COMPUTATIONAL VISION:
Image gradients and edges

Master in Artificial Intelligence

Department of Mathematics and Computer Science

2023-2024

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, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (feature detection and matching)

Outline

- Edge definition
- Discrete operators for edge extraction
  - Classical: Prewitt and Sobel
- Effects of noise: Gaussian filter
  - Derivative of Gaussian filter
  - Laplacian of Gaussian
- Canny edge detector

Edge detection

- **Goal**: map image from 2D array of pixels to a **set of curves or line segments or contours**. Locating structural features.

- **Why?**

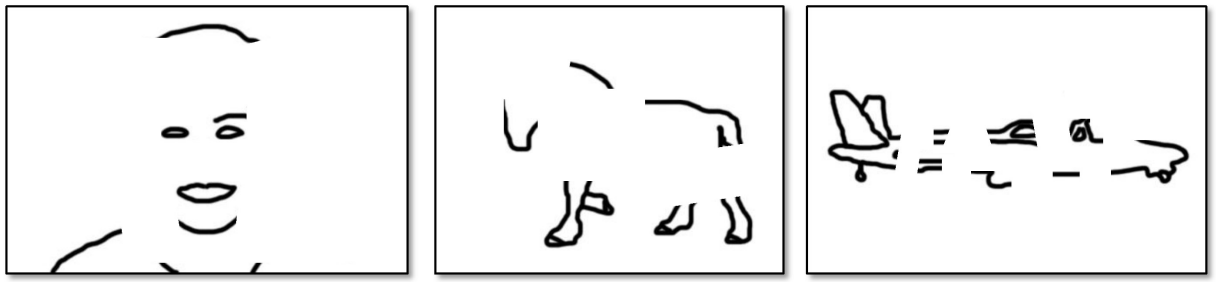
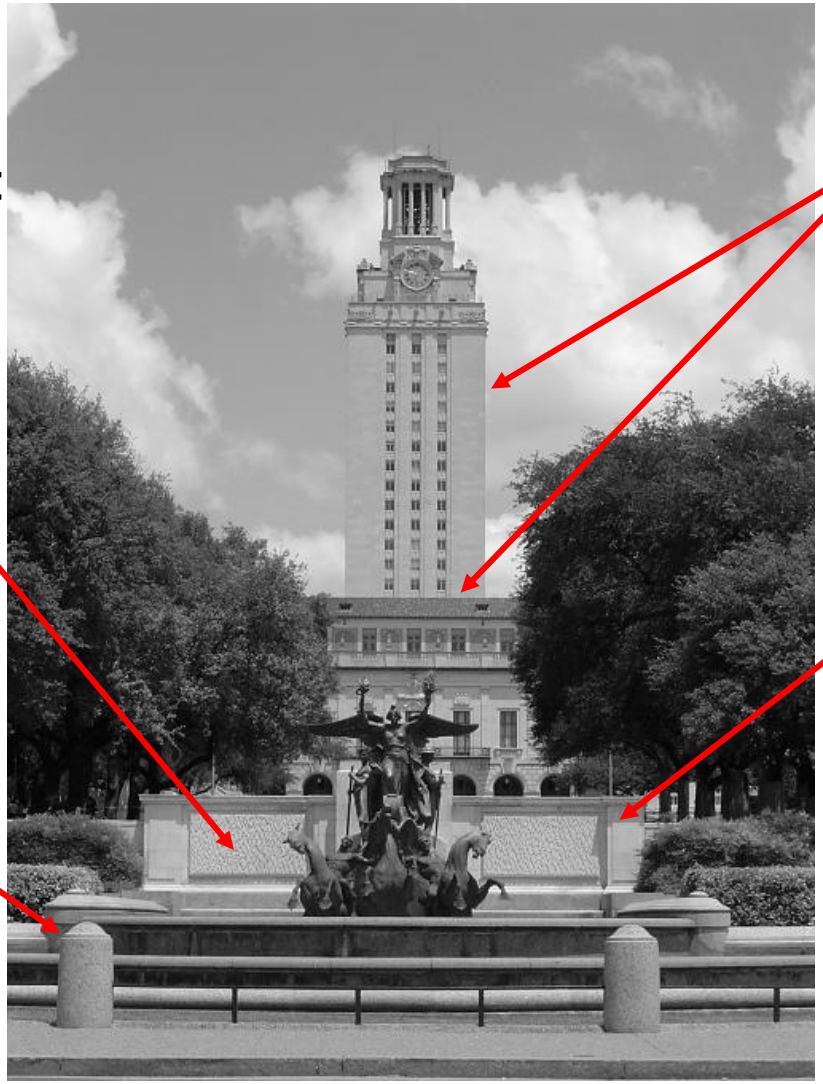


Figure from J. Shotton et al., PAMI 2007

Because the contours define the silhouette and are very informative about the content of the image!

What does cause an edge?



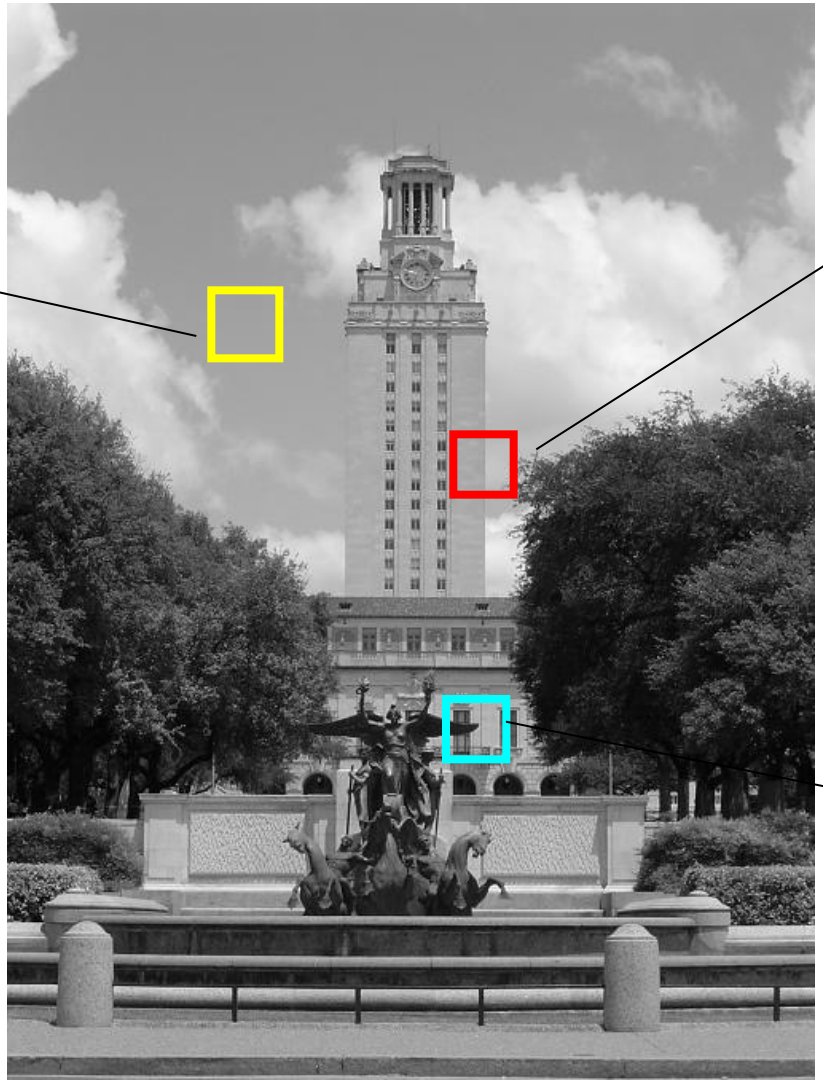
Reflectance change: appearance information, texture

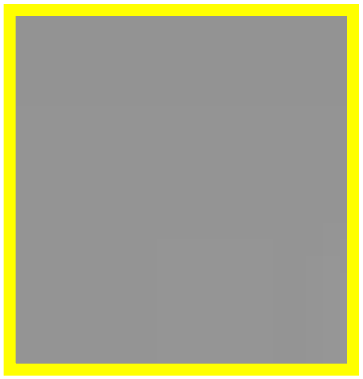
Depth discontinuity: object boundary

Cast shadows

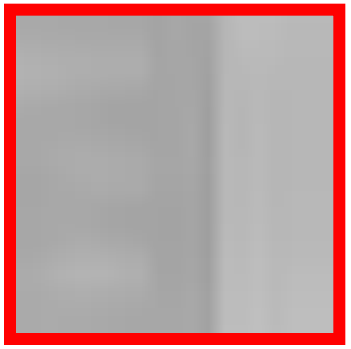
Change in surface orientation: shape

Edges/gradients and invariance





Homogenous area



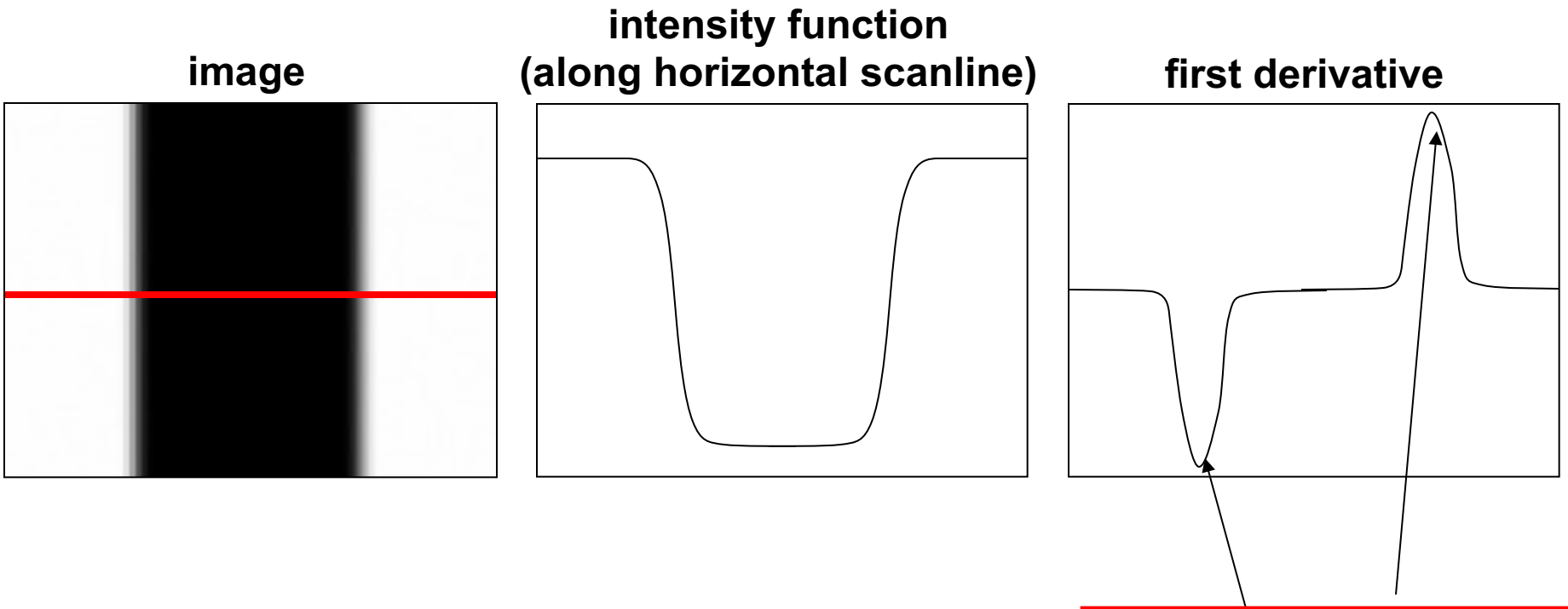
Smooth gradient

Hard gradient

Edges are defined by changes, opposite to invariance (homogeneous areas).

Derivatives and edges

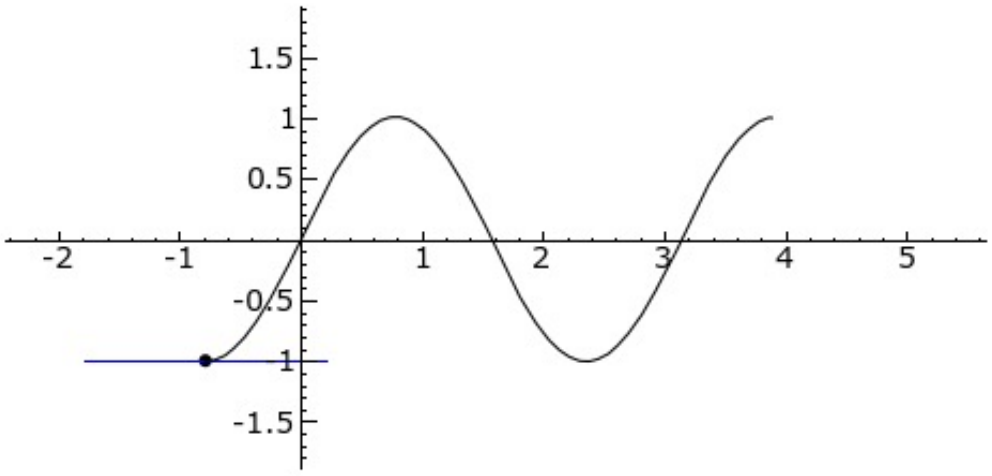
An edge is a place of rapid change in the image intensity function.



Source: L. Lazebnik

Derivative (Reminder)

The derivative of a function at a point c is the slope of the tangent line at this point.



Plot of $f(x) = \sin(2x)$ from $-\pi/4$ to $5\pi/4$;

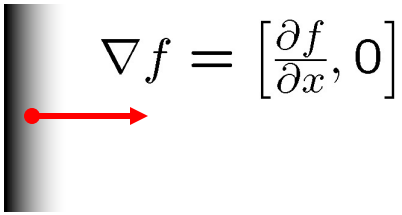
Image gradient

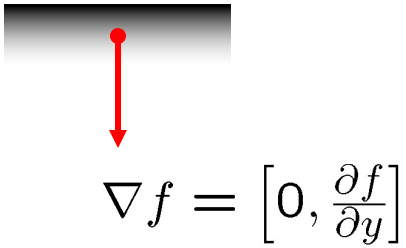
The gradient of an image is the vector:

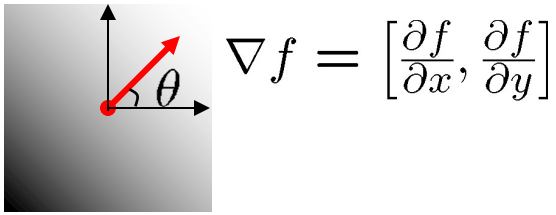
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

And points in the direction of most rapid change in intensity.

Examples:







Slide credit Steve Seitz

First derivatives: discrete operators.

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

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

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

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

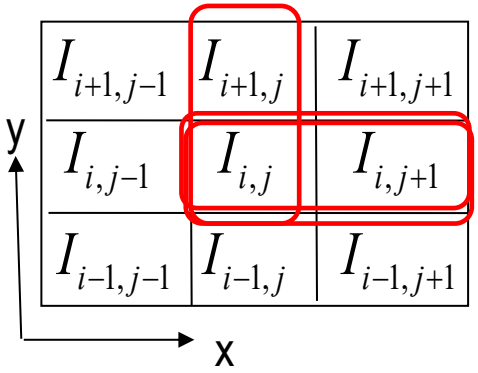
How to compute it on the image?

First derivatives: discrete operators.

Take discrete derivative (**finite differences**)

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

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

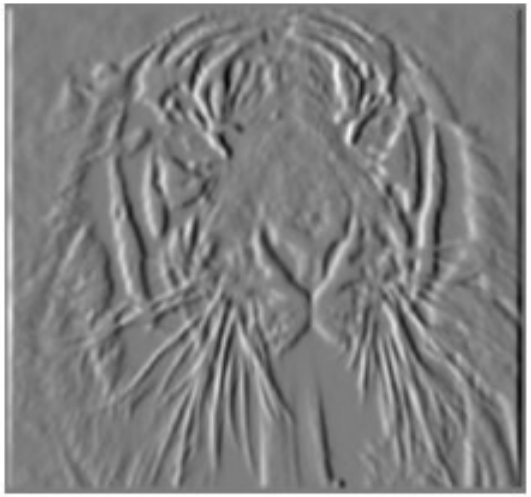


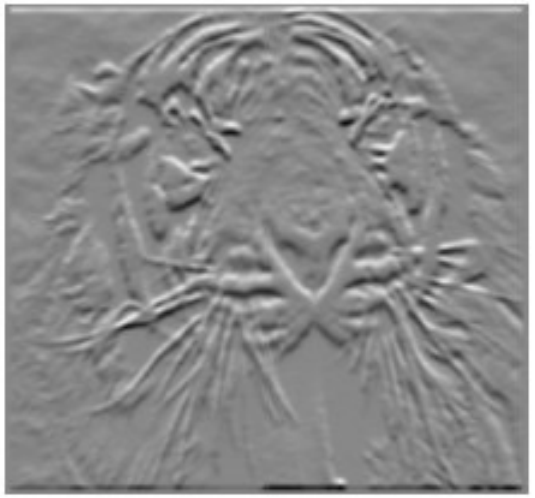
(given that f is the image I)

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

Partial derivatives of an image

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





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

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

-1	1

1
-1

$I_{i+1,j}$

$I_{i,j}$

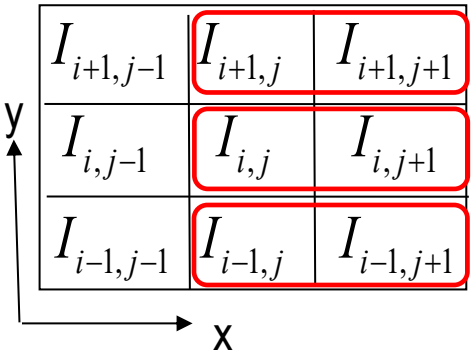
Which one does show changes with respect to x and y?

First derivatives: discrete operators.

But how to be less sensitive to noise?

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

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



An alternative is: getting the average of the 3 derivatives in order to be less sensitive to noise:

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

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

Example:

noise

0 100 100

0 255 100

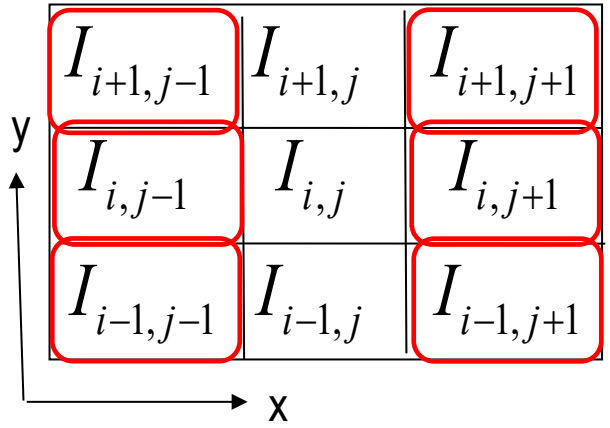
0 100 100

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 convolution mask to compute the derivatives?

Prewitt:

x =

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

y =

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

Finite difference filters: Sobel and Prewitt

Two common operators:

Prewitt: x=

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

y=

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

Sobel: x=

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

y=

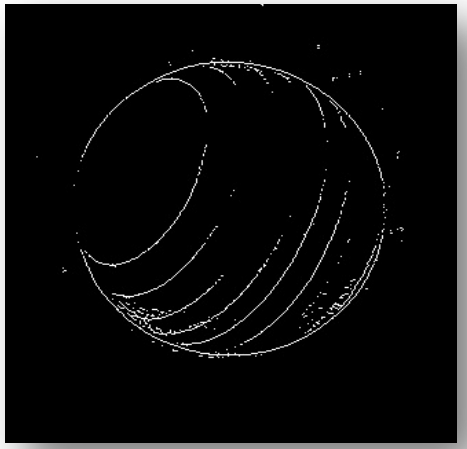
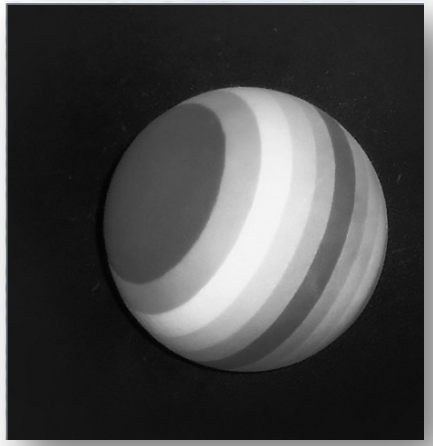
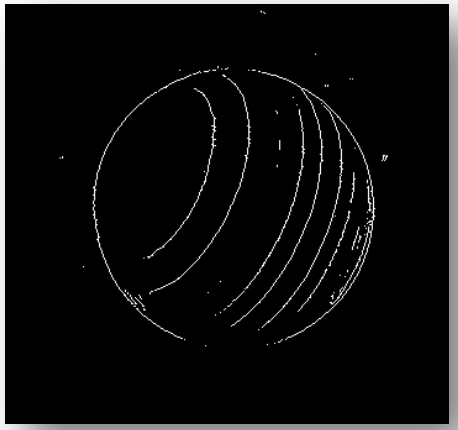
1	2	1
0	0	0
-1	-2	-1

What is the difference between ***Prewitt*** and ***Sobel*** operators?

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

Assorted finite difference filters

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

How to apply Sobel or Prewitt in Python to detect the edges?

→**`skimage.filters`**

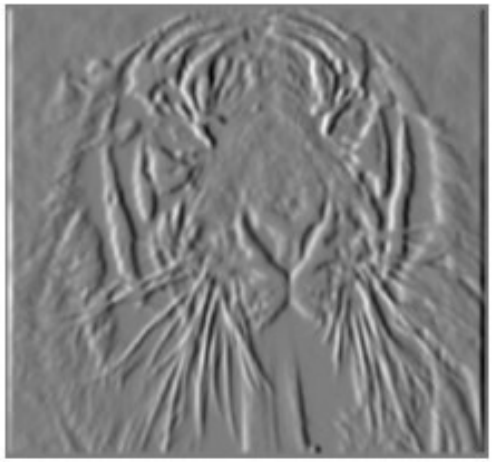
Image gradient

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

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

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

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



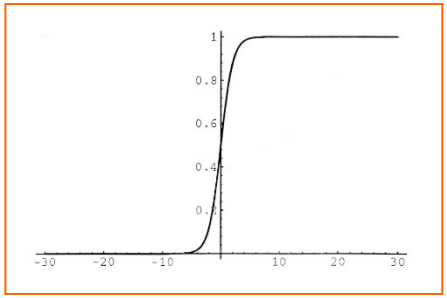
Which aspect (magnitude or direction) is invariant to image contrast?

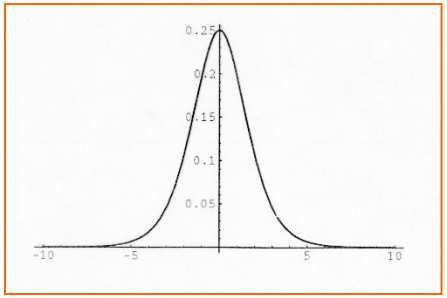
Slide credit Steve Seitz

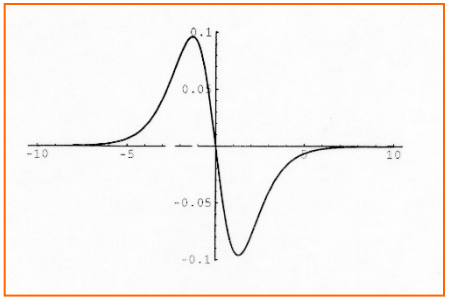
Discrete operators: second derivatives

What do you expect about the second derivatives?

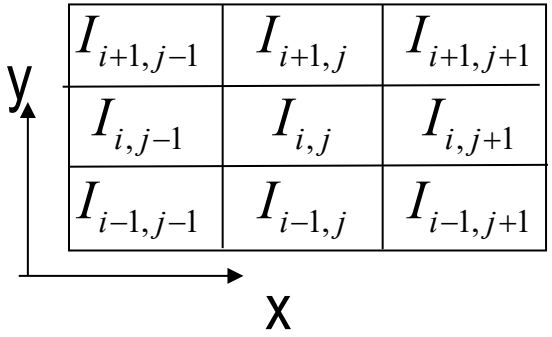
$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$







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.



The second derivative is the derivative of the first derivative:

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

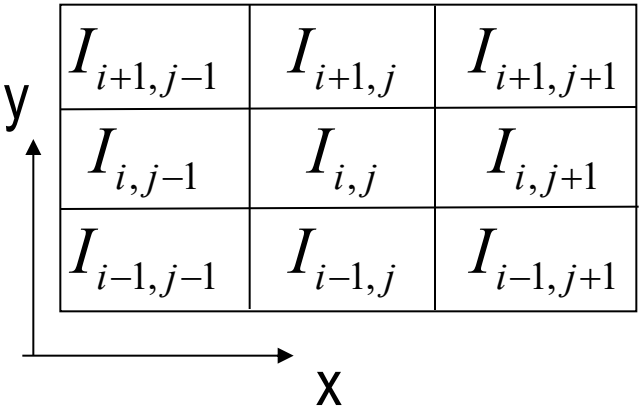
$$\frac{\partial^2 I}{\partial x^2} = (I_{i,j-1} - 2I_{i,j} + I_{i,j+1}) \rightarrow$$

1 -2 1

Discrete operators: Laplacian

Let's define:

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



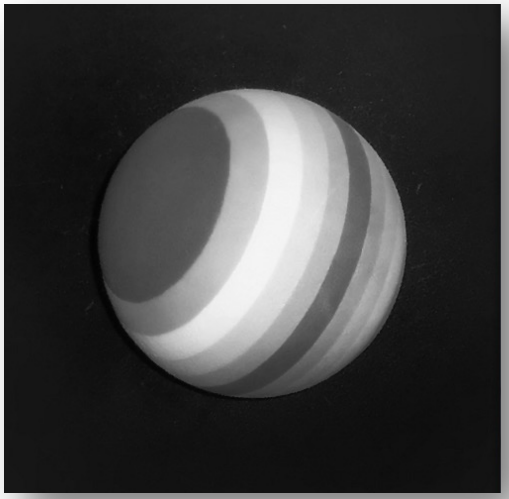
Laplacian mask:

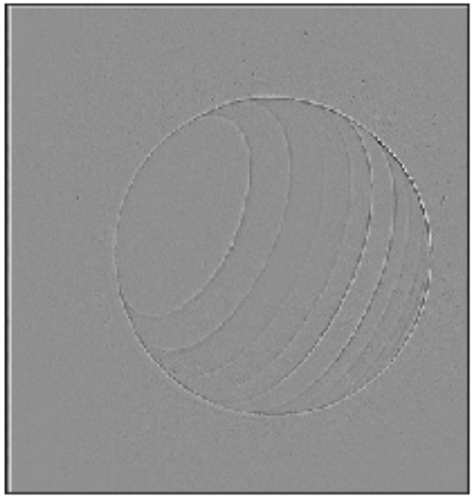
$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Discrete operators: Laplacian

Laplacian mask:

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





Skimage:

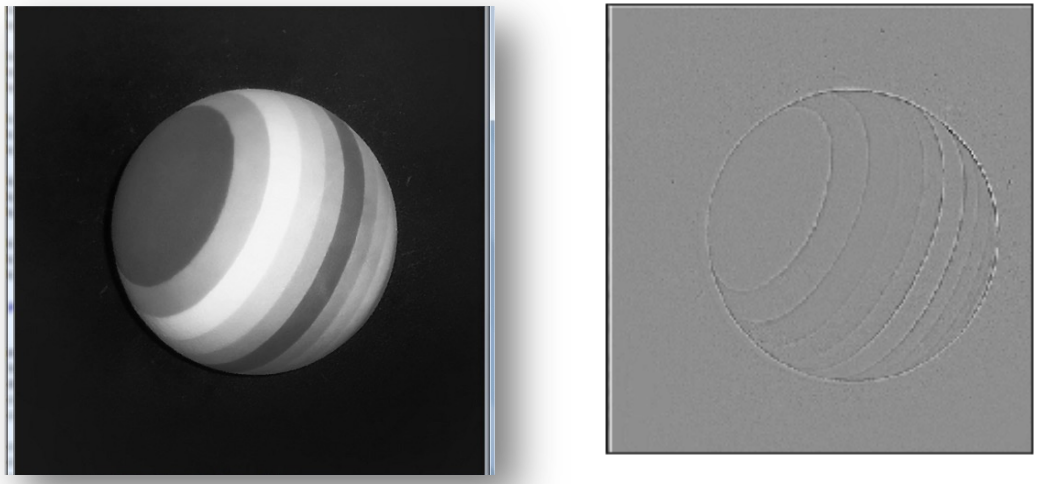
```
>> mask=numpy.array([[0,1,0],[1,-4,1],[0,1,0]], dtype='float')

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

Discrete operators: Laplacian

Including the diagonal neighbours:

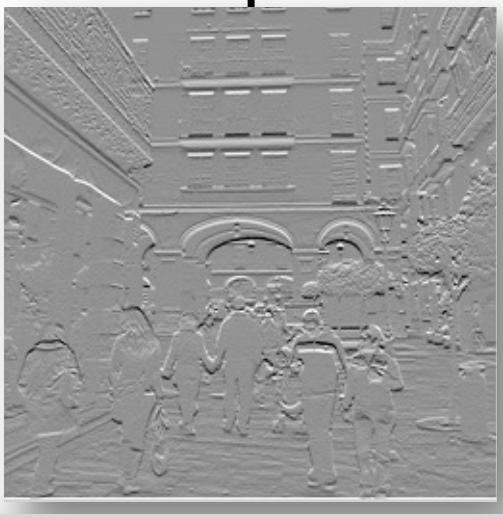
Laplacian:

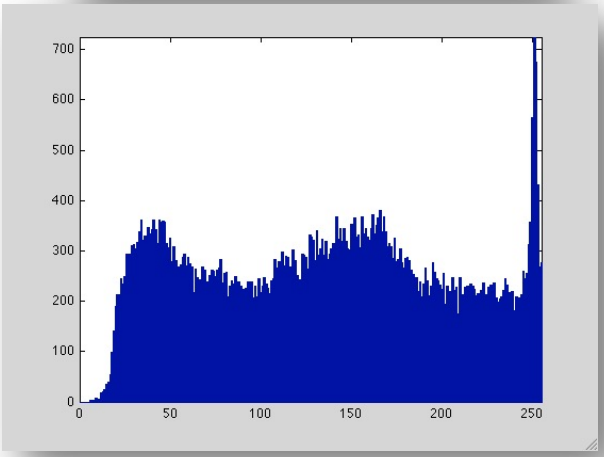
$$\begin{matrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{matrix}$$

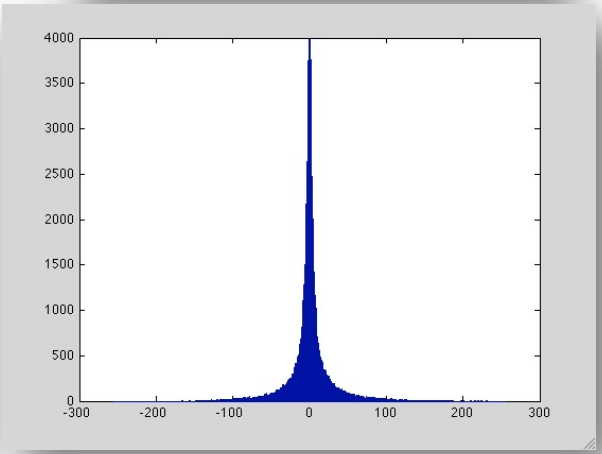


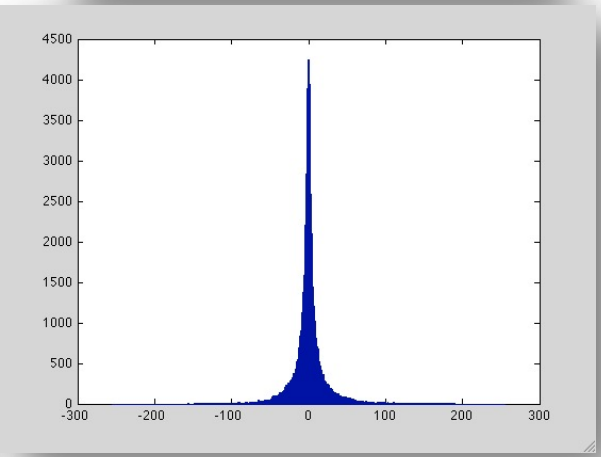
21

Histogram of x and y edge maps









Horizontal and vertical derivatives distributions.

Why are they positive and negative?

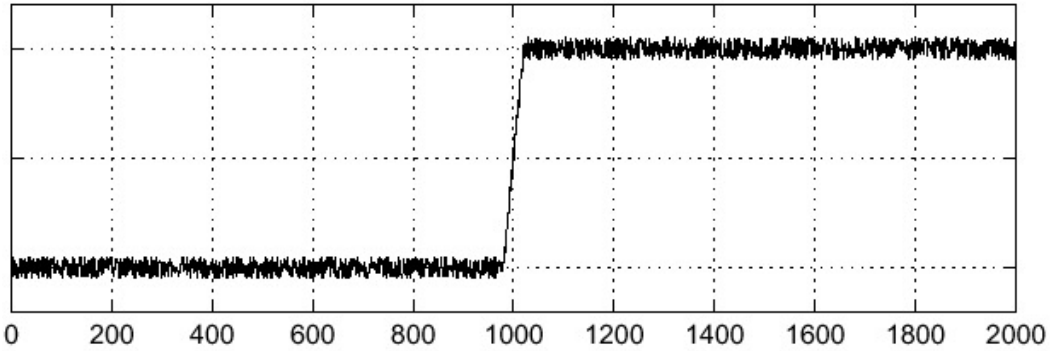
Why the maxima is in 0? Can I store them in a uint8 type image?

Effects of noise

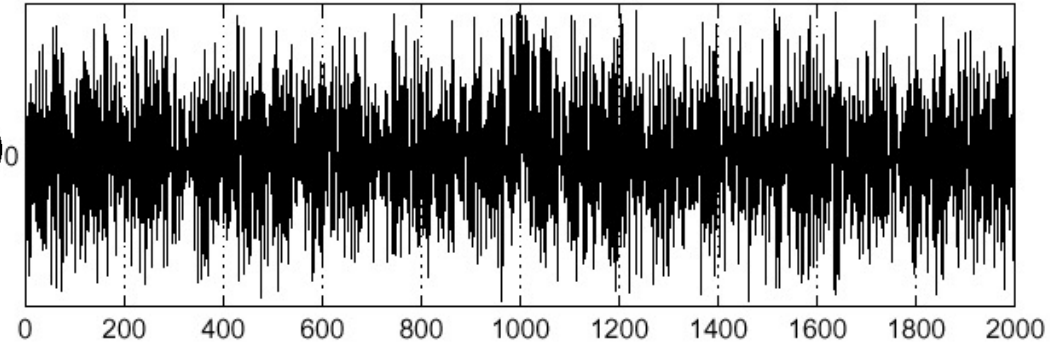
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal (function f)

$f(x)$



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

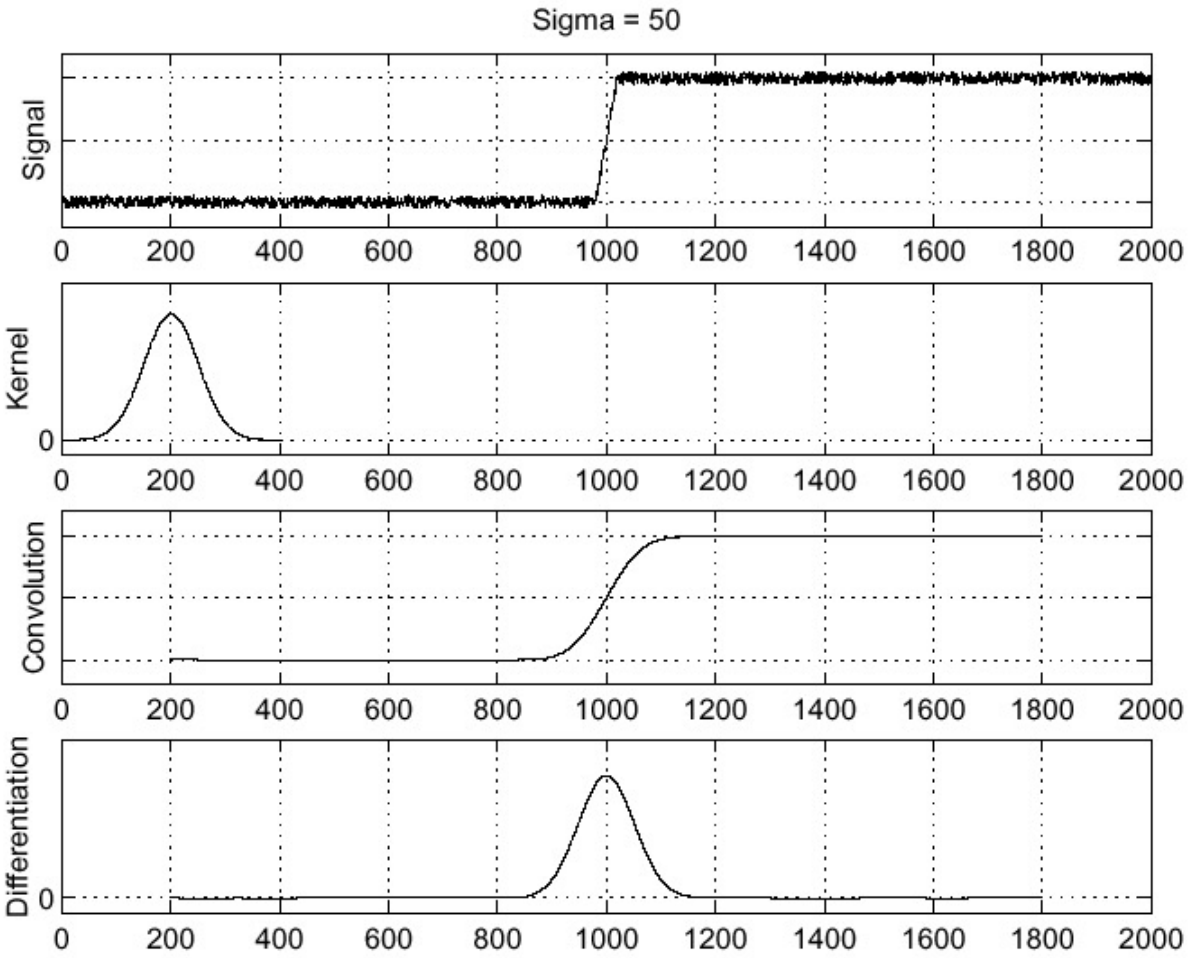


Where is the edge?

Slide credit Steve Seitz

Solution: smooth using Gaussian filter

f



Gaussian filter: h

Smoothing

Convolution: $h \star f$

Edge enhancement:

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

Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

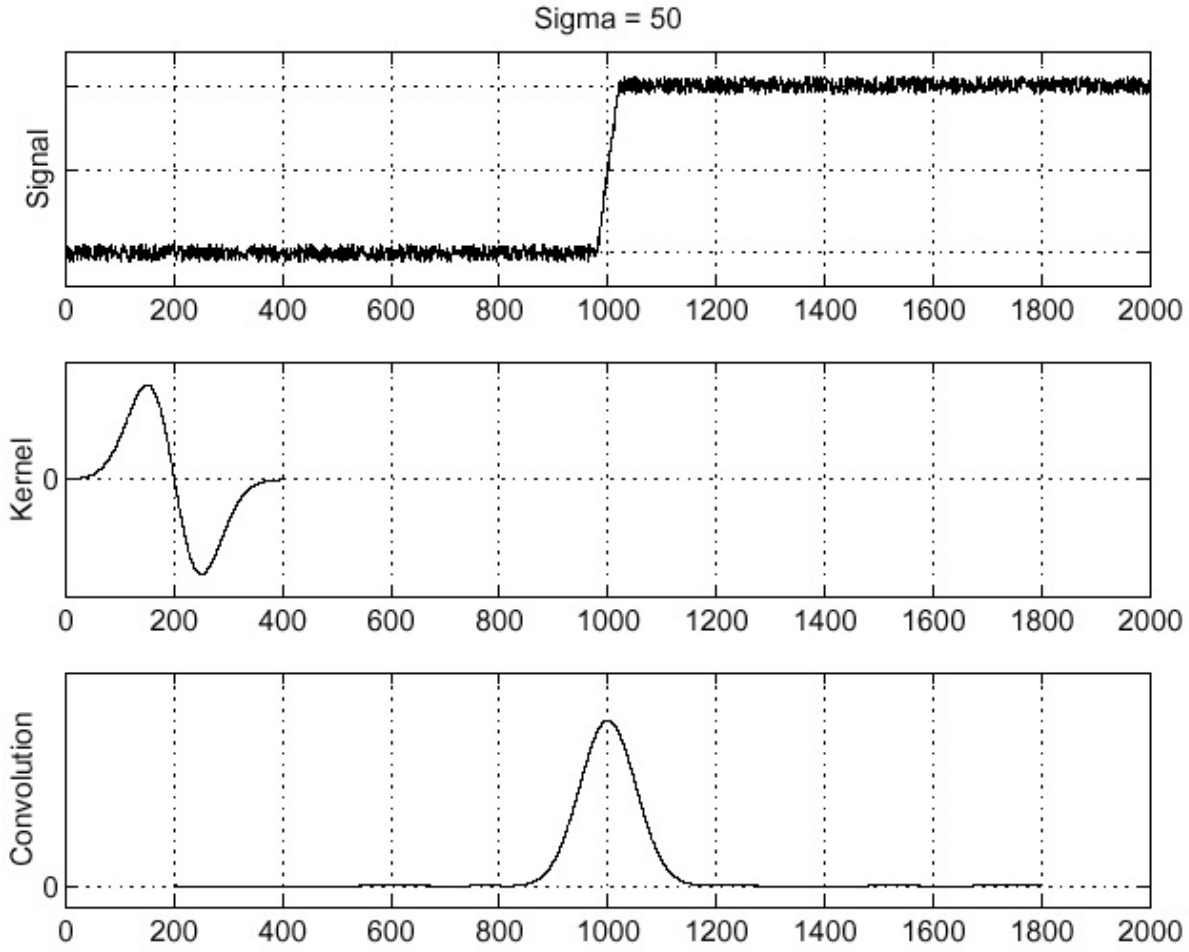
Differentiation property of convolution: $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$

f

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

Derivative of Gaussian filter

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



Slide credit Steve Seitz

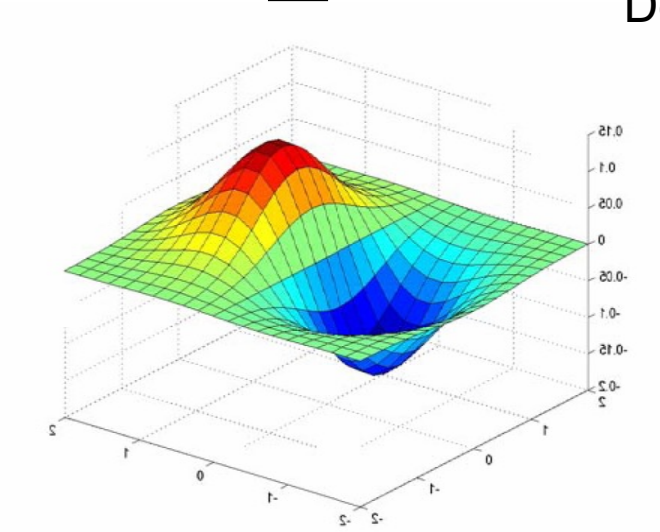
Derivative of Gaussian (DoG) filter

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

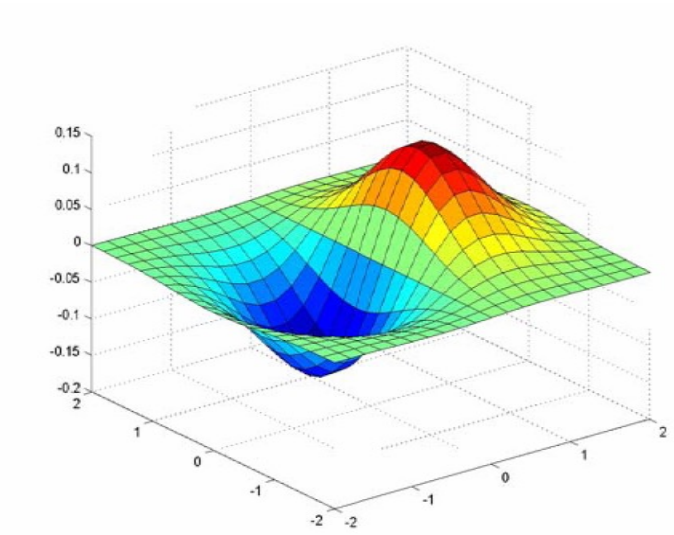
$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix}$$

Gaussian filter

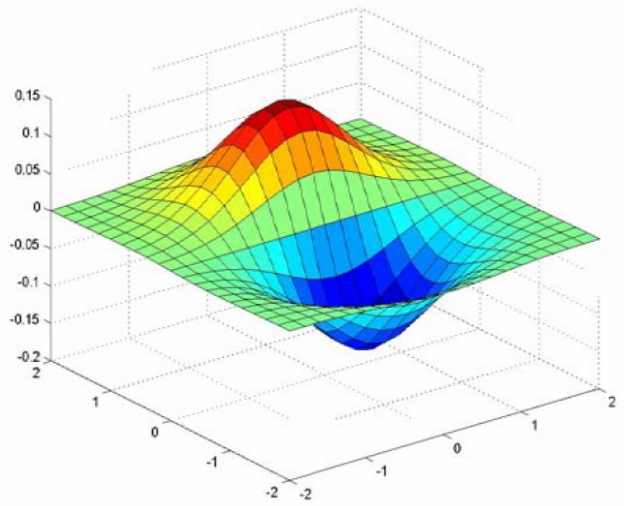
Derivative filter



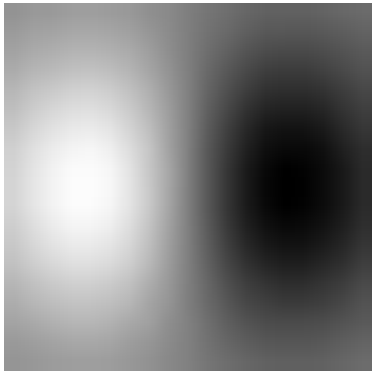
Derivative of Gaussian filters

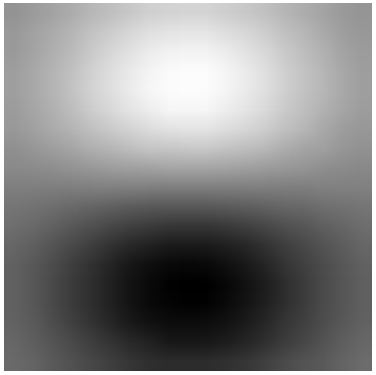


***x*-direction**



***y*-direction**





Source: L. Lazebnik

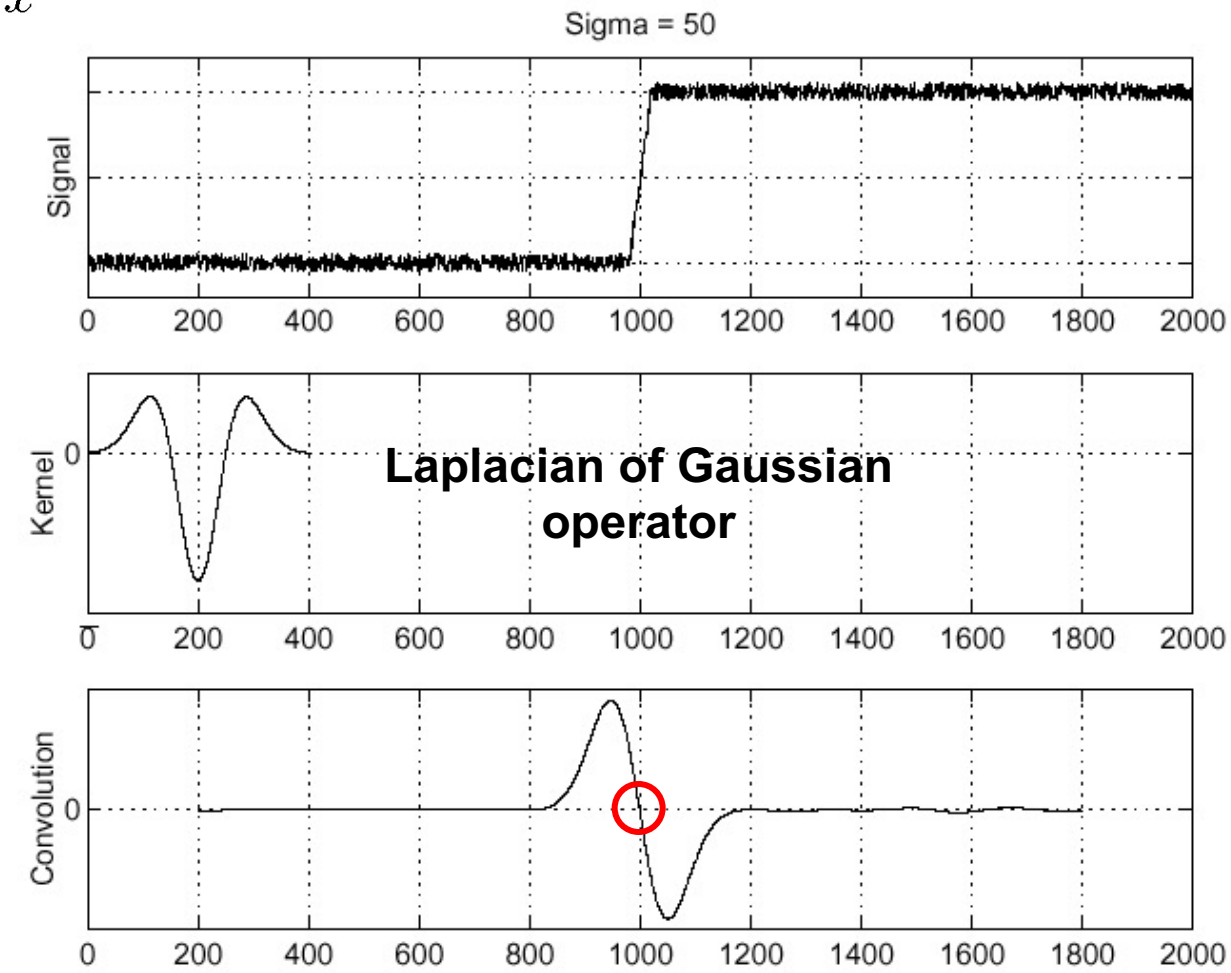
Laplacian of Gaussian (LoG) filter

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

f

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

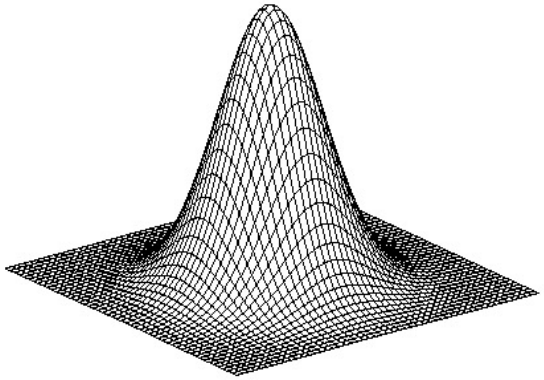
$(\frac{\partial^2}{\partial x^2} h) \star f$



Where is the edge? Look for zero-crossing on bottom graph.

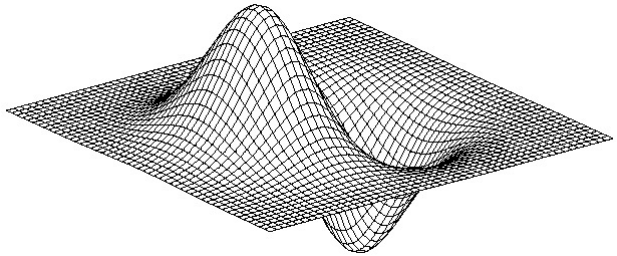
Slide credit: Steve Seitz

2D Gradient and derivatives



Gaussian

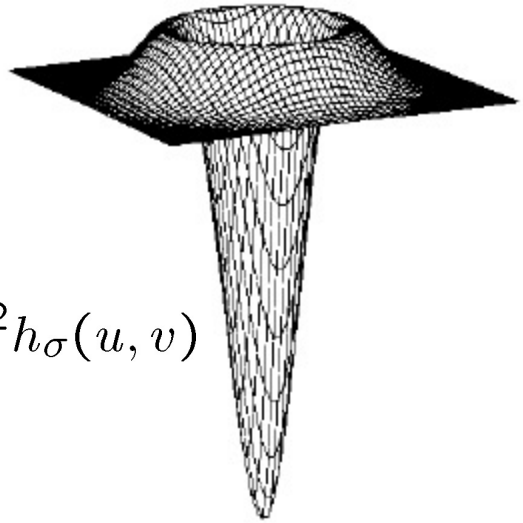
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Derivative of Gaussian

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

Laplacian of Gaussian



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

- ∇^2 is the Laplacian operator:

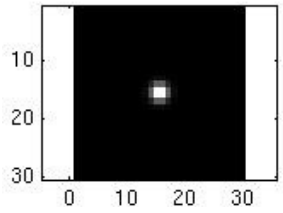
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

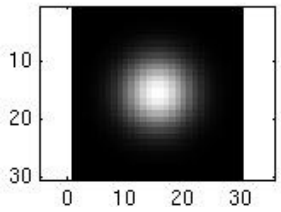
Slide credit: Steve Seitz

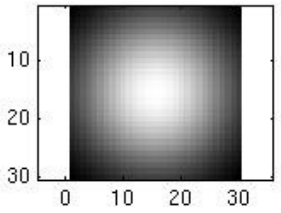
Smoothing with a Gaussian

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

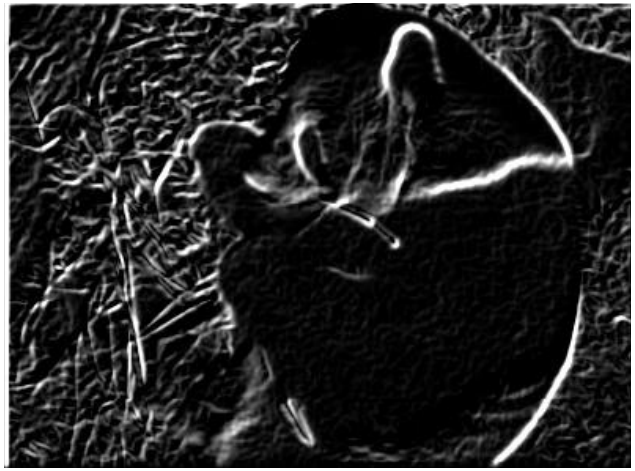
...







Effect of σ on derivatives



σ = 1 pixel

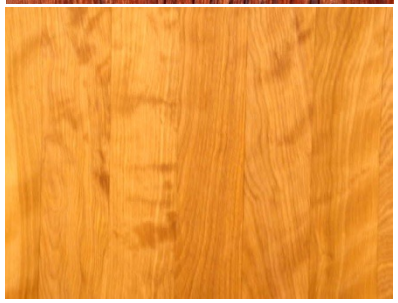
σ = 3 pixels

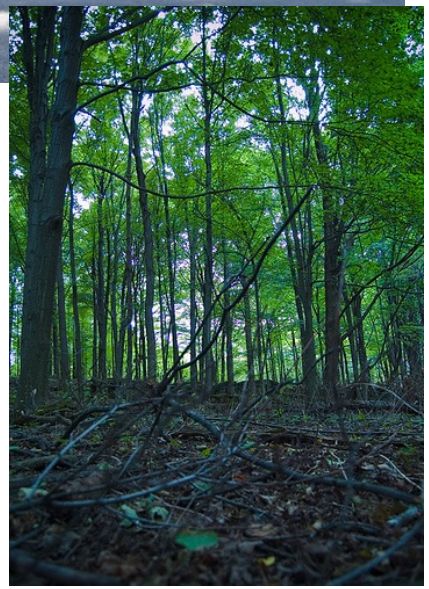
The apparent structures differ depending on Gaussian's scale parameter:

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

So, what scale to choose?

It depends what we're looking for.

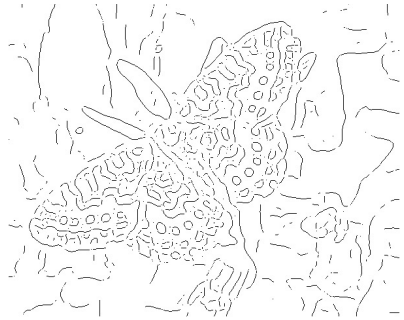




We need previous knowledge about the right scale.
Or manage different scales.

From Gradients to edges

Primary edge detection steps:



1. Smoothing: suppress noise

   (e.g. by applying a Gaussian filter)

2. Edge enhancement: filter for contrast detection

   (e.g. applying finite difference filters as Sobel)

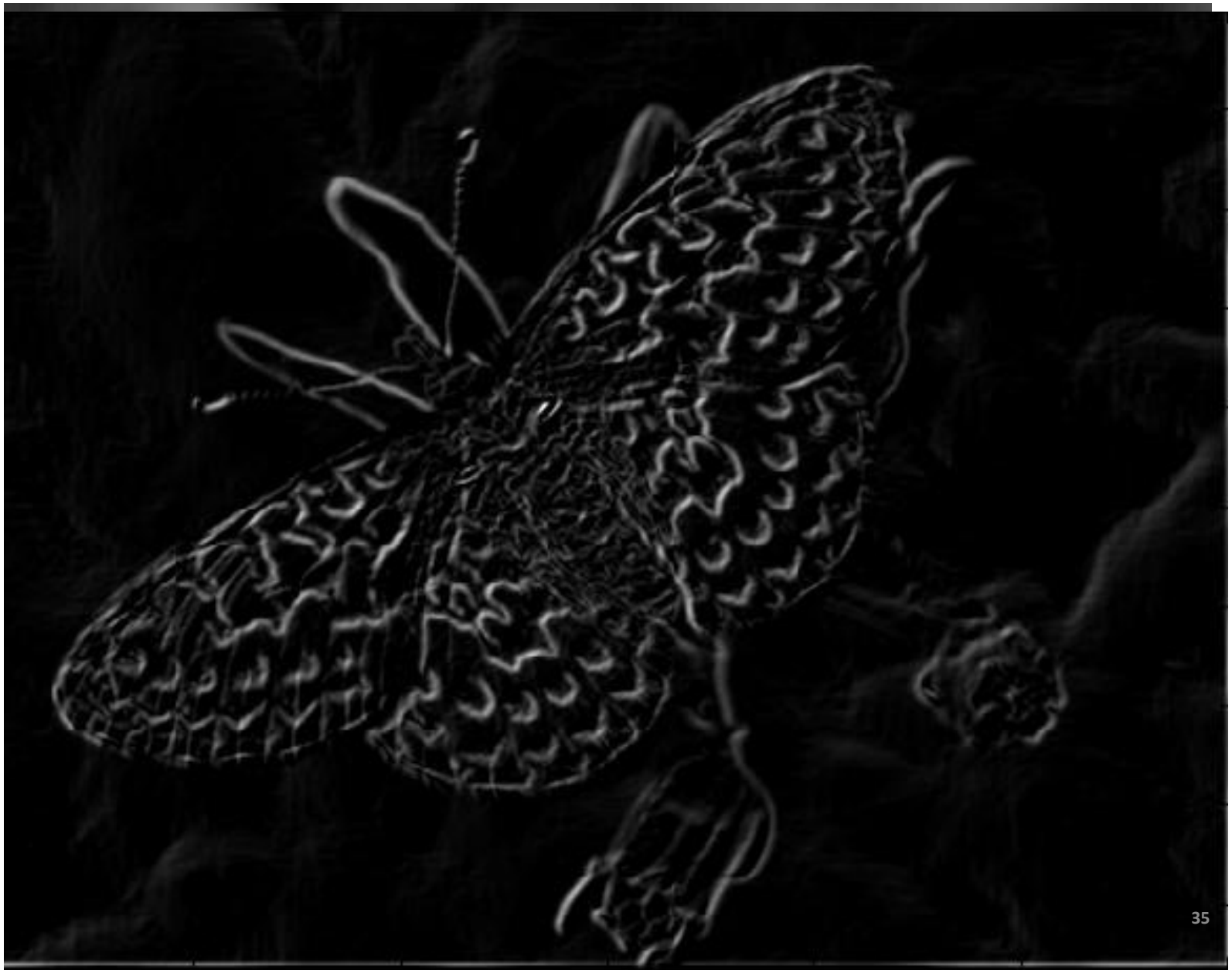
Or a single step: using DoG.

3. Edge localization

   - Determine which local maxima from filter output are actually edges vs. noise.
   - **Binarization (thresholding)**: all pixels with output value
     - higher than a threshold → assign 1,
     - lower than the threshold → assign 0.

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 filter
- 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

J. Canny, “A Computational Approach to Edge Detection,” IEEE PAMI, vol. 8, pp. 679-698, 1986.

Source: D. Lowe, L. Fei-Fei

The Canny edge detector

Original image (Lena)

Slide credit: Steve Seitz

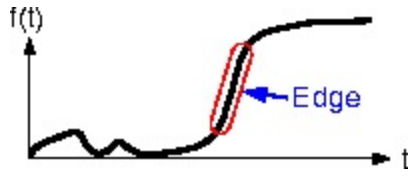
The Canny edge detector

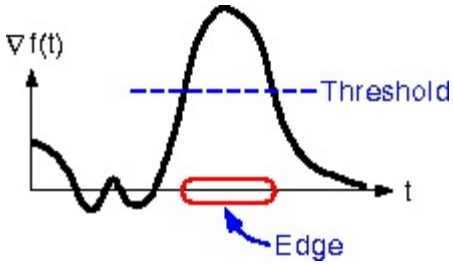
Norm of the gradient

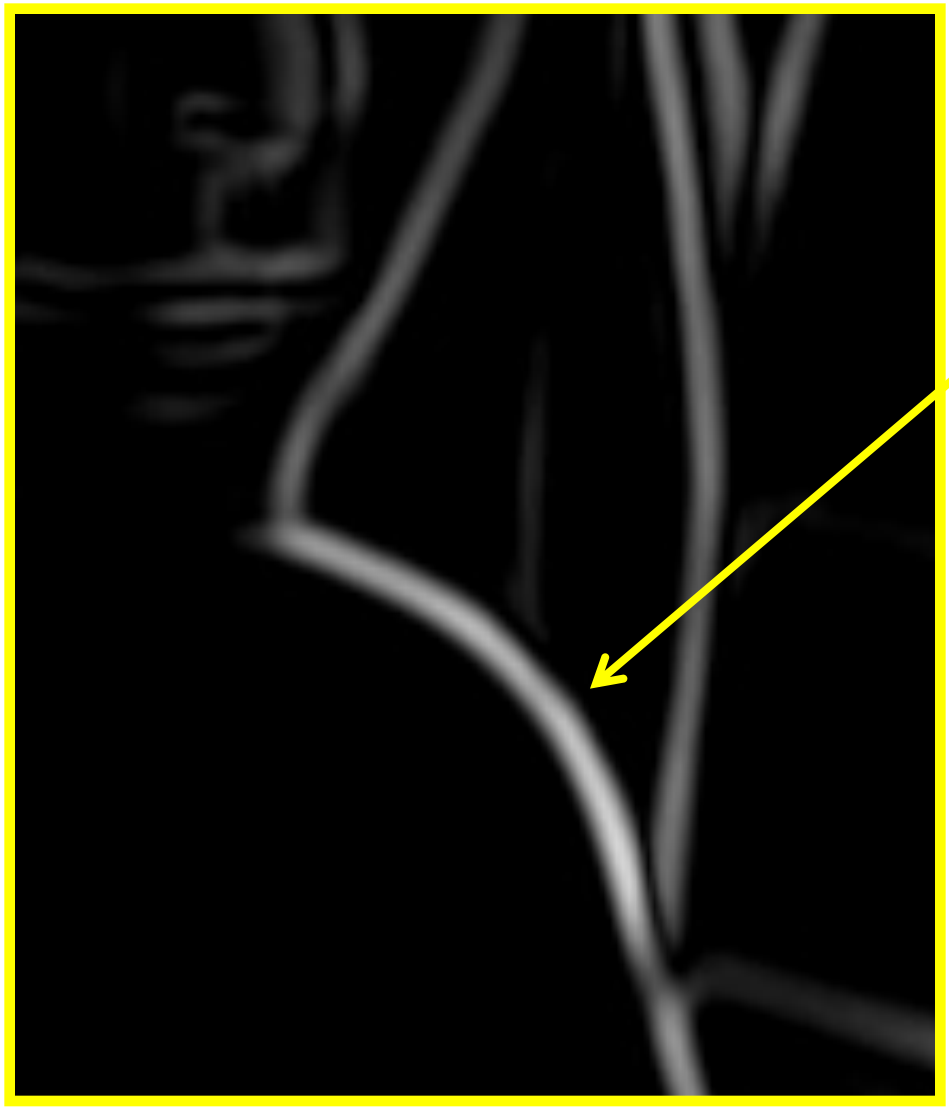
The Canny edge detector

Thresholding

The Canny edge detector

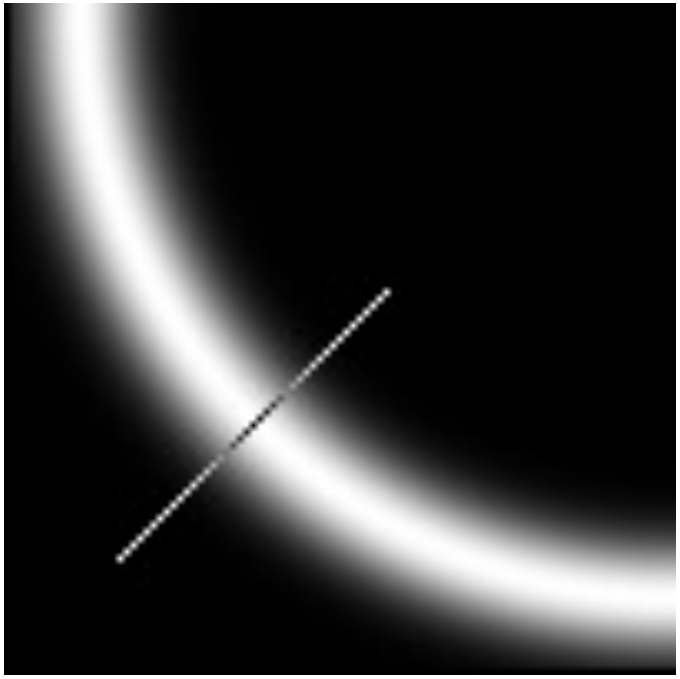






How to turn these thick regions of the gradient into curves?

Non-maximum suppression



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

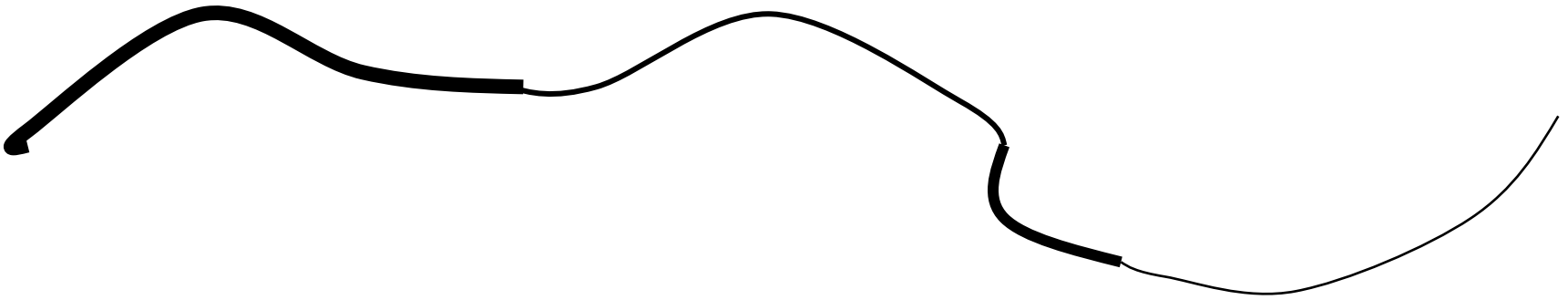
The Canny edge detector

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

Canny in Skimage: **skimage.feature.canny**

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


Source: D. Lowe, L. Fei-Fei

Canny edge detector

Low-level edges vs. perceived contours

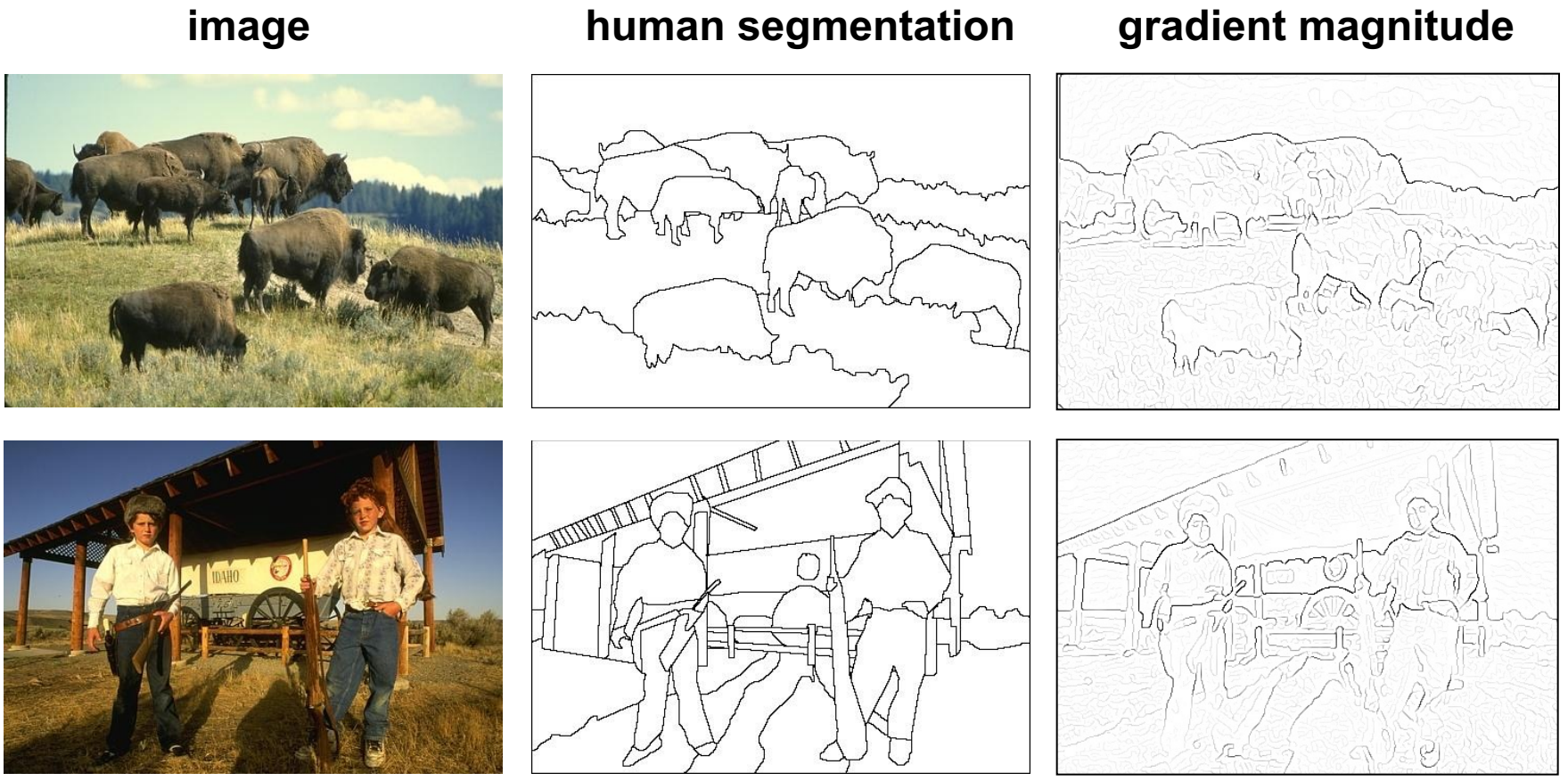
Background

Texture

Shadows

Is Canny edge detector distinguishing these contours?

Low-level edges vs. perceived contours

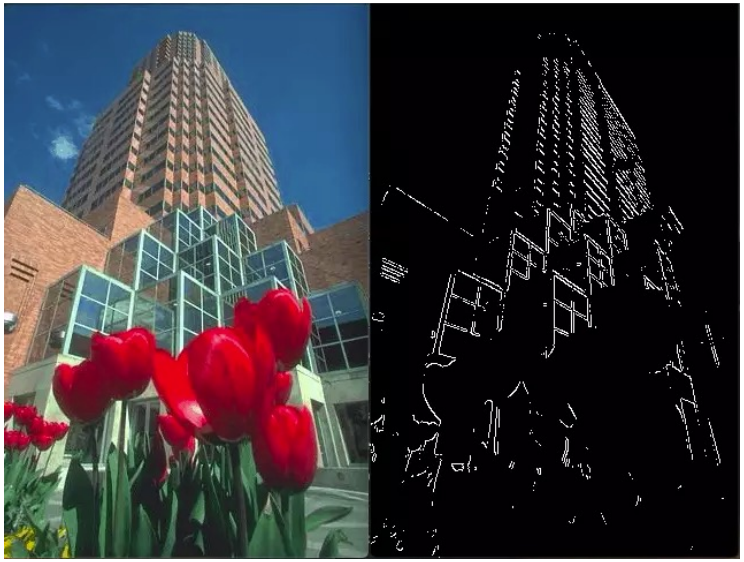


- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Source: L. Lazebnik

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.

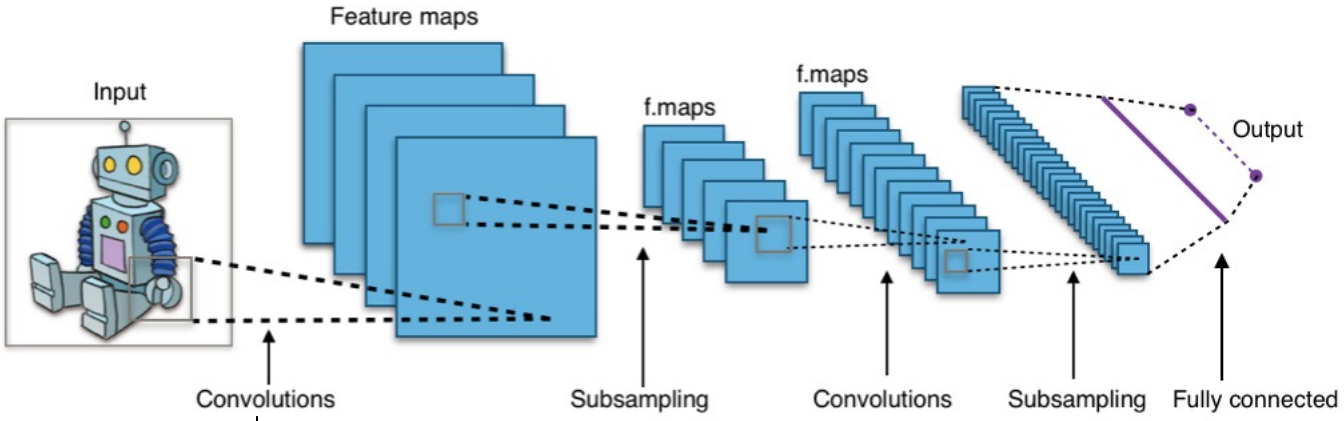
There are alternatives as the one in OpenCV: *Holistically-Nested Edge Detection* (HED), an **end-to-end deep neural network**.
[Saining Xie, Zhuowen Tu "HED" 2015]

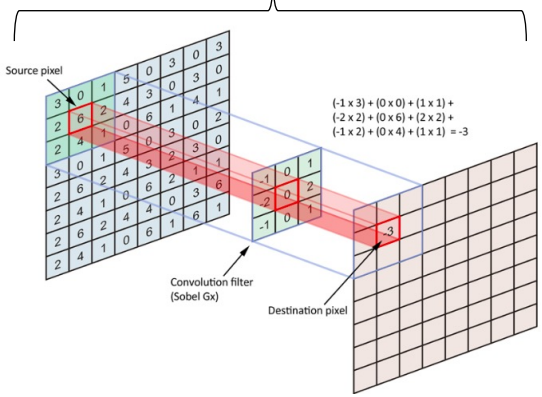
Summary

- 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 Derivative of 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: **classical edge detector**
    - Assures continuous and thin contours due to the hysteresis and the thinning steps.
    - Needs parameters
- Still the question about the object contours is not solved!

For the next lessons:
Relation between linear filters and CNN?

- **Convolutional** Neural Network (CNN)

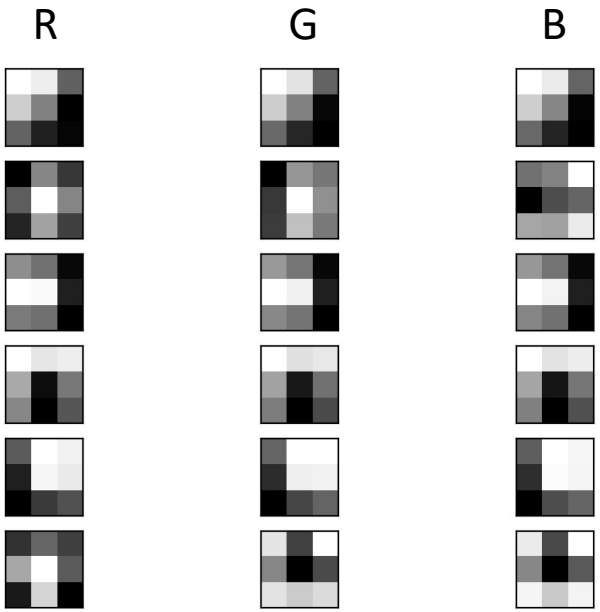




- The activation maps, called feature maps, capture the result of applying the filters to the input (the input image or another feature map).
- The convolution filter is learned during training.

Convolution filters of a CNN

- Visualization of the learned filters:



Plot of the first 6 filters from VGG16 with one subplot per channel.
Source

Similars to the derivatives filters, filters extracting some local structure as countours, corners and so on.

Feature maps of a CNN

- Visualization of the fature maps or activation maps:

Input image

Visualization of the feature maps extracted from the first convolutional layer in the VGG16 Model

Source

Different versions of the bird image with different features highlighted.

Bibliography

- Source of slides: K. Grauman, UT-Austin

Other Slides Source:

- P. Radeva
- L. Lazebnik

- The following chapters of the book "Computer Vision: Algorithms and Applications" by Richard Szeliski provide additional information:
  - Chapter 3. Image processing: Point operators & linear filtering.
  - Chapter 4. Feature detection and matching: Edges.

Lecture videos

FILTERS

- 102 - Linear Quiz
  Until 125 - Median Filter
  Except 117

Total time: 27,57 minutes.

EDGES:

- 143 – Intro
  Until 164 - Linearity Property Quiz Solution
  Except 160, 161.
- 167 - Derivative of Gaussian Filter 2D
  Until 179 - Single 2D Edge Detection Filter
  Except 173, 174, 175.

Total time: 36,63 minutes.

Canny: 171 + 172

From *Introduction to Computer Vision*:

https://www.udacity.com/course/introduction-to-computer-vision--ud810

Final Test

- **Go to:** *Socrative.com*
  - Choose Student Login
  - Room Name: COMPUTERVISION
  - Enter your name: It can be anonymous.

COMPUTATIONAL VISION:
Image gradients and edges

Master in Artificial Intelligence

Department of Mathematics and Computer Science

UNIVERSITAT DE BARCELONA