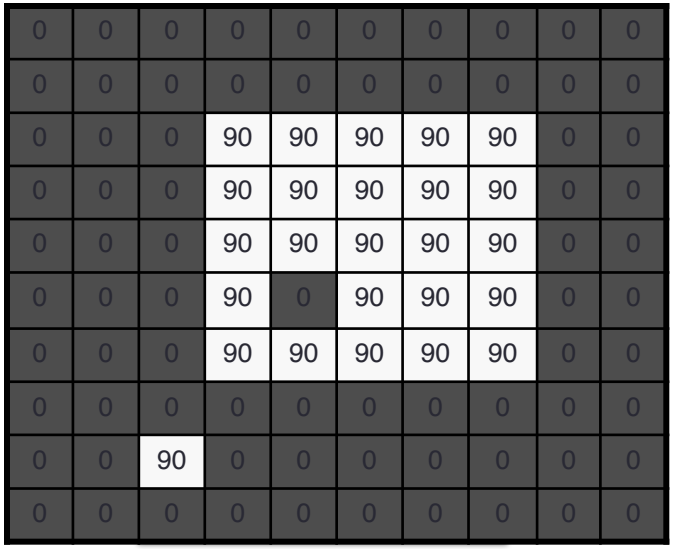
IMAGE GRADIENTS AND EDGES

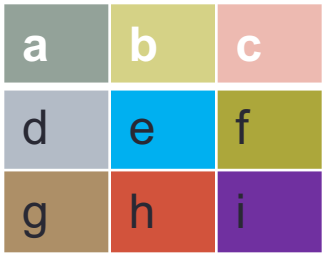
Class 3: Artificial Vision

Source of most slides: Kristen Grauman, UT-Austin

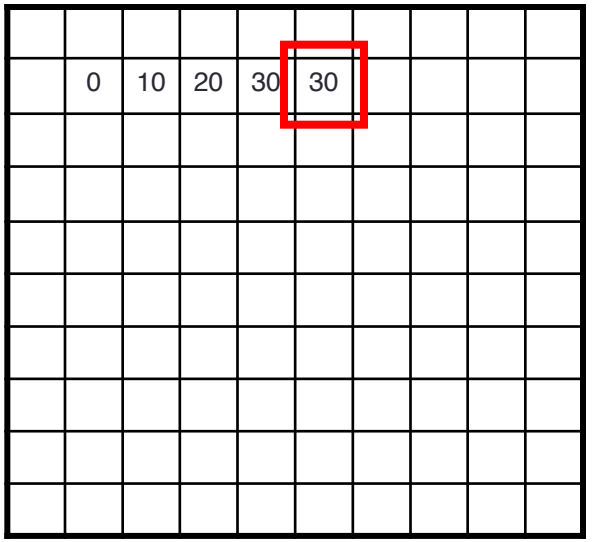
Reminder: Linear filter

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





"box filter"



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

Last lecture

Answers next page

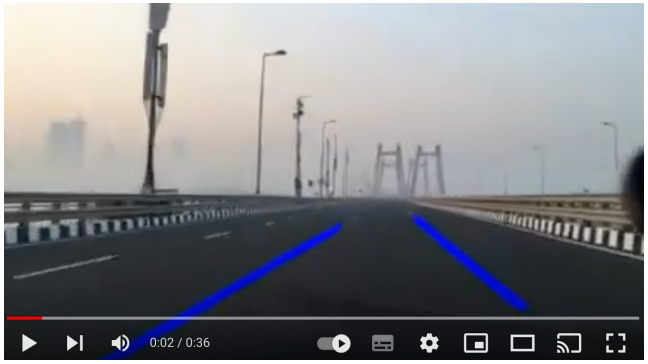
- Linear filters and convolution useful for
  - Image smoothing, removing noise
    - ...box.....filter → más importancia al pixel que estamos estudiando y a los vecinos.
    - Alternative (.ponderate.…) filter [1,4,16...] → gaussian
    - Impact of scale / width (……) of the smoothing filter
- Mean filter: → máscara de 1s → promedio ó máscara ponderada
  - Smoothing pixel values by …….. them
  - Uniform average vs. …….. average
- Median filter: → Valores con 0 o 100. → ordenaba los píxeles pero no hace un promedio. No suaviza el contorno, pero lo guarda muy bien.
  - a linear or non-linear filter, ?!
  - Does it assure no averaging grey levels?
  - Does it assure no inventing new grey-levels?
  - Is it edge-preserving?

Last lecture

- Linear filters and convolution useful for
  - Image smoothing, removing noise
    - Box filter
    - Alternative 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
  - it assures no averaging grey levels
  - it assures no inventing new grey-levels
  - It is edge-preserving

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?
- What is a Canny edge detector?
- Application: hybrid images



Review

original image filtered

Which filter is applied:

→ horizontal haciendo suavización horizontal

vertical → $f = 1/9 \times [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]^T$

horizontal → or $f = 1/9 \times [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]$?!

→ Nos podemos salir del rango de la imagen

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 (edges, etc) –
  - 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?**

edge → cambio brusco de valores de gris
contorno → límites de los objetos.

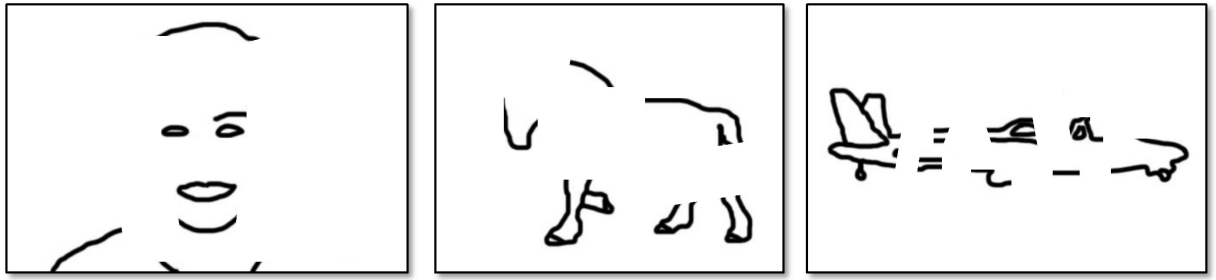
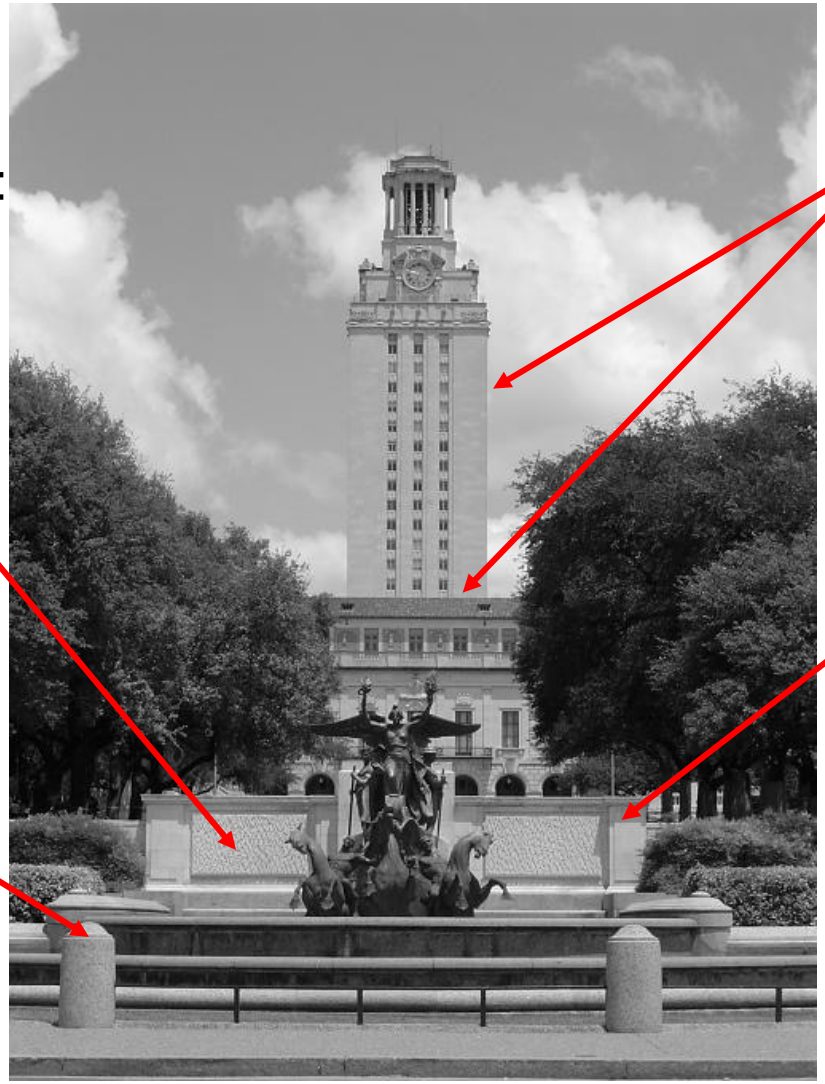


Figure from J. Shotton et al., PAMI 2007

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

What does cause an edge?



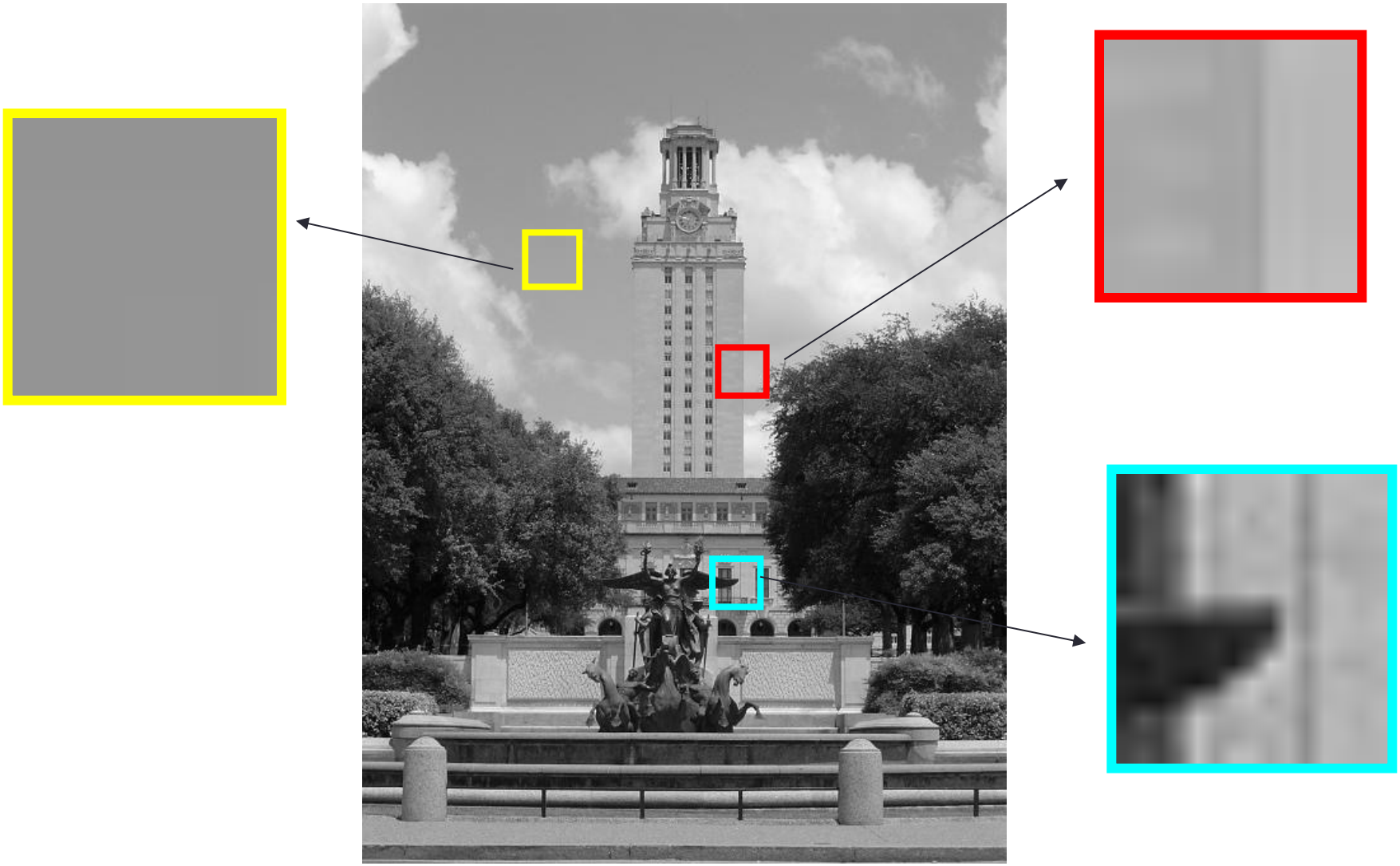
Reflectance change: appearance information, texture

Depth discontinuity: object boundary

Cast shadows

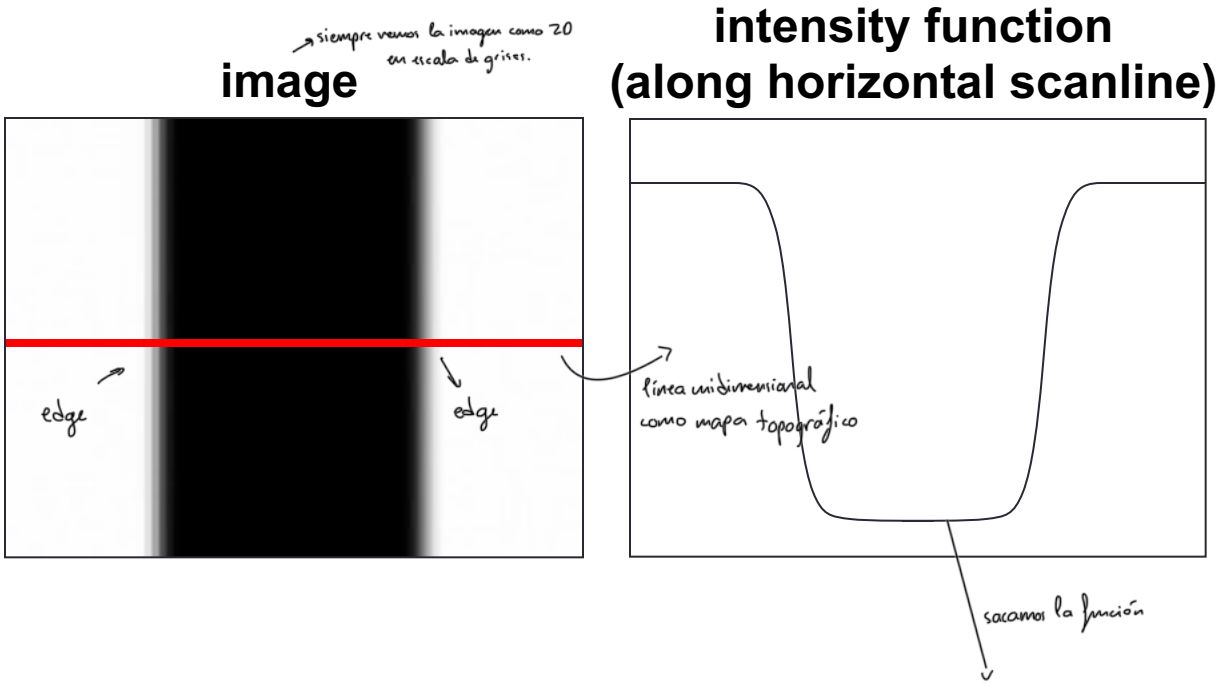
Change in surface orientation: shape

Edges/gradients and invariance



Derivatives and edges

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



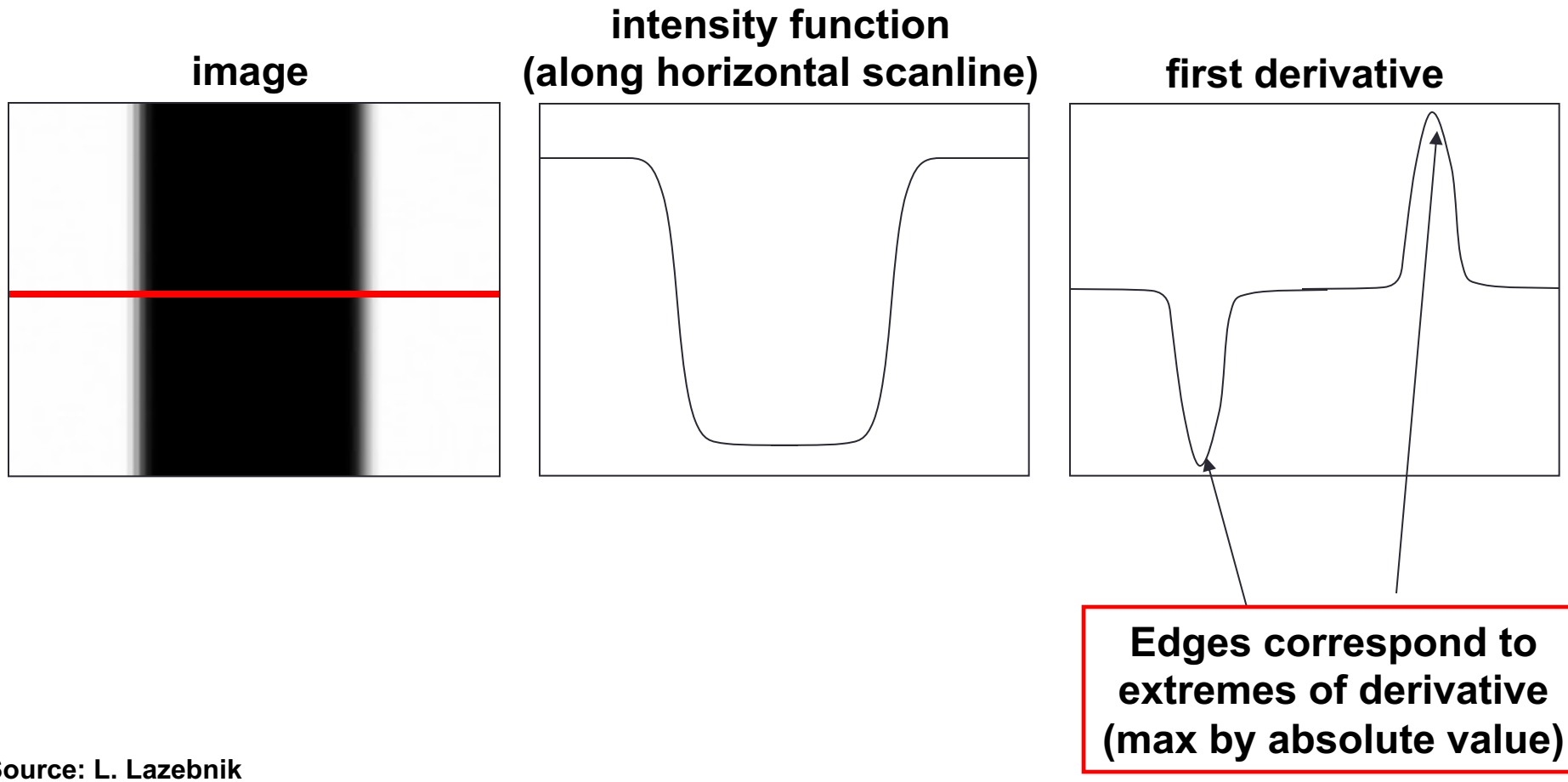
How to detect the points (pixels) of maximal change?

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

Source: L. Lazebnik

Derivatives and edges

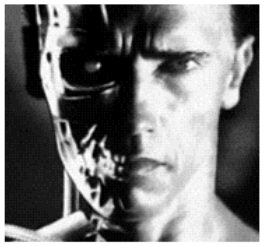
An edge is a place of rapid change in the image intensity function.

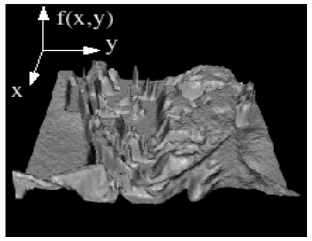


Source: L. Lazebnik

Derivatives with convolution

But… the image is 2D!





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

diferencia entre 2 puntos

pixel actual

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

derivada respecto a x

como mínimo 1 (diferencia de distancia entre 2 píxeles vecinos).

→ No tiene sentido porque estamos hablando de 3 colores diferentes. Calcular la derivada respecto a los canales no tiene sentido (podemos tener RBG en vez de RGB).

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

Derivatives with convolution

But… the image is a discrete matrix!

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

como mínimo 1

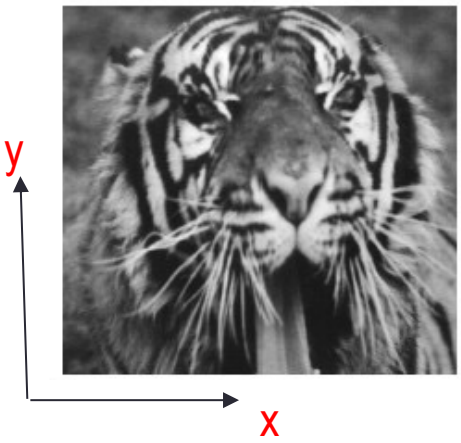
We can approximate it using finite differences:

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

⇒ derivada de la imagen

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

a: f[x-1, y+1] no está, por tanto 0.

0	~~b~~ 0	0
0	-1	1
0	0	0

Si aplicamos esta máscara tendremos la derivada de la imagen respecto a x.

↓ obtenemos derivada parcial en la dirección x.

↓ queremos obtener este resultado mediante convolution y la máscara?

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

→ lo tiene la e, pero la tenemos en negativo

Recall:

→ Por cada píxel multiplicamos sus vecinos por [a, b, c, d, e, f, g, h, i]

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

Partial derivatives of an image

a	b	c
d	e	f
g	h	i

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

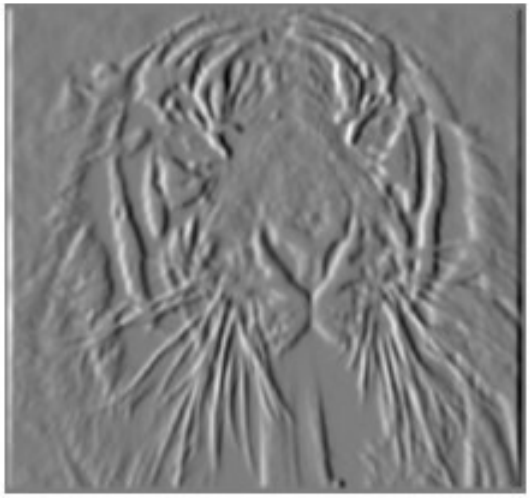
-1	1

Partial derivatives of an image

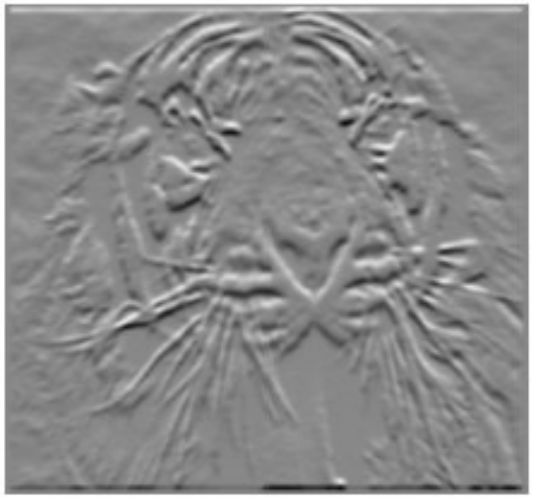
respecto y $\frac{\delta f(x,y)}{\delta y} = \frac{f(x,y+1) - f(x,y)}{1}$ (*)

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

-1	1



o destaco contornos verticales o horizontales



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

abajo ↓ arriba ↓

-1
1

? or

1
-1

depende de si la y es para ↑ arriba o para abajo ↓

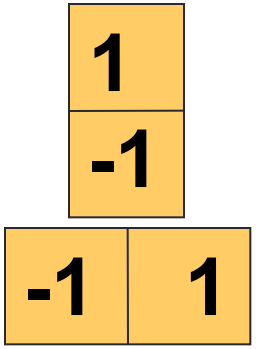
Which derivative does show changes with respect to x?

First derivatives: discrete operators.

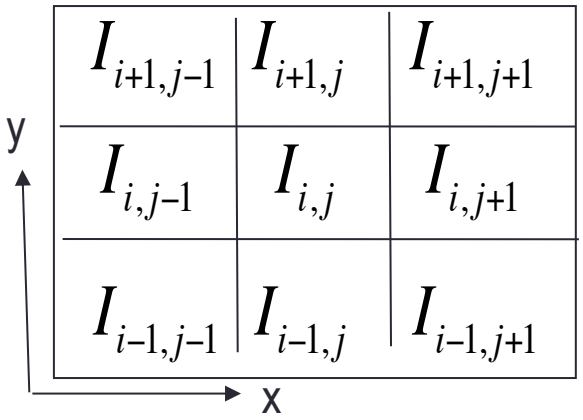
Given that the function f to derive our image I is:

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

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



Masks



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

derivada horizontal → resalta contornos verticales ↘ y viceversa

oscuro claro

0	10	10
0	10	10
0	10	10

First derivatives: discrete operators.

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

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

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

Hacemos el promedio de las derivadas para suavizar.

What happens when there is noise in the image?

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

y

x

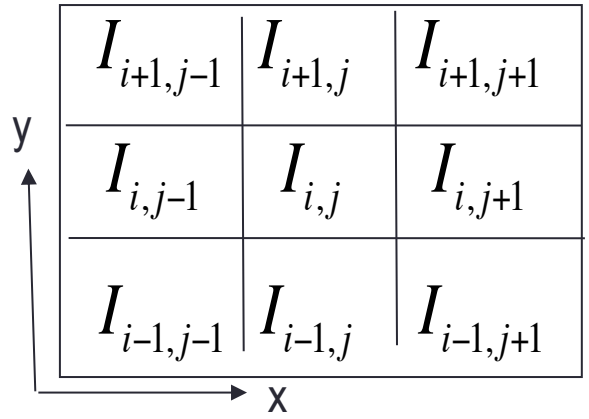
0	0	0
0	25	0
0	0	0

valor muy diferente, seguramente sea ruido y no lo queremos.

First derivatives: discrete operators.

How to be less sensitive to noise?

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



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:

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

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

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

y

x

What would be the convolution mask to compute the derivatives?

misma máscara que antes pero mejorada. Centrada en el pixel y haciendo el promedio con los pixeles vecinos.

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

Valores ponderados para dar más prioridad al pixel.

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

funciona mejor porque da más prioridad al pixel y menos al vecino.

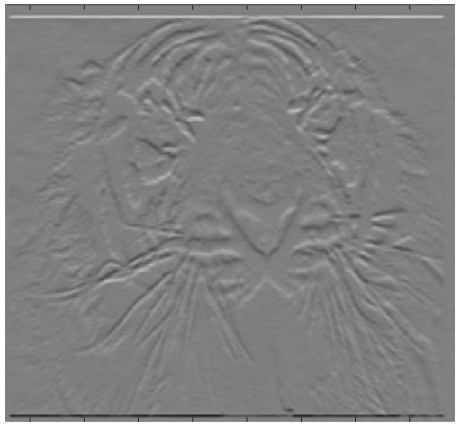
Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

→ resalta contornos horizontales

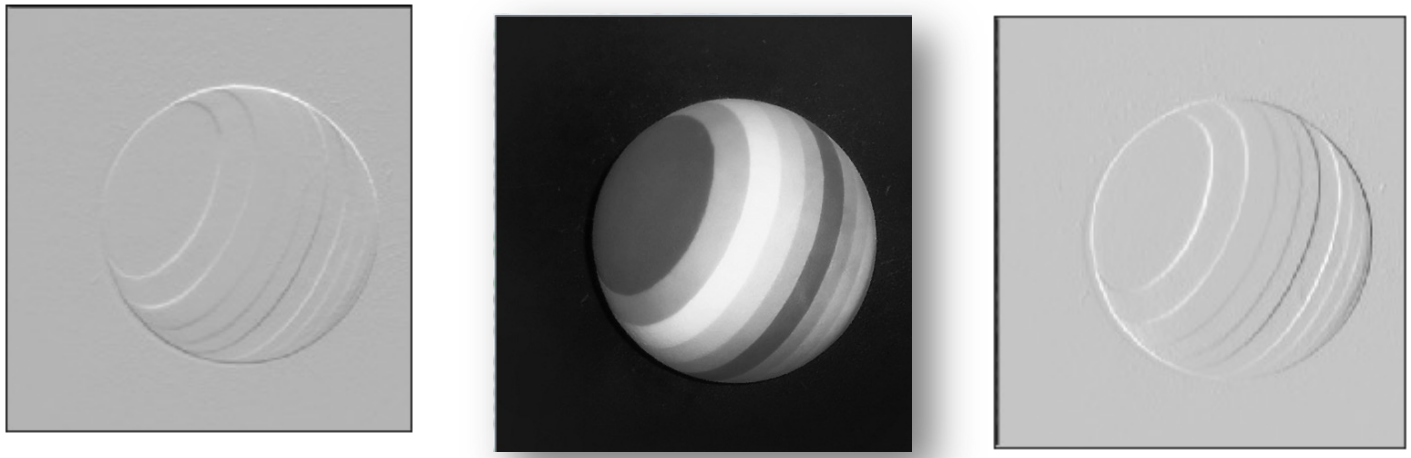
What is the difference between the Prewitt and Sobel operators?

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

Which do you expect to be better?



Assorted finite difference filters



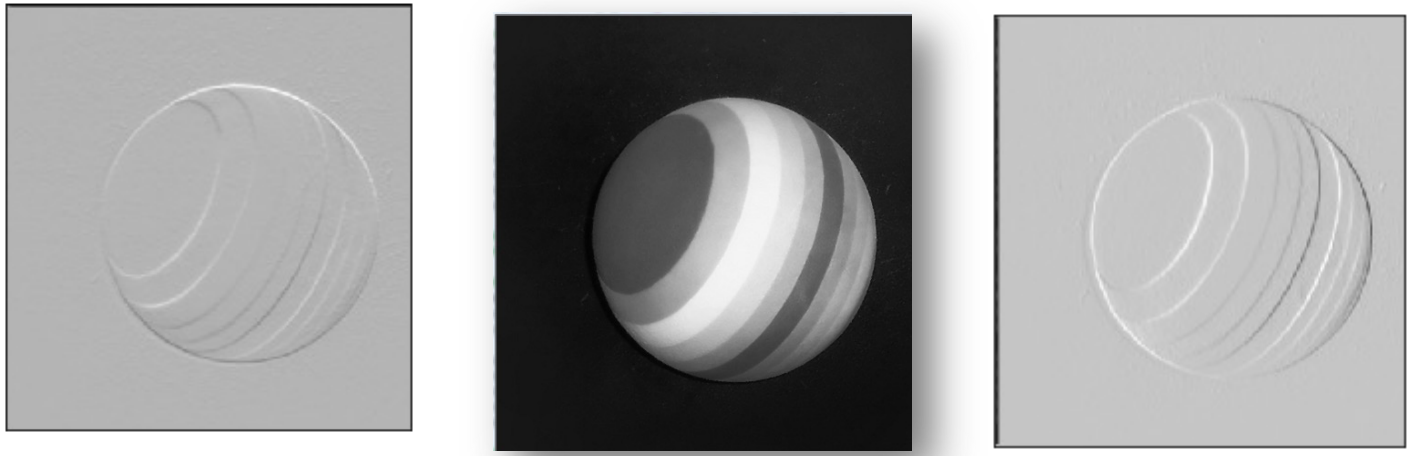
Determine to which Sobel masks (wrt x and y) do these images correspond to?!

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)?

ya está implementado en .sobel.

la combinación horizontal, vertical, diagonal...

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

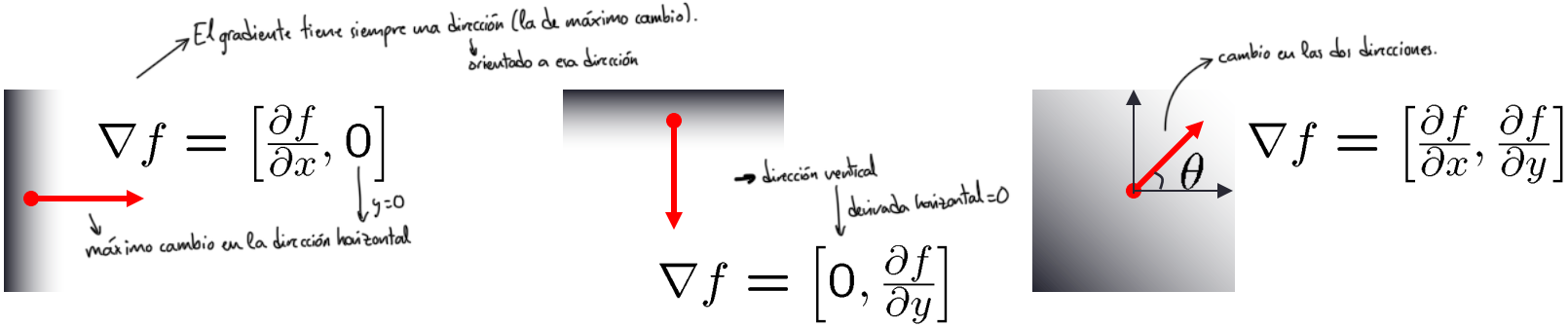
Image gradient

The gradient of an image:

vector compuesto por la derivada parcial en x y la derivada parcial en y.

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

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



Slide credit Steve Seitz

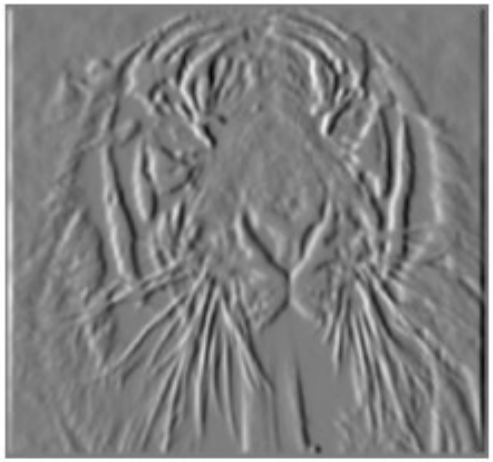
Image gradient

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

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

es un vector

mayor contraste → mayor magnitud de gradiente



Slide credit Steve Seitz

Image gradient

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

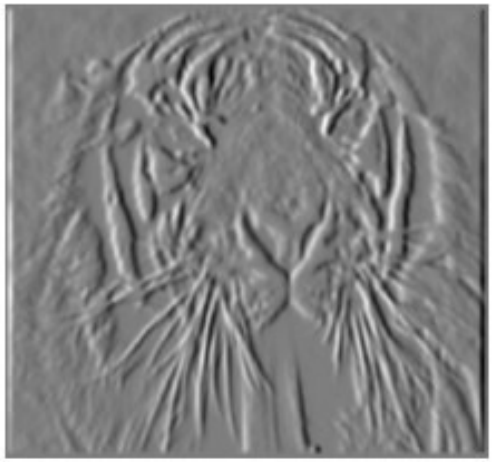
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

arctan

división

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

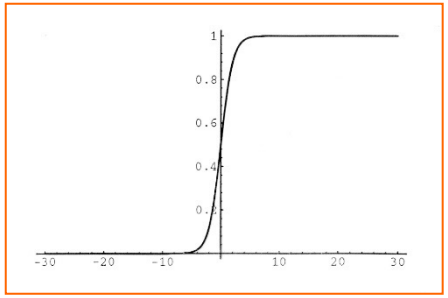
Si aumentamos el contraste ⇒ la magnitud del gradiente será mayor
Si hacemos las zonas claras, más claras; y las zonas oscuras más oscuras no cambia la dirección del gradiente. } cambiamos magnitud pero no dirección.

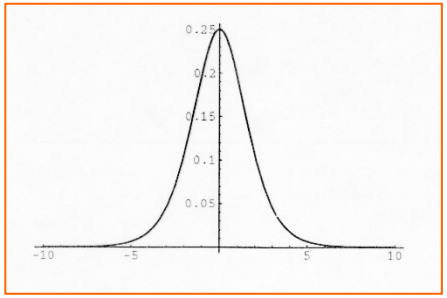


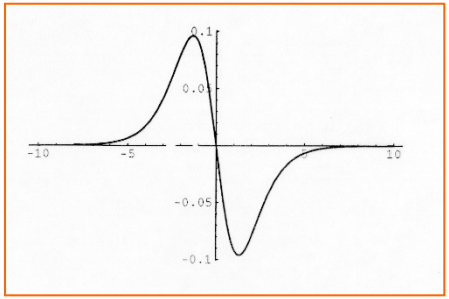
Slide credit Steve Seitz

Discrete operators: second derivatives

What do you expect about the second derivatives?





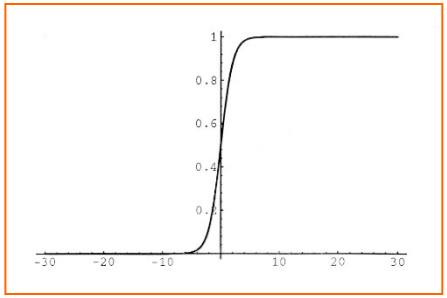
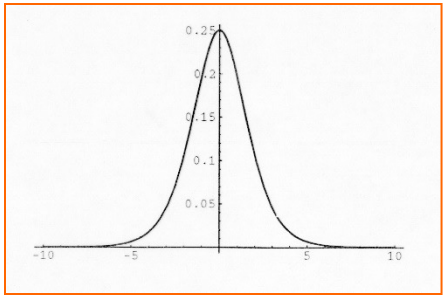
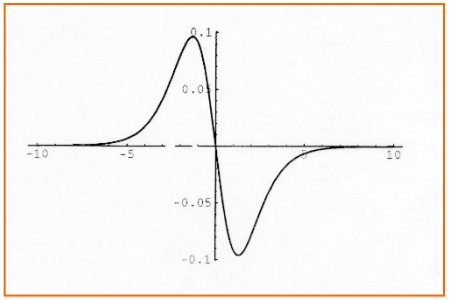


$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

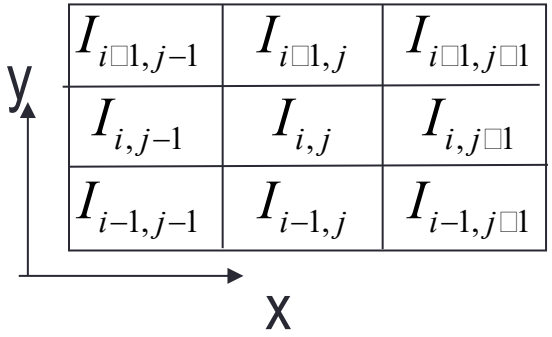
Given an edge, the first derivative has an extreme, and the second one has a zero-crossing in the edge and extremes before and after the edge.

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.



The second derivative is the derivative of the first derivative:

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

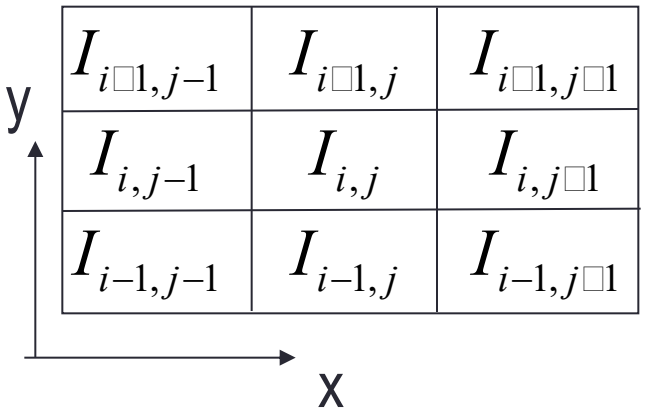
$$\frac{\partial^2 I}{\partial x^2} = (I_{i,j-1} - 2I_{i,j} + I_{i,j+1}) \rightarrow$$

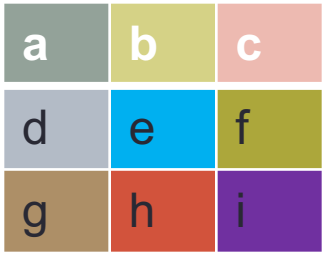
1 -2 1

Discrete operators: Laplacian

Let's define:

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





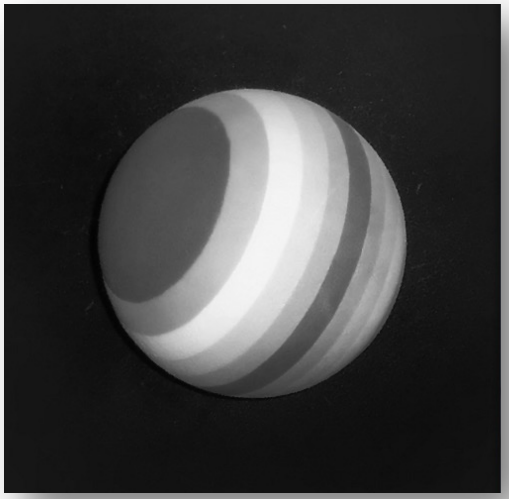
Laplacian mask:

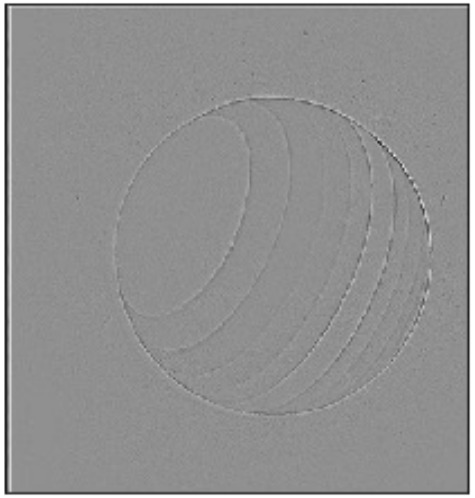
$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Discrete operators: Laplacian

Laplacian mask:

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





Skimage:

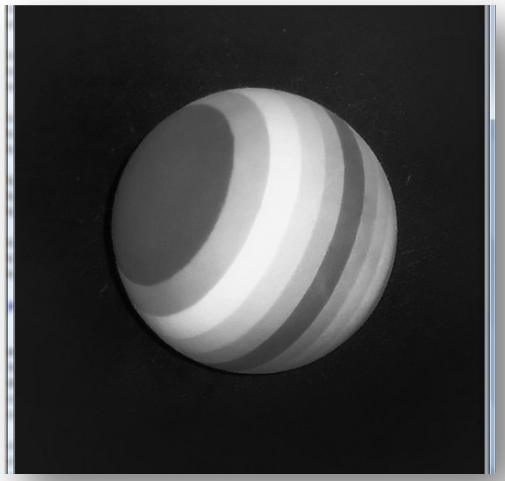
```
>> mask=numpy.array([[0,1,0],[1,-4,1],[0,1,0]], dtype='float')

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

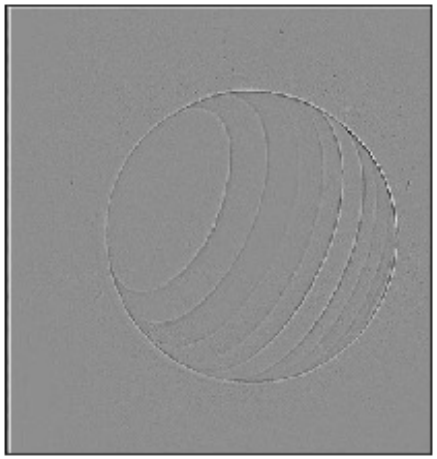
Discrete operators: Laplacian

Including the diagonal neighbours:

Laplacian:

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



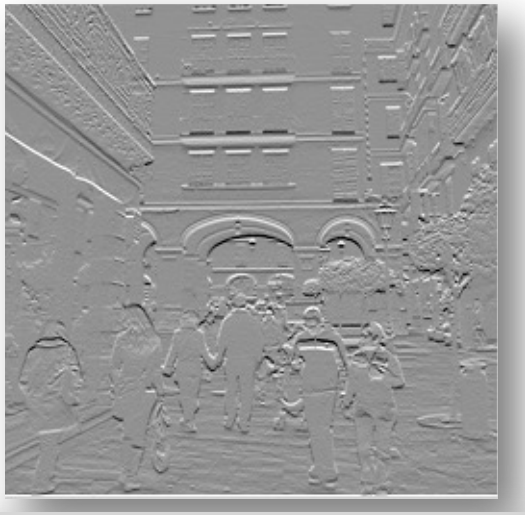


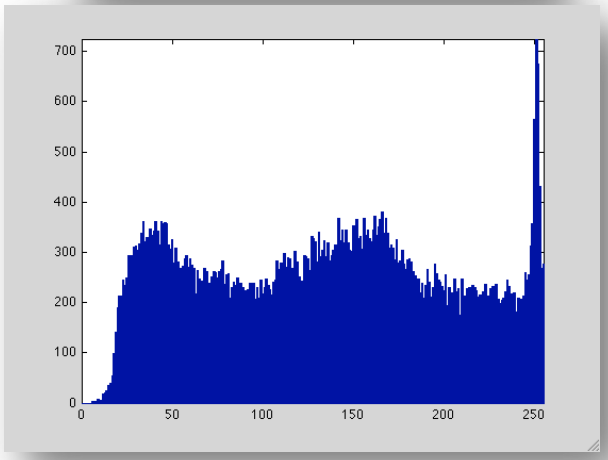
Mask properties

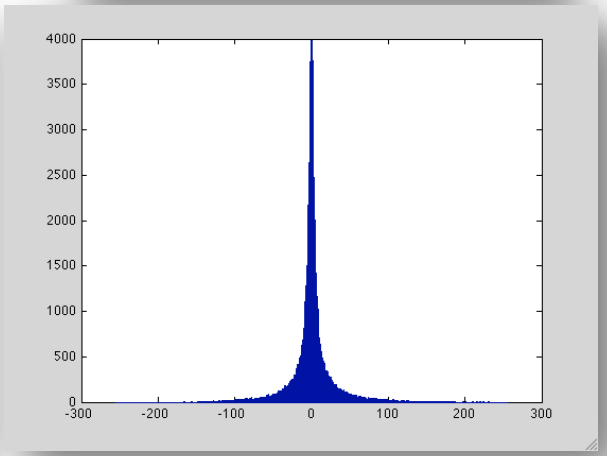
- Test on smoothing
  - Values should be _____
  - Sum to ___→ constant regions same as input
  - Amount of smoothing _____ to mask size
  - Remove "_____-frequency" components; "_____-pass" filter

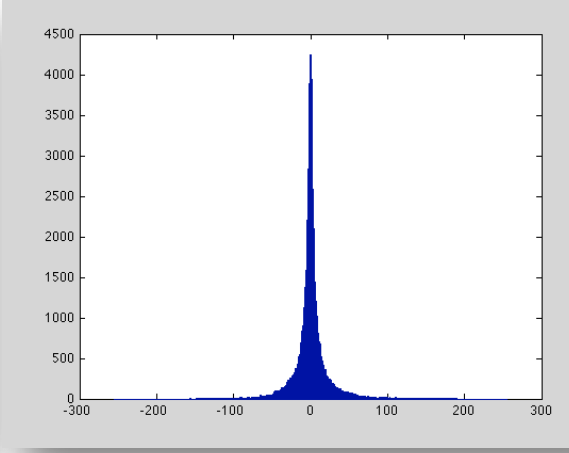
- Test on derivatives
  - _____ signs used to get high response in regions of high contrast
  - Sum to ___ → no response in constant regions
  - _____ value at points of high contrast

Explain the histogram of x and y edge maps









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

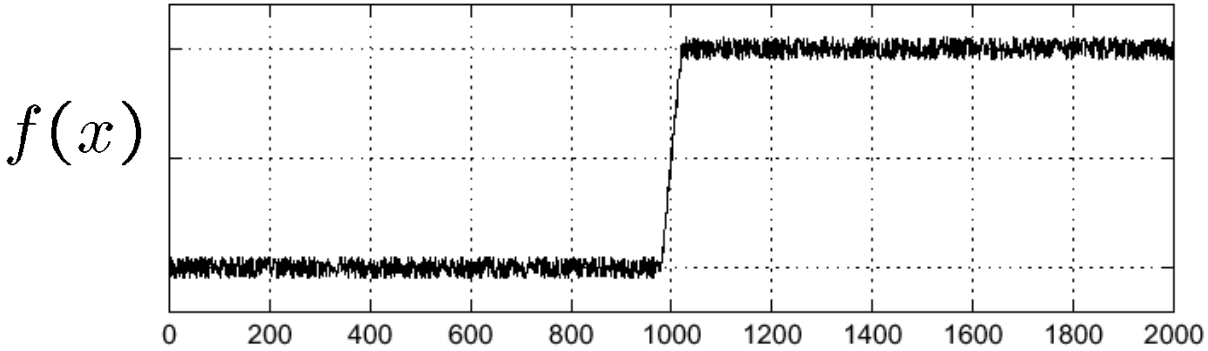
Today

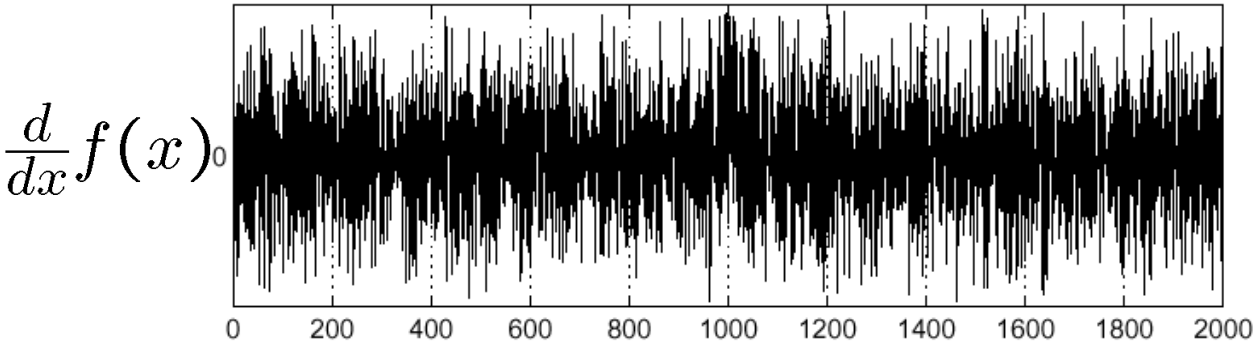
- What is an image derivative and its relation to the edges
  - How to implement different edge detectors and what is their difference?
  - First and second derivatives, image gradient and Laplacian
- How to get image derivatives with a Gaussian
- Canny edge detector
- Application: hybrid images

Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal





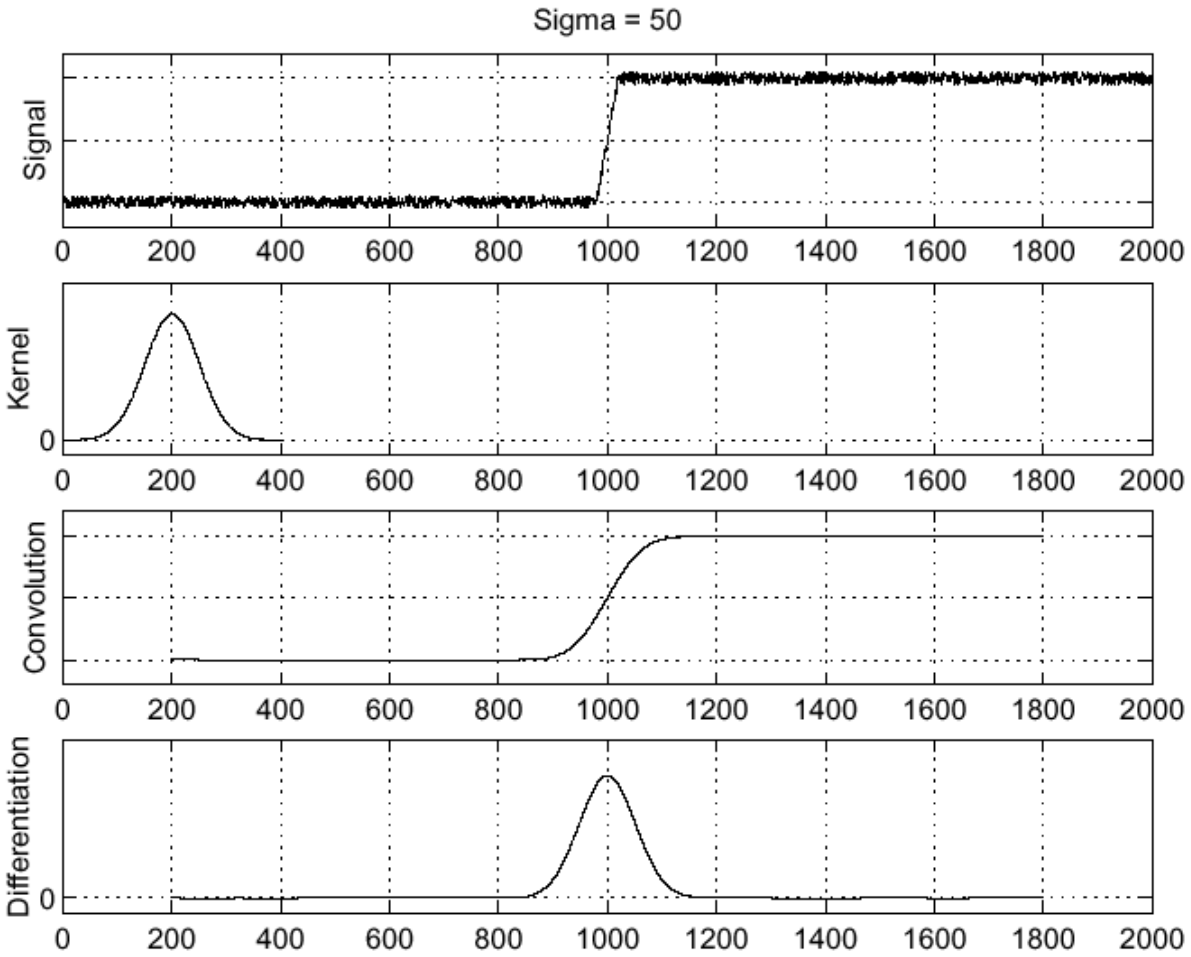
Where is the edge?

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

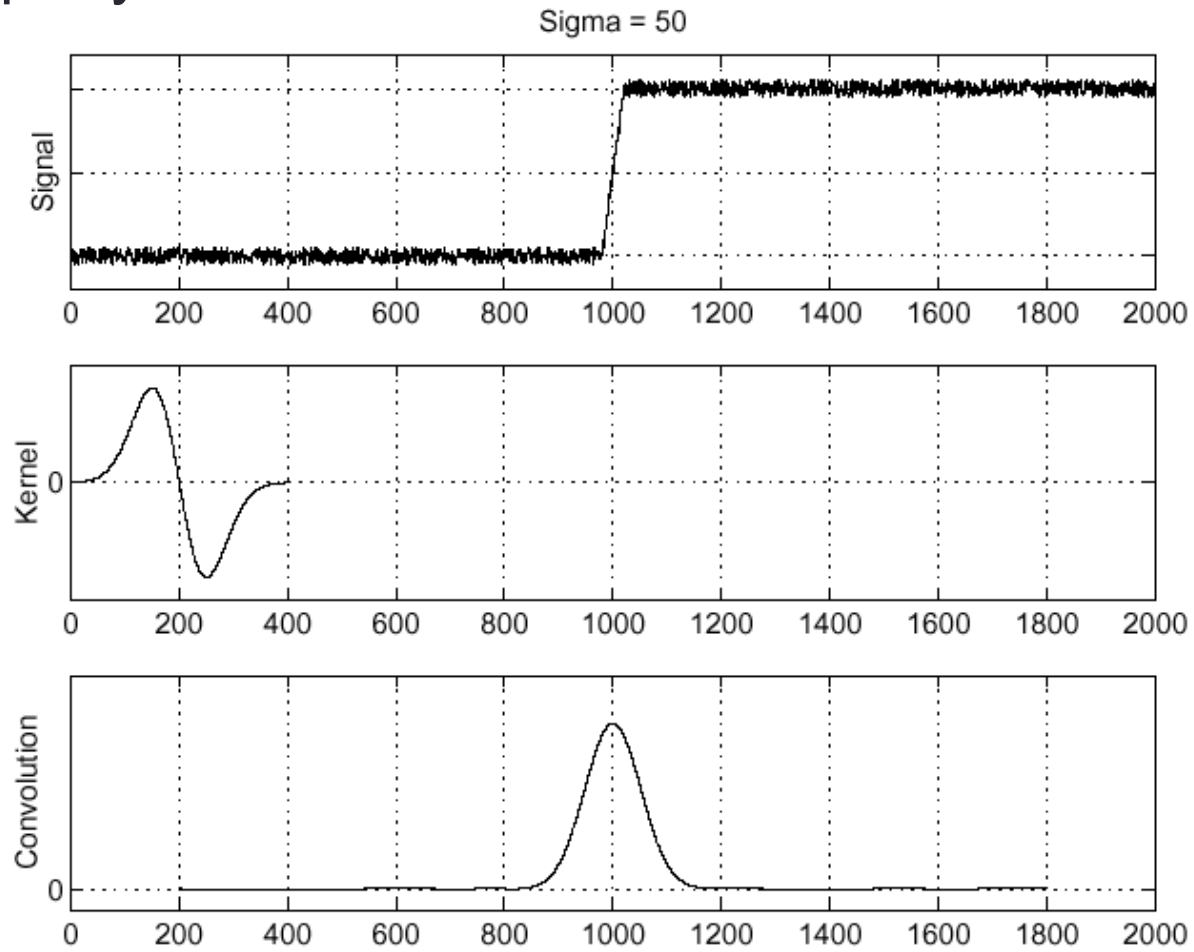
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

Differentiation property of convolution.

f

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

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



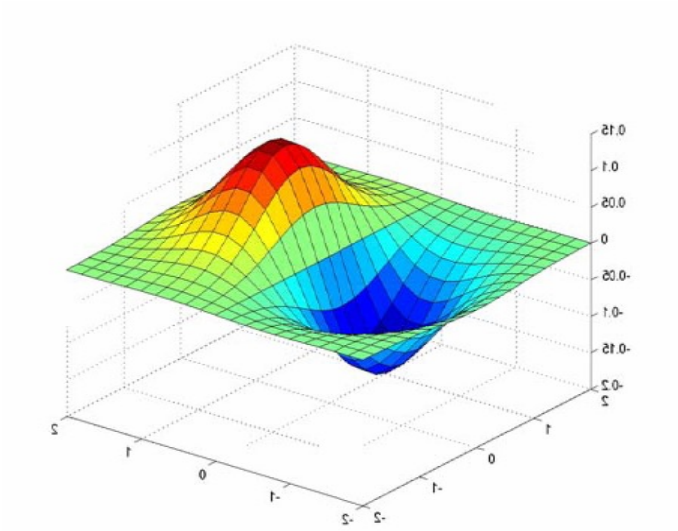
Slide credit Steve Seitz

Derivative of Gaussian filters

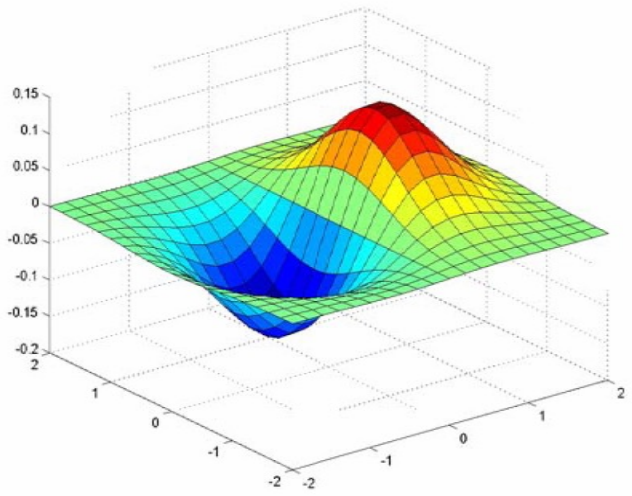
$$(I \otimes g) \otimes h \quad \square \quad I \otimes (g \otimes h)$$

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

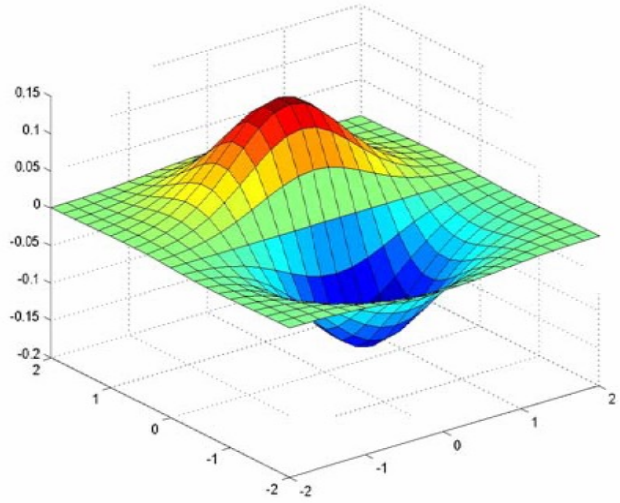
$$\begin{bmatrix} \end{bmatrix} \otimes \square 1 \quad -1 \quad \begin{bmatrix} \end{bmatrix}$$



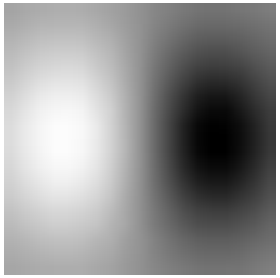
Derivative of Gaussian filters

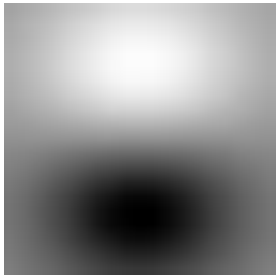


***x*-direction**



***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!

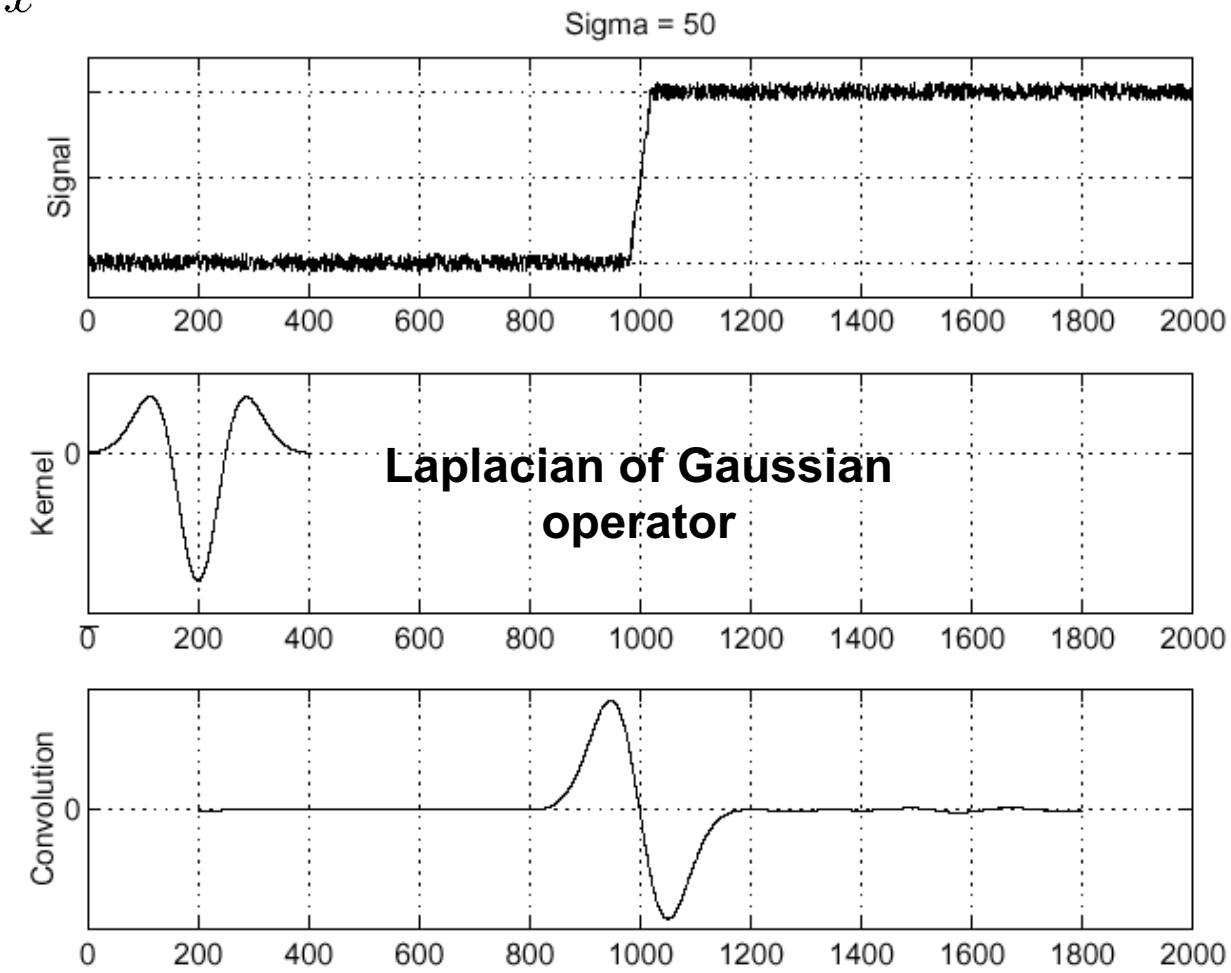
Laplacian of Gaussian

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

f

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

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

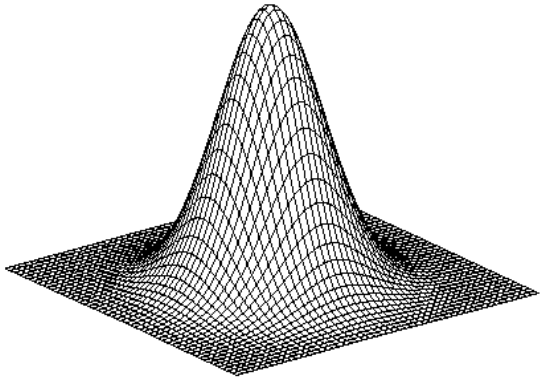


Where is the edge?

Zero-crossings of bottom graph

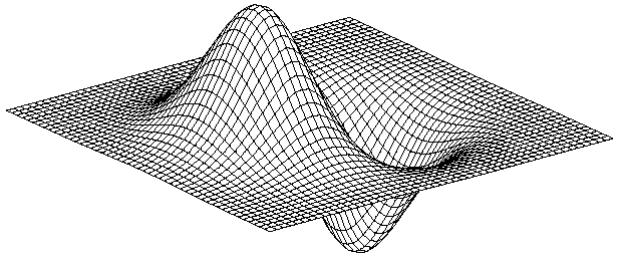
Slide credit: Steve Seitz

2D edge detection filters



Gaussian

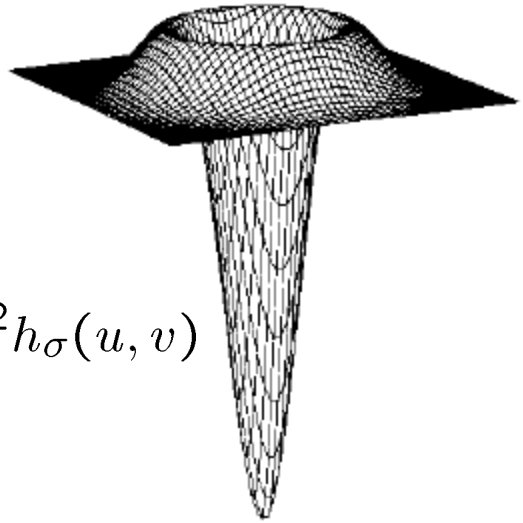
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Derivative of Gaussian

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

Laplacian of Gaussian



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

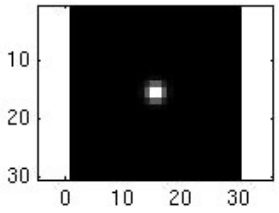
- ∇^2 is the Laplacian operator:

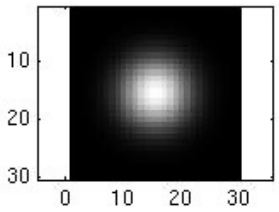
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

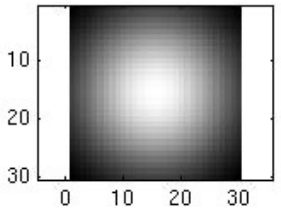
Slide credit: Steve Seitz

Smoothing with a Gaussian

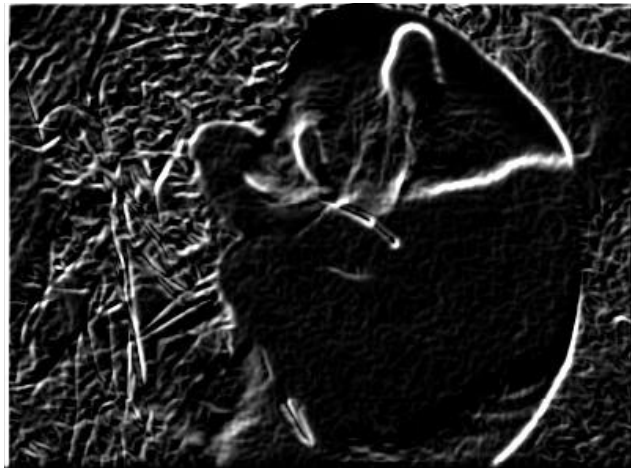
Recall: parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.







Effect of σ on derivatives



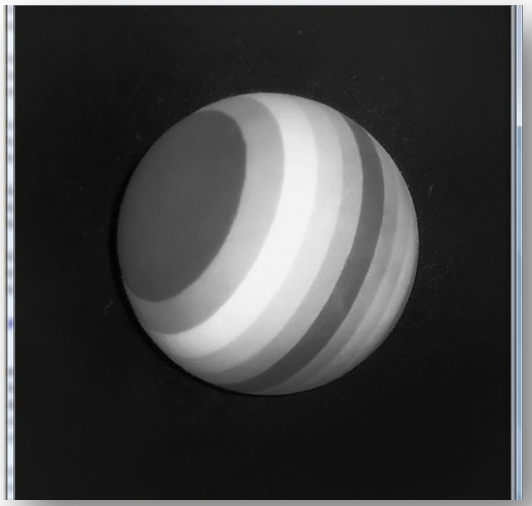
σ = 1 pixel

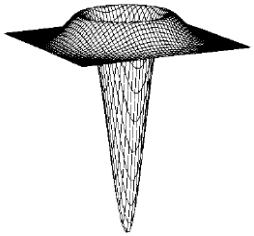
σ = 3 pixels

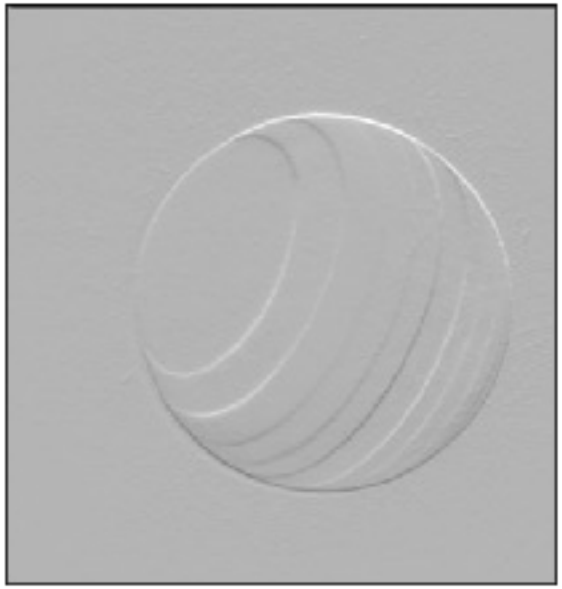
The apparent structures differ depending on Gaussian's scale parameter.

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

Laplacian





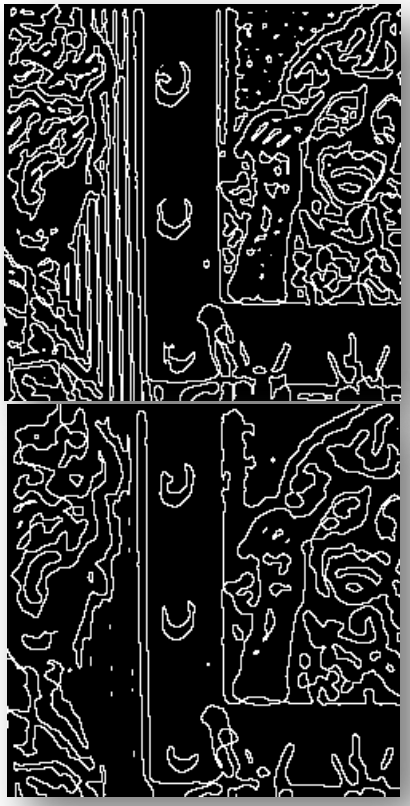


$$\nabla^2 h(x,y) = \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) h(x,y)$$

Skimage:

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

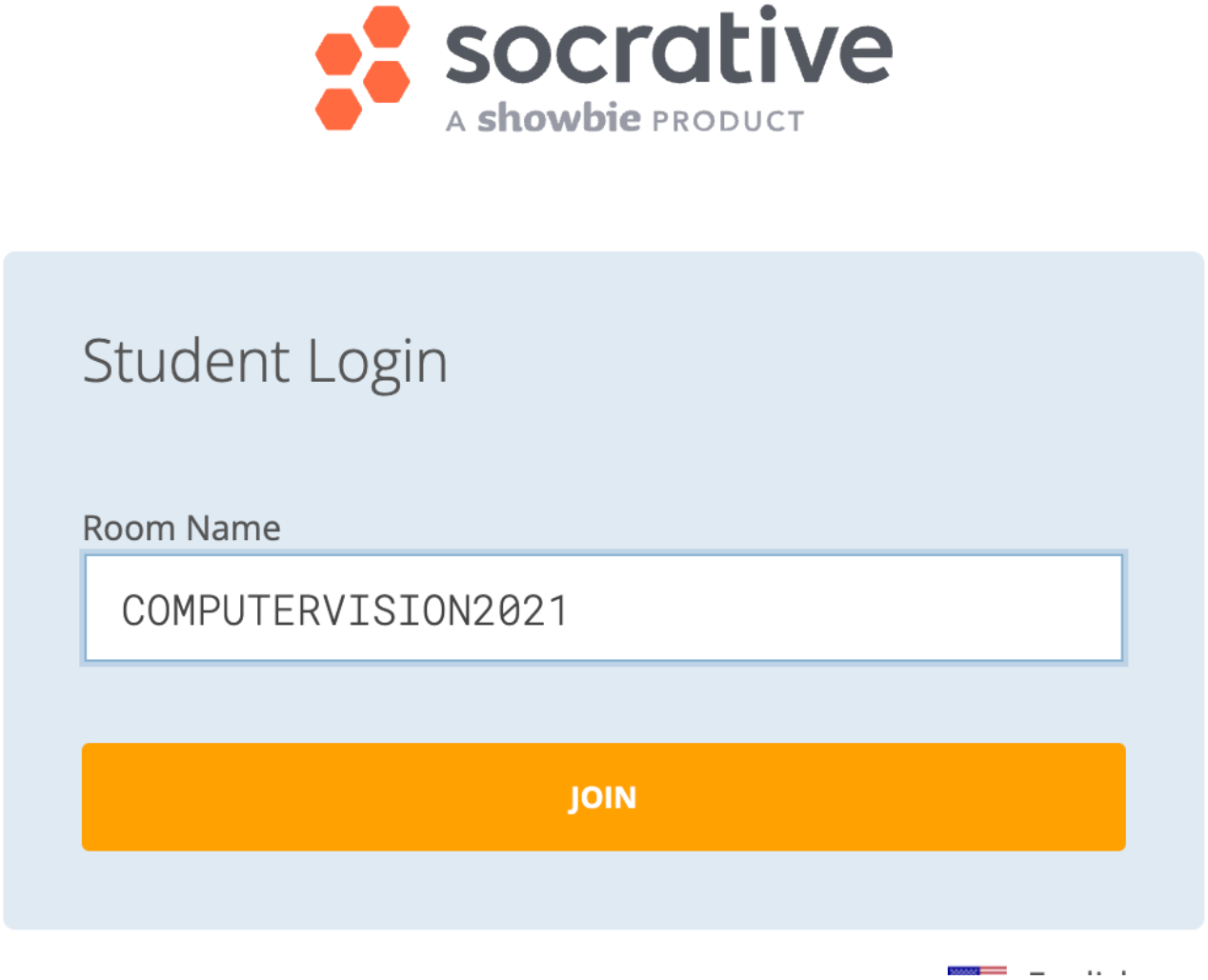
Laplacian and zero-crossings



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).



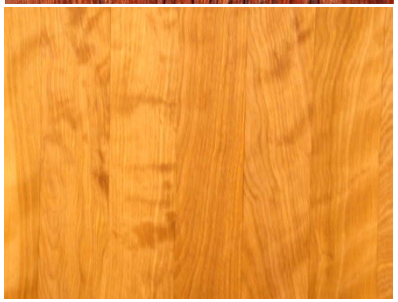
b.socrative,com/login/student

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.

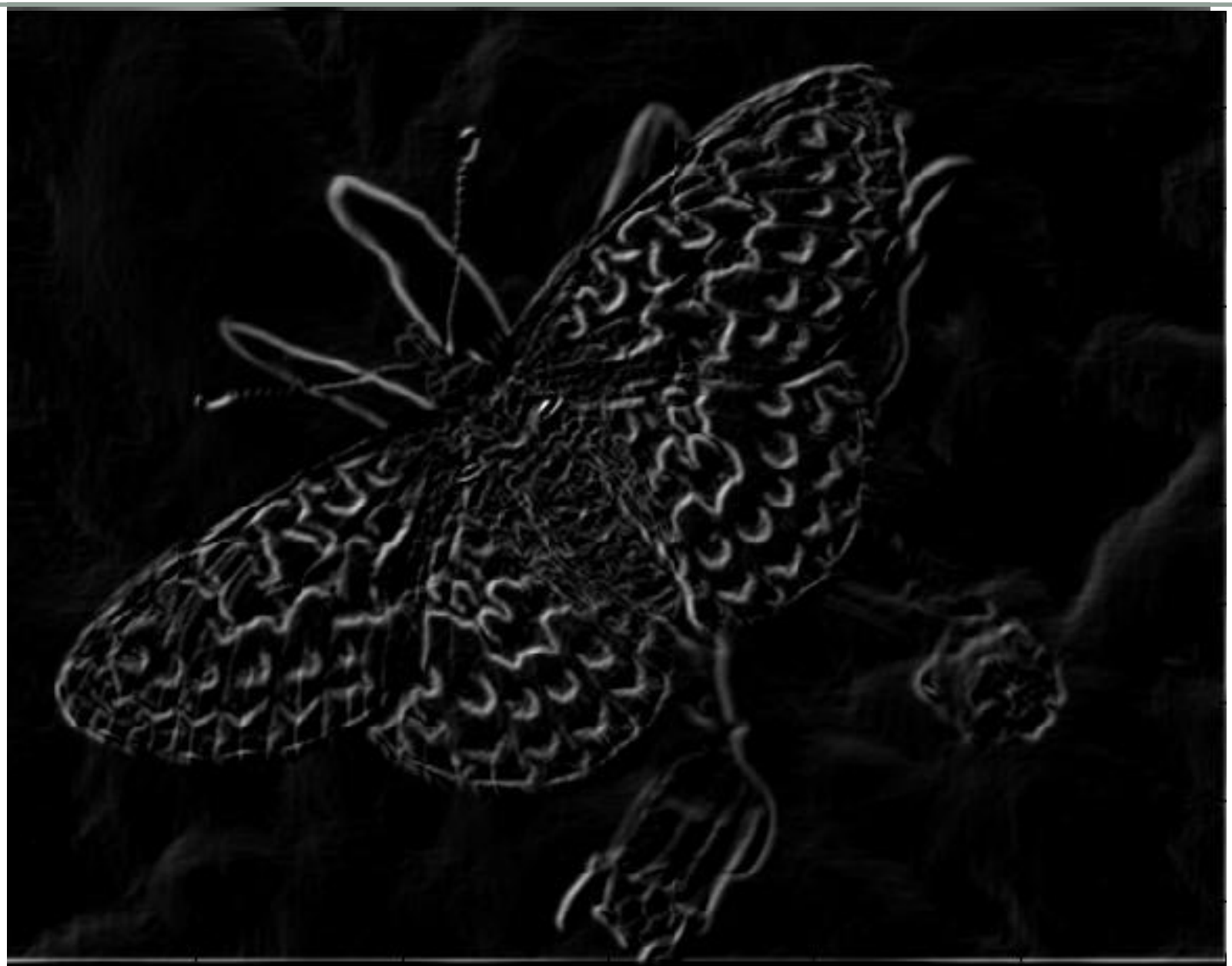


We need previous knowledge about the right scale.

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

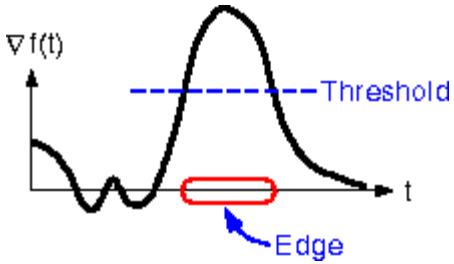
The Canny edge detector

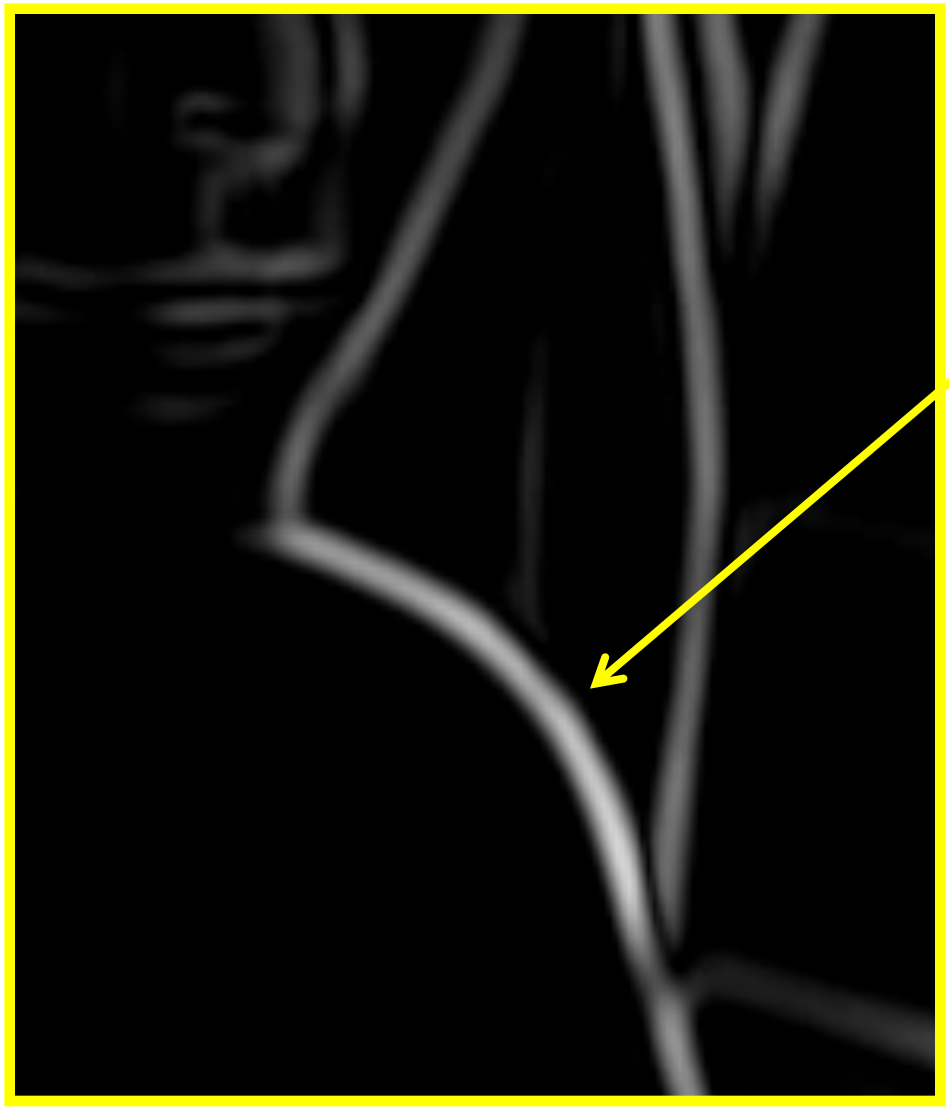
norm of the gradient

The Canny edge detector

thresholding

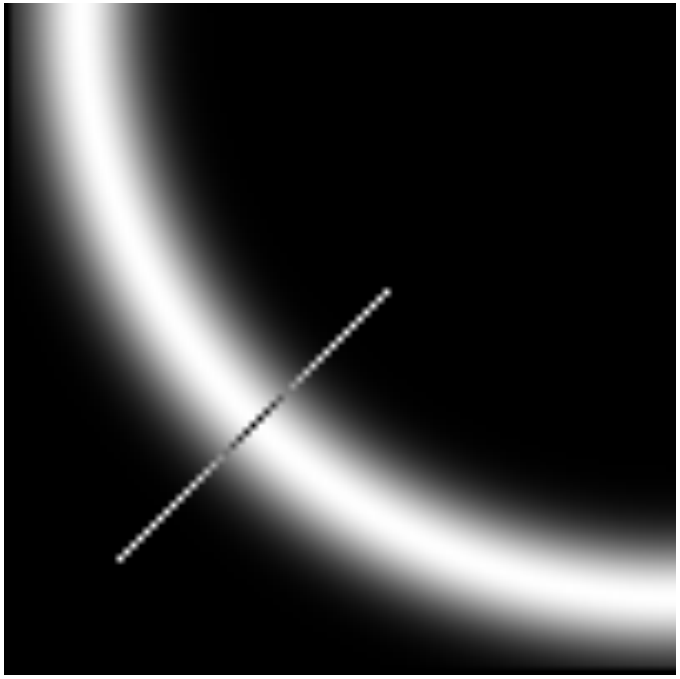
The Canny edge detector





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

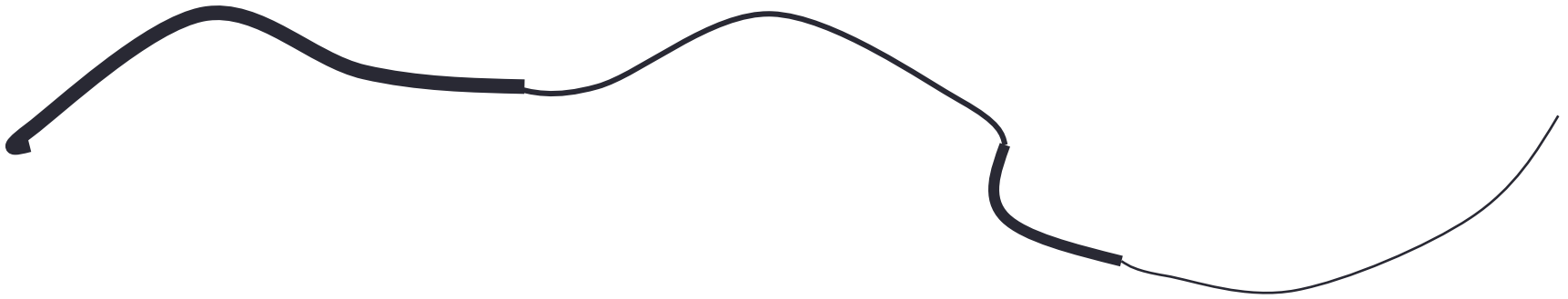
The Canny edge detector

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

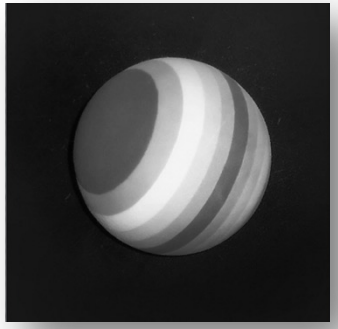
1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. **Non-maximum suppression**:
   - Thin wide “ridges” down to single pixel width
4. **Linking and thresholding** (**hysteresis**):
   - Define two thresholds: low and high
   - Use the high threshold to start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

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

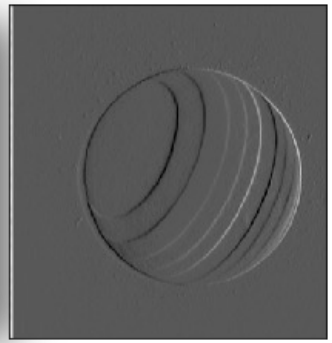
Source: D. Lowe, L. Fei-Fei

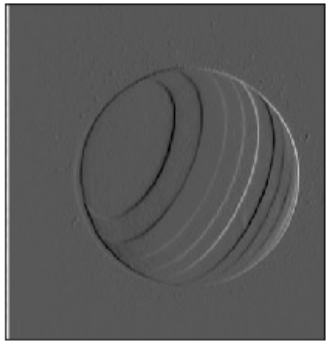
Exercices (comparison)

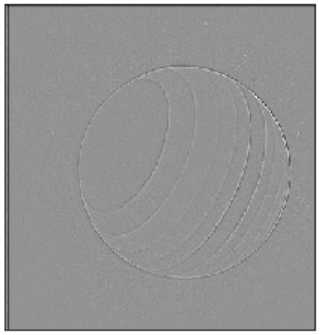
- Determine the operator applied:

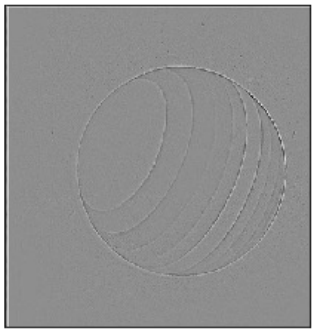


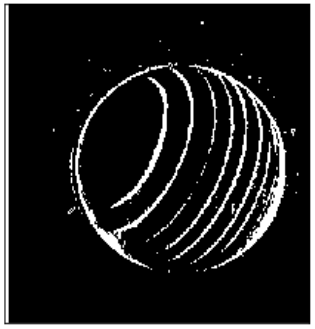
Original image

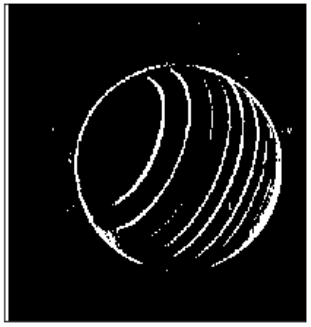


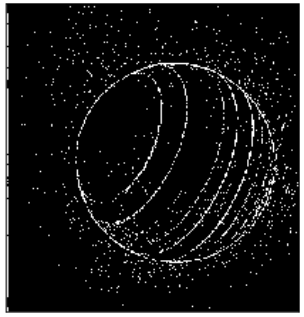


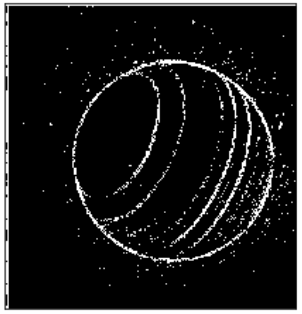












Sobel with a mask

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')

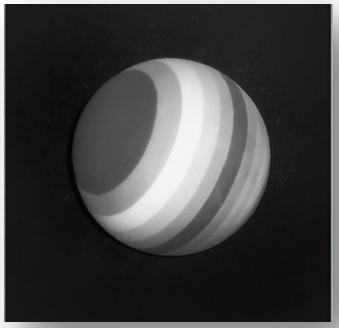
Laplacian with a mask

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

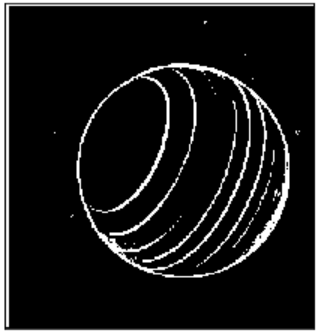
im_Lapl=convolve(im,mask)

Exercices (comparison)

- Determine the operator applied:



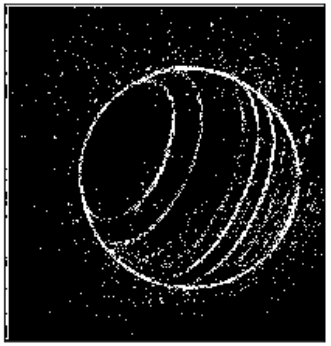
Original image



Sobel

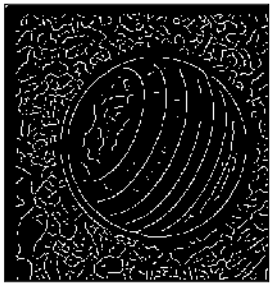
im2=filters.sobel(im)

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



Laplacian of a Gaussian

im2=filters.laplace(im)

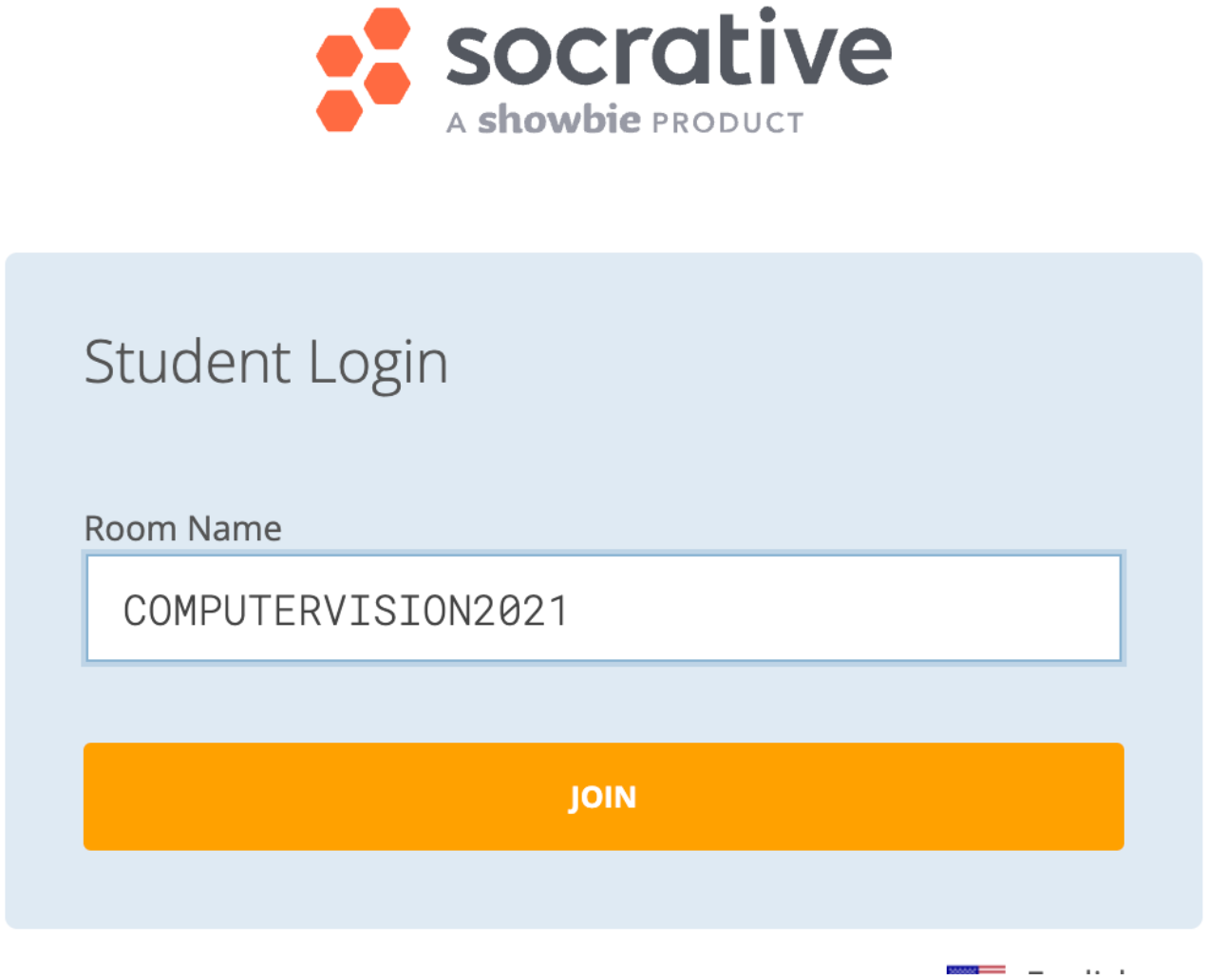


Canny sigma=4

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

Canny sigma=6.5

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



b.socrative,com/login/student

Low-level edges vs. perceived contours

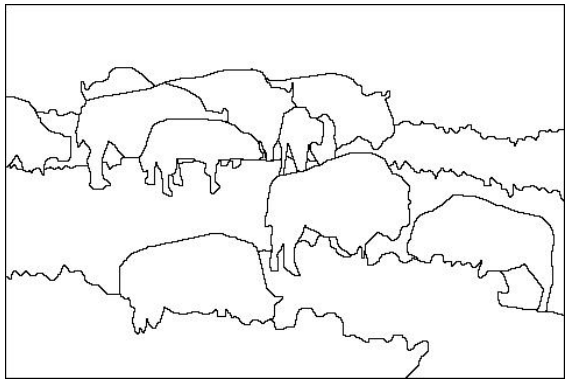
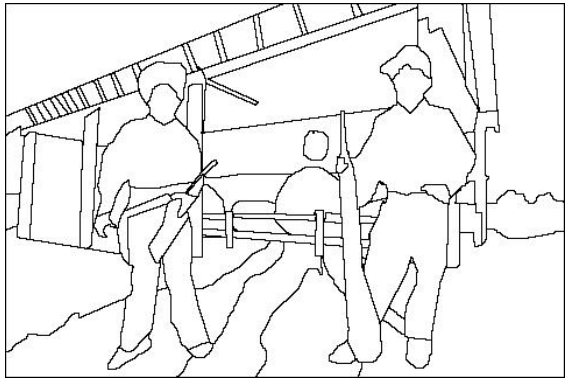
Background

Texture

Shadows

Is Canny edge detector distinguishing these contours?

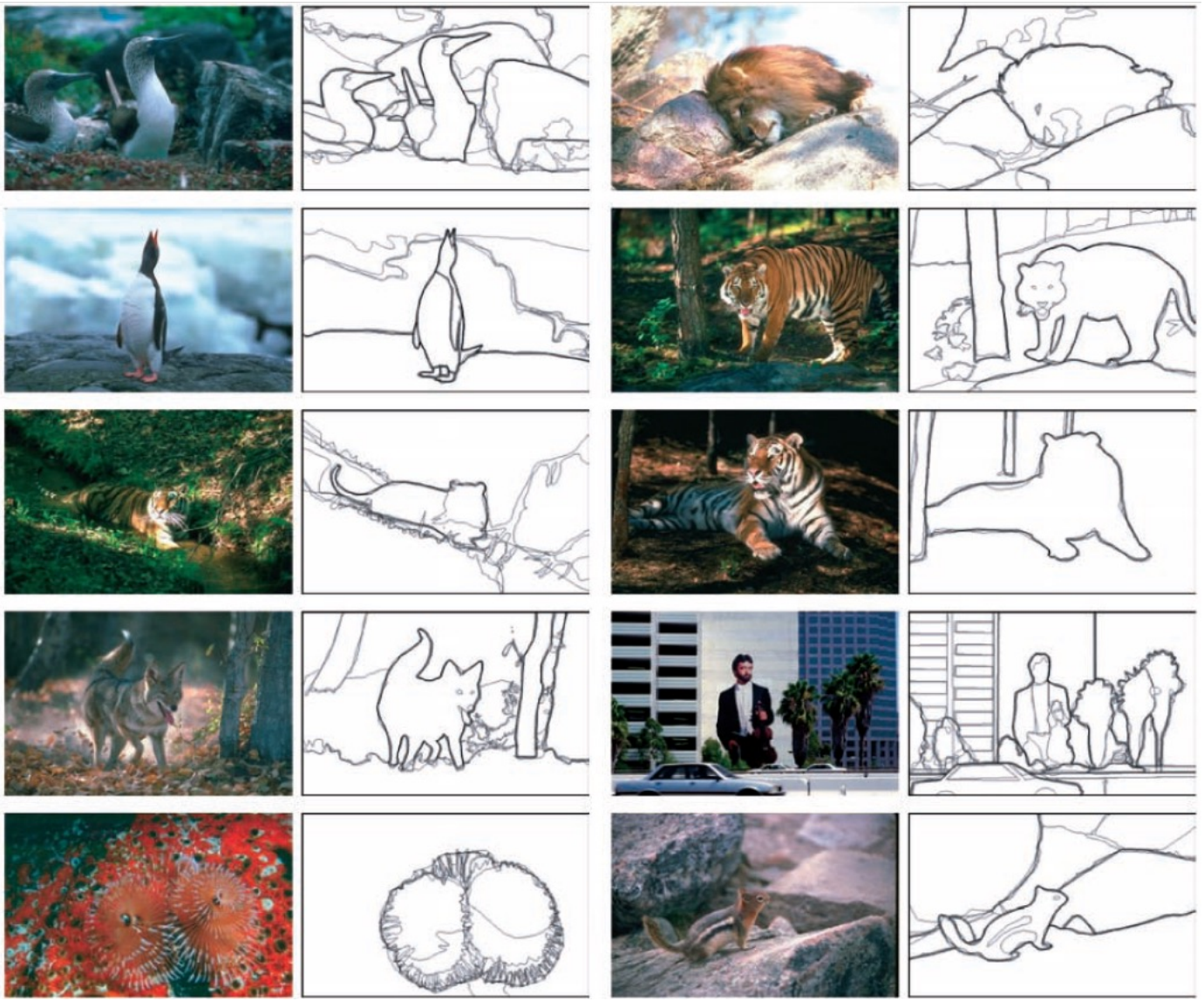
Low-level edges vs. perceived contours

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

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

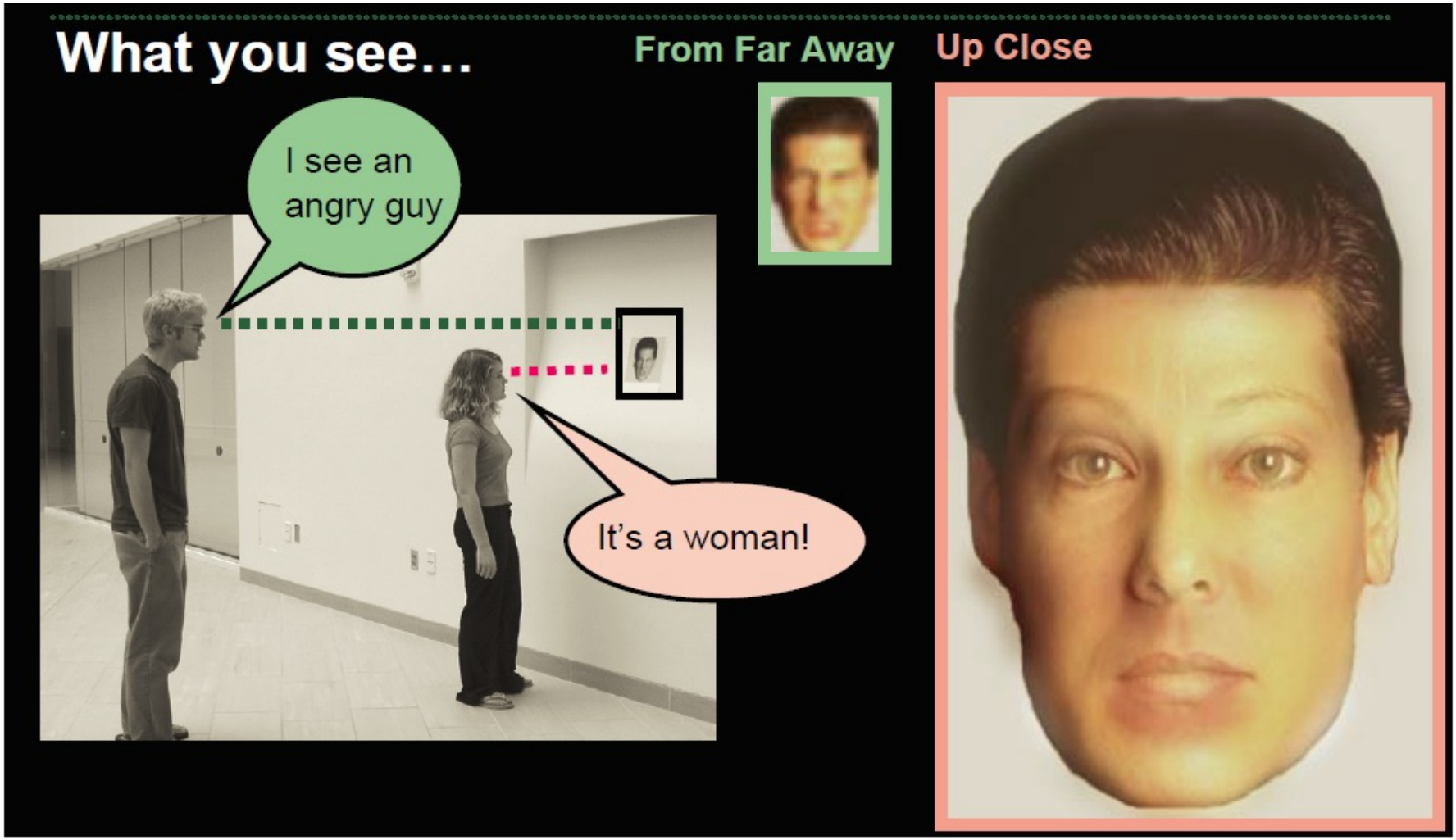
Squeeze

Qué derivada utilizarías para encontrar:

a) La linea contínua separadora

b) El final del carril?

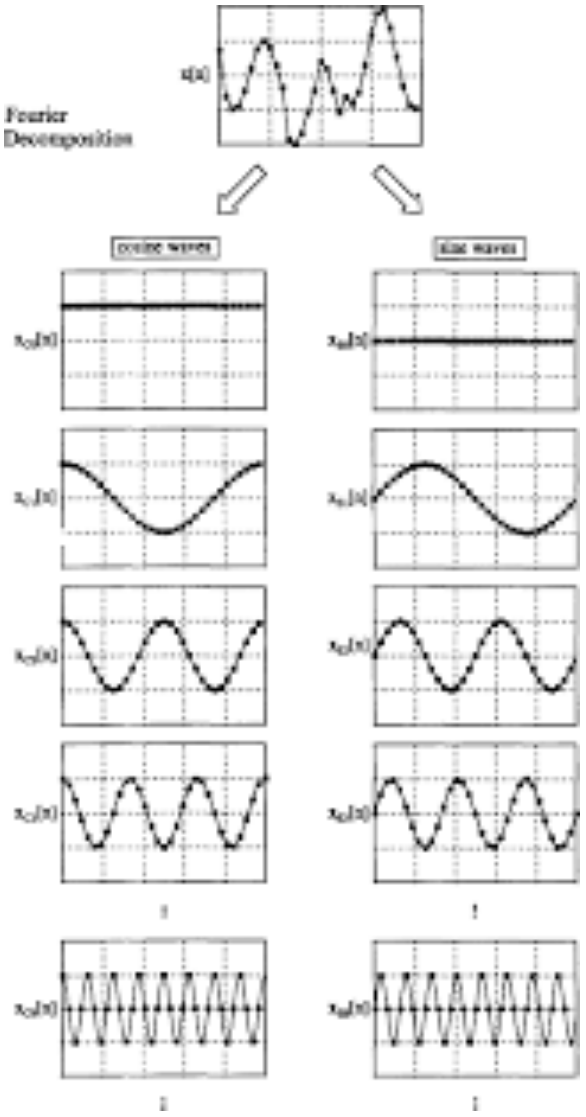
Application: Hybrid Images



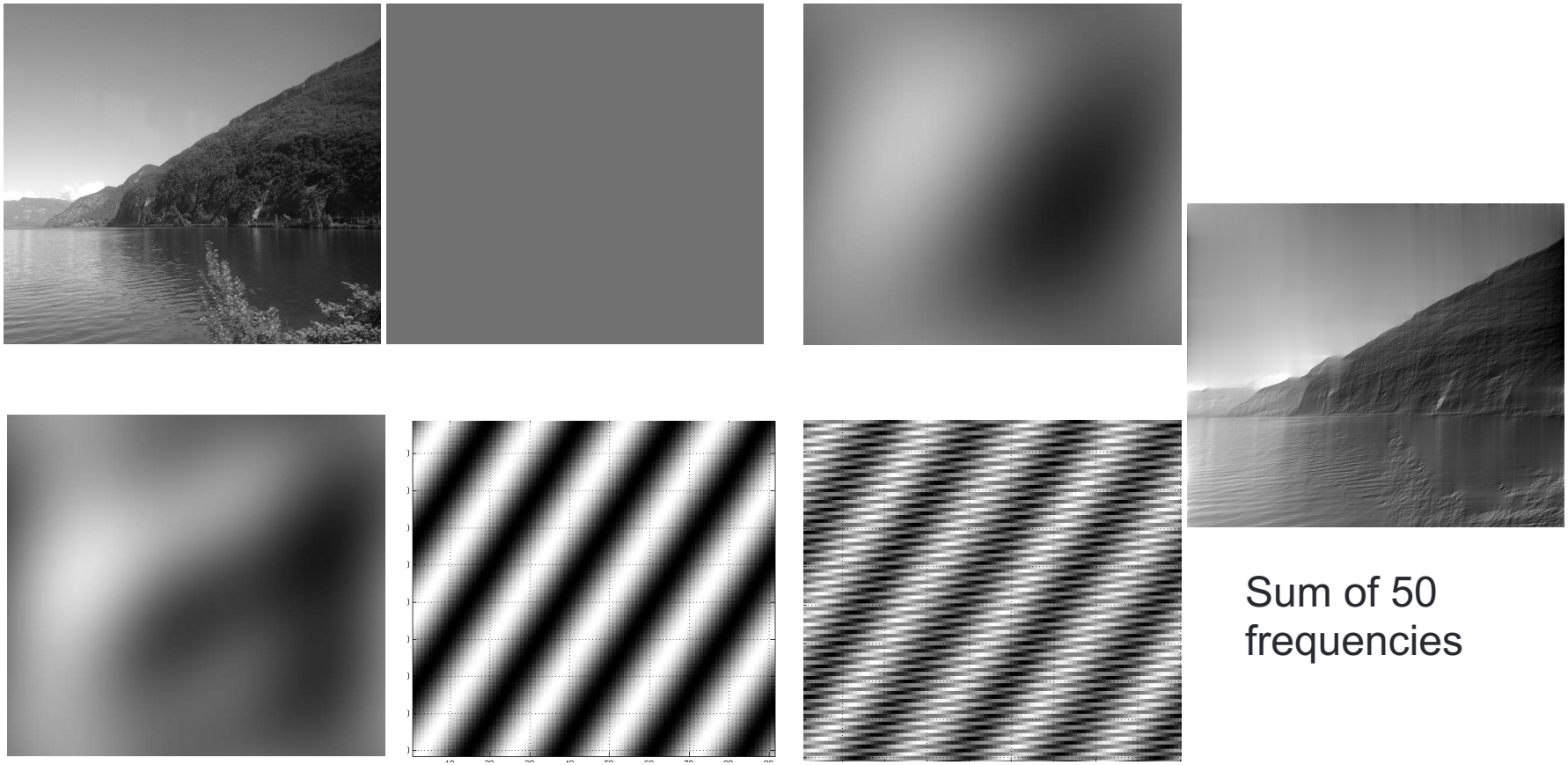
Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

Fourier decomposition

Impulse noise



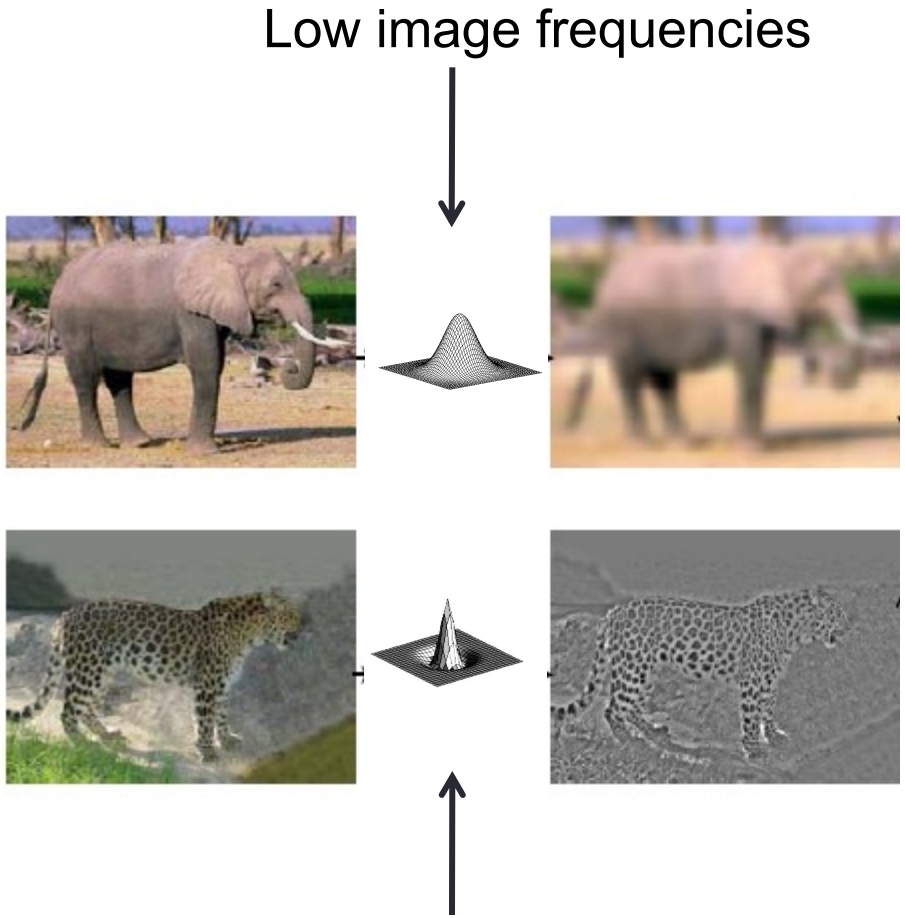
Intuition about low and high image frequencies



Sum of 50 frequencies

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

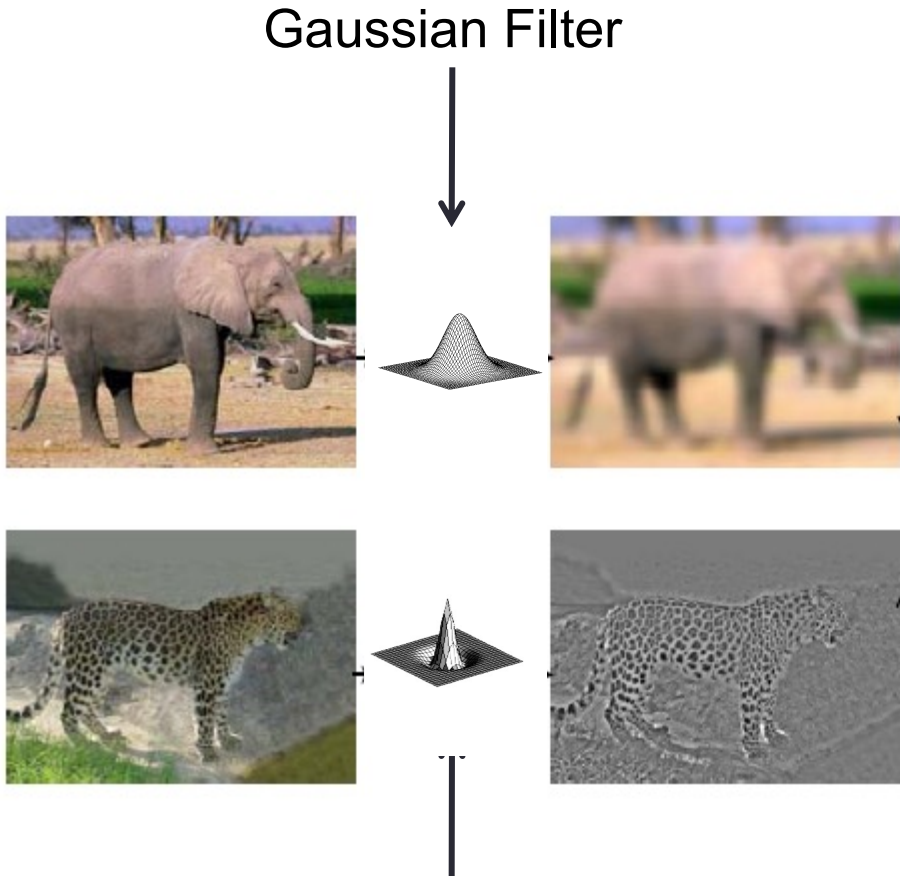
Application: Hybrid Images



How to extract them?

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

Application: Hybrid Images



Low frequencies:

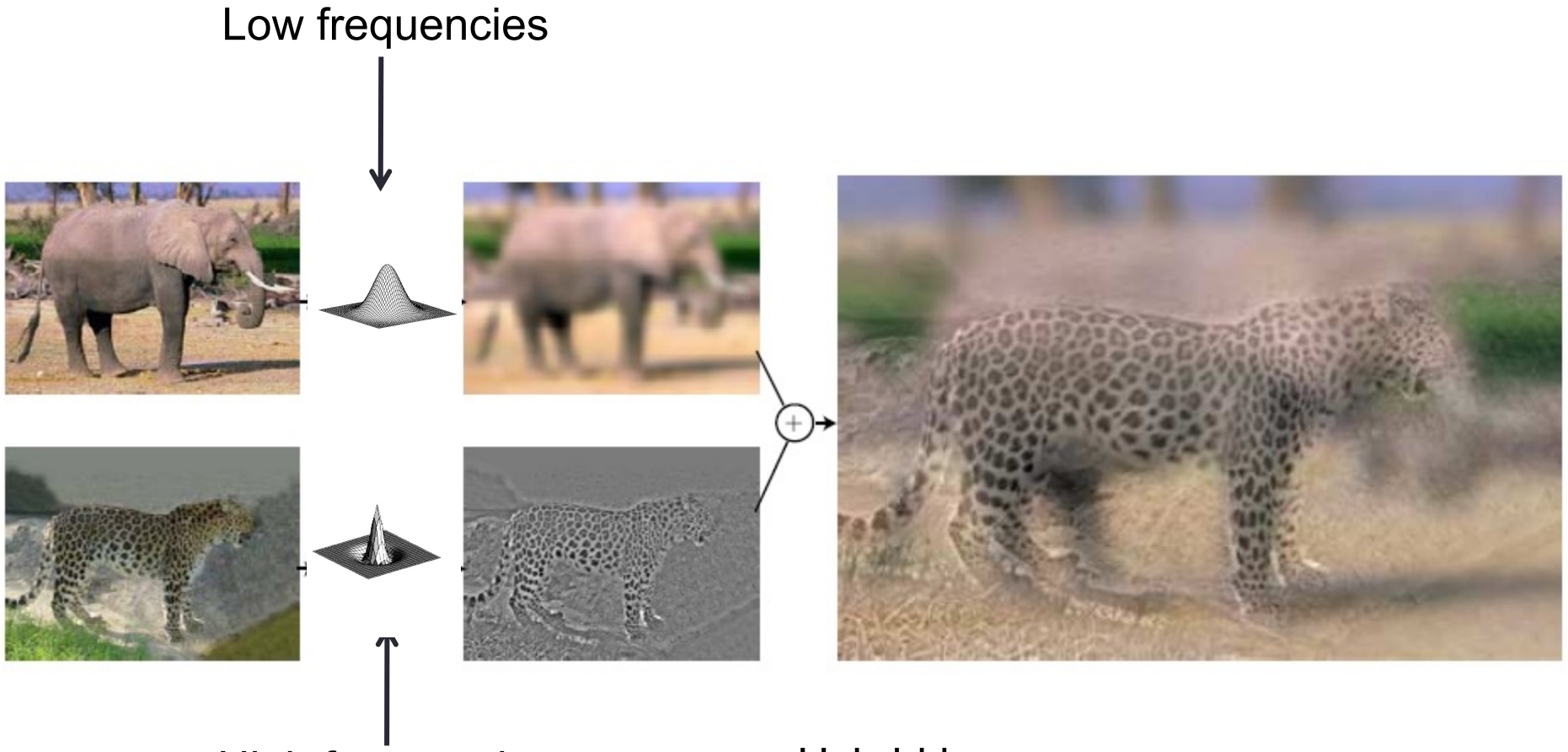
L(I1) -> Gaussian smoothing

High frequencies:

H(I2)=I2 - L(I2)

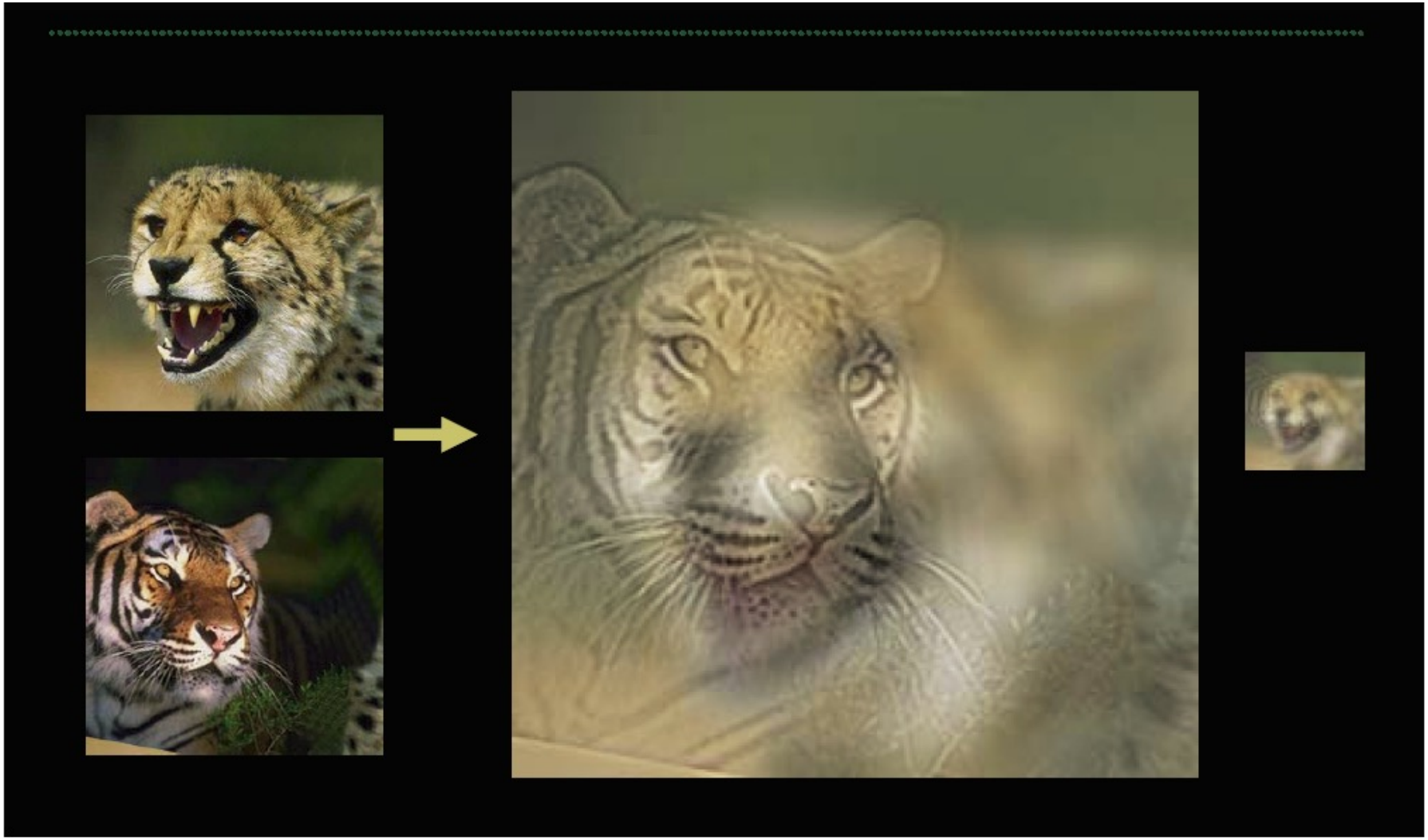
A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006

Application: Hybrid Images

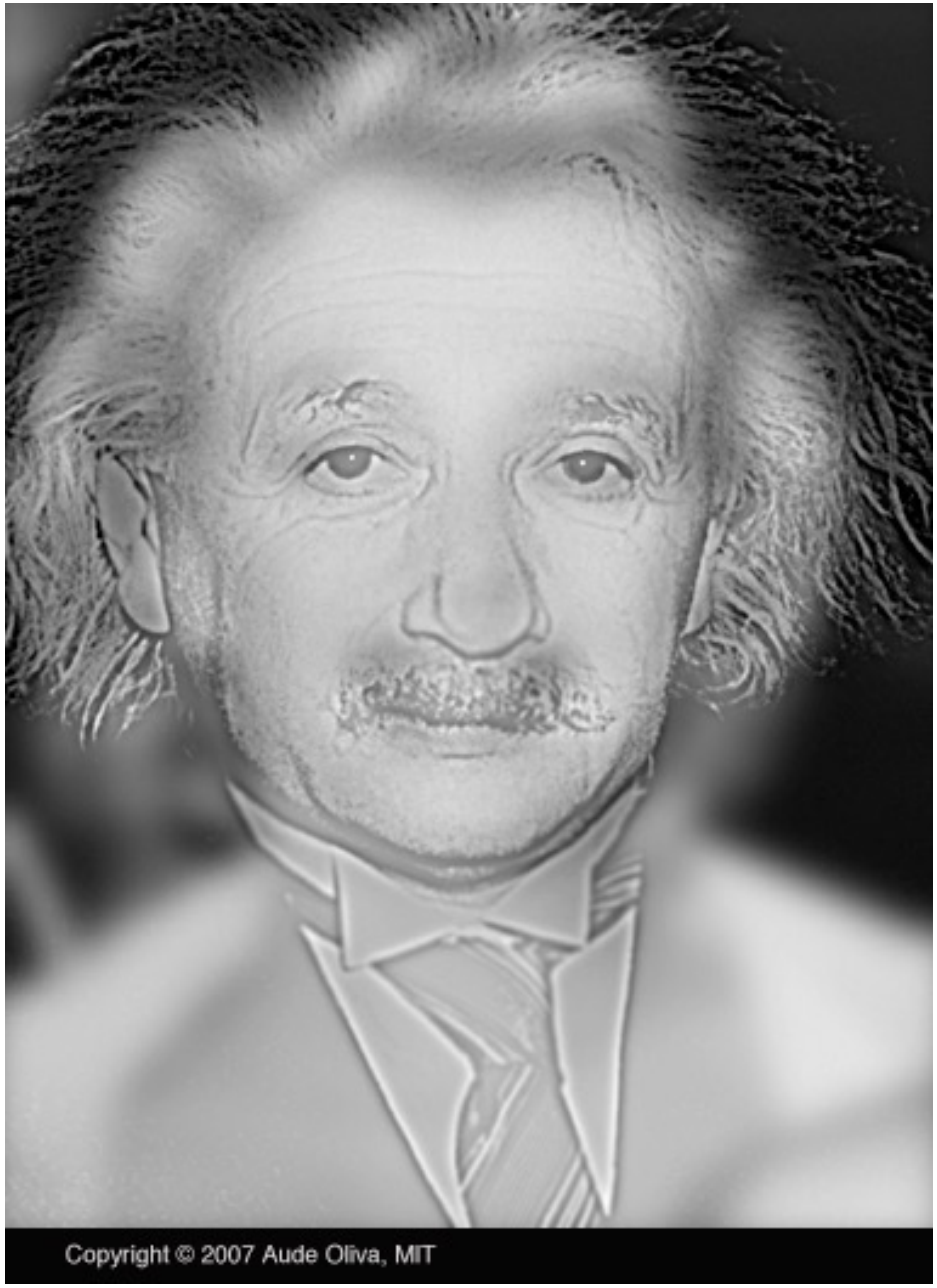


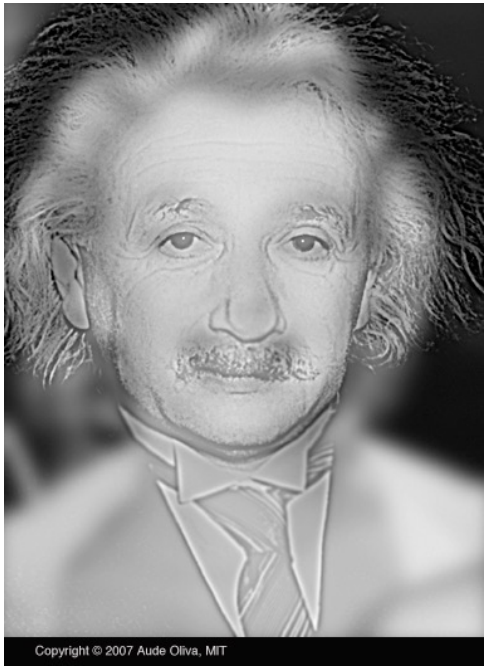
Note: In order to emphasize the effect, use different sigma parameters in the Gaussian

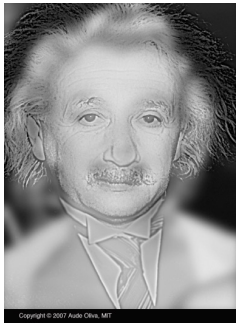
A. Oliva, A. Torralba, P.G. Schyns, *"Hybrid Images,"* SIGGRAPH 2006



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



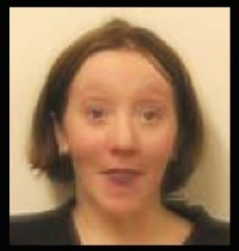




http://cvcl.mit.edu/hybrid_gallery/gallery.html

Changing expression

Sad ⟷ Surprised



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

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!