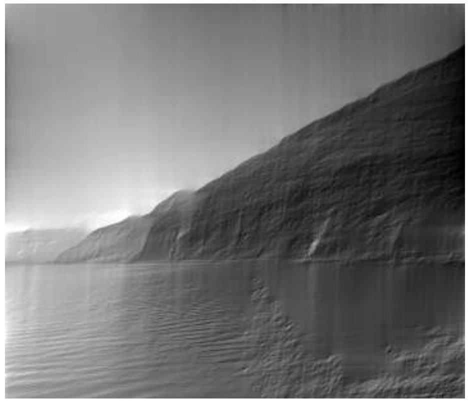
- What is an image derivative and its relation to the edges
  - How to implement different edge detectors and what is their difference?
  - First and second derivatives, image gradient and Laplacian
- How to get image derivatives with a Gaussian
- Canny edge detector
- Application: hybrid images

Application: Hybrid Images



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

Intuition about low and high image frequencies

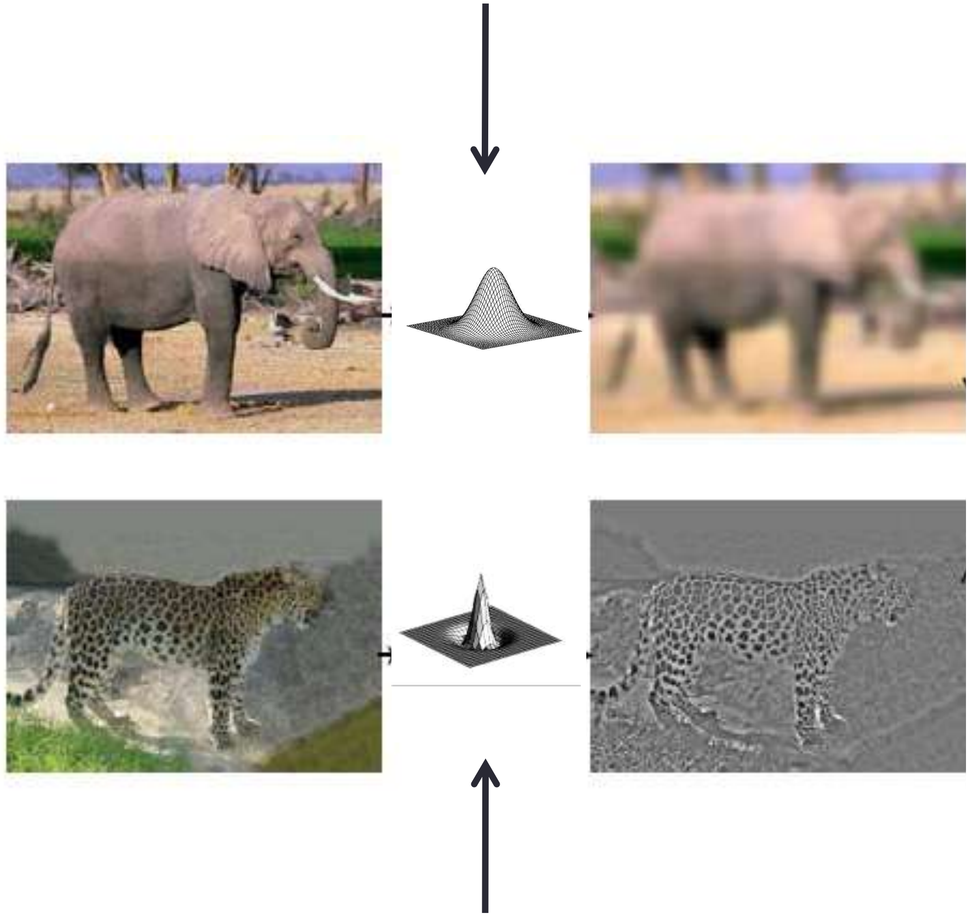






Sum of 50 frequencies

Low frequencies -> pixel values that are changing slowly over space.
High frequency -> pixel values that are rapidly changing in space.

Application: Hybrid Images

Low image frequencies

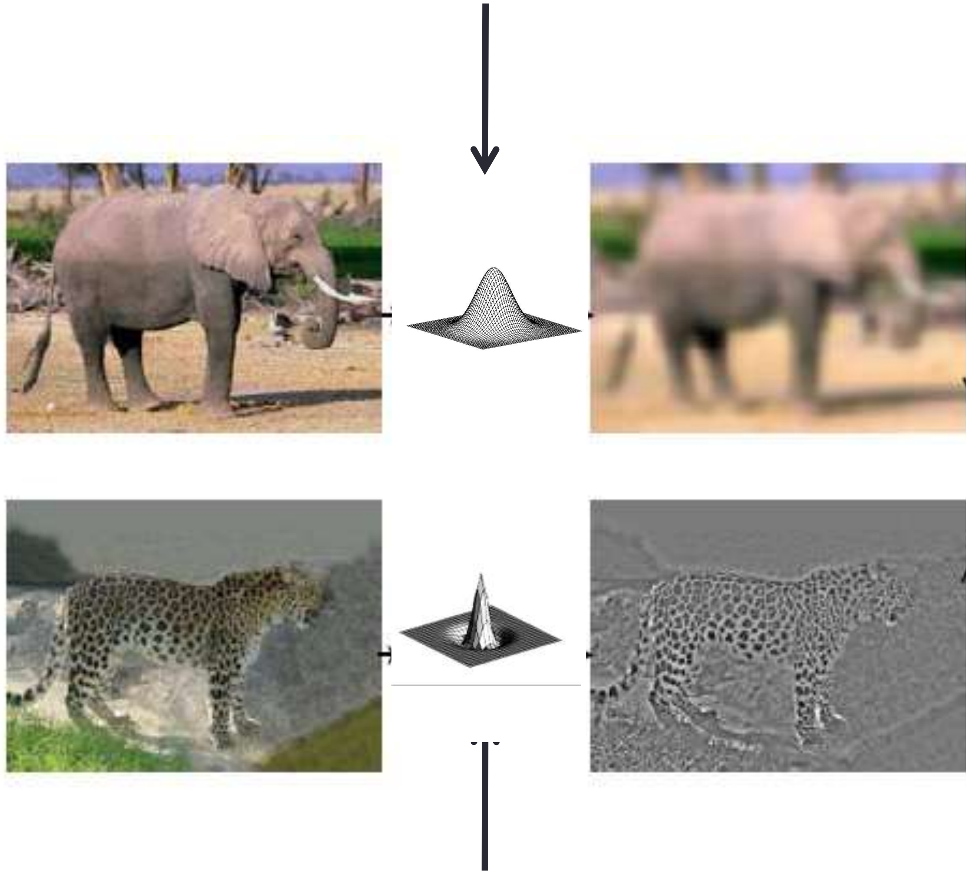


How to extract them?

High image frequencies

Application: Hybrid Images

Gaussian Filter



Laplacian Filter or the difference btw the image and its low frequencies

Low frequencies:

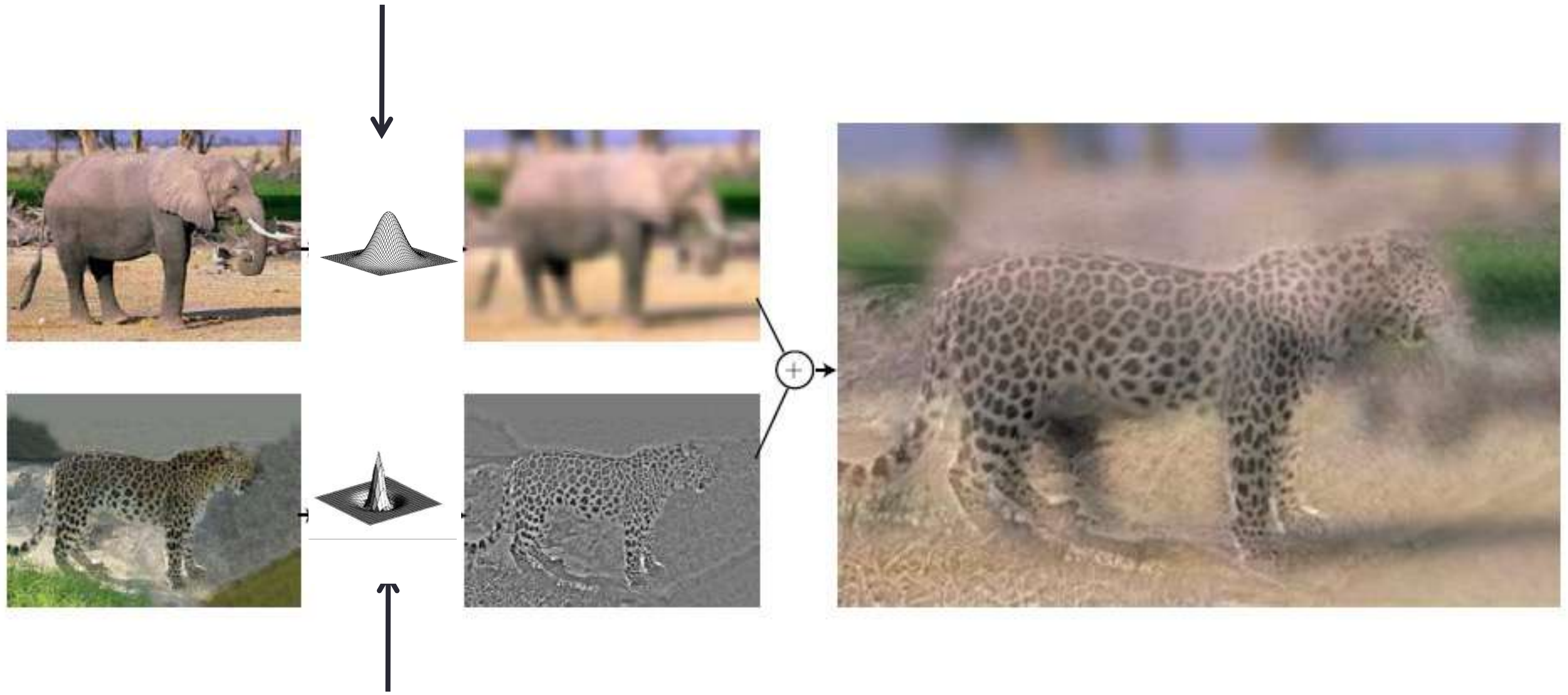
L(I1) -> Gaussian smoothing

High frequencies:

H(I2)=I2 - L(I2)

Application: Hybrid Images

Low frequencies

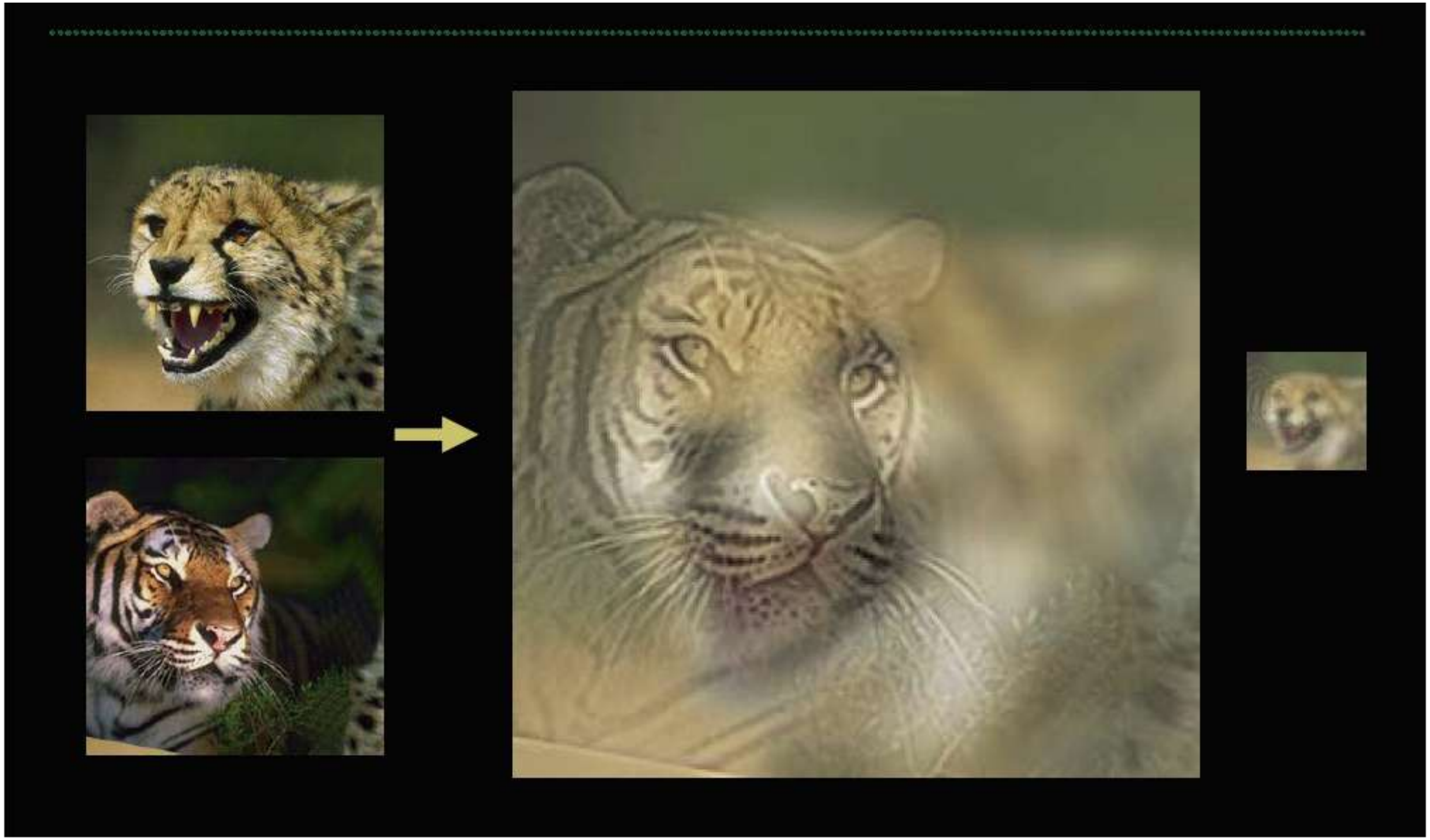


High frequencies

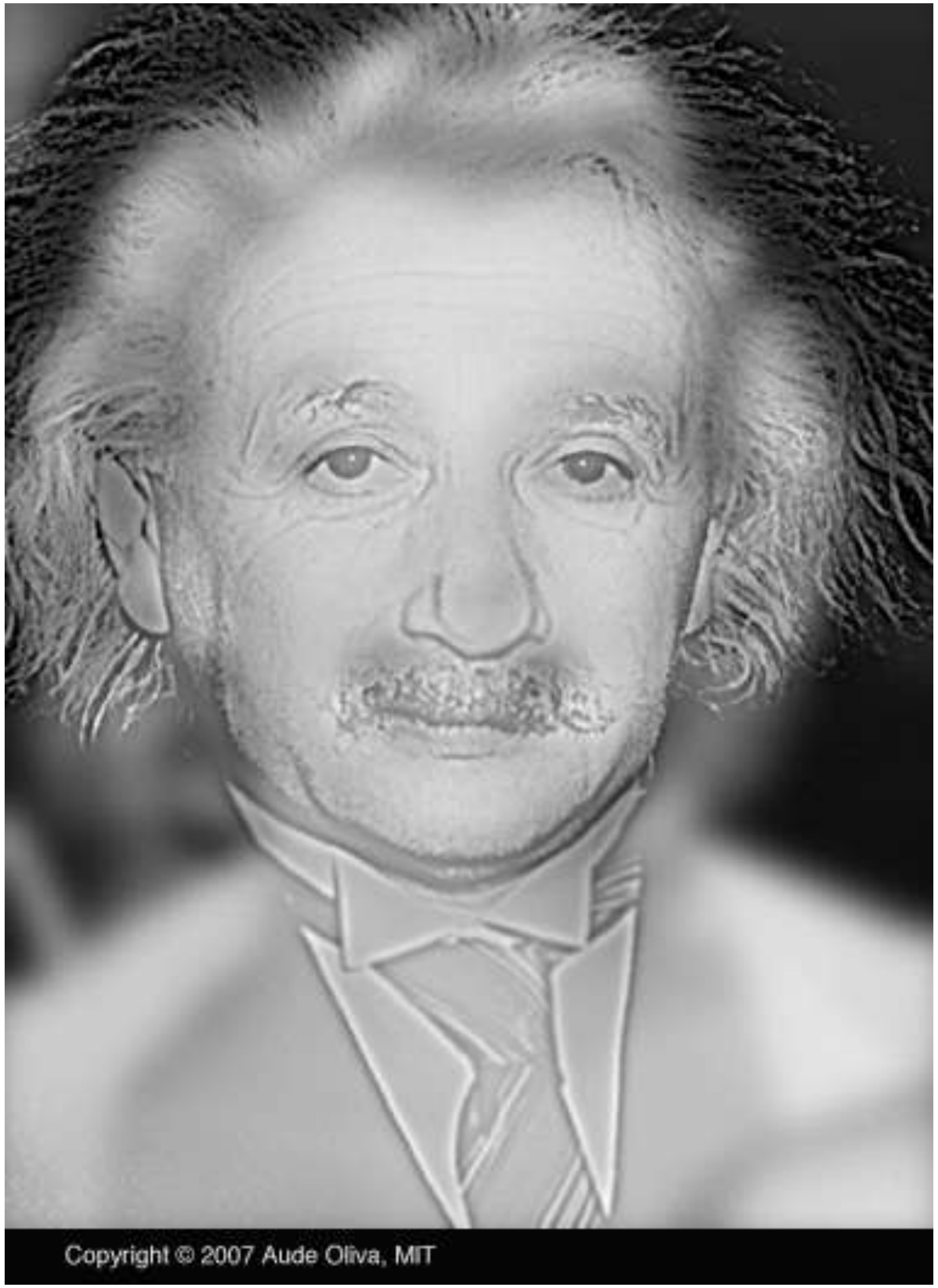
Hybrid image:
Hybrid(I1,I2)=L(I1)+H(I2)

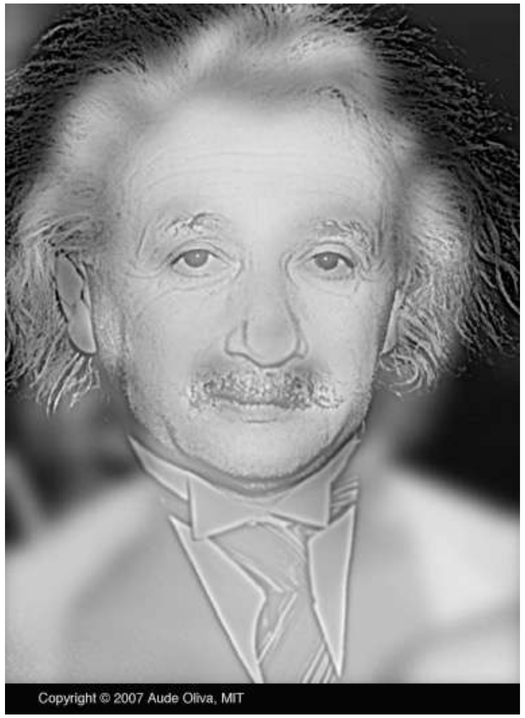
Note: In order to emphasize the effect, use different sigma parameters in the Gaussian

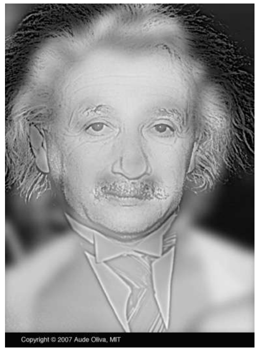
A. Oliva, A. Torralba, P.G. Schyns, “Hybrid Images,” SIGGRAPH 2006

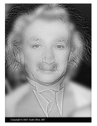


Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006







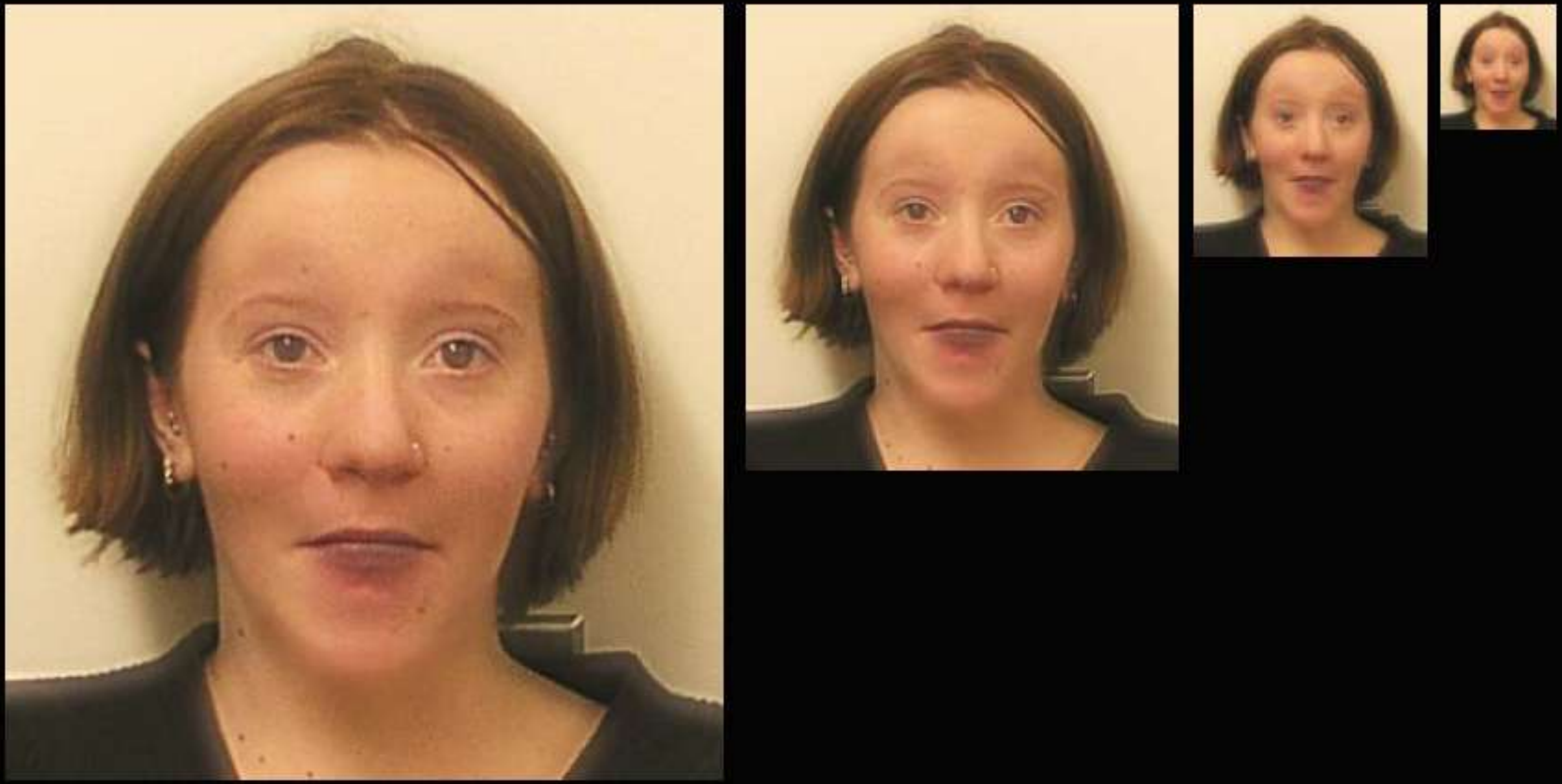


http://cvcl.mit.edu/hybrid_gallery/gallery.html

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!

Let’s do some test

- Go to socrative.com
- Room: ComputerVision2122