מבוא לראייה ממוחשבת – 22928
2016א

מנחה: אמיר אגוזי
egozi5@gmail.com

מפגש מס' 1

הנושאים להיום:

- אדמיניסטרציה
- מבוא לקורס
- תמונות
- צבע
- רעש ופילטרים מרחביים
- איתור סף – edge detection

אדמיניסטרציה

- **מנחה** (ומרכז): אמיר אגוזי, egozi5@gmail.com
  - שעות קבלה טלפוניות - שלישי 17:00-19:00, 050-2773676.
- **חומר הקורס**
  - 12 הרצאות של ד"ר טל הסנר (הרצאה בשבוע)
  - בנוסף לחומרים נוספים שיתפרסמו באתר
- **מפגשי הנחיה**
  - כל שבועיים (בדר"כ) = שתי הרצאות/יחידות
  - יש לצפות בהרצאות הרלוונטיות
  - במפגשים - דיון והרחבה בנושאי ההרצאות

מטלות

- 3 מטלות – ניקוד כל אחת – 5%.
- 2 מטלות תיכנות ב – Python, מטלה 1 – שאלות פתוחות.
- חובה להגיש 2 מתוך 3.
- הגשה דרך מערכת מטלות בלבד.

פרויקט מסכם - תחרות

- התמודדות עם dataset "אמיתי"
- מימוש של שיטה קיימת
- פיתוח ומימוש רעיונות מקוריים
- הערכה יחסית

Python

- הוראות התקנה ל-Anaconda באתר.
- עבודה בעיקר עם OpenCV
- חבילות נוספות:
  - Numpy, matplotlib, scikit-learn
  - Scikit-image
  - Scipy.ndimage
  - PIL

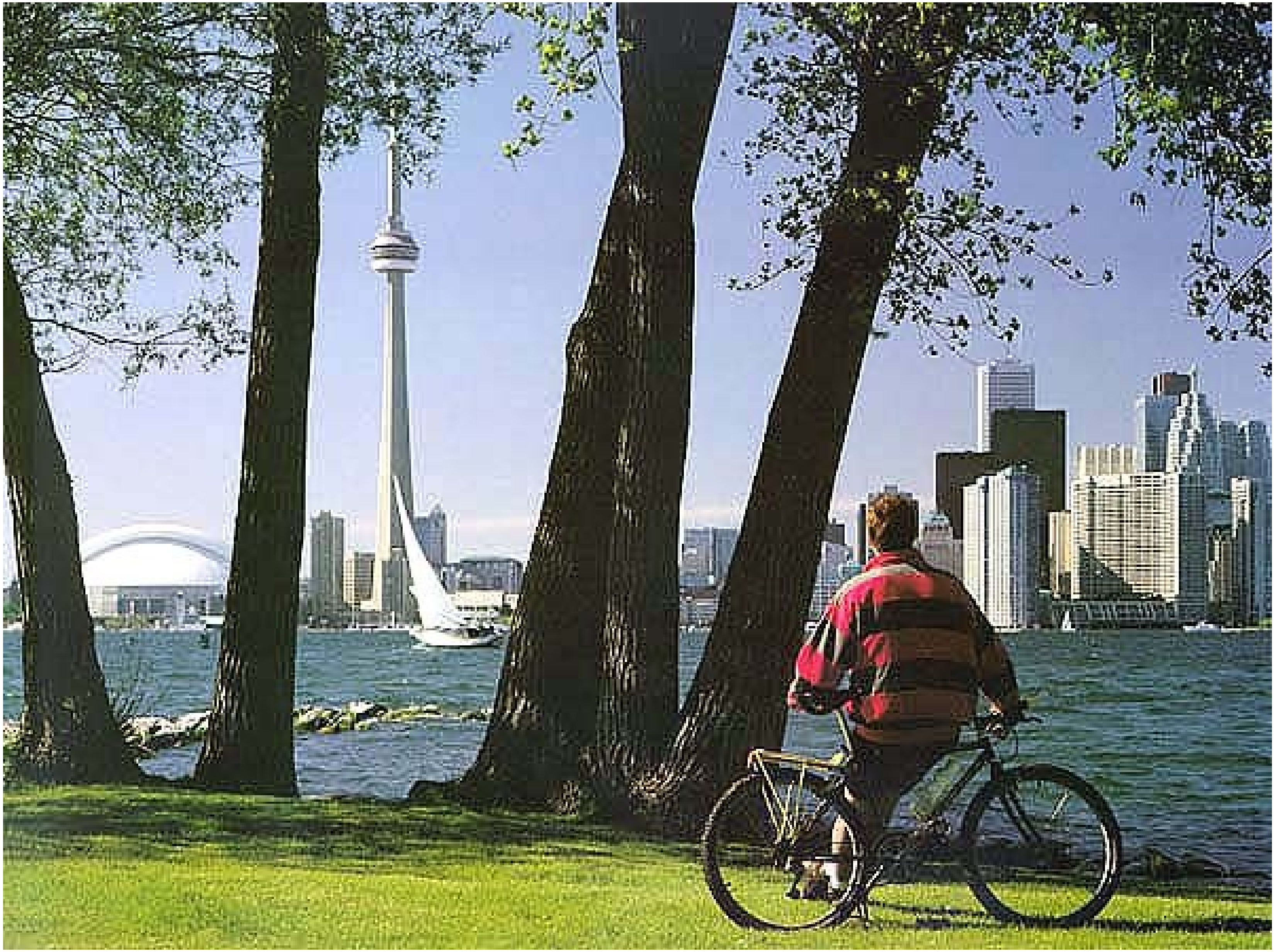
What is computer vision?

The goal of computer vision is to develop algorithms that allow computer to “see”.

Also called:

- Image understanding
- Image analysis
- Machine vision

General visual perception is hard

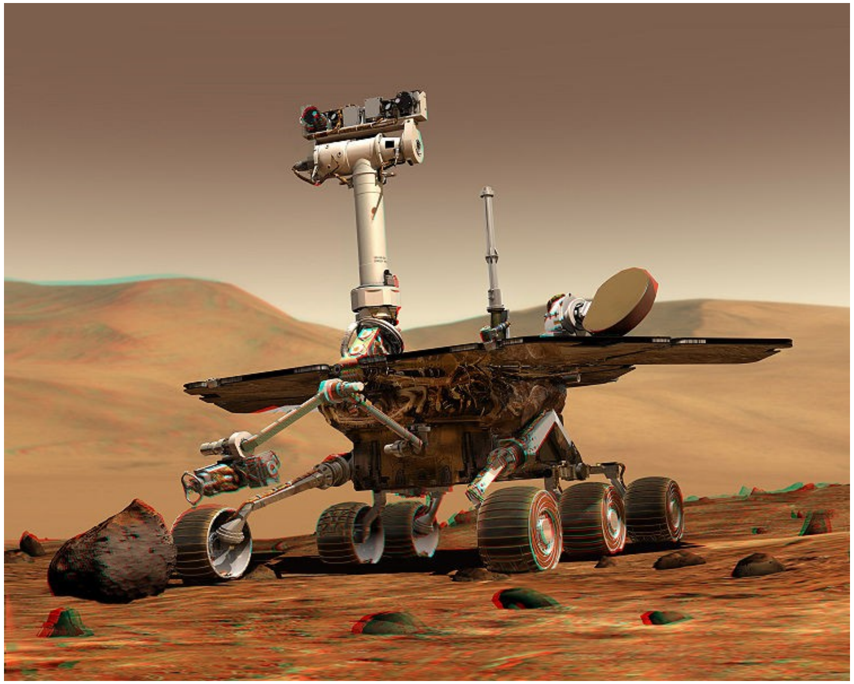


Our time

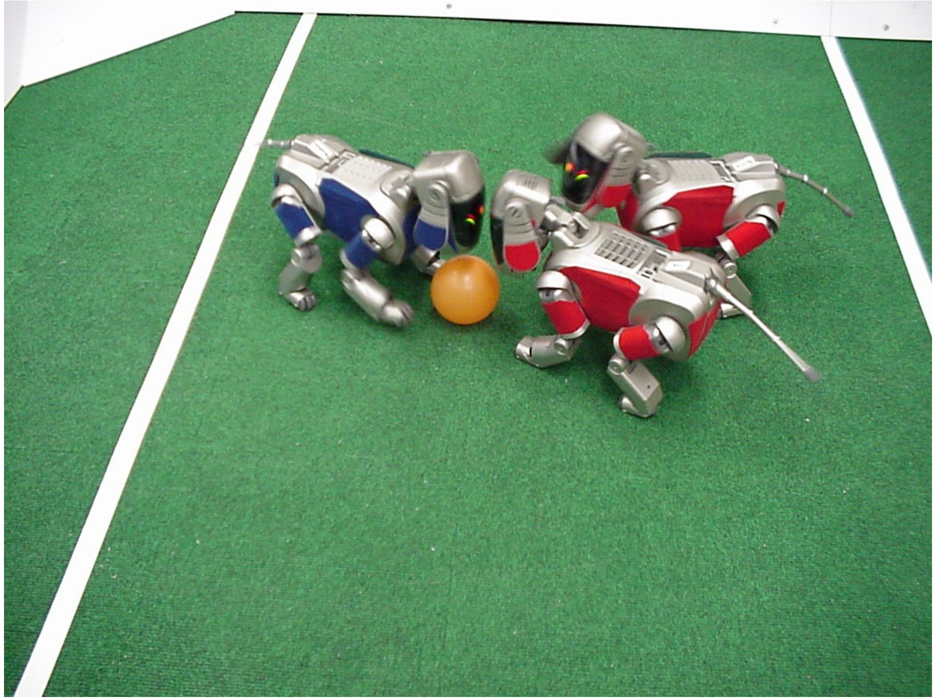
- It is a good time to do computer vision now, because:
  - Powerful computers
  - Inexpensive cameras
  - Algorithm improvements
  - Understanding of vision systems

Computer vision's applications

Robotics



NASA's Mars Spirit Rover
http://en.wikipedia.org/wiki/Spirit_rover



http://www.robocup.org/

Adapted from S. Seitz

3D Maps

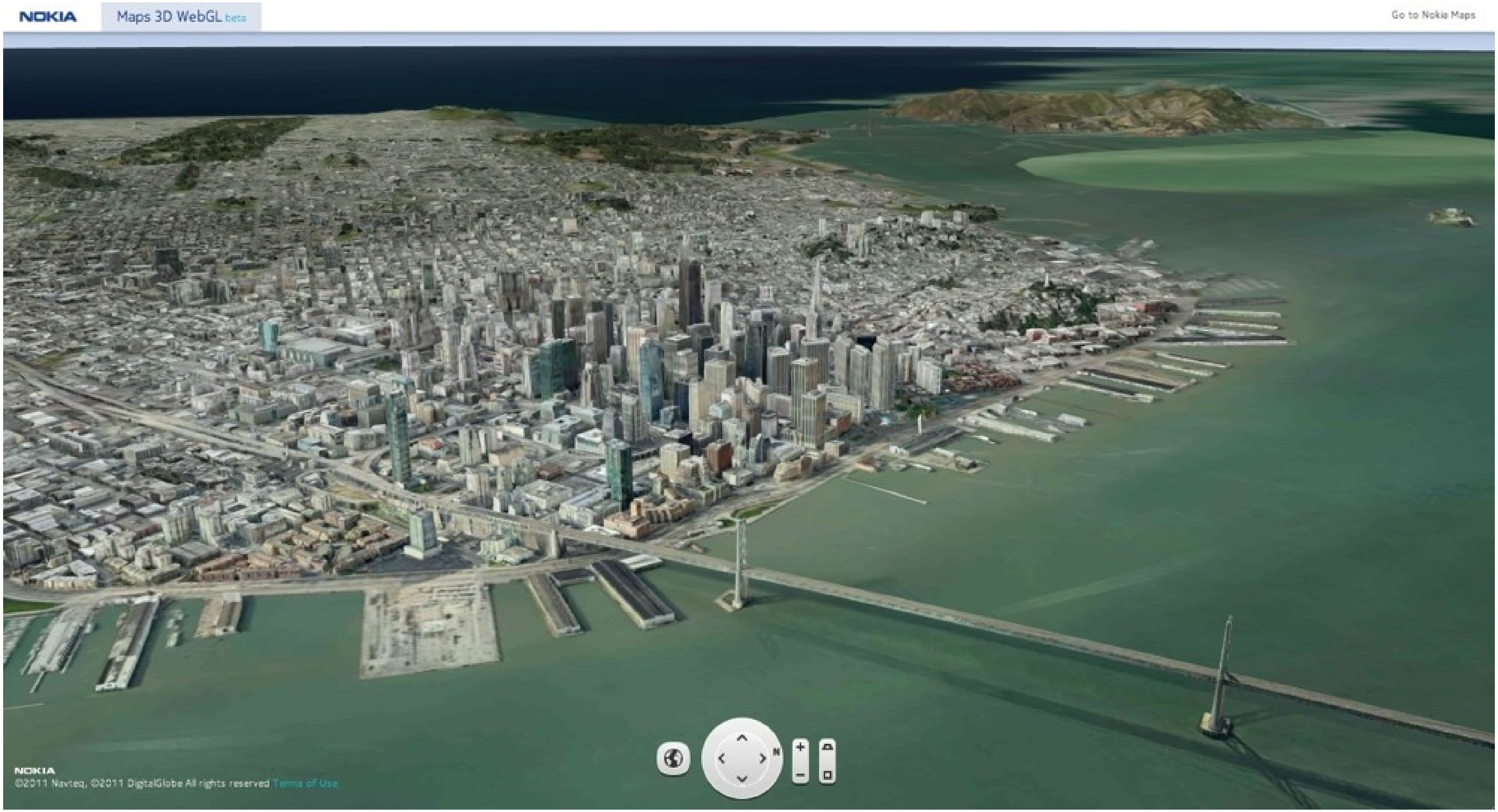


Image from Nokia's Maps 3D WebGL
(see also: Google Maps GL, Google Earth)

Adapted from S. Seitz

Motion capture

Microsoft's XBox Kinect

Adapted from S. Seitz

Face detection

Most digital cameras detect faces

Adapted from S. Seitz

Object recognition

[Google Goggles](#)
[Bing Vision](#)

Adapted from S. Seitz

Special effects: shape capture

The Matrix movies, ESC Entertainment, XYZRGB, NRC

Adapted from S. Seitz

Sports

Sportvision first down line
Nice explanation on www.howstuffworks.com

Adapted from S. Seitz

Smart cars

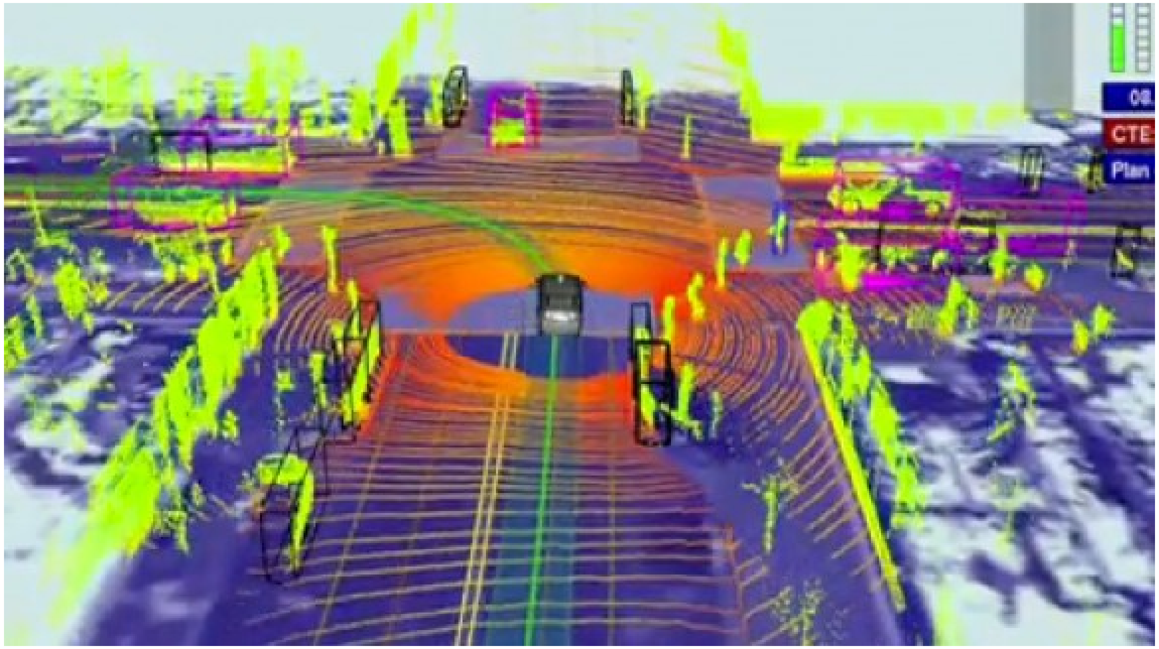
Slide content courtesy of Amnon Shashua

Mobileye

- Vision systems currently in high-end BMW, GM, Volvo models

Adapted from S. Seitz

Self-driving cars



“Our self-driving cars have now traveled nearly 200,000 miles on public highways in California and Nevada, 100 percent safely. They have driven from San Francisco to Los Angeles and around Lake Tahoe, and have even descended crooked Lombard Street in San Francisco. They drive anywhere a car can legally drive.”

- Sebastian Thrun, Google

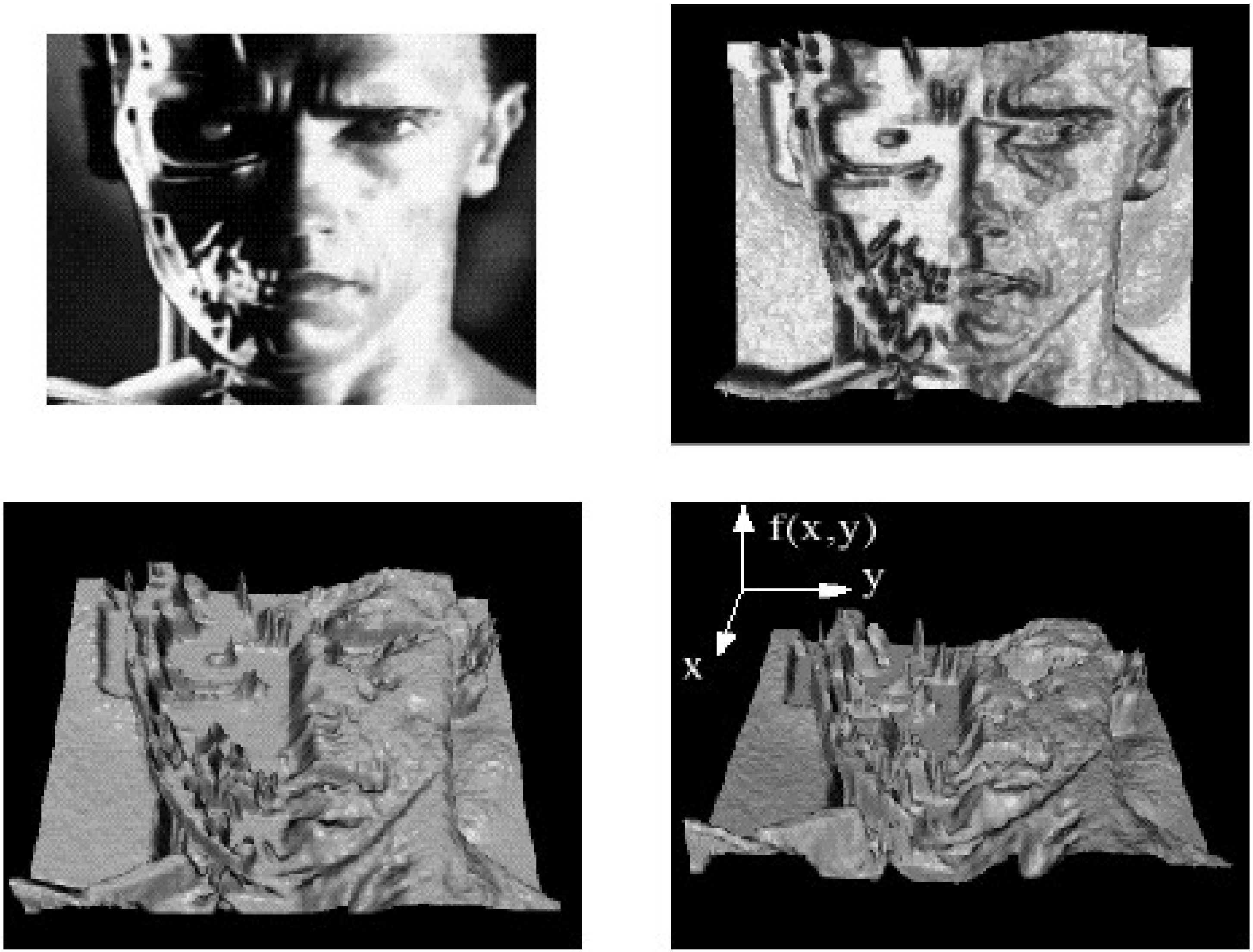
Adapted from S. Seitz

Computer vision's applications

- To learn more about vision applications and companies
  - David lowe maintains an excellent overview of vision companies
    - http://www.cs.ubc.ca/spider/lowe/vision.html

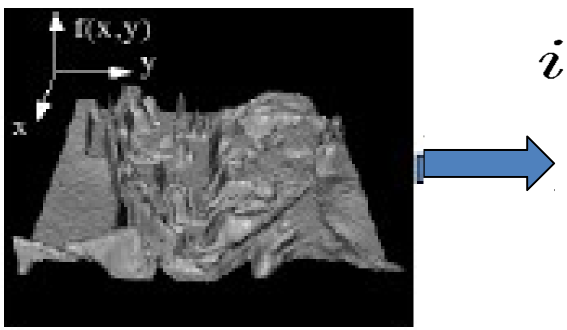
In the beginning ...

Images as functions



Digital images

- In computer vision we operate on **digital** (**discrete**) images:
  - **Sample** the 2D space on a regular grid
  - **Quantize** each sample (round to nearest integer)
- Image thus represented as a matrix of integer values.

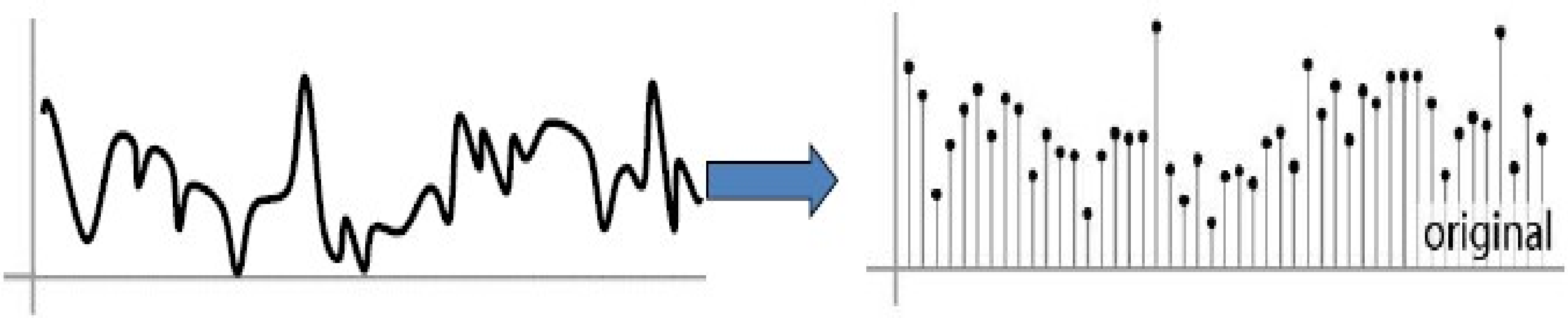


j

i

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

2D

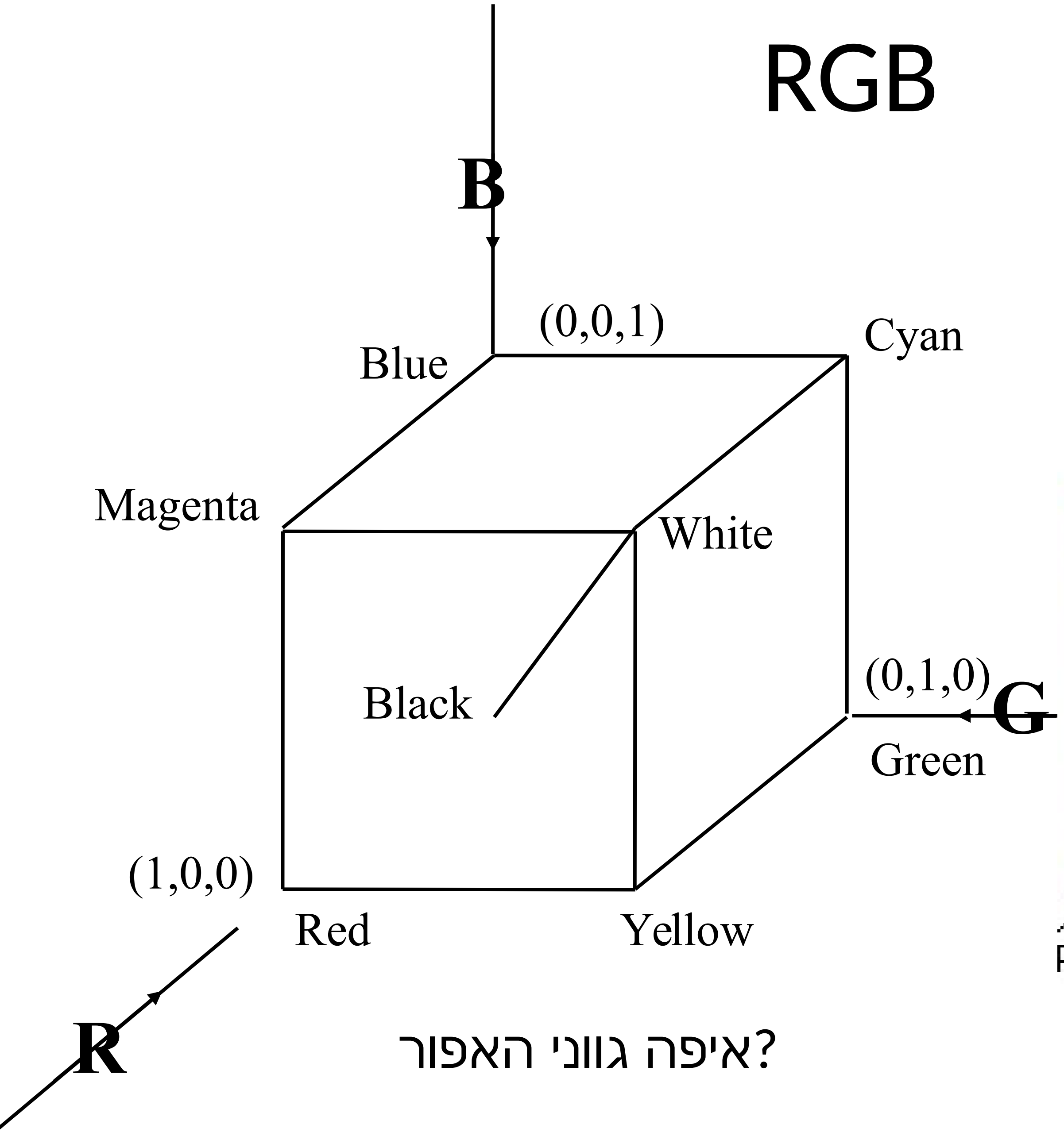


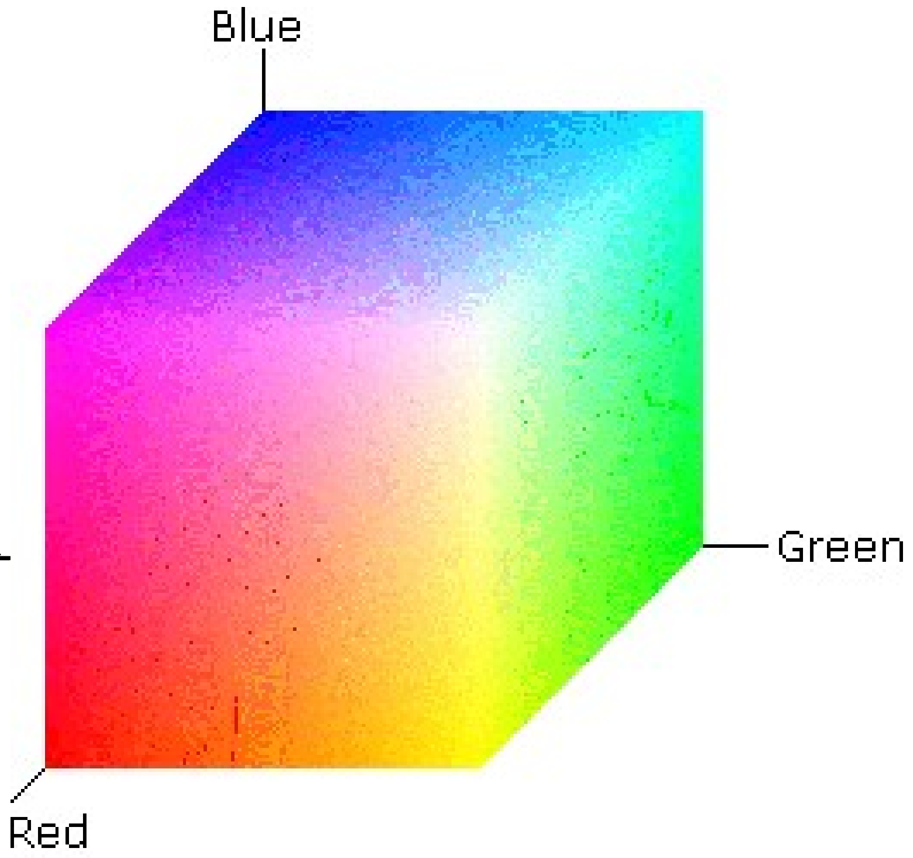
1D

Adapted from S. Seitz

צבע

RGB





איפה גווני האפור?

Color Spaces

- “facilitate the specification of colors in some standard, generally accepted way. In essence, [...] a specification of a coordinate system and a subspace within that system where each color is represented by a single point”

Gonzalez & Woods

- RGB, CIE-XYZ, HSV, YIQ, CIE-Lab, YCbCr, …

HSV color space

- **H**ue, **S**aturation, **V**alue (Brightness)
- Nonlinear – reflects topology of colors by coding **hue** as an angle
- Matlab: `hsv2rgb`,
- `rgb2hsv`.

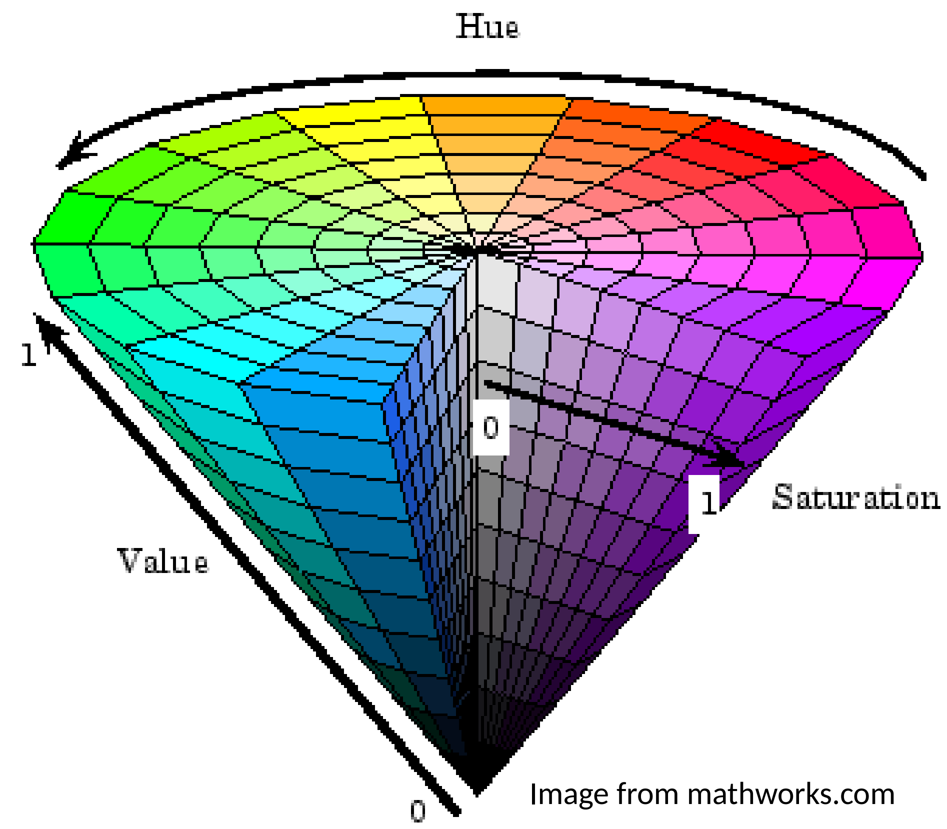
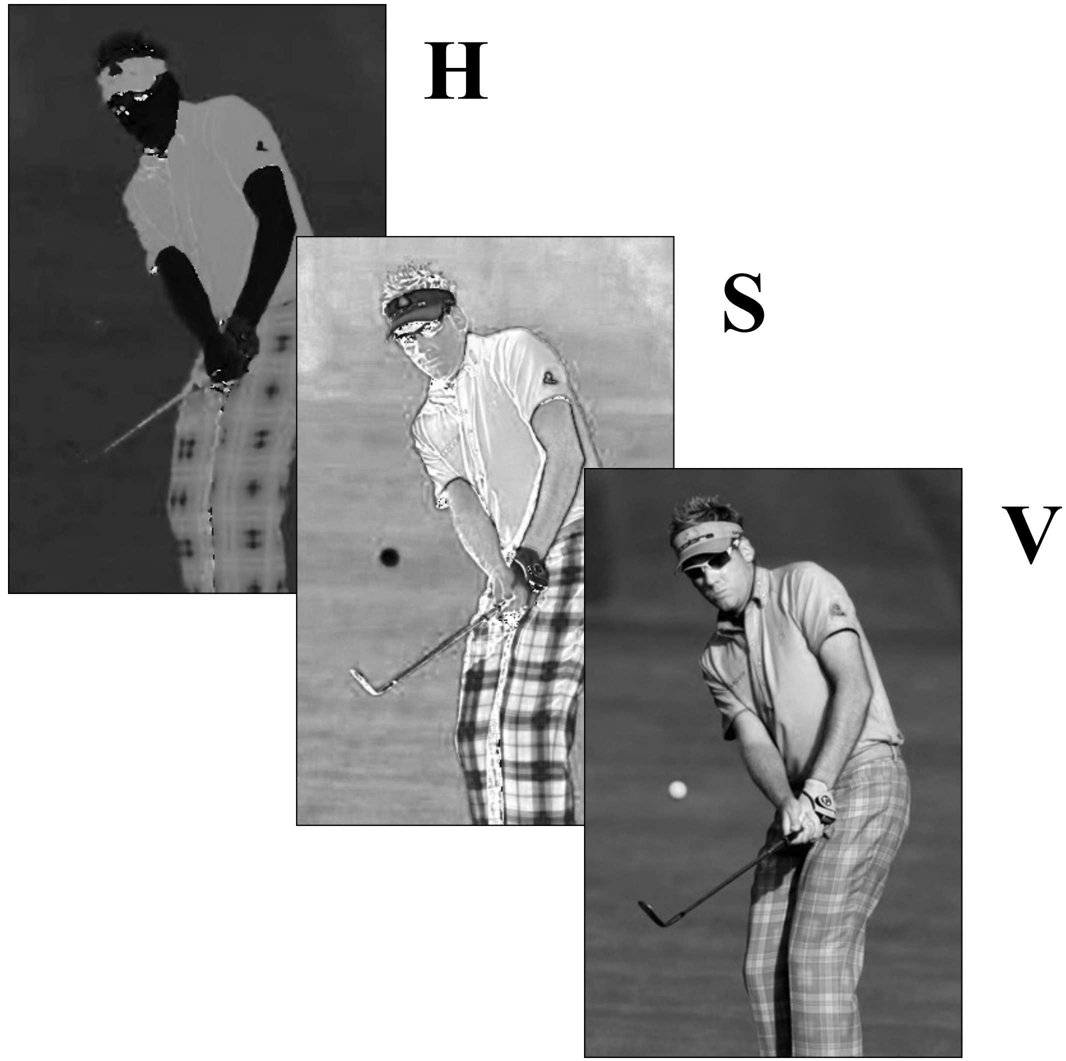


Image from mathworks.com

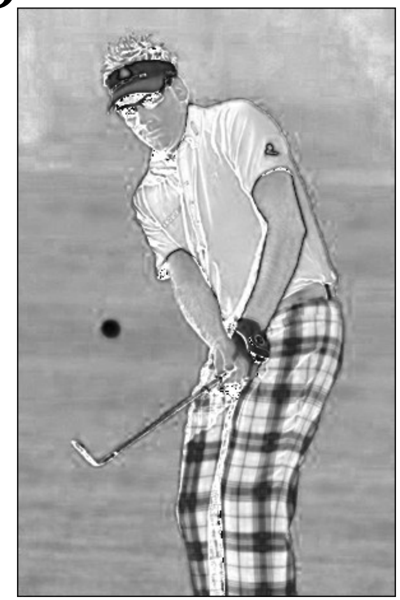
HSV



HSV

H

S



V

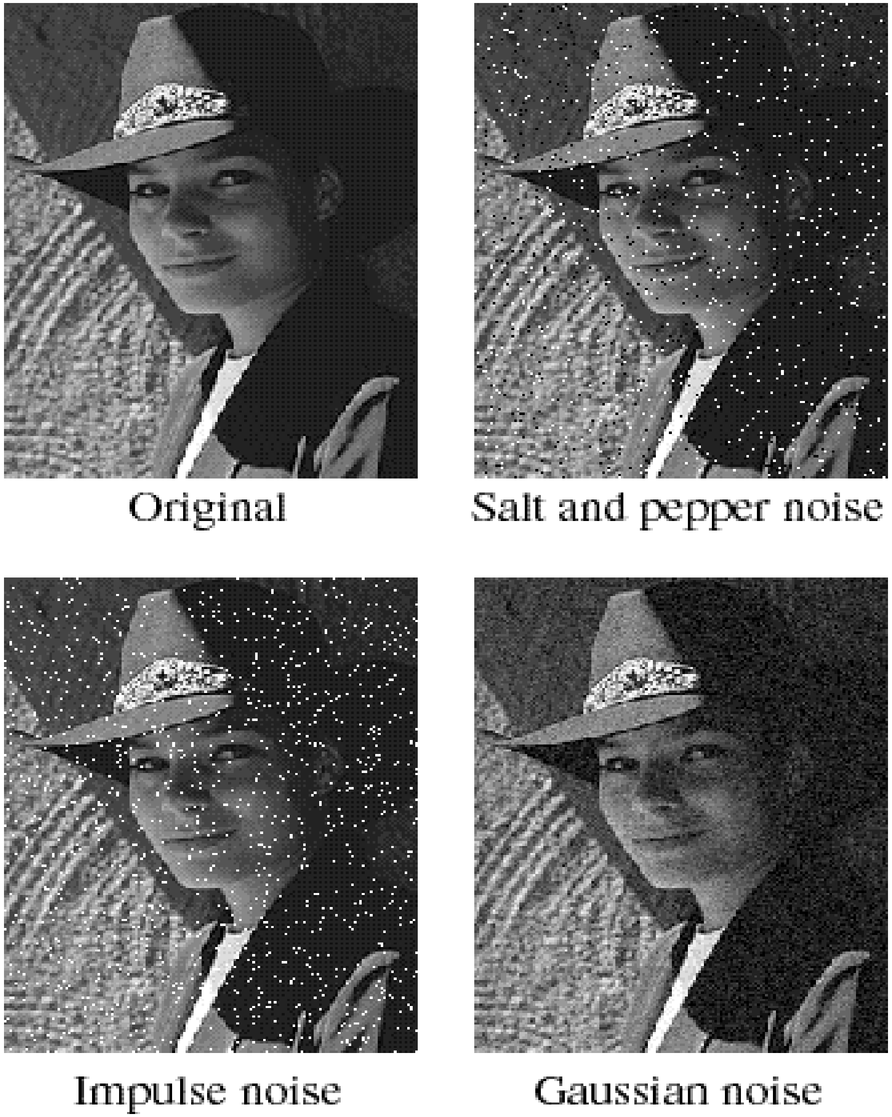
Python:

```
img = cv2.imread("golf.png")
# Convert BGR to HSV
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
plt.imshow(hsv[:,:,0],cmap='hsv')
```

רעש ופילטרים

Common types of noise

- **Salt and pepper noise**: random occurrences of black and white pixels
- **Impulse noise:** random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution



Source: S. Seitz

Linear filters

- Form a new image whose pixels are a weighted sum of the original pixel values, using the same set of weights at each point

Box filter

- Mask with positive entries, that sum to 1.
- Replace each pixel with an average of its neighborhood.
- If all weights are equal, it is called a **box filter**.

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Correlation filtering

Say the averaging window size is *(2k+1)x(2k+1)*:

$$G[i,j] = \underbrace{\frac{1}{(2k+1)^2}}_{\text{Attribute uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]}_{\text{Loop over all pixels in neighborhood around image pixel } F[i,j]}$$

Now generalize to allow different weights depending on neighboring pixel's relative position:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underbrace{H[u,v]}_{\text{Non-uniform weights}} F[i+u, j+v]$$

Correlation filtering

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

This is called cross-correlation, denoted

$$G = H \otimes F$$

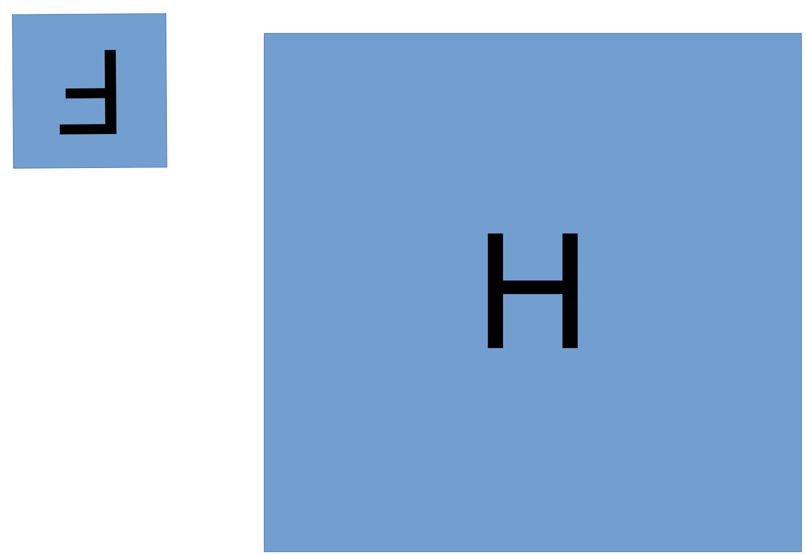
Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter "kernel" or "mask" H[u,v] is the prescription for the weights in the linear combination.

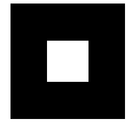
Convolution

$$G[i,j] = H * F$$
$$= \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

- Flip the filter in both dimension.



Smoothing by averaging

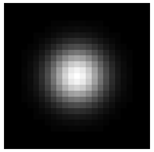


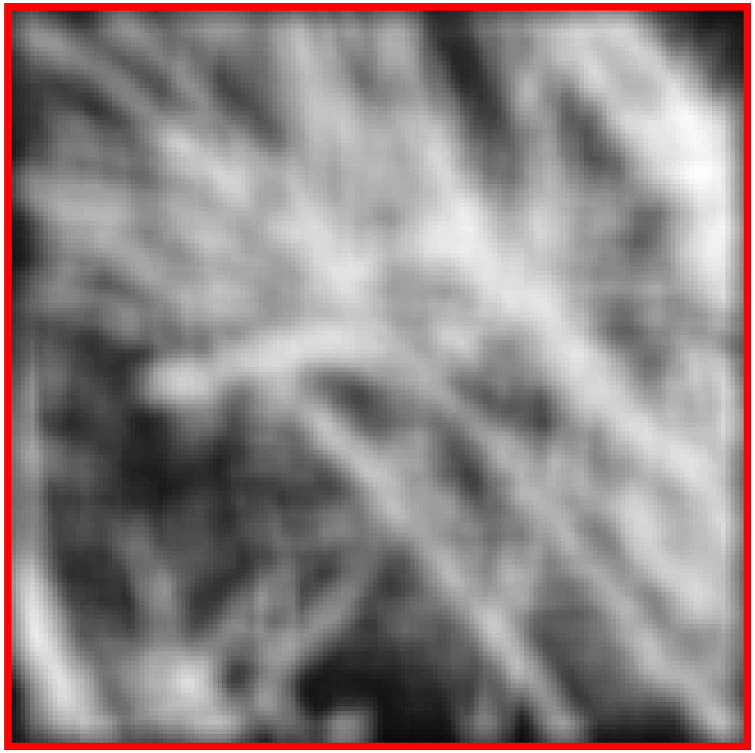
depicts box filter:
white = high value, black = low value

original

filtered

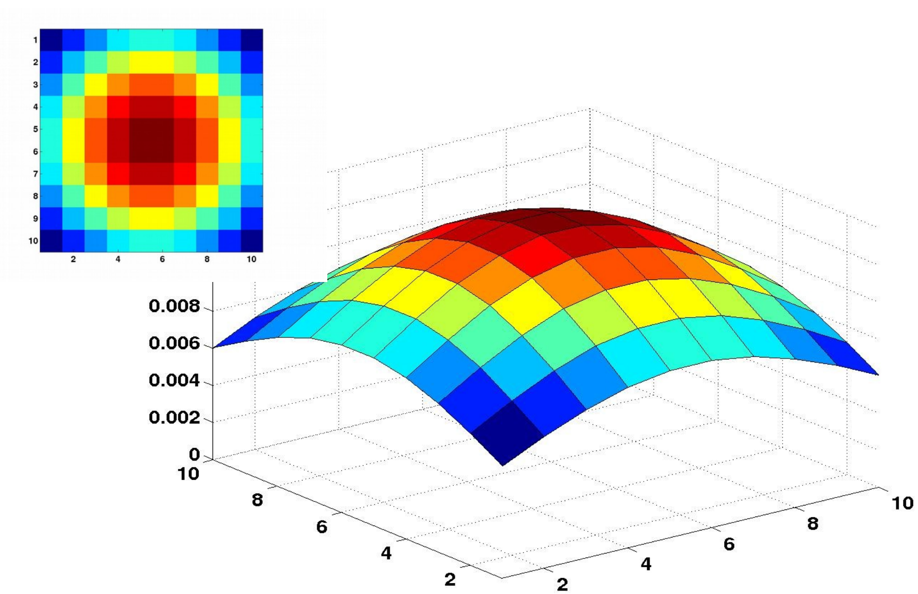
Smoothing with a Gaussian



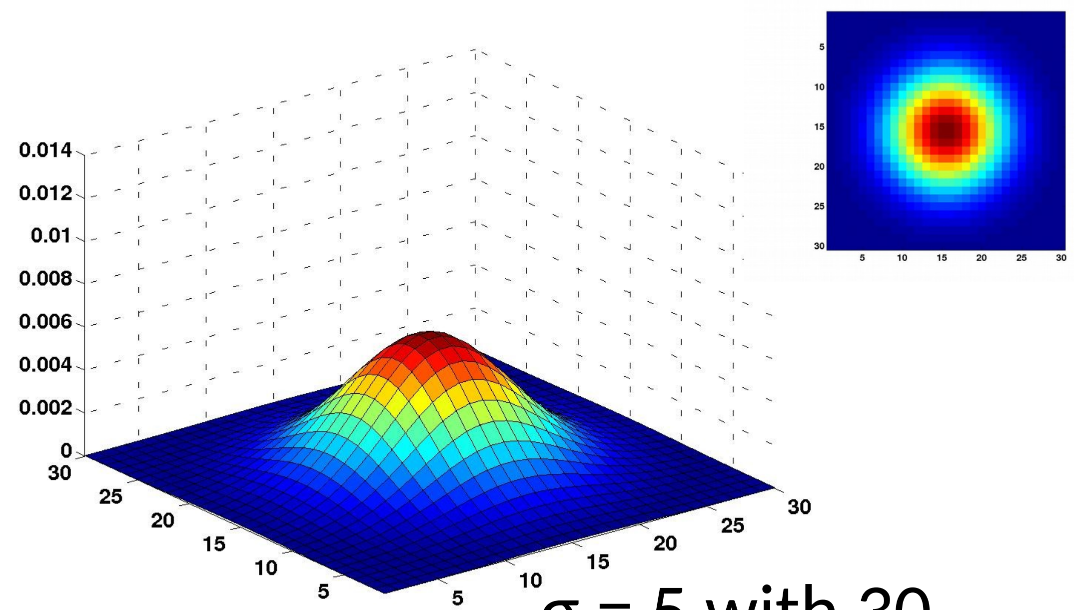


Gaussian filters

- What parameters matter here?
- **Size** of kernel or mask
  - Note, Gaussian function has infinite support, but discrete filters use finite kernels



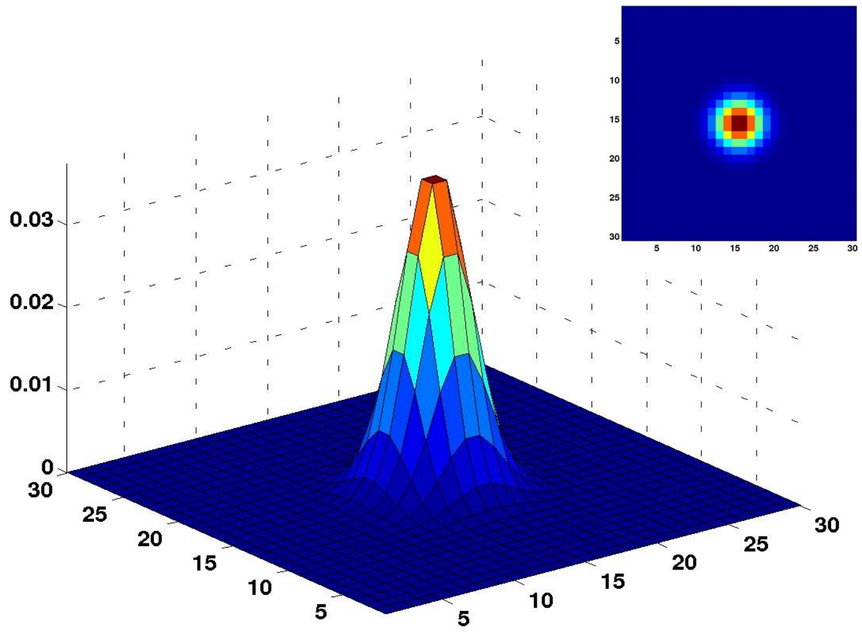
σ = 5 with 10 x 10 kernel



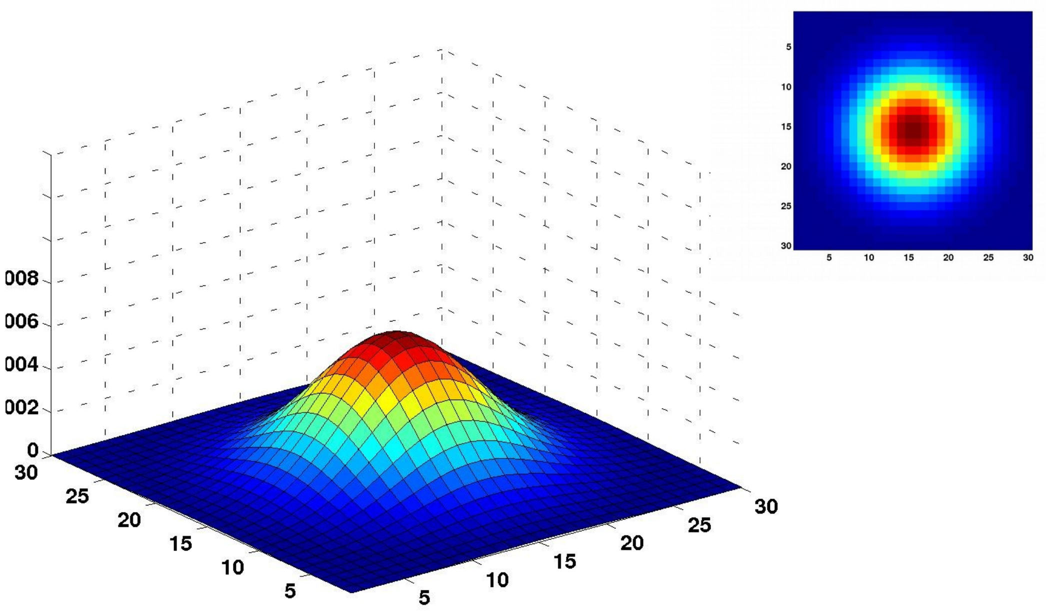
σ = 5 with 30 x 30 kernel

Gaussian filters

- What parameters matter here?
- **Variance** of Gaussian: determines extent of smoothing



σ = 2 with 30 x 30 kernel



σ = 5 with 30 x 30 kernel

Efficient Implementation - Separability

- In some cases, filter is separable,

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- Thus, using the associativity property,

$$(f * g) * h = f * (g * h)$$

- compute in two steps.

Separability of the Gaussian filter

- The 2D Gaussian is separable:

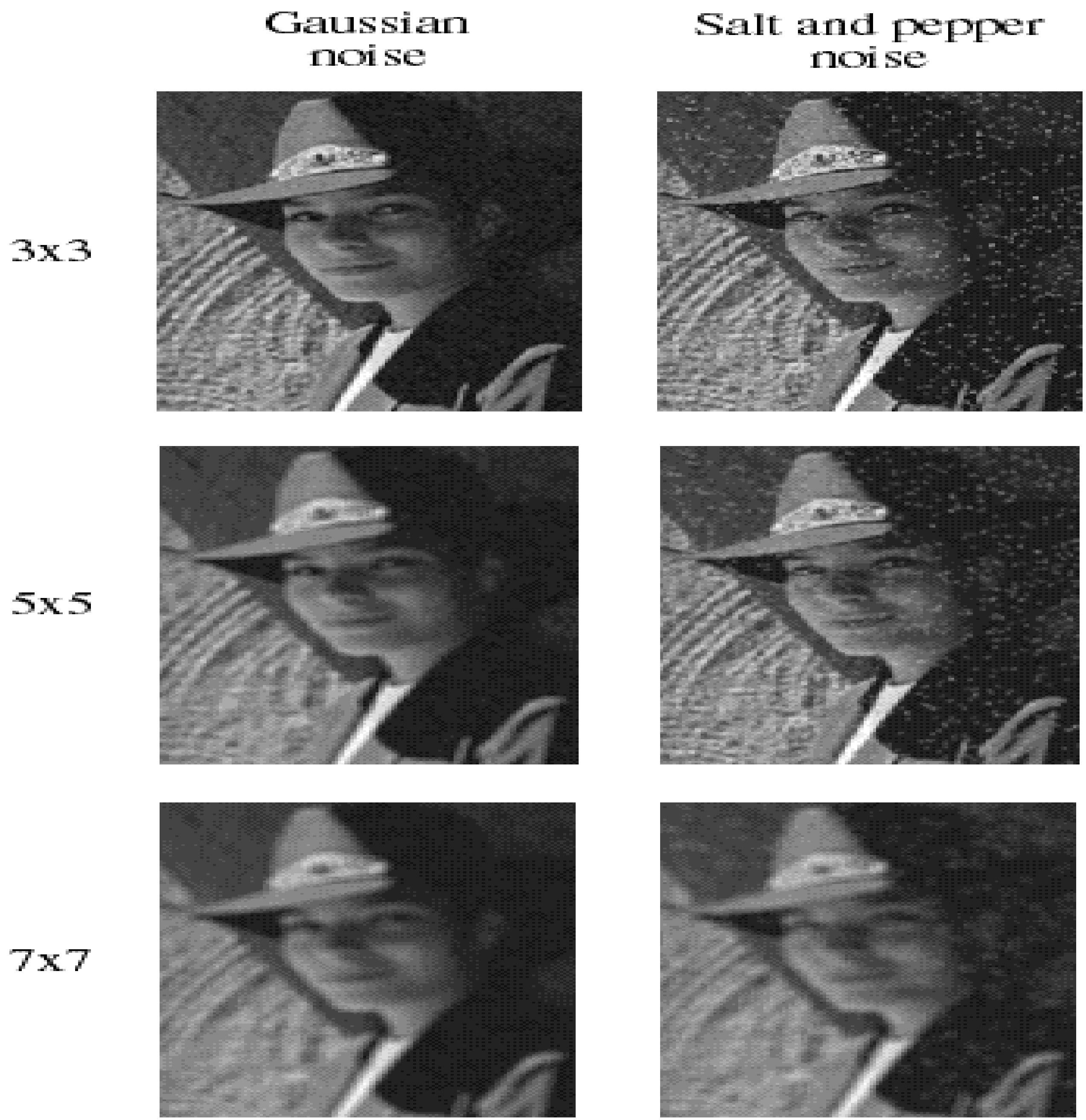
$$
\begin{aligned}
G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}} \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi\sigma^2}}\exp^{-\frac{y^2}{2\sigma^2}}\right)
\end{aligned}
$$

- What about the Box-filter?

Properties of convolution

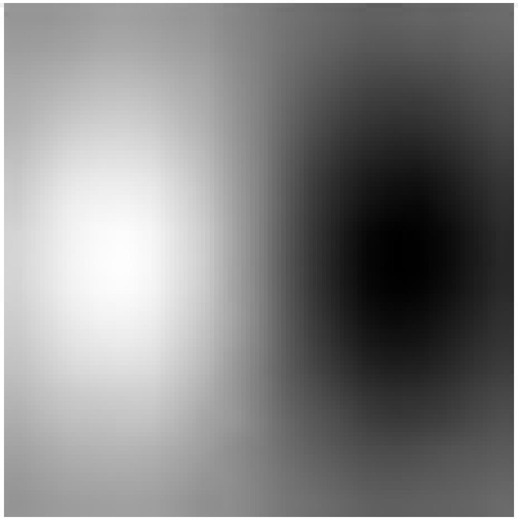
- Linear & shift invariant
- Commutative: $$f * g = g * f$$
- Associative $$(f * g) * h = f * (g * h)$$
- Identity:
- unit impulse $$e = [\ldots, 0, 0, 1, 0, 0, \ldots] \Rightarrow f * e = f$$
- Differentiation: $$\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g$$

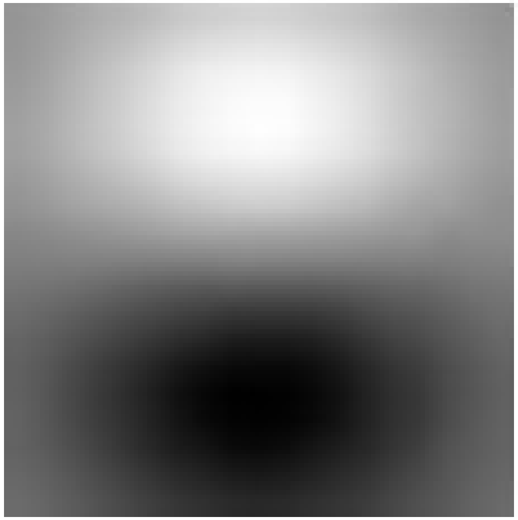
Effect of smoothing filters



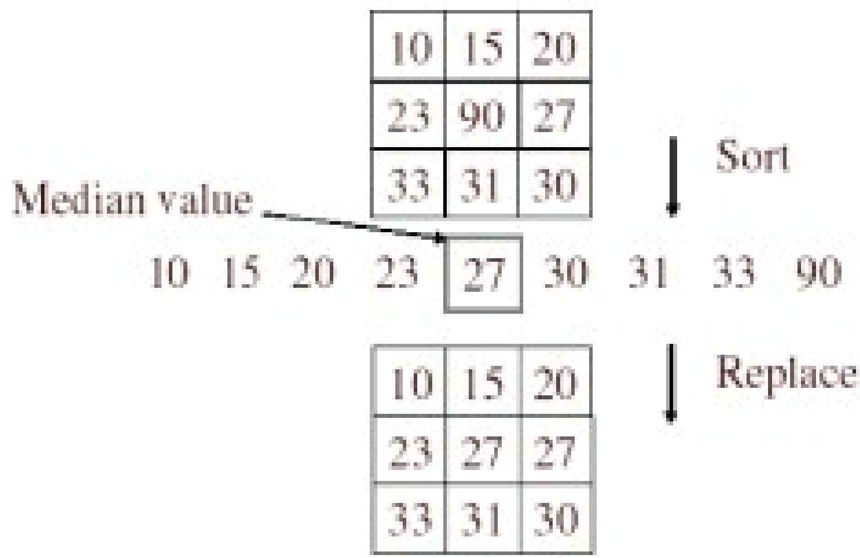
Filter as templates

- Applying a filter can be seen as taking dot-product between an image area and a vector.
- Filter emphasizes similar areas.
  - Highest response for regions that “look the most like the filter”





Median filter

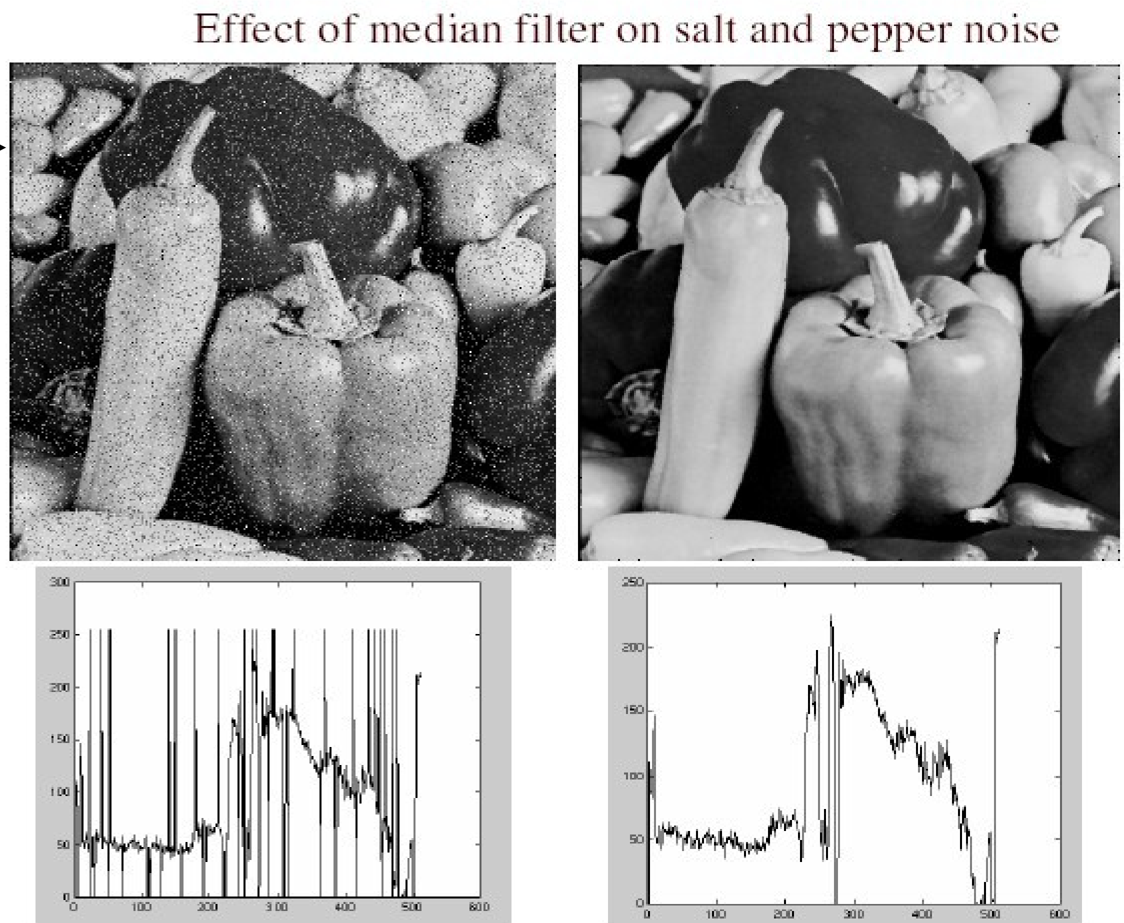


- Nonlinear
- No new pixel values introduced
- Remove spikes: good for impulse, salt&pepper noise
- Drawback: produces unexpected artifacts, erase fine lines and/or inverts parallel fine lines black → white and white → black, erodes sharp edges, slow.

Median filter

Salt and pepper noise

Median filtered

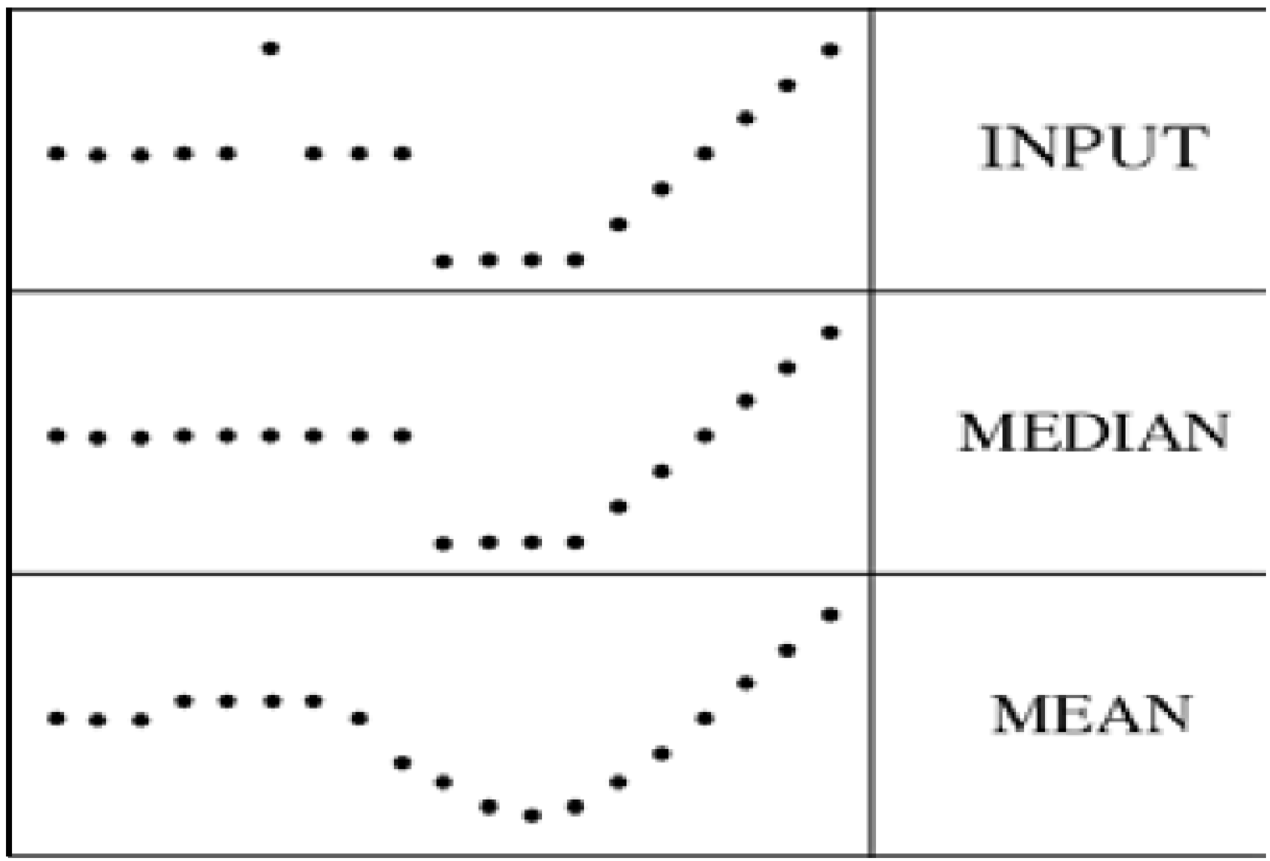


Plots of a row of the image

Source: M. Hebert

Median filter

- Median filter is edge preserving



Edge detection

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

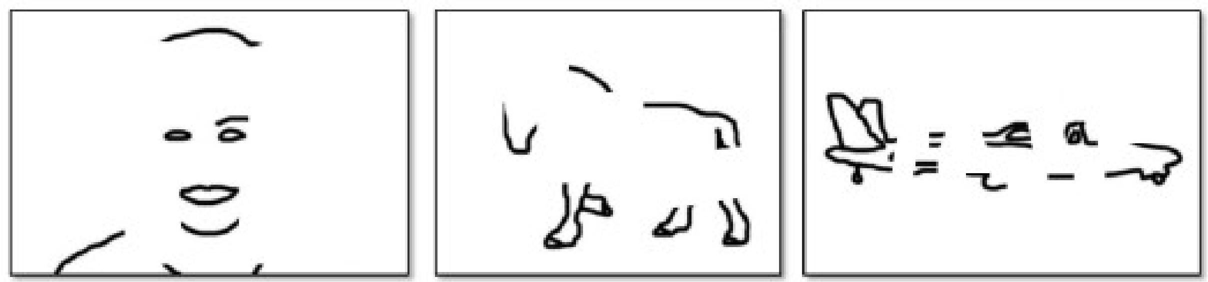
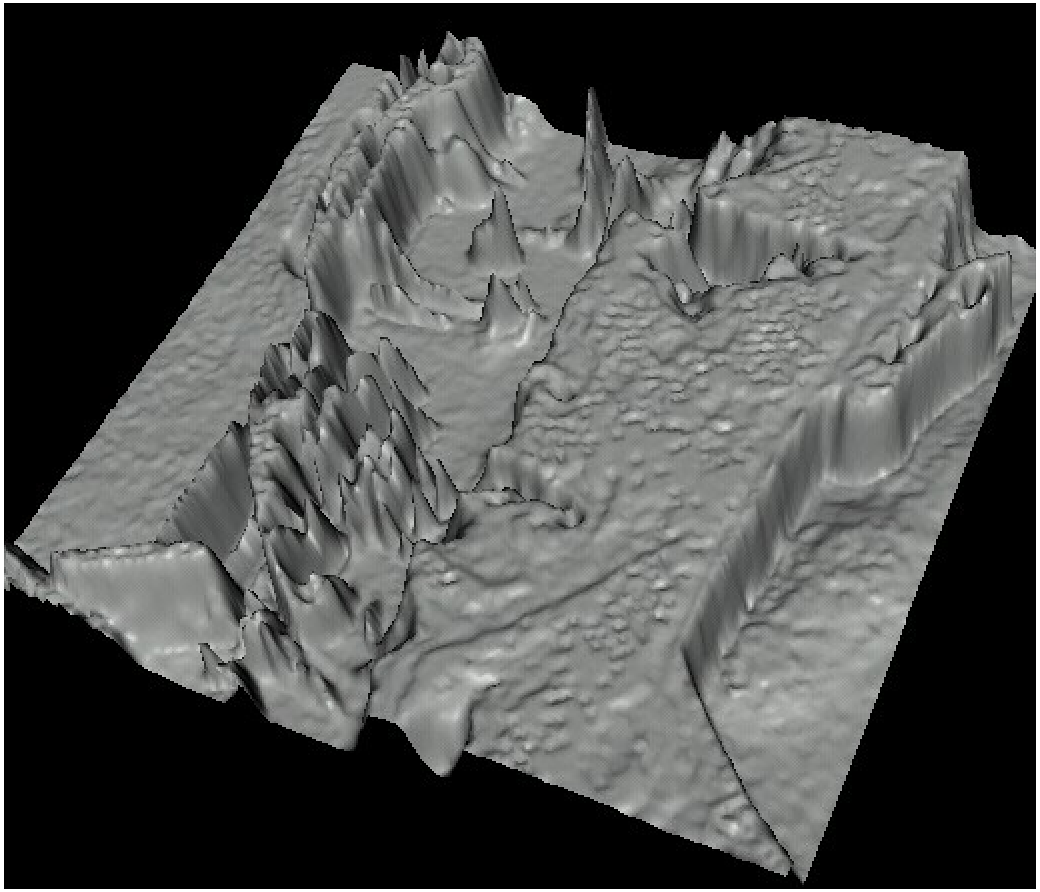


Figure from J. Shotton et al., PAMI 2007

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

Recall : Images as functions



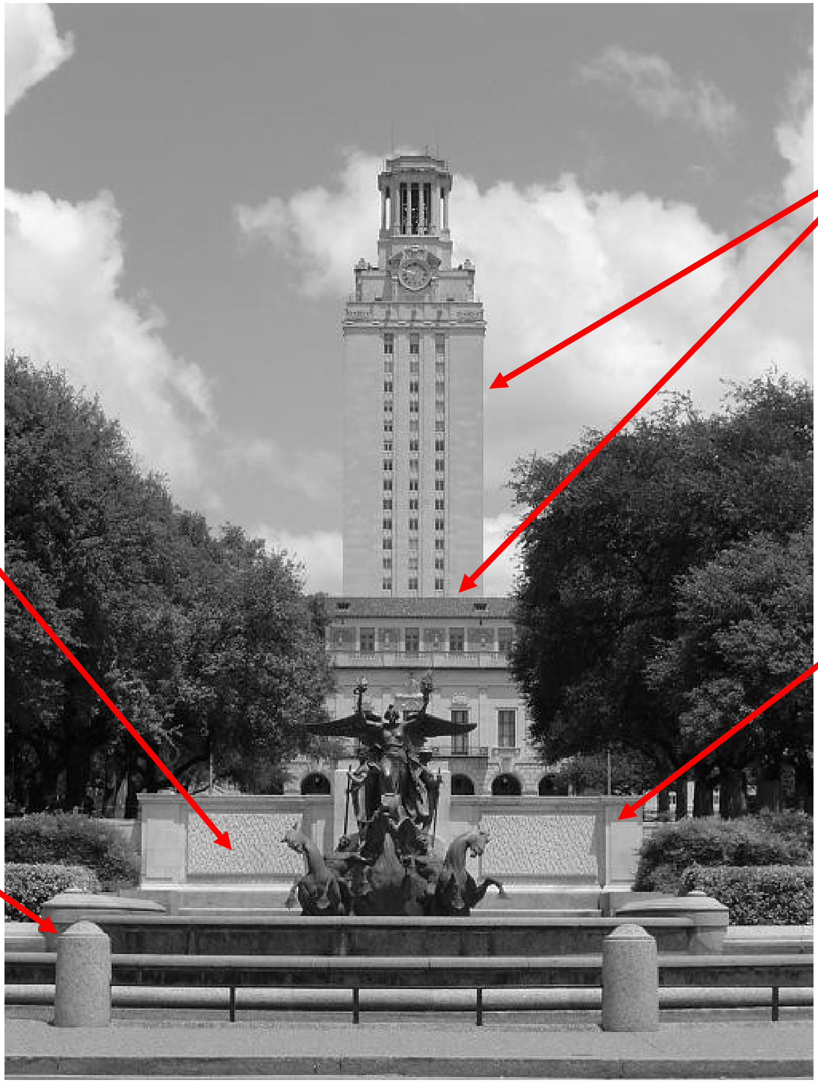
- Edges look like steep cliffs

Source: S. Seitz

What can cause an edge?

Reflectance change: appearance information, texture

Change in surface orientation: shape

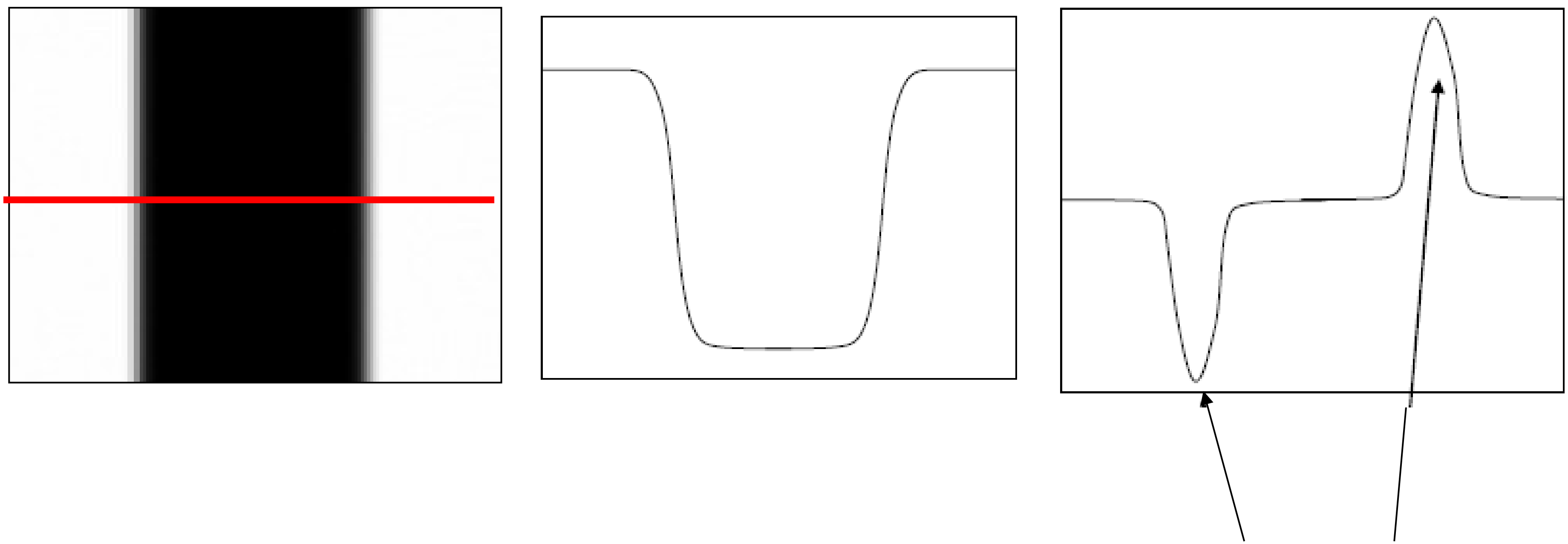


Depth discontinuity: object boundary

Cast shadows

Derivatives and edges

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



Source: L. Lazebnik

Differentiation and convolution

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

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

For discrete data, we can approximate using finite differences:

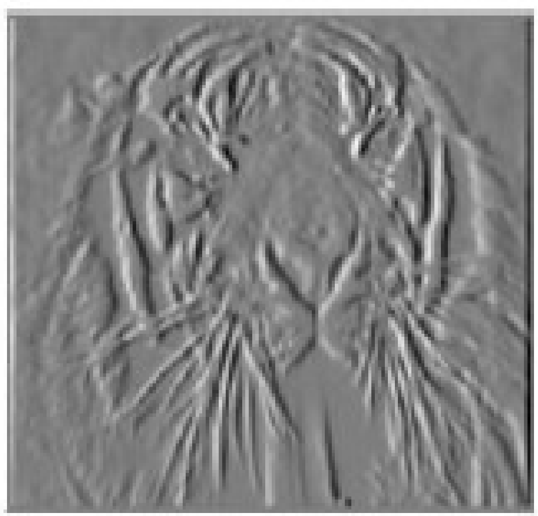
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

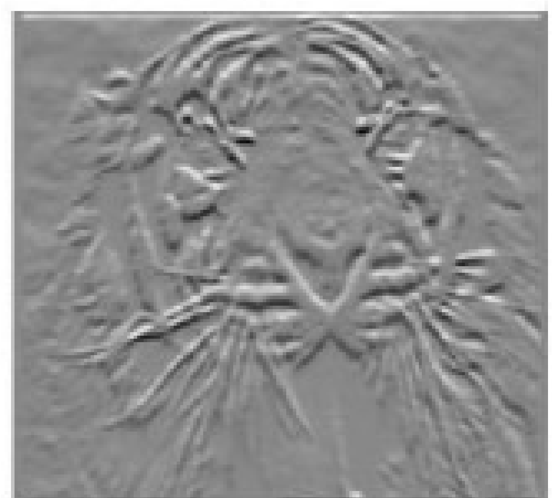
To implement above as convolution, what would be the associated filter?

Partial derivatives of an image

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

-1	1





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

-1
1

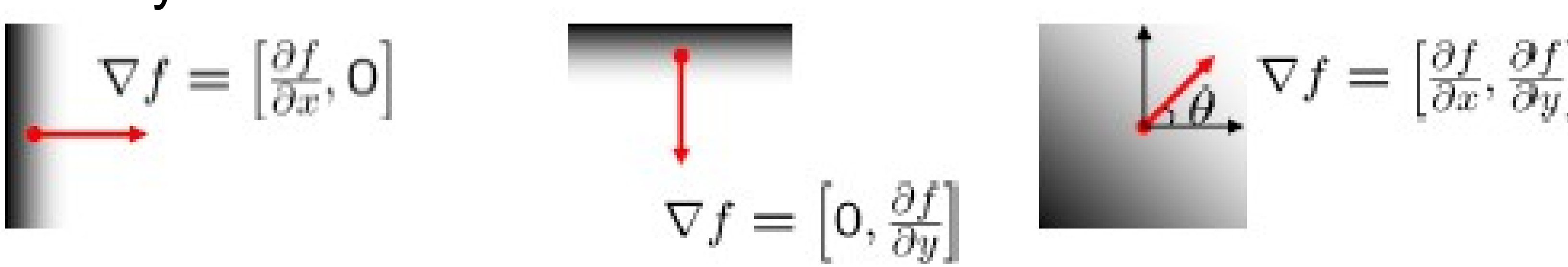
Which shows changes with respect to x?

Image gradient

- The gradient of an image

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

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



The gradient direction is the orientation of the edge normal:

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

- The edge strength is given by the gradient magnitude:

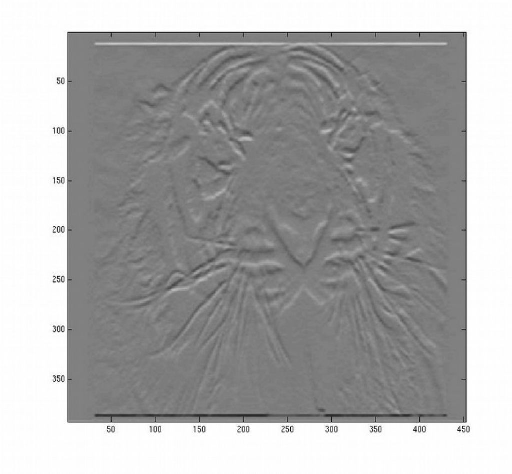
$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Assorted finite difference filters

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

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

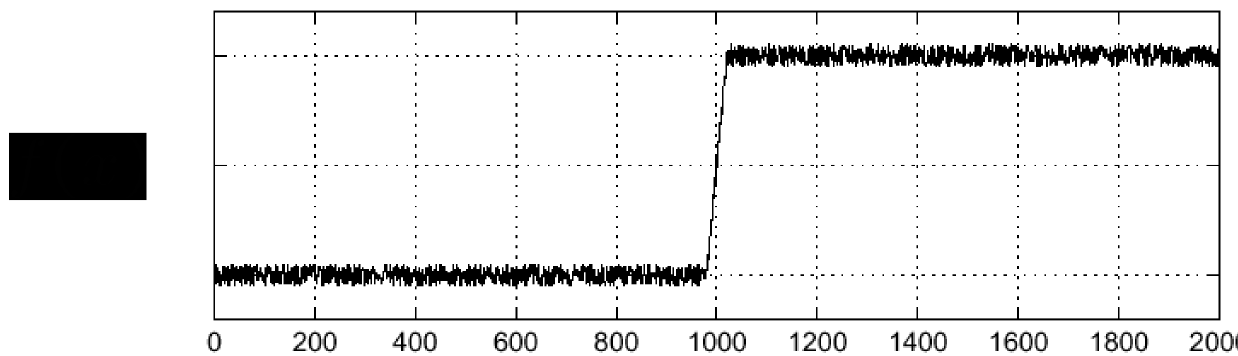
Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$



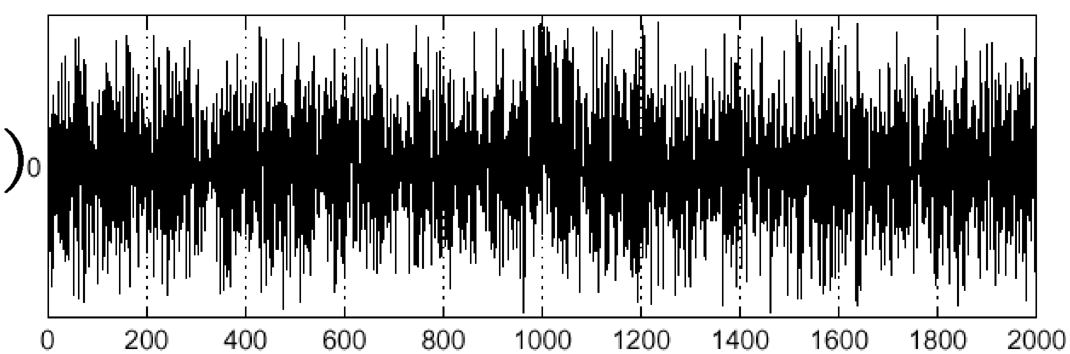
Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal



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



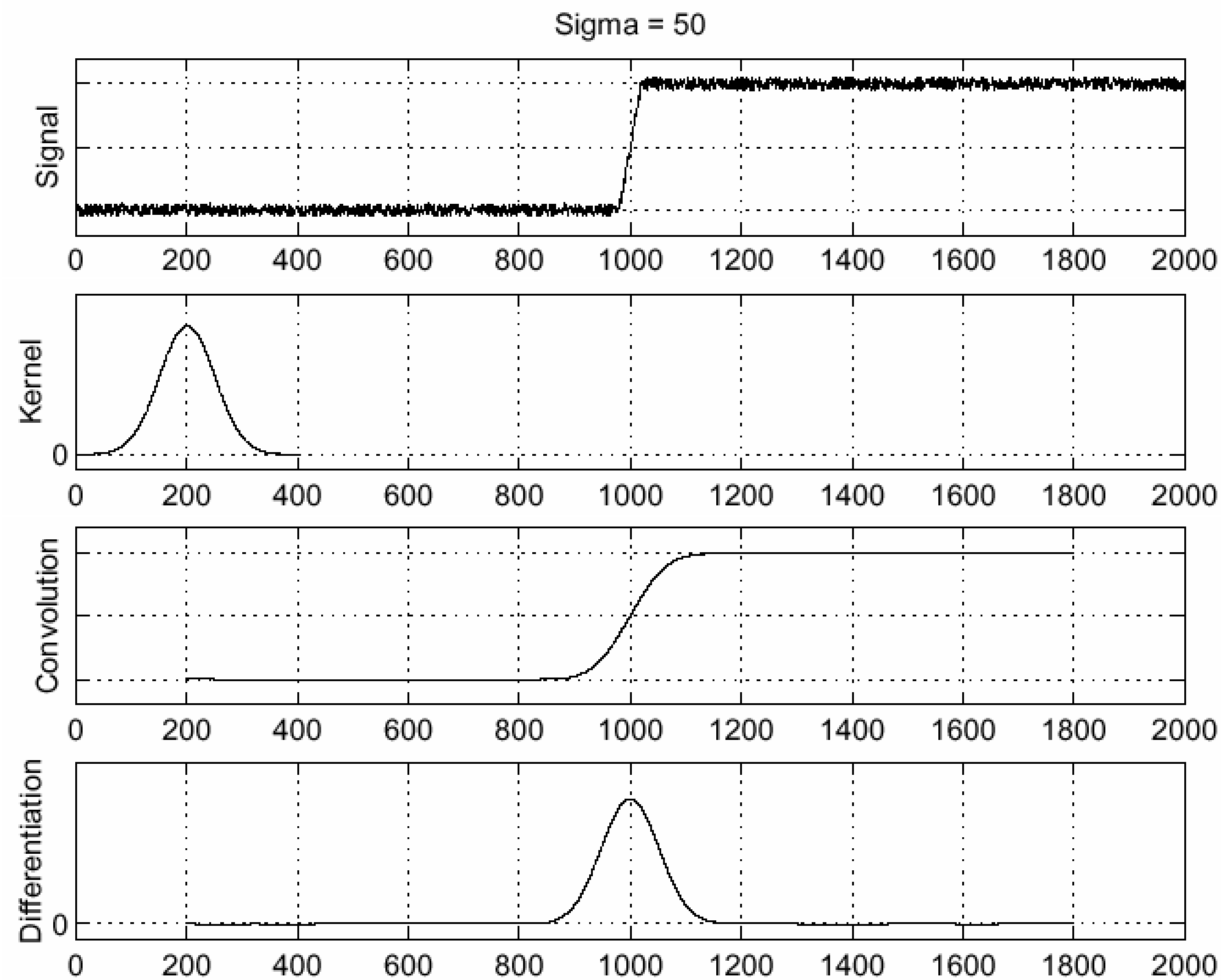
Where is the edge?

Solution: smooth first

[redacted]

[redacted]

[redacted]



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

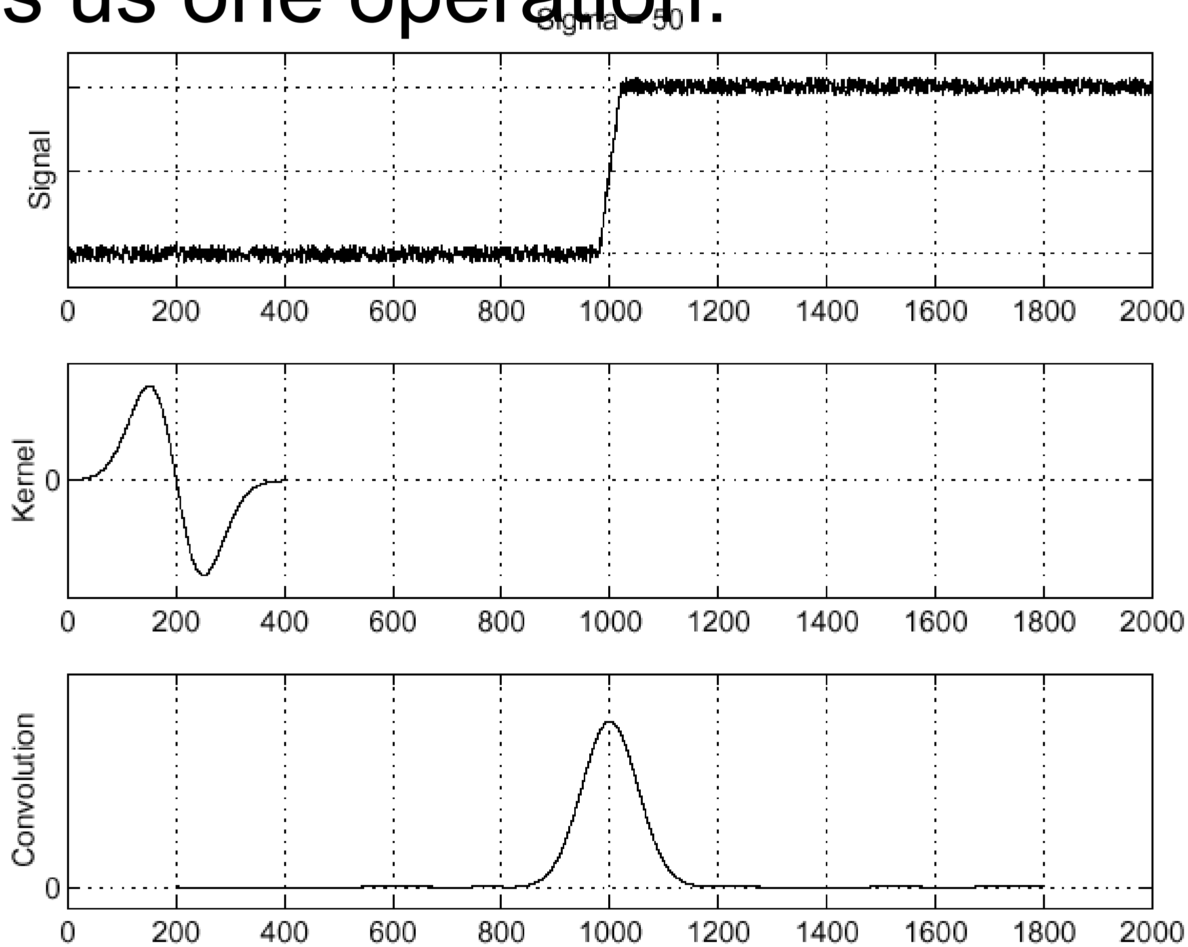
Derivative theorem of convolution

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

This saves us one operation:

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

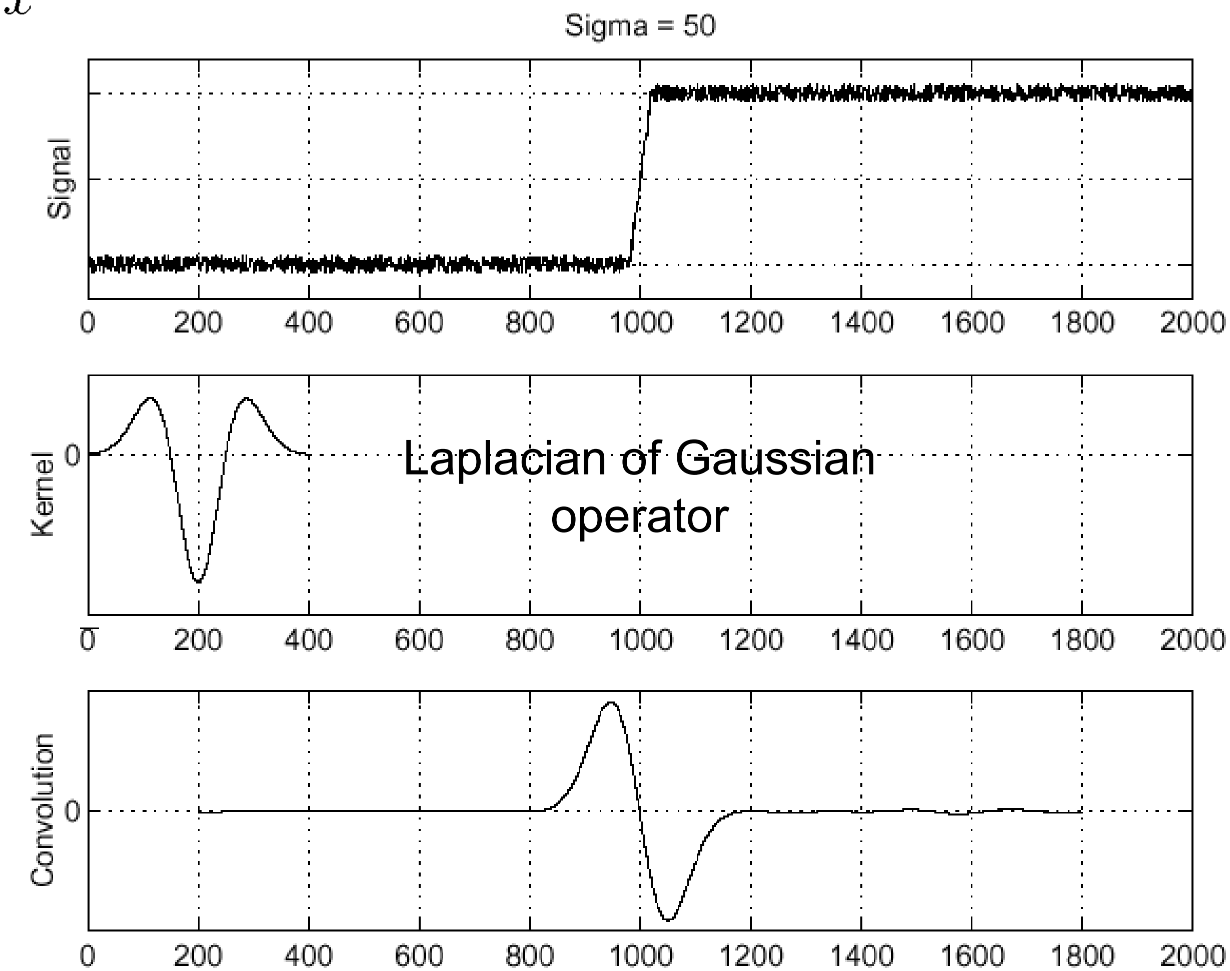
$(\frac{\partial}{\partial x}h) \star f$



How can we find (local) maxima of a function?

Laplacian of Gaussian

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

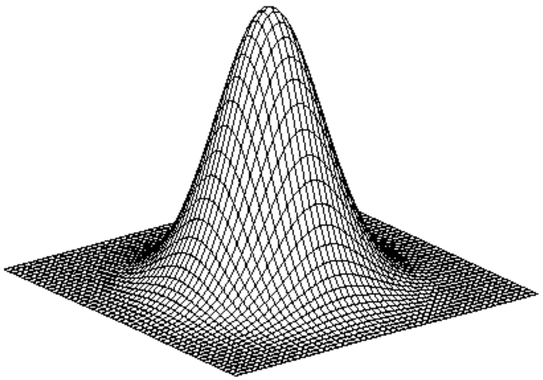


$\frac{\partial^2}{\partial x^2}h$

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

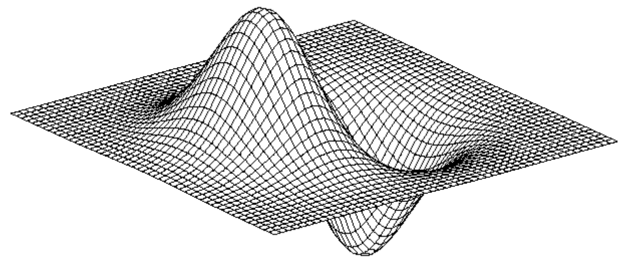
Where is the edge? Zero-crossings of bottom graph

2D edge detection filters



Gaussian

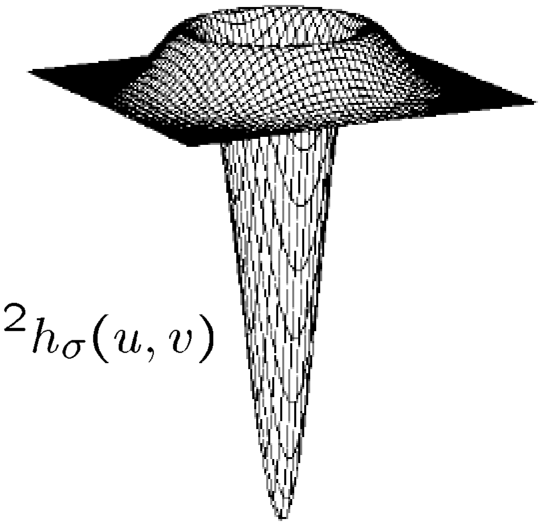
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian

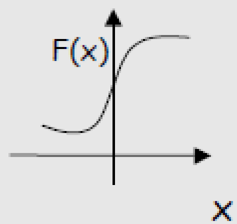
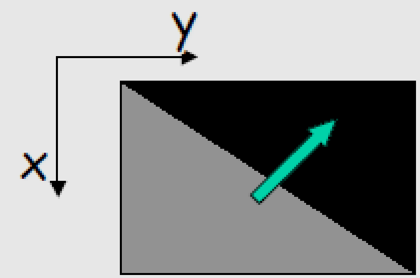


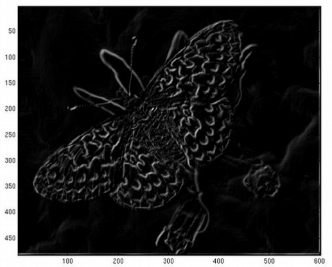
$$\nabla^2 h_\sigma(u,v)$$

[redacted] is the Laplacian operator:

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

Edge detection summary

	1D	2D
step edge	$I(x)$ 	$I(x,y)$ 
1st deriv	$\left\lvert \frac{dI(x)}{dx} \right\rvert > Th$	$\lvert\nabla I(x,y)\rvert = (I_x{}^2(x,y) + I_y{}^2(x,y))^{1/2} > Th$ $\tan\theta = I_x(x,y) / I_y(x,y)$
2nd deriv	$\frac{d^2 I(x)}{dx^2} = 0$	$\nabla^2 I(x,y) = I_{xx}(x,y) + I_{yy}(x,y) = 0$ Laplacian



Gradients -> edges

Primary edge detection steps:

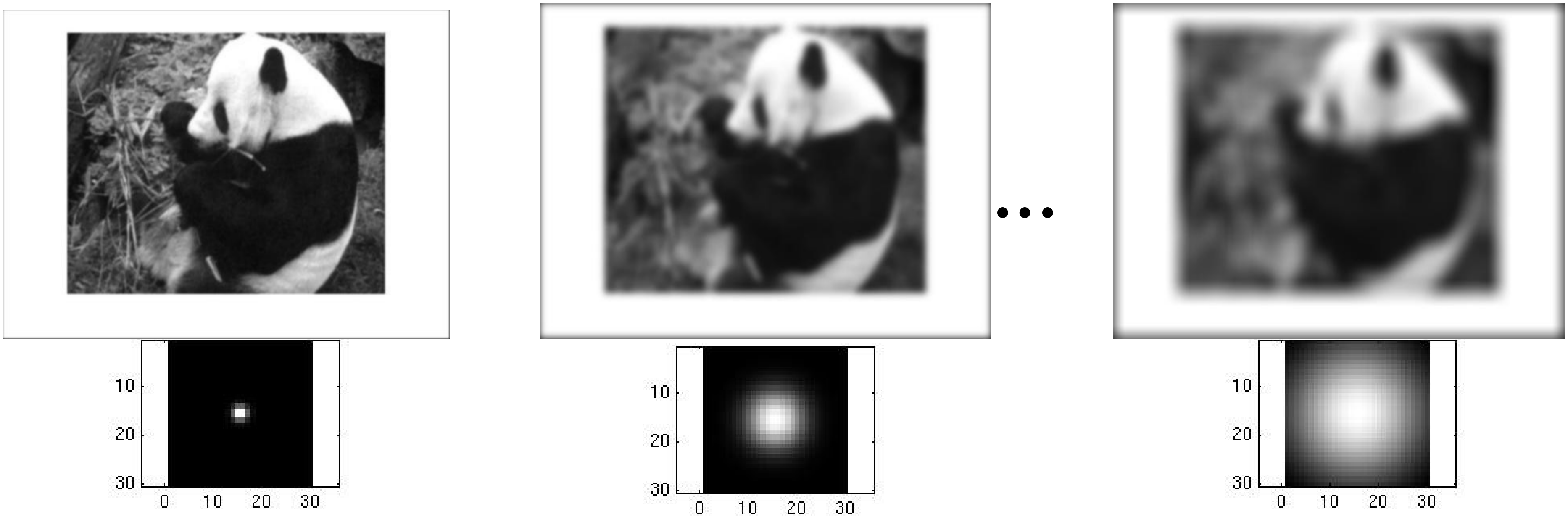
1. Smoothing: suppress noise
2. Edge enhancement: filter for contrast
3. Edge localization

Determine which local maxima from filter output are actually edges vs. noise

- Threshold, Thin

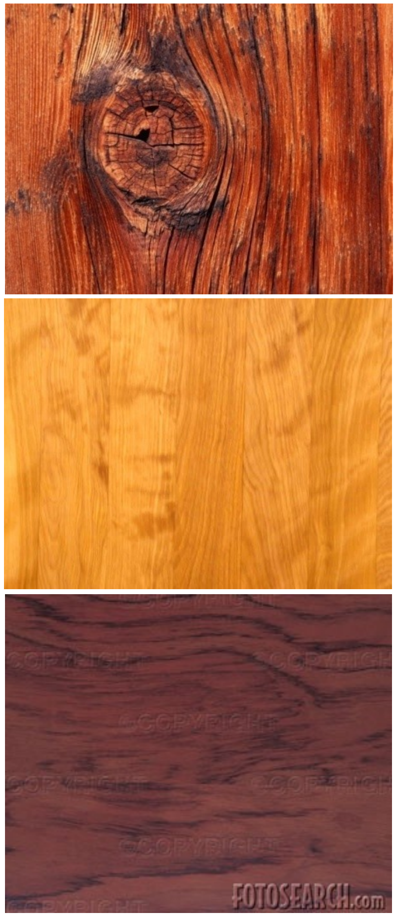
Smoothing with a Gaussian

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



So, what scale to choose?

It depends what we're looking for.



Too fine of a scale…can't see the forest for the trees.

Too coarse of a scale…can't tell the maple grain from the cherry.

Canny edge detector

- Filter image with derivative of Gaussian
- Find magnitude and orientation of gradient
- **Non-maximum suppression**:
  - Thin multi-pixel 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

Source: D. Lowe, L. Fei-Fei

The Canny edge detector

original image (Lena)

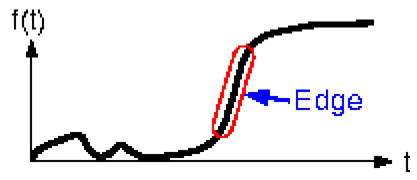
The Canny edge detector

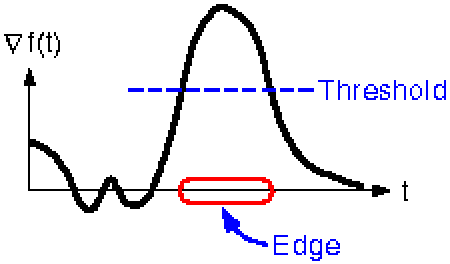
norm of the gradient

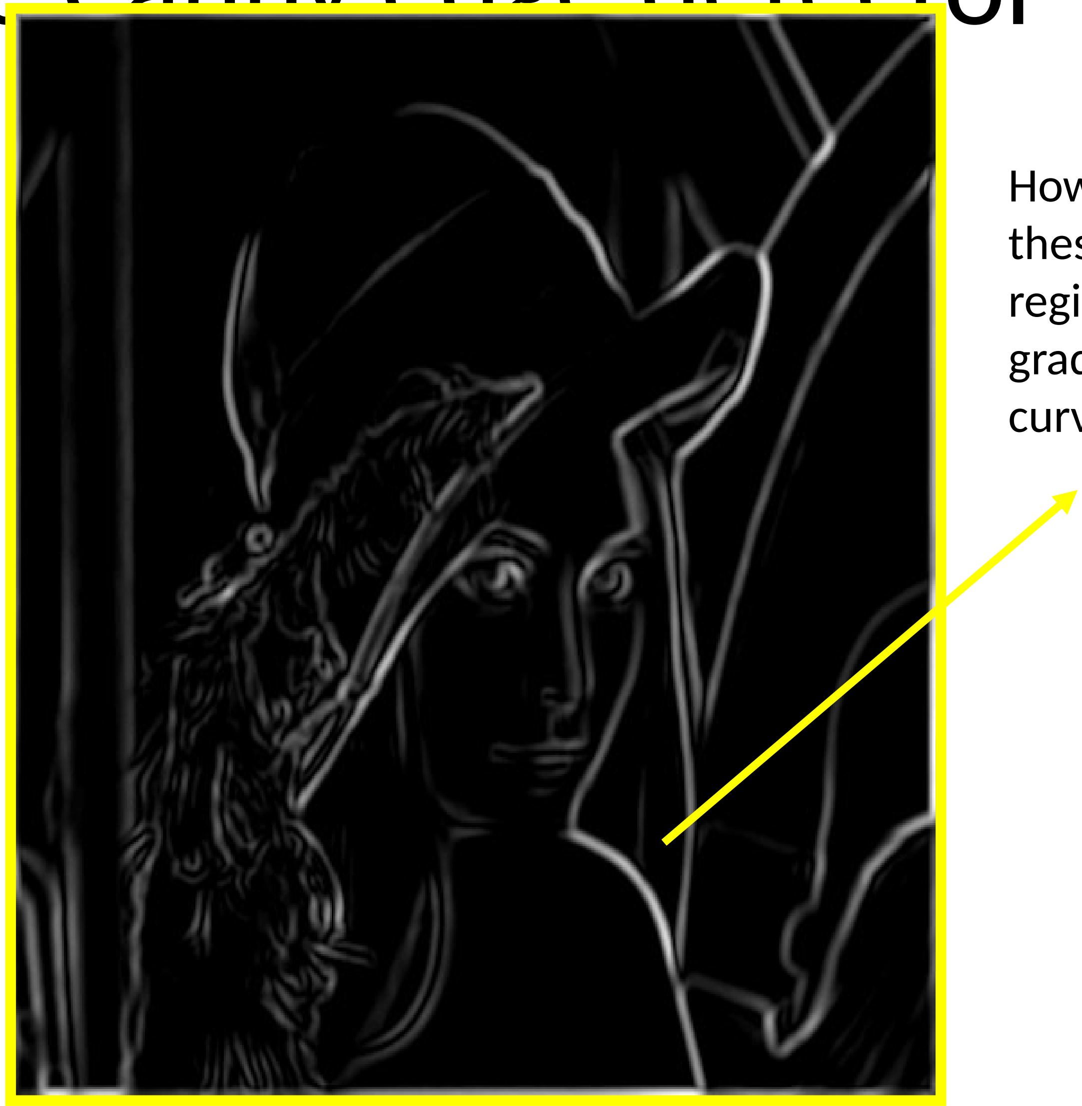
The Canny edge detector

thresholding

The Canny edge detector

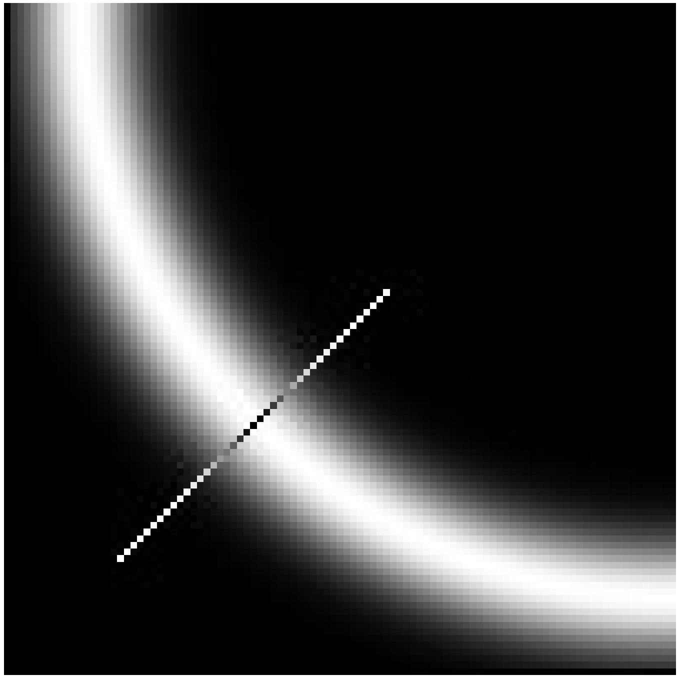


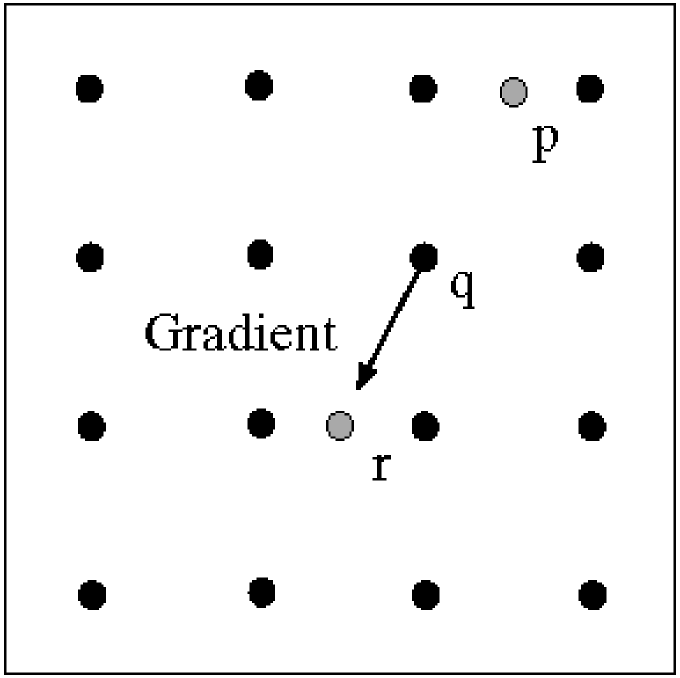




How to turn these thick regions of the gradient into curves?

Non-maximum suppression





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

- requires checking interpolated pixels p and r

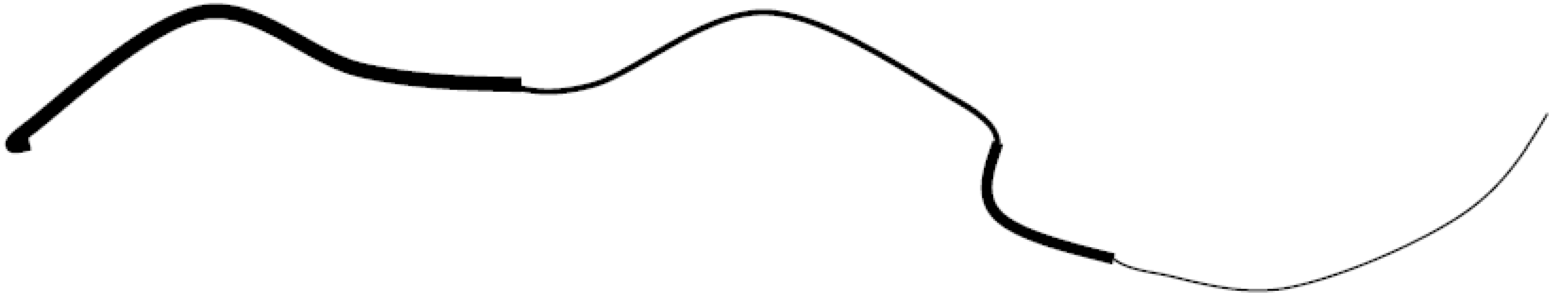
The Canny edge detector

thinning
(non-maximum suppression)

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

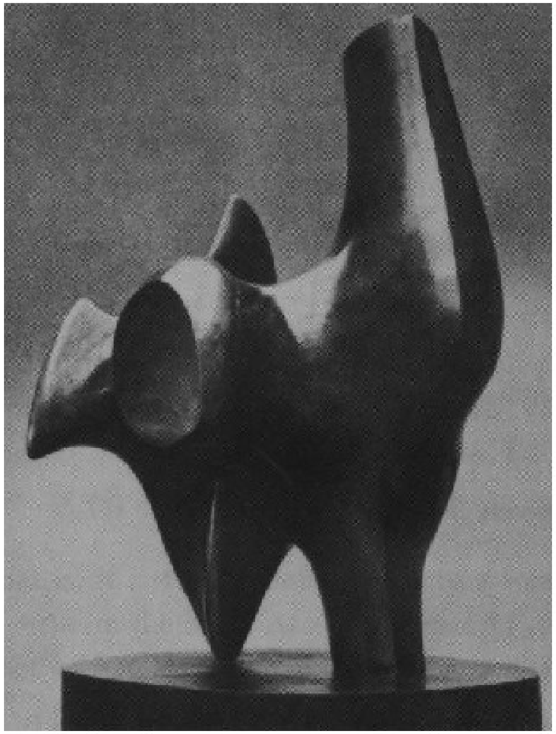
Hysteresis thresholding

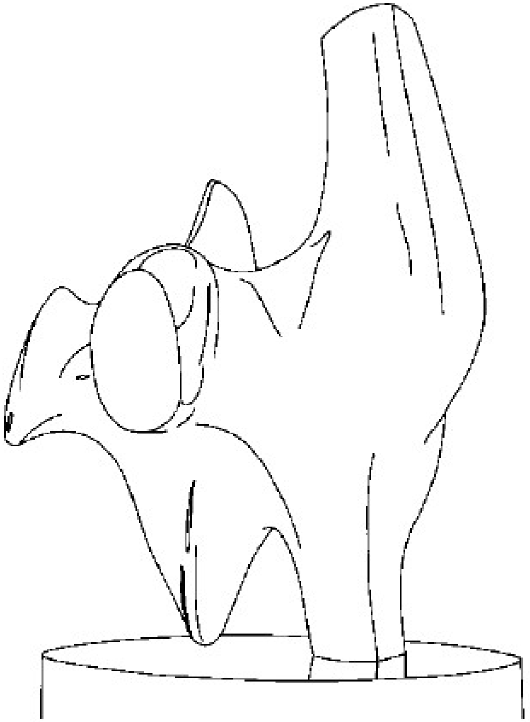
- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.



Source: S. Seitz

Summary





Summary