E9 205 Machine Learning for Signal Processing
23-8-17

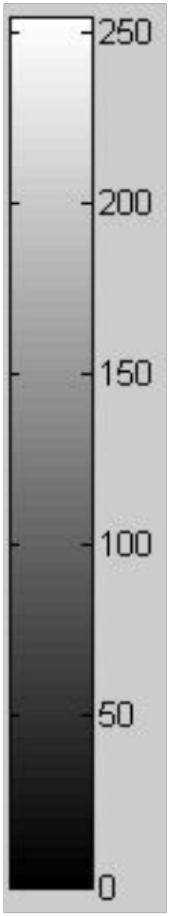
Outline

- Basics for Image Processing
  - Filtering
  - Smoothing
  - Edge Detection
- Scale Invariant Feature Transform (SIFT)

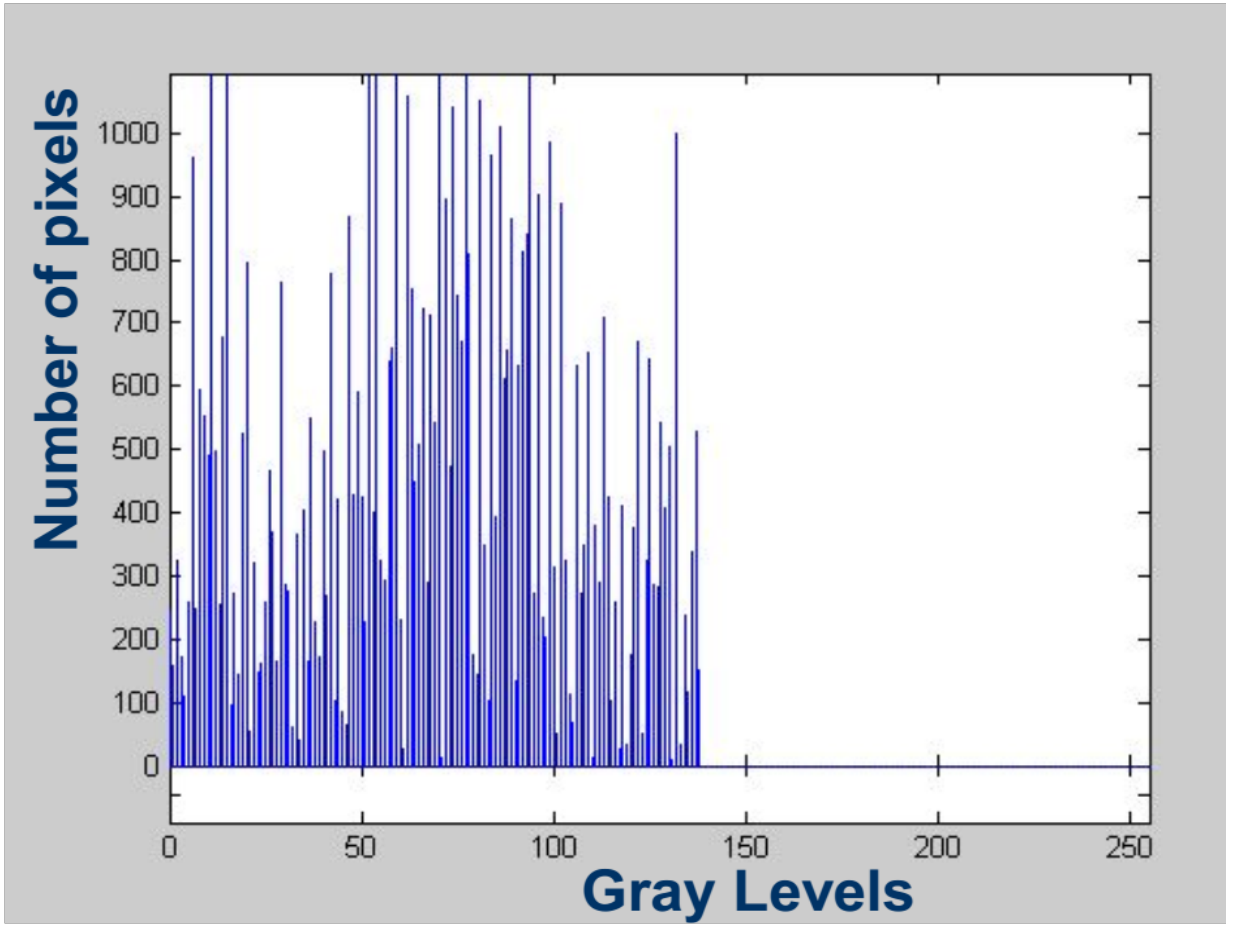
Reference: UCF, Computer Vision Course
Link: http://crcv.ucf.edu/courses/CAP5415/Fall2014/index.php

Gray Scale Image

10	5	9				
				100		



Histogram



- Histogram captures the distribution of gray levels in the image.
- How frequently each gray level occurs in the image

Image filtering

- Image filtering: compute function of local neighborhood at each position

  Really important!

  - Enhance images
  - Denoise, resize, increase contrast, etc.
  - Extract information from images
  - Texture, edges, distinctive points, etc.
  - Detect patterns
  - Template matching

Discrete Derivative

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x)$$

$$\frac{df}{dx} = \frac{f(x) - f(x-1)}{1} = f'(x)$$

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

$$f(x) = 10 \quad 15 \quad 10 \quad 10 \quad 25 \quad 20 \quad 20 \quad 20$$

$$f'(x) = 0 \quad 5 \quad -5 \quad 0 \quad 15 \quad -5 \quad 0 \quad 0$$

$$f''(x) = 0 \quad 5 \quad 10 \quad 5 \quad 15 \quad -20 \quad 5 \quad 0$$

Discrete Derivative Finite Difference

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$ Backward difference

$$\frac{df}{dx} = f(x) - f(x+1) = f'(x)$$ Forward difference

$$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$ Central difference

Backward difference	[-1 1]
Forward difference	[1 -1]
Central difference	[-1 0 1]

Derivatives in 2 Dimensions

Given function $f(x, y)$

Gradient vector $$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Gradient magnitude $$\left|\nabla f(x, y)\right| = \sqrt{f_x^2 + f_y^2}$$

Gradient direction $$\theta = \tan^{-1} \frac{f_x}{f_y}$$

Derivatives of Images

Derivative masks

$$f_x \Rightarrow \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad f_y \Rightarrow \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Correlation

$$f \otimes h = \sum_k \sum_l f(k,l)h(k,l)$$

f = Image

h = Kernel

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$\otimes$

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$$f \otimes h = f_1h_1 + f_2h_2 + f_3h_3 + f_4h_4 + f_5h_5 + f_6h_6 + f_7h_7 + f_8h_8 + f_9h_9$$

Convolution

$$f * h = \sum_k \sum_l f(k,l)h(-k,-l)$$

f = Image

h = Kernel

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$X - flip$

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

$Y - flip$

f

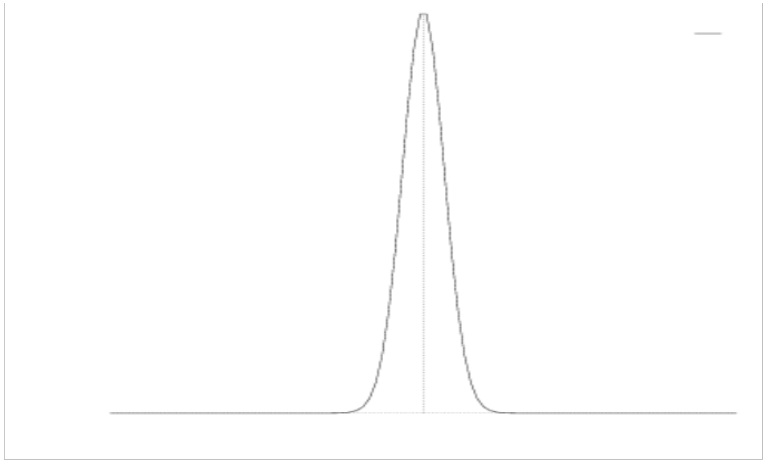
f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

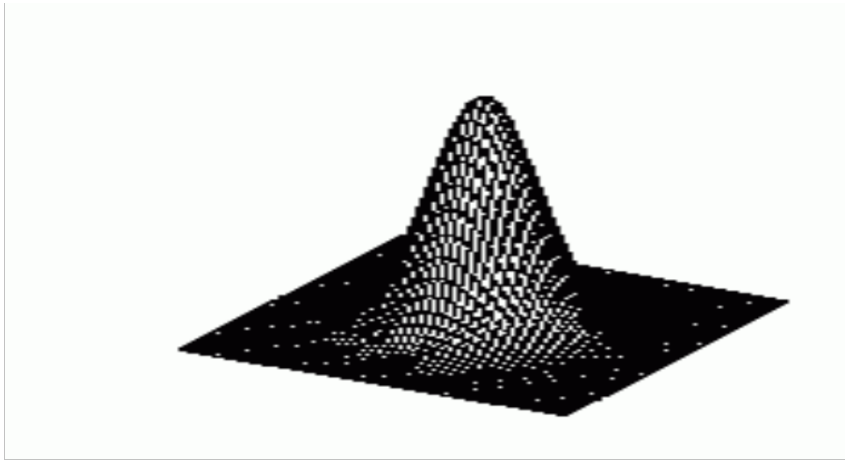
$*$

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

$$f * h = f_1h_9 + f_2h_8 + f_3h_7 + f_4h_6 + f_5h_5 + f_6h_4 + f_7h_3 + f_8h_2 + f_9h_1$$

Gaussian Filter





$$g(x) = e^{\frac{-x^2}{2o^2}}$$

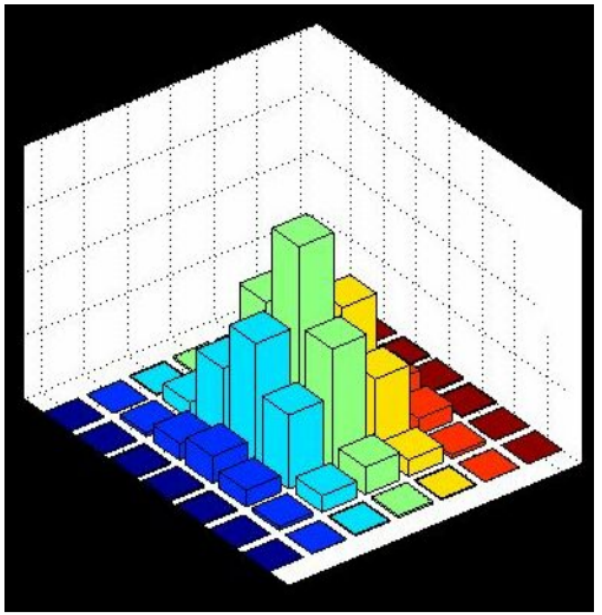
$$g(x,y) = e^{\frac{-(x^2+y^2)}{2o^2}}$$

$$g(x) = [.011 \quad .13 \quad .6 \quad 1 \quad .6 \quad .13 \quad .011]$$

$$\sigma = 1$$

Gaussian Filtering

*

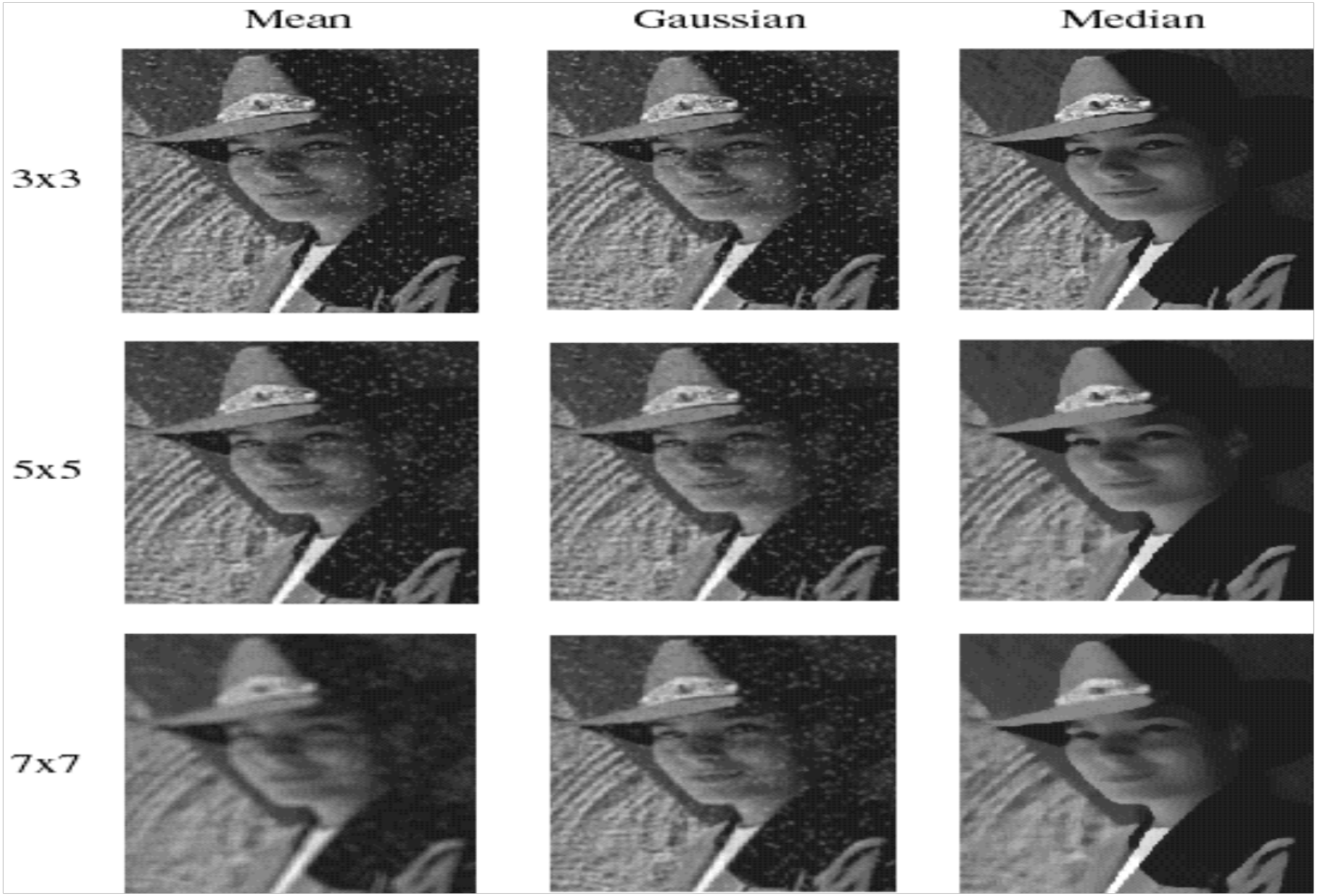


=

Gaussian Smoothing

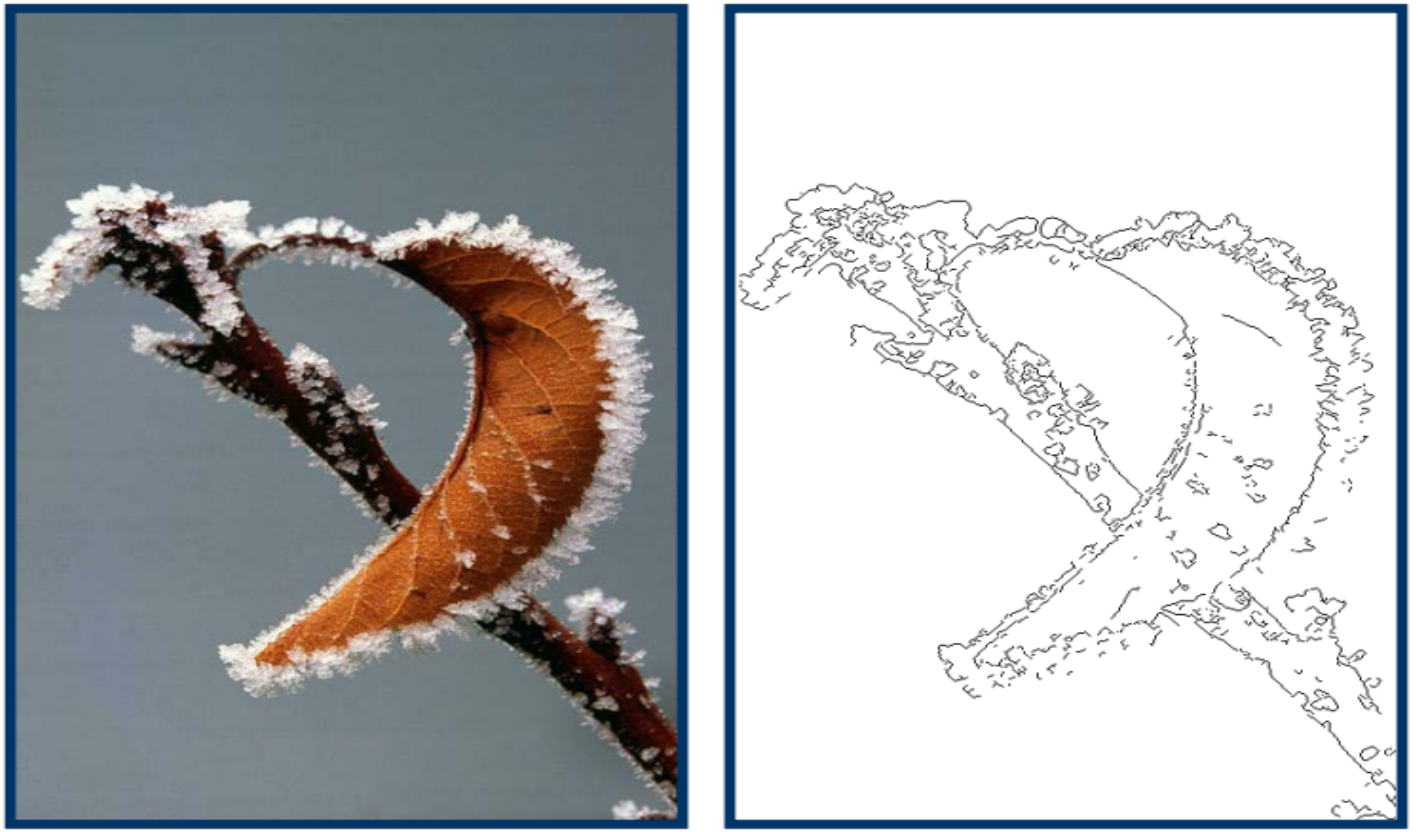
Smoothing by Averaging

Median Filter: operates over a window by selecting the median intensity in the window



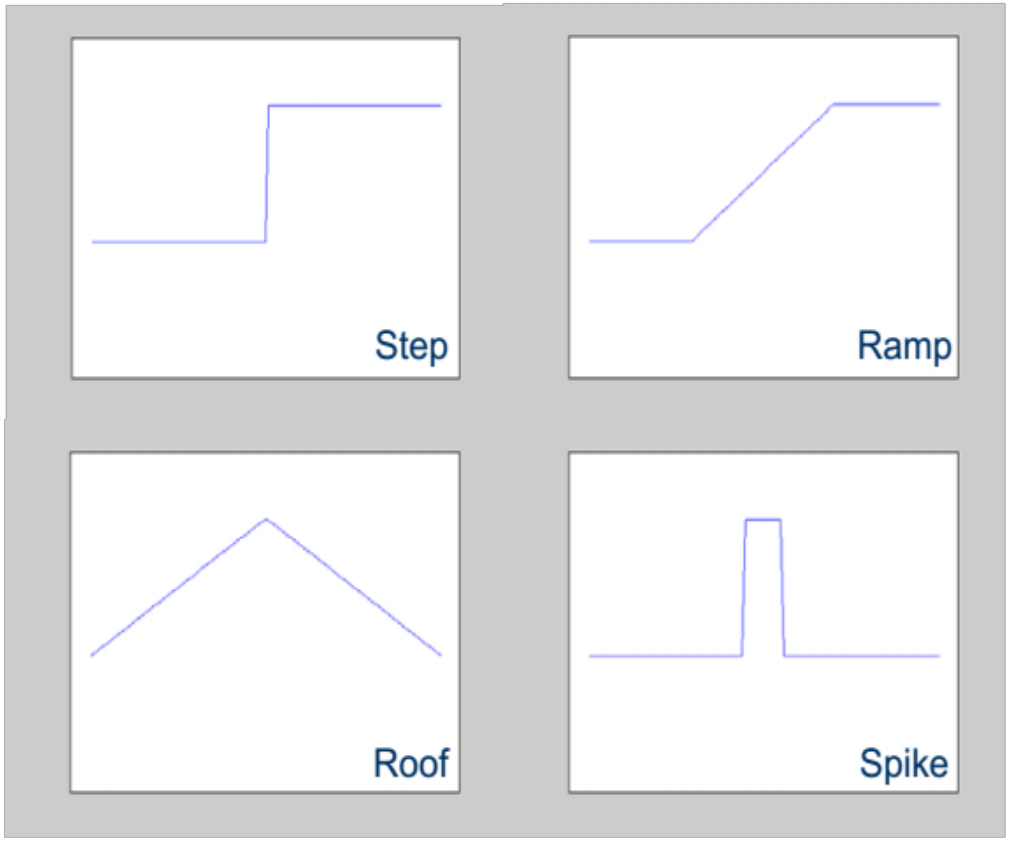
Edge Detection

- Goal: Identify sudden changes (discontinuities) in an image



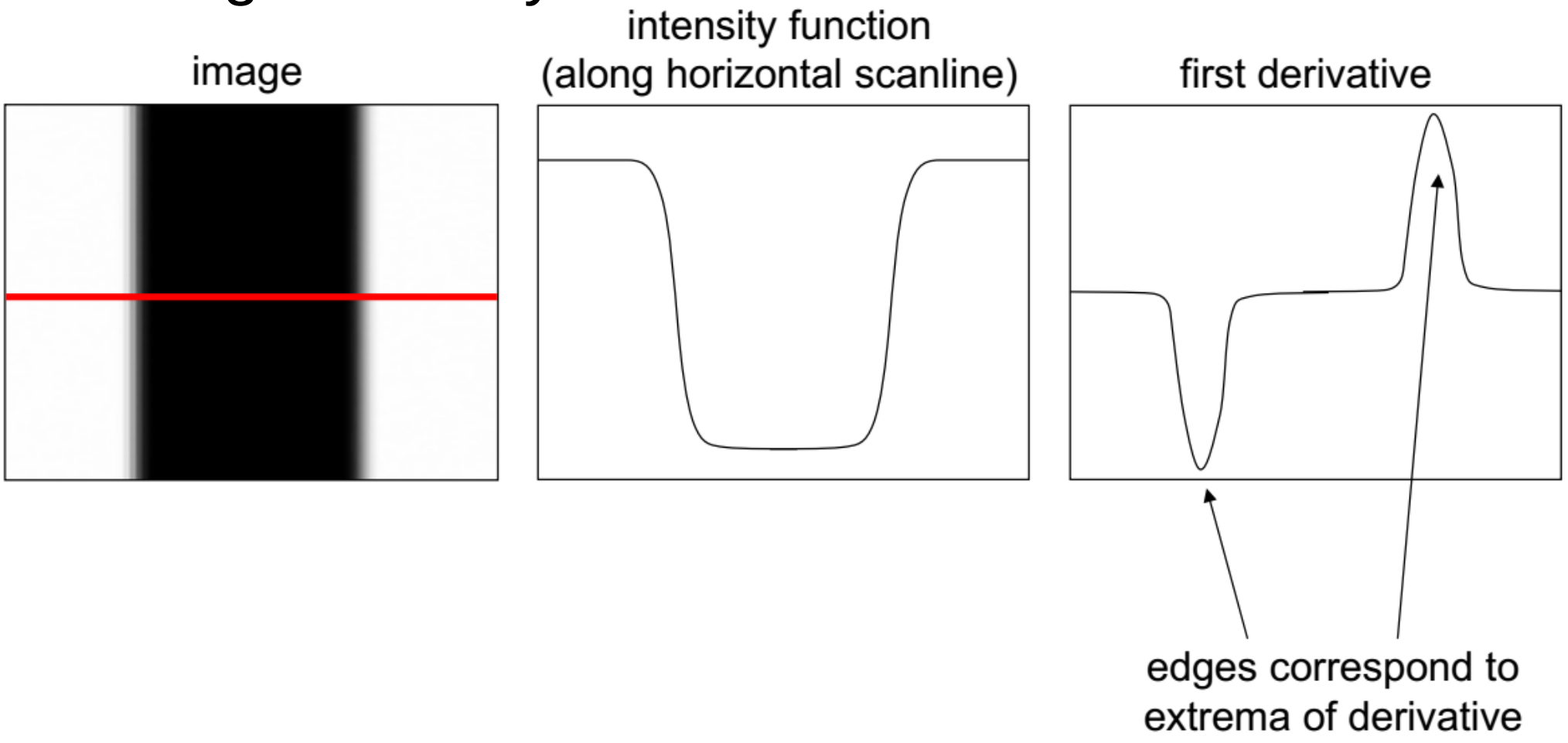
What is an Edge?

- Discontinuity of intensities in the image
- Edge models
  - Step
  - Roof
  - Ramp
  - Spike

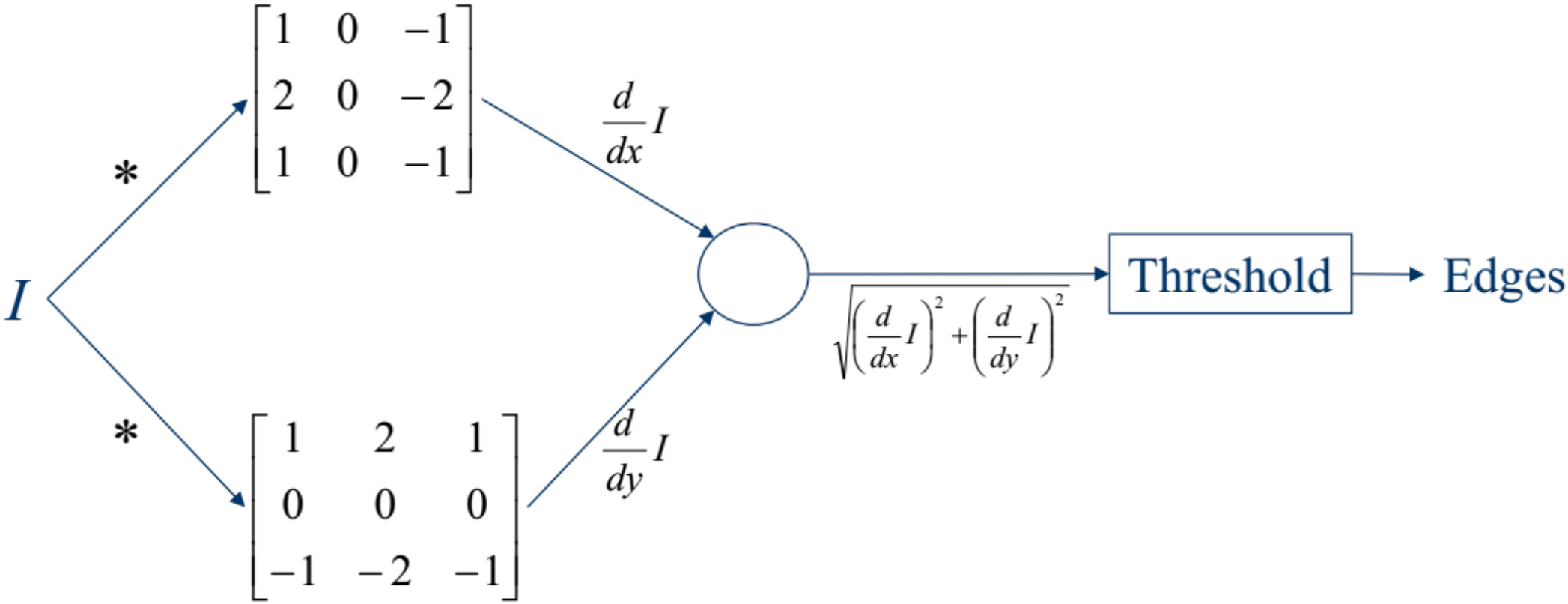


Characterizing edges

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

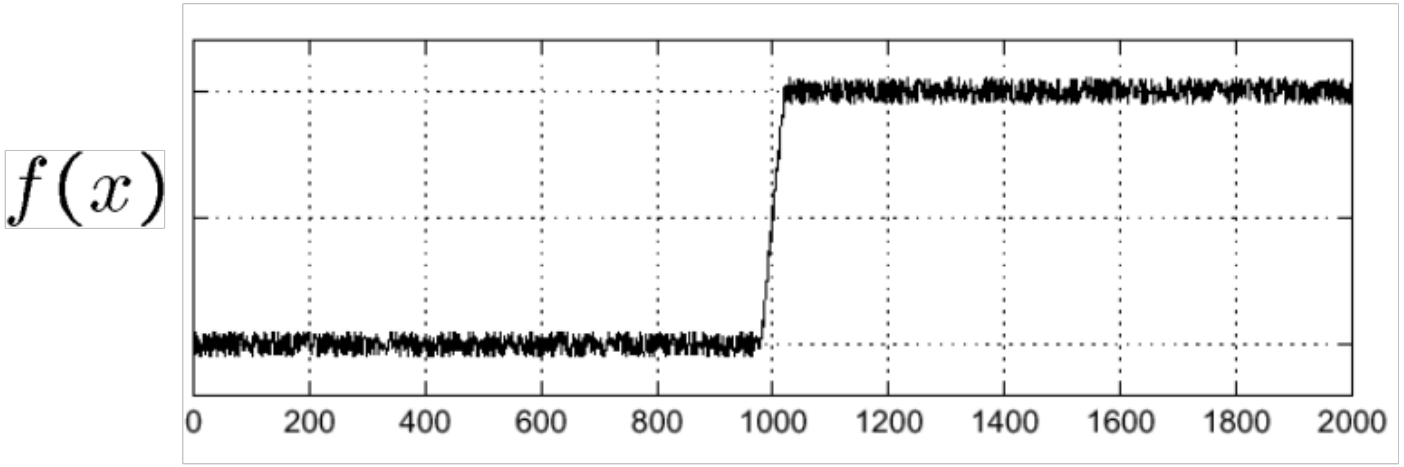


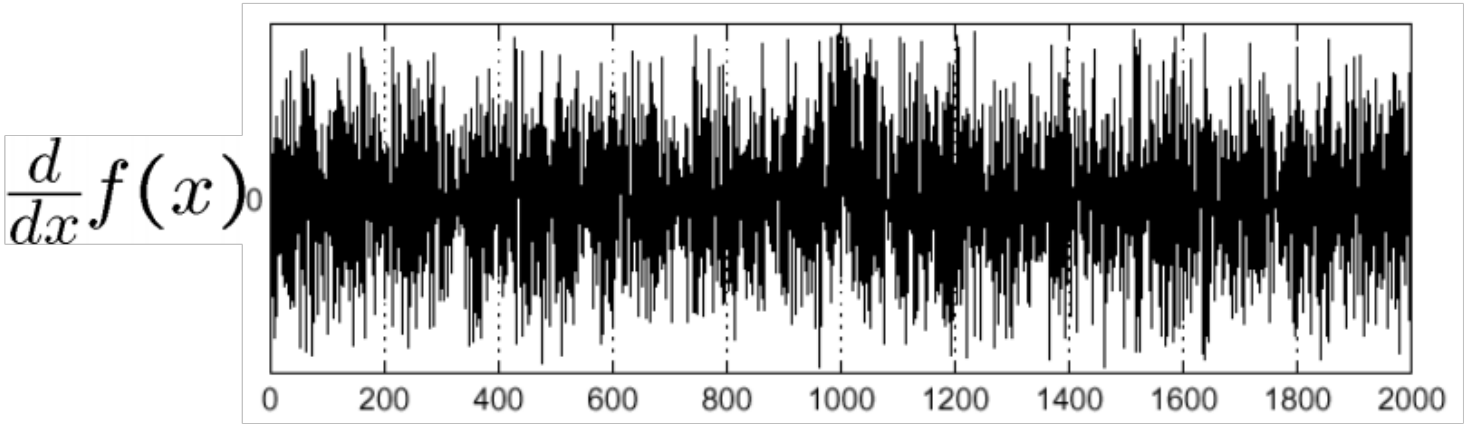
Sobel Edge Detector



Effects of noise

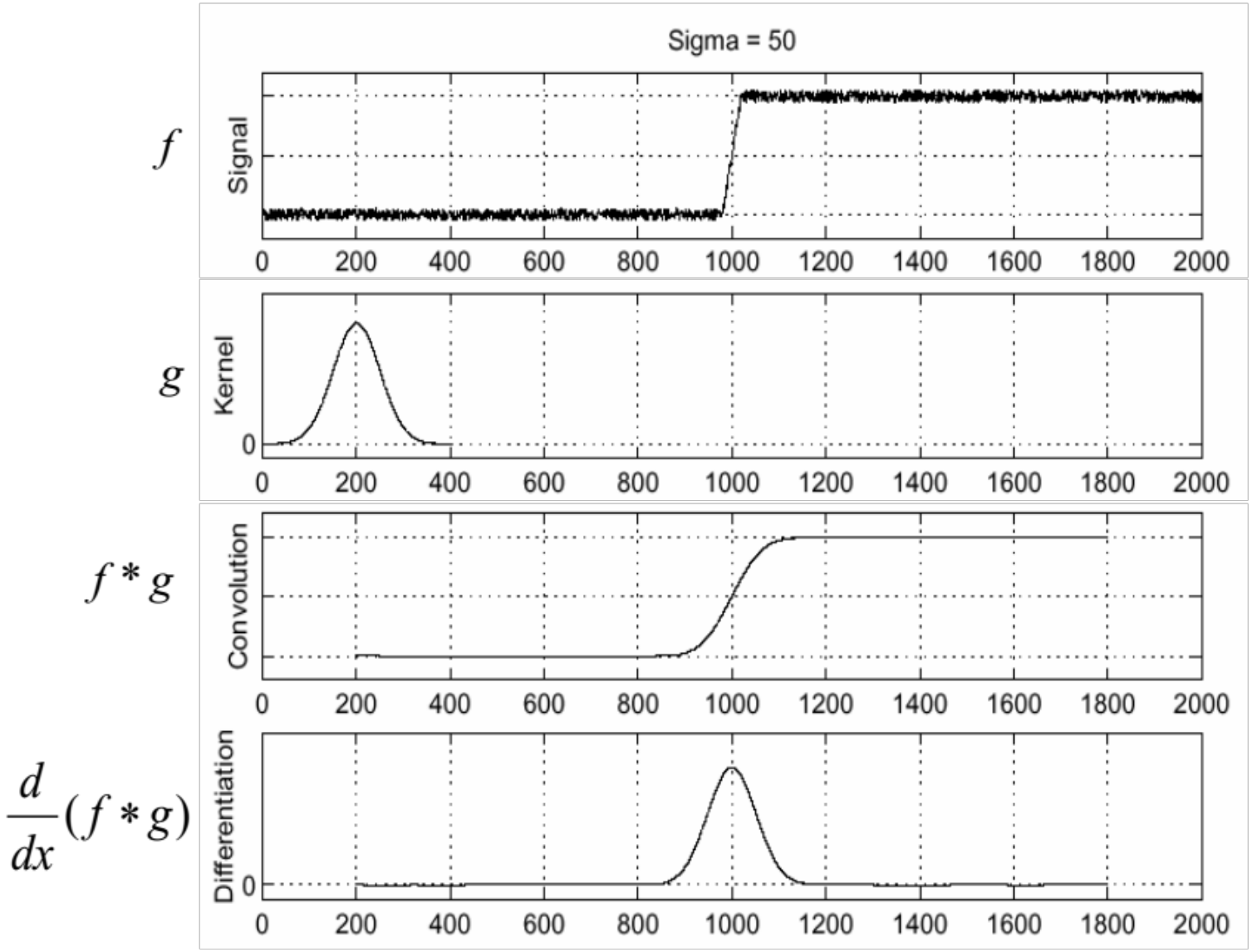
- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal





Where is the edge?

Solution: smooth first



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

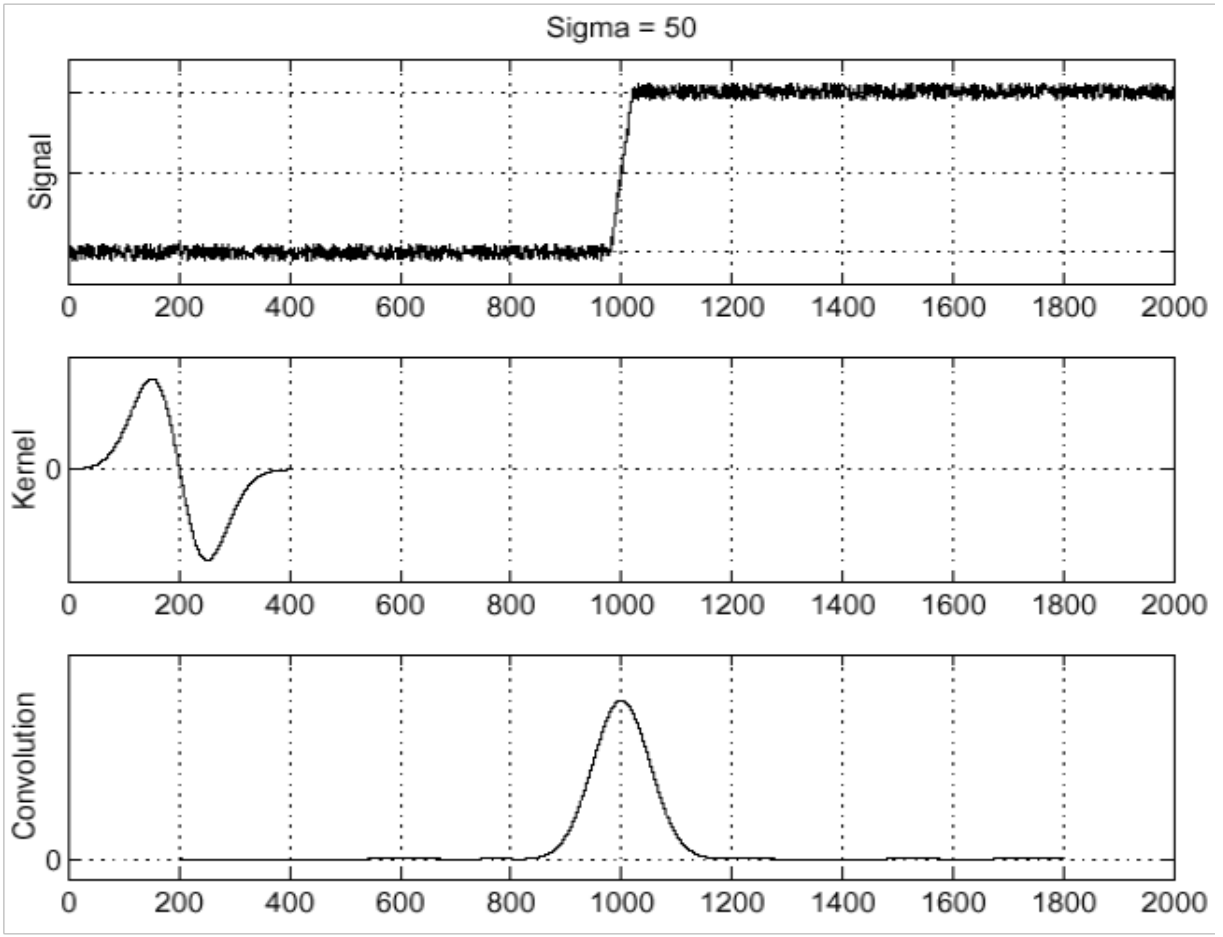
Convolution derivative property

$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

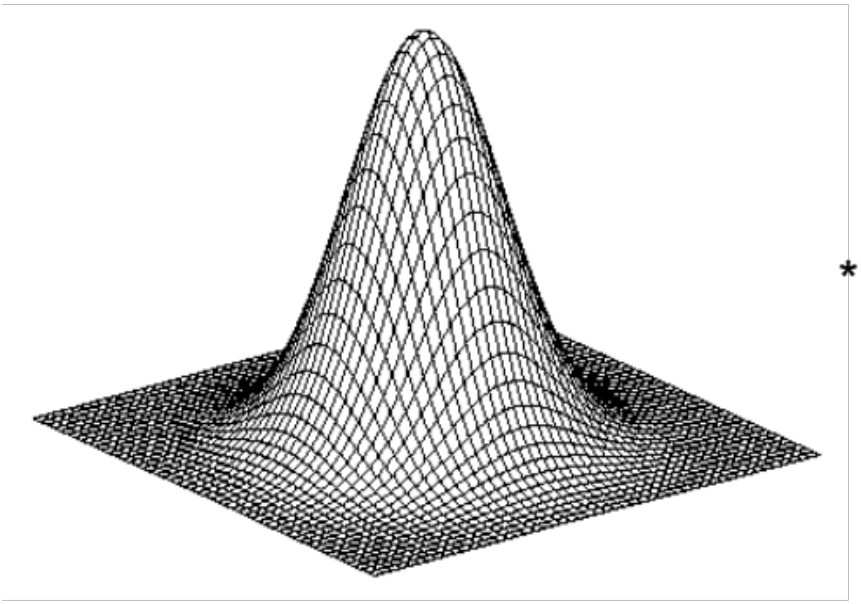
f

$\frac{d}{dx} g$

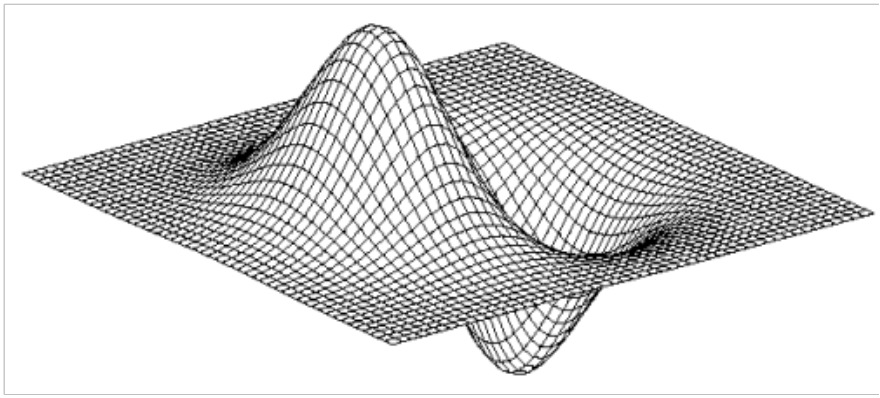
$f * \frac{d}{dx} g$



Derivative of Gaussian filter



* [1 -1] =



Canny Edge Detector

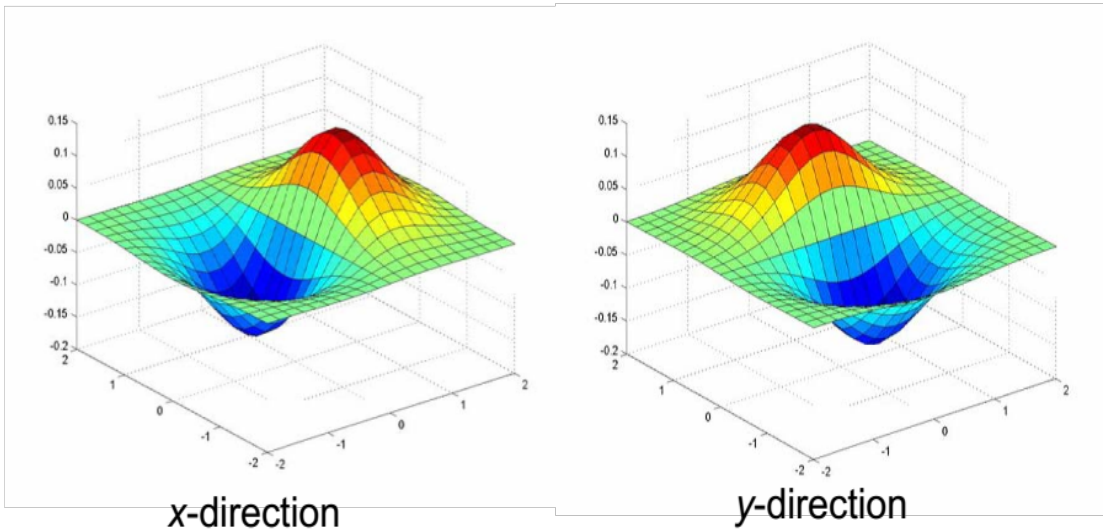
- Steps
  - Smooth image with Gaussian filter
  - Compute derivative of filtered image
  - Find magnitude and orientation of gradient
  - Apply “Non-maximum Suppression”
  - Apply “Hysteresis Threshold

3.Gradient magnitude and gradient direction

(S_x, S_y) Gradient Vector

$$\text{magnitude} = \sqrt{(S_x^2 + S_y^2)}$$

$$\text{direction} = \theta = \tan^{-1}\frac{S_y}{S_x}$$



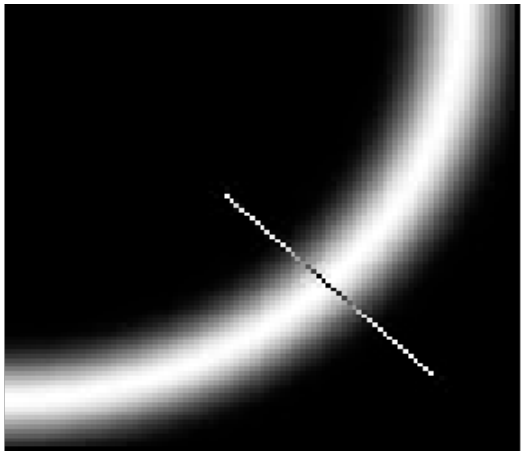
x-direction

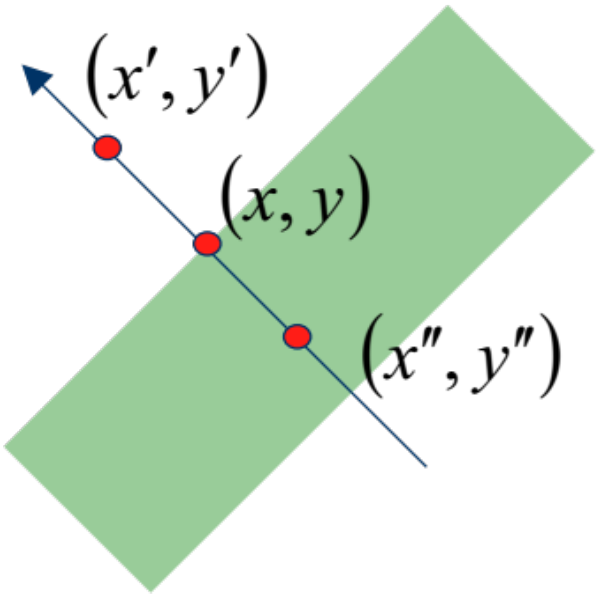
y-direction

Gradient Magnitude

gradient direction

3.Non-Maximum Suppression





$$M(x,y)=\begin{cases}|\nabla S|(x,y) & \text{if } |\nabla S|(x,y)>|\Delta S|(x',y') \\ & \&\ |\Delta S|(x,y)>|\Delta S|(x'',y'') \\ 0 & \text{otherwise}\end{cases}$$

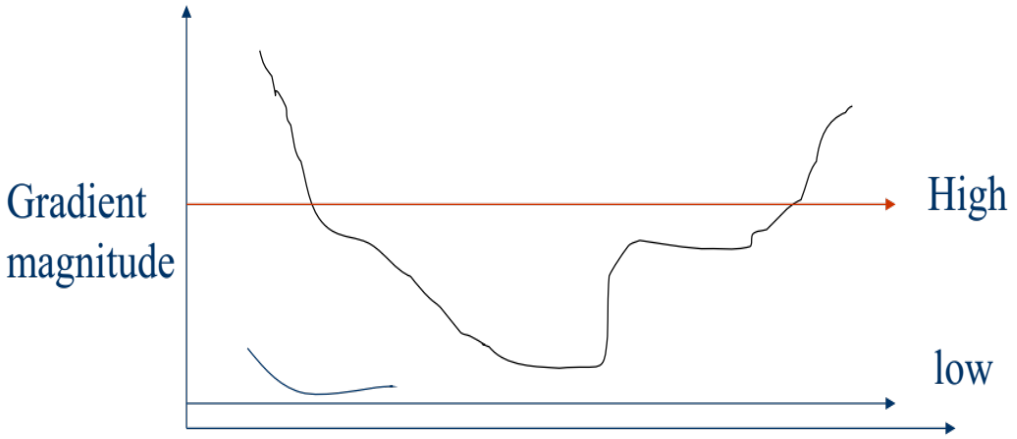
x' and **x**" are the neighbors of **x** along normal direction to an edge

Before Non-Maximum Suppression

After Non-Maximum Suppression

4.Hysteresis Thresholding

- If the gradient at a pixel is
  - above “High”, declare it as an ‘edge pixel’
  - below “Low”, declare it as a “non-edge-pixel”
  - between “low” and “high”
    - Consider its neighbors iteratively then declare it an“edge pixel” if it is connected to an ‘edge pixel’ directly.



Effect of σ (Gaussian kernel spread/size)

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

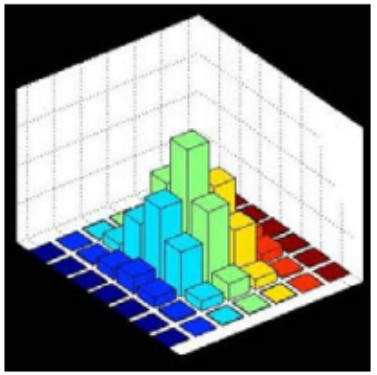
The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features

Source: S. Seitz

Gaussian smoothing

$$\underbrace{\vec{S}}_{\text{smoothed image}} = \underbrace{\vec{g}}_{\text{Gaussian filter}} * \underbrace{\vec{I}}_{\text{image}} \qquad g = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



Find Laplacian

$$\Delta^2 S = \overbrace{\frac{\partial^2}{\partial x^2} S}^{\text{second order derivative in } x} + \overbrace{\frac{\partial^2}{\partial y^2} S}^{\text{second order derivative in } y}$$

- ∇ is used for gradient (first derivative)
- Δ^2 is used for Laplacian (Second derivative)

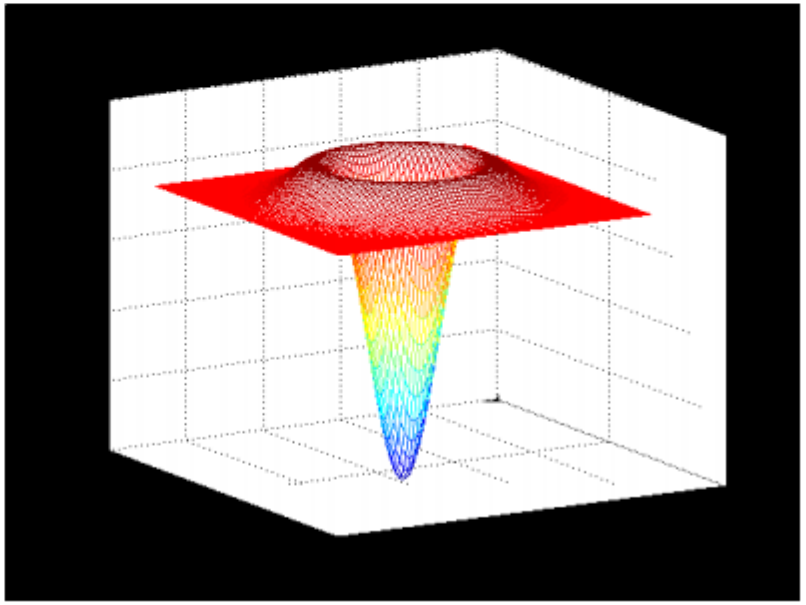
Deriving the Laplacian of Gaussian (LoG)

$$\Delta^2 S = \Delta^2 (g * I) = (\Delta^2 g) * I$$

$$g = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$g_x = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}} \left(-\frac{2x}{2\sigma^2}\right)$$

$$\Delta^2 g = -\frac{1}{\sqrt{2\pi}\sigma^3} \left(2 - \frac{x^2+y^2}{\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$



Approximation of LoG by Difference of Gaussians:

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \Delta^2 G$$

$$\text{Typical values}: \sigma = 1.6;\ k = \sqrt{2}$$

Smooth image by Gaussian filter ➔ S

Apply Laplacian to S

- Used in mechanics, electromagnetics, wave theory, quantum mechanics and Laplace equation

Find zero crossings

- Scan along each row, record an edge point at the location of zero-crossing.
- Repeat above step along each column

I $I * \left(\Delta^2 g\right)$ Zero crossings of $\Delta^2 S$



SIFT - Key Point Extraction

- Stands for Scale Invariant Feature Transform
- Similar to the one used in primate visual system (human, ape, monkey, etc.)
- Transforms image data into scale invariant coordinates

D. Lowe. Distinctive image features from scale-invariant key points., International Journal of Computer Vision 2004.

Objective

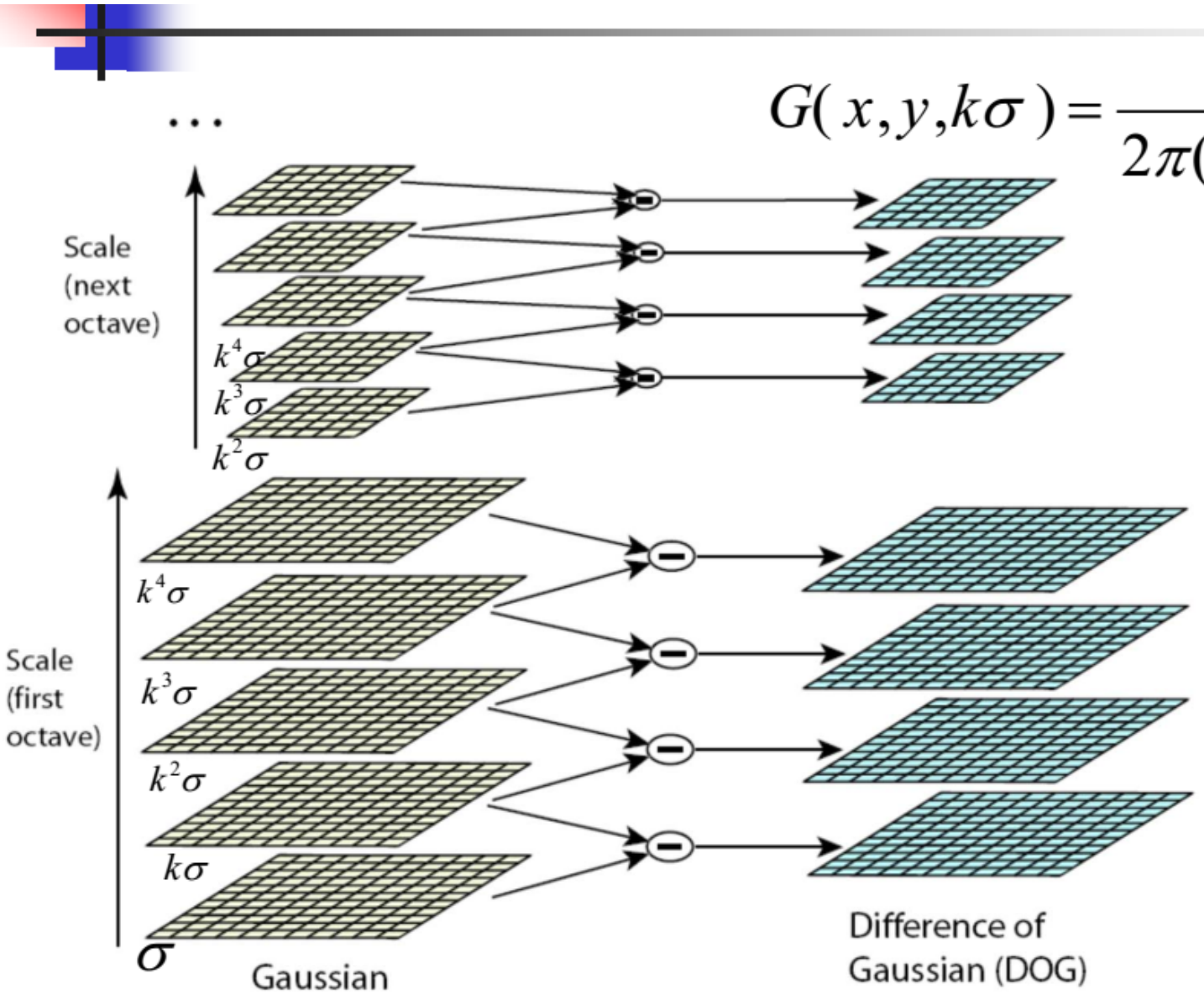
- Extract distinctive invariant features
  - Correctly matched against a large database of features from many images
- Invariance to image scale and rotation
- Robustness to
  - Affine (rotation, scale, shear) distortion,
  - Change in 3D viewpoint,
  - Addition of noise,
  - Change in illumination.

Steps for Extracting Key Points (SIF Points)

- Scale space peak selection
  - Potential locations for finding features
- Key point localization
  - Accurately locating the feature key points
- Orientation Assignment
  - Assigning orientation to the key points
- Key point descriptor
  - Describing the key point as a high dimensional vector (128) (SIFT Descriptor)

Building a Scale Space

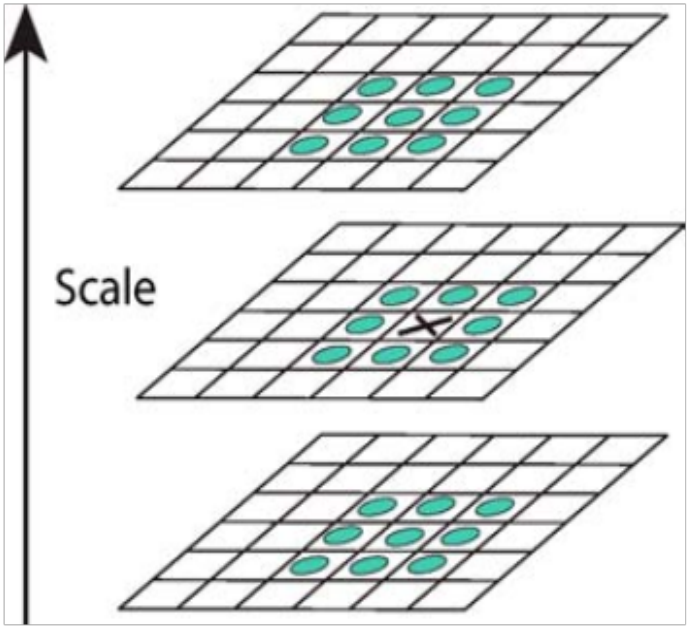
$$G(x, y, k\sigma) = \frac{1}{2\pi(k\sigma)^2} e^{-(x^2+y^2)/2k^2\sigma^2}$$



$$k = \sqrt{2}$$

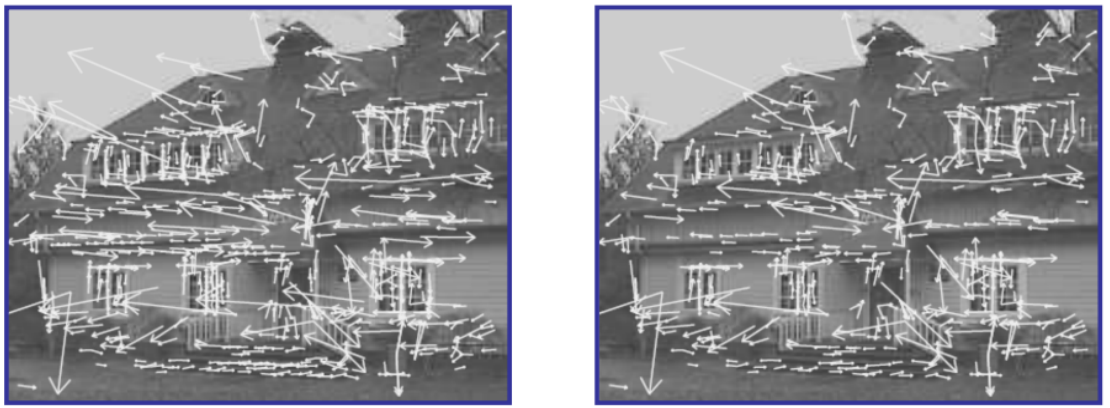
Scale Space Peak Detection

- Compare a pixel (X) with 26 pixels in current and adjacent scales (Green Circles)
- Select a pixel (X) if larger/smaller than all 26 pixels
- Large number of extrema, computationally expensive
  - Detect the most stable subset with a coarse sampling of scales



Key Point Localization

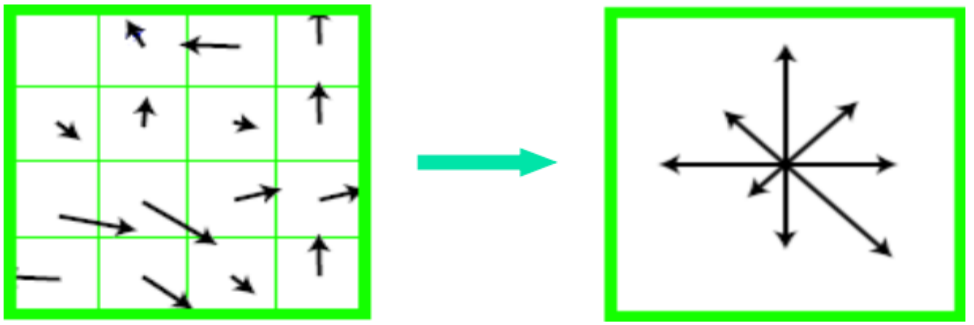
- Candidates are chosen from extrema detection
- 2 steps
  - Initial Outlier Rejection (Taylor Series)
  - Further Outlier Rejection (principal curvatures)



from 729 key points to 536 key points.

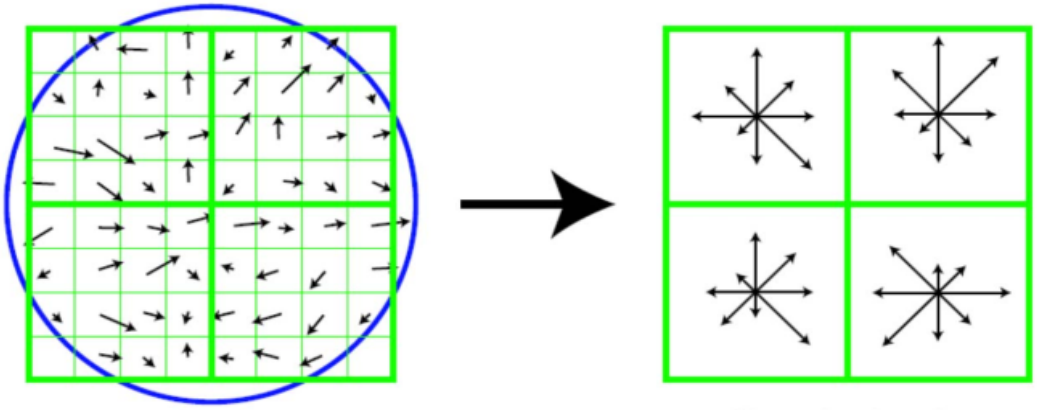
Orientation Assignment

- To achieve rotation invariance
- Compute central derivatives, gradient magnitude and direction of L (smooth image) at the scale of key point (x,y)
- Create a weighted direction histogram in a neighborhood of a key point (36 bins)
- Select the peak as direction of the key point



SIFT Descriptor

- Compute relative orientation and magnitude in a 16x16 neighborhood at key point
- Form weighted histogram (8 bin) for 4x4 regions
  - Weight by magnitude and spatial Gaussian
  - Concatenate 16 histograms in one long vector of 128 dimensions



Regarding Course Programming Assignments

- Python
  - Basics (PythonLearn, http://www.pythonlearn.com/slides.php)
  - Terminal or IDE: Pycharm, Spyder, Jupyter Notebook.
  - Libraries:
    - Scikit-learn: Built on NumPy, SciPy, and matplotlib
    - Theano, Keras, Tensorflow for Deep Learning

- Extra Class for Python tutorial??

Thank you