Fundamentals of Data Science

Basic Concepts and Terminology for Image Processing and Computer Vision

Including 2 case studies:

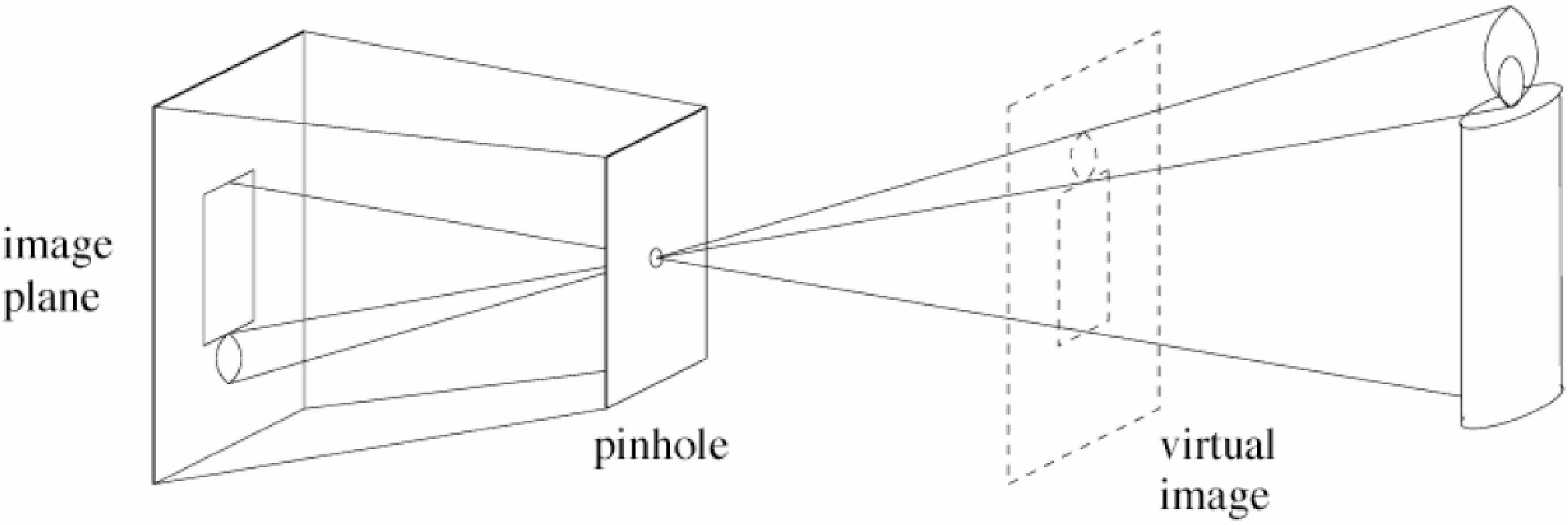
- Recovery of 3D structure

- Object Recognition

SAPIENZA
UNIVERSITÀ DI ROMA

Pinhole Camera (Model)

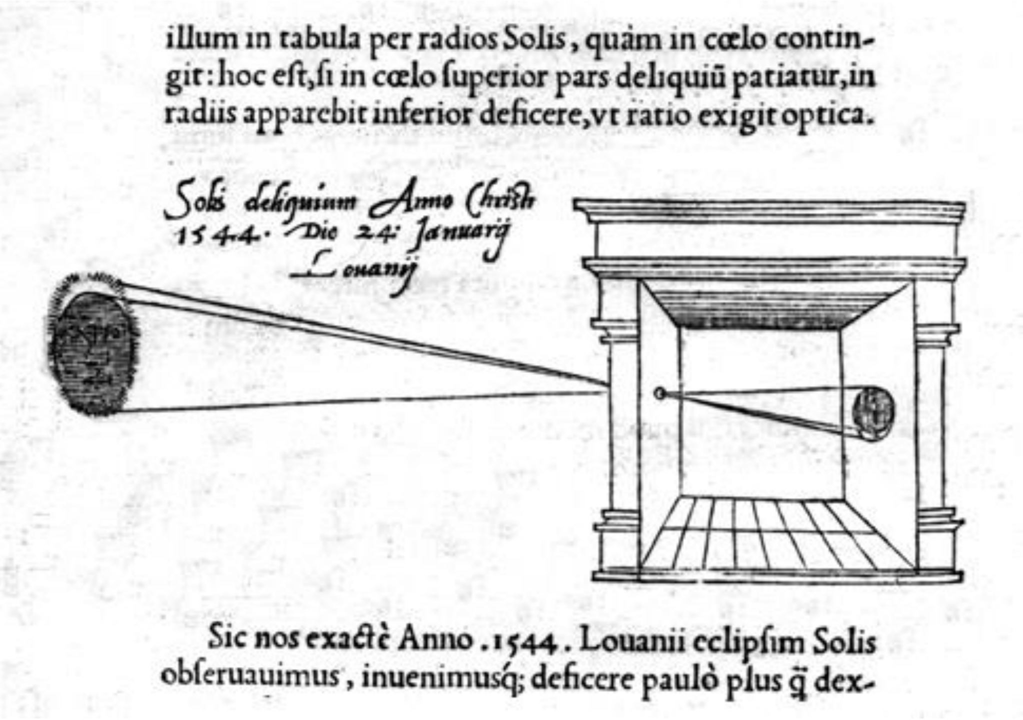
- (simple) standard and abstract model today
  - box with a small hole in it



Camera Obscura

- around 1519, Leonardo da Vinci (1452 - 1519)
  - http://www.acmi.net.au/AIC/CAMERA_OBSCURA.html

"when images of illuminated objects … penetrate through a small hole into a very dark room … you will see [on the opposite wall] these objects in their proper form and color, reduced in size … in a reversed position owing to the intersection of the rays"

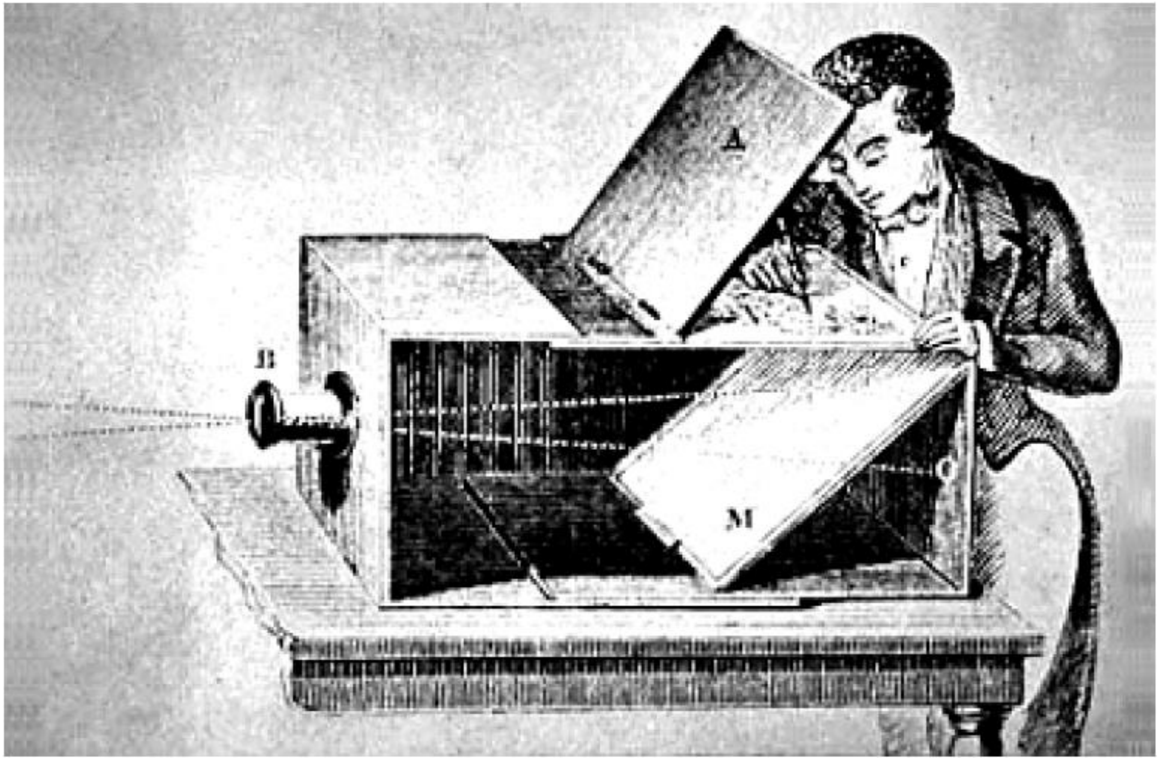

illum in tabula per radios Solis, quàm in cœlo contin-
git: hoc eſt, ſi in cœlo ſuperior pars deliquiũ patiatur, in
radiis apparebit inferior deficere, vt ratio exigit optica.

Solis deliquium Anno Christi
1544. Die 24. Januarij
Louanij

Sic nos exactè Anno .1544. Louanii eclipſim Solis
obſeruauimus, inuenimusq; deficere paulò plus q̃ dex-

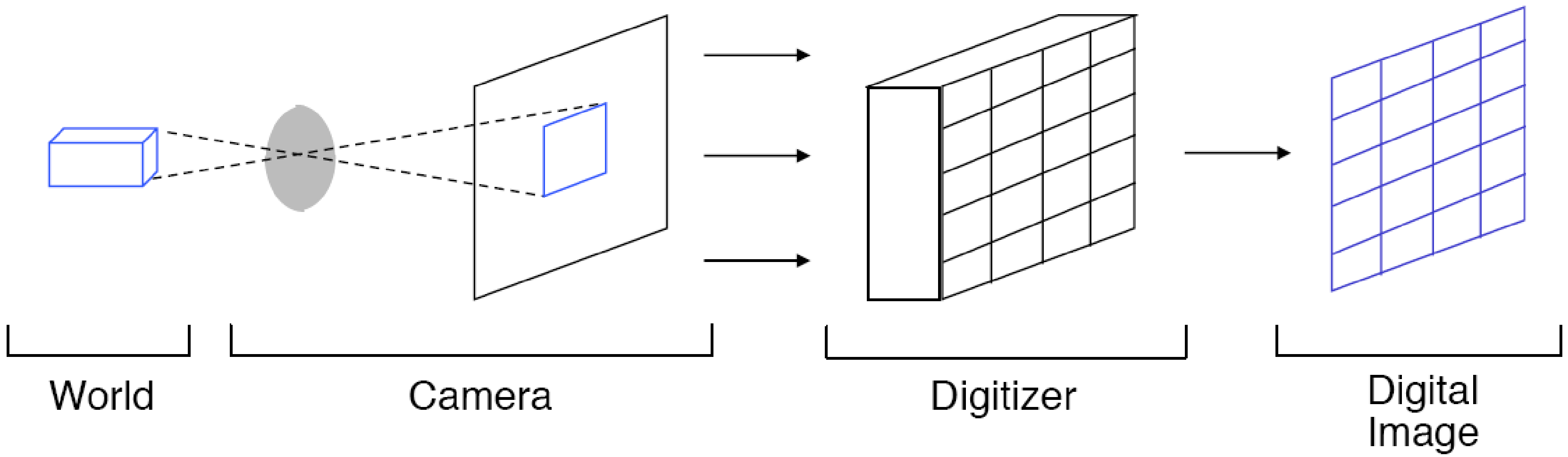
Principle of pinhole....

- ...used by artists
  - (e.g. Vermeer 17th century, dutch)
- and scientists



Digital Images

- Imaging Process:
  - (pinhole) camera model
  - digitizer to obtain digital image



(Grayscale) Image

- ‘Goals’ of Computer Vision
  - how can we recognize fruits from an array of (gray-scale) numbers?
  - how can we perceive depth from an array of (gray-scale) numbers?
  - …
- ‘Goals’ of Graphics
  - how can we generate an array of (gray-scale) numbers that looks like fruits?
  - how can we generate an array of (gray-scale) numbers so that the human observer perceives depth?
  - …

- computer vision = the problem of ‘inverse graphics’ …?

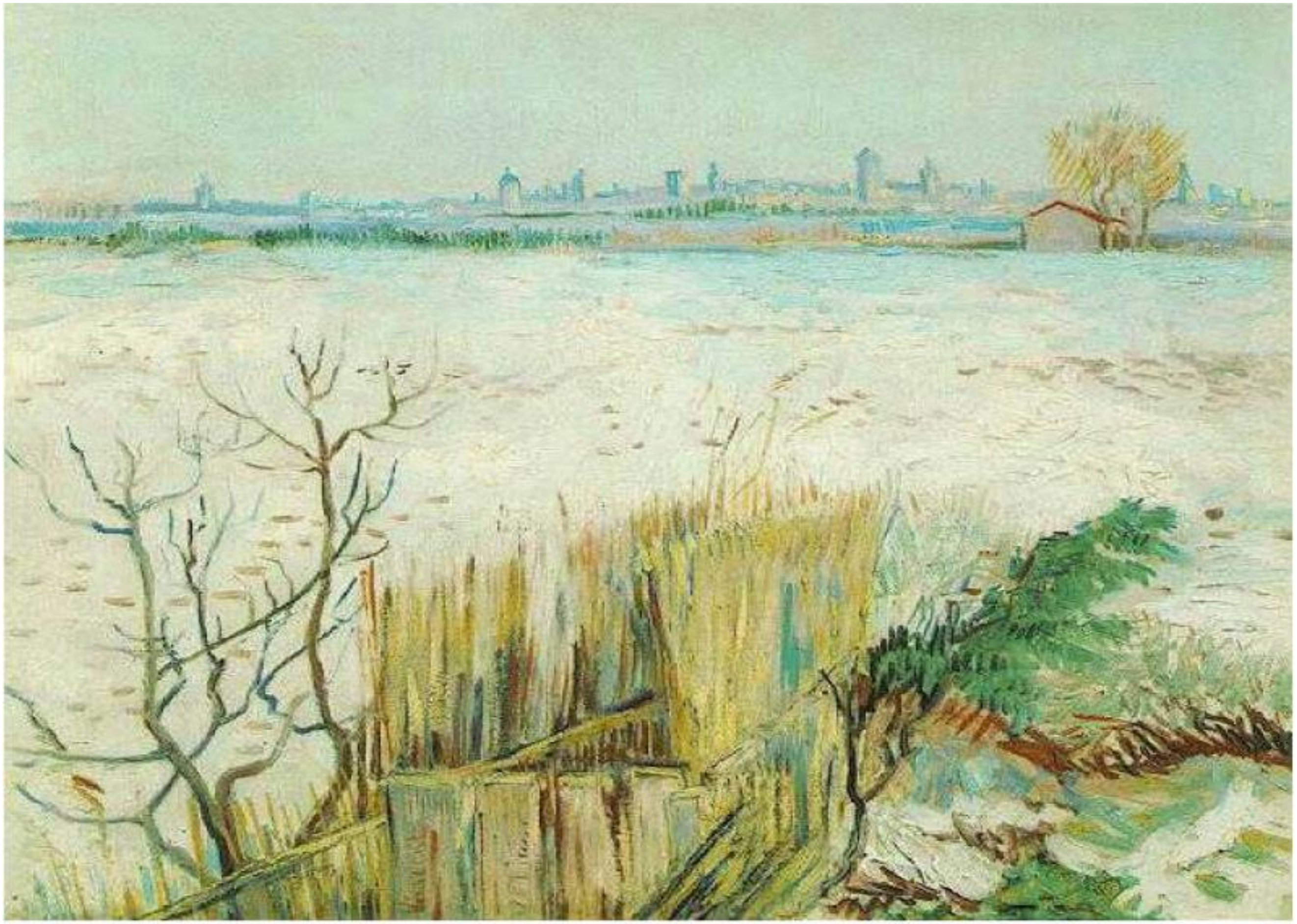
y = \ x =	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
41	210	209	204	202	197	247	143	71	64	80	84	54	54	57	58
42	206	196	203	197	195	210	207	56	63	58	53	53	61	62	51
43	201	207	192	201	198	213	156	69	65	57	55	52	53	60	50
44	216	206	211	193	202	207	208	57	69	60	55	77	49	62	61
45	221	206	211	194	196	197	220	56	63	60	55	46	97	58	106
46	209	214	224	199	194	193	204	173	64	60	59	51	62	56	48
47	204	212	213	208	191	190	191	214	60	62	66	76	51	49	55
48	214	215	215	207	208	180	172	188	69	72	55	49	56	52	56
49	209	205	214	205	204	196	187	196	86	62	66	87	57	60	48
50	208	209	205	203	202	186	174	185	149	71	63	55	55	45	56
51	207	210	211	199	217	194	183	177	209	90	62	64	52	93	52
52	208	205	209	209	197	194	183	187	187	239	58	68	61	51	56
53	204	206	203	209	195	203	188	185	183	221	75	61	58	60	60
54	200	203	199	236	188	197	183	190	183	196	122	63	58	64	66
55	205	210	202	203	199	197	196	181	173	186	105	62	57	64	63

1. Case Study:
Human & Art - Recovery of 3D Structure

1. Case Study:
Human & Art - Recovery of 3D Structure

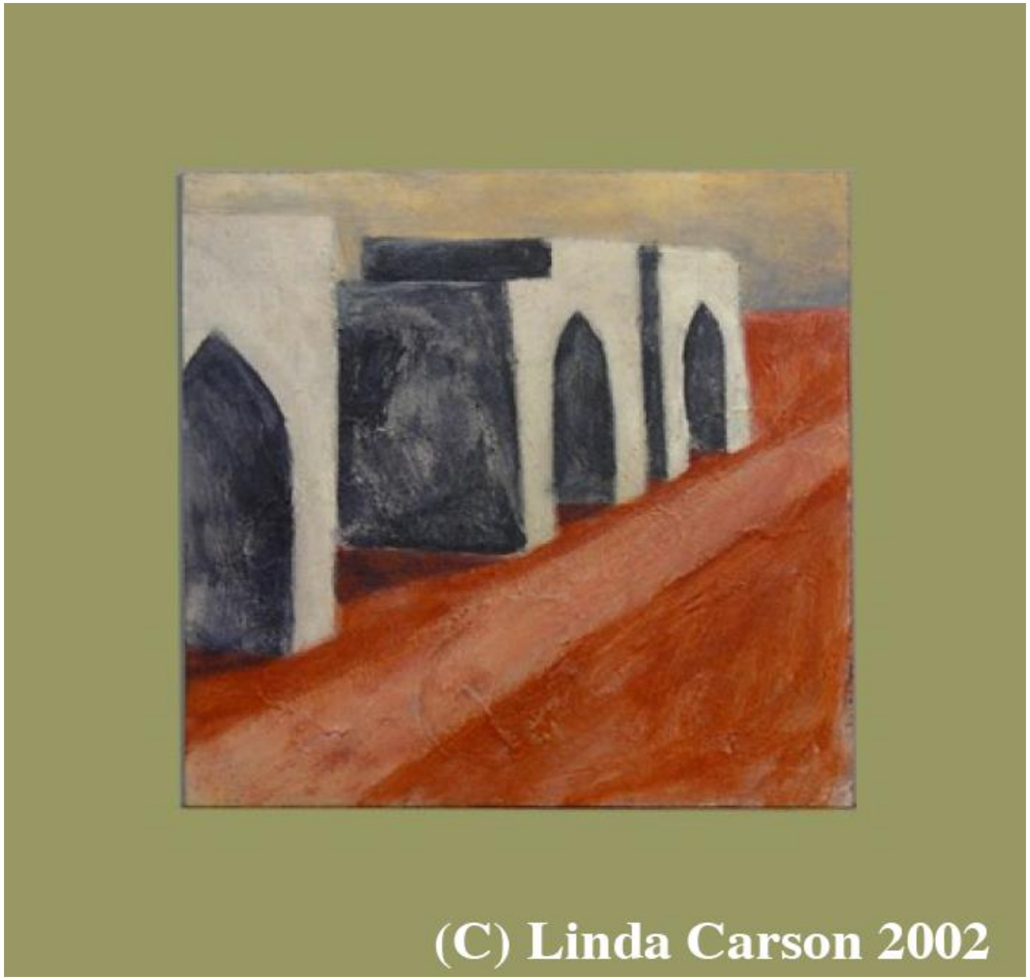
Vincent van Gogh *Interior of a Restaurant at Arles* 1888

1. Case Study:
Human & Art - Recovery of 3D Structure

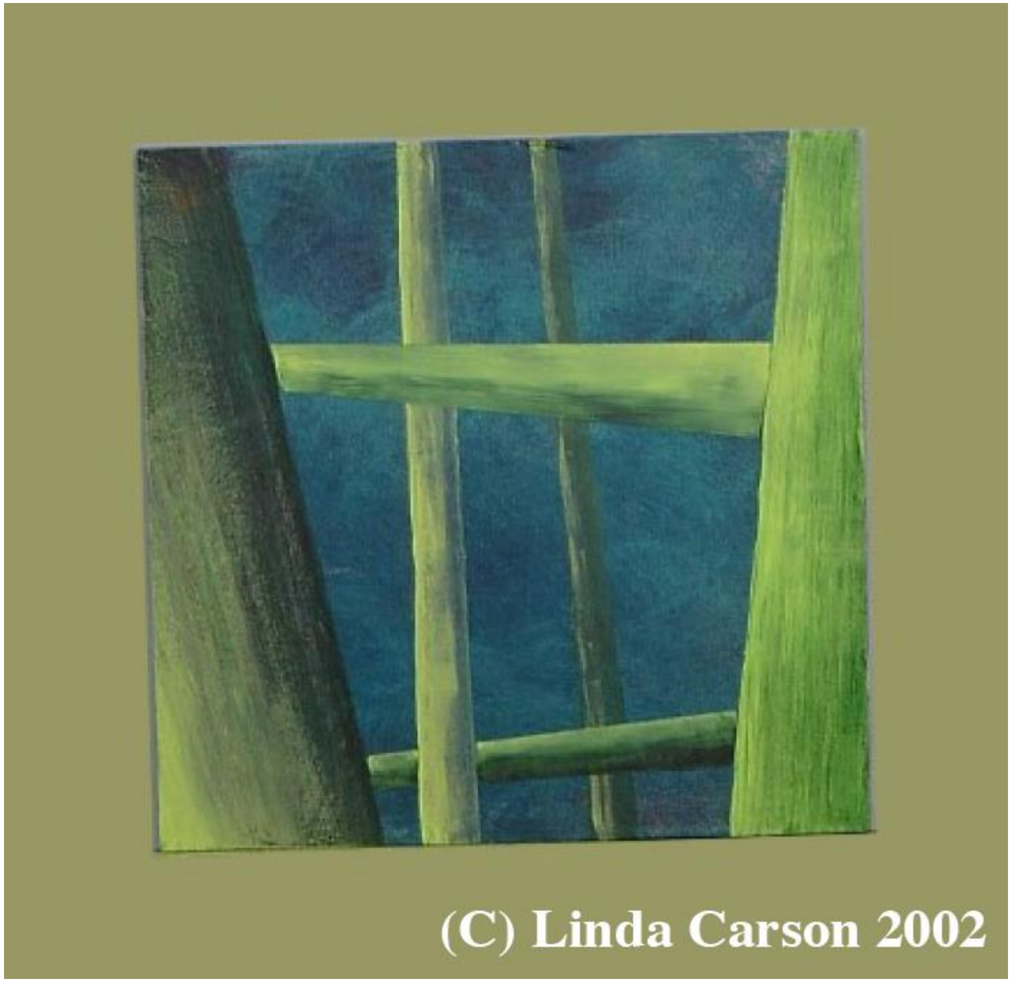


Vincent van Gogh *Snowy Landscape with Arles in the Background* 1888

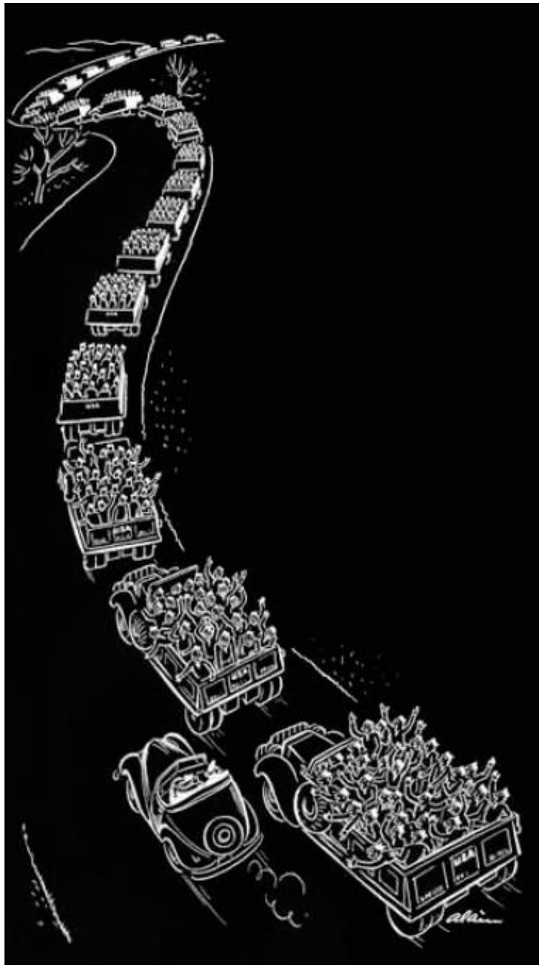
1. Case Study:
Human & Art - Recovery of 3D Structure

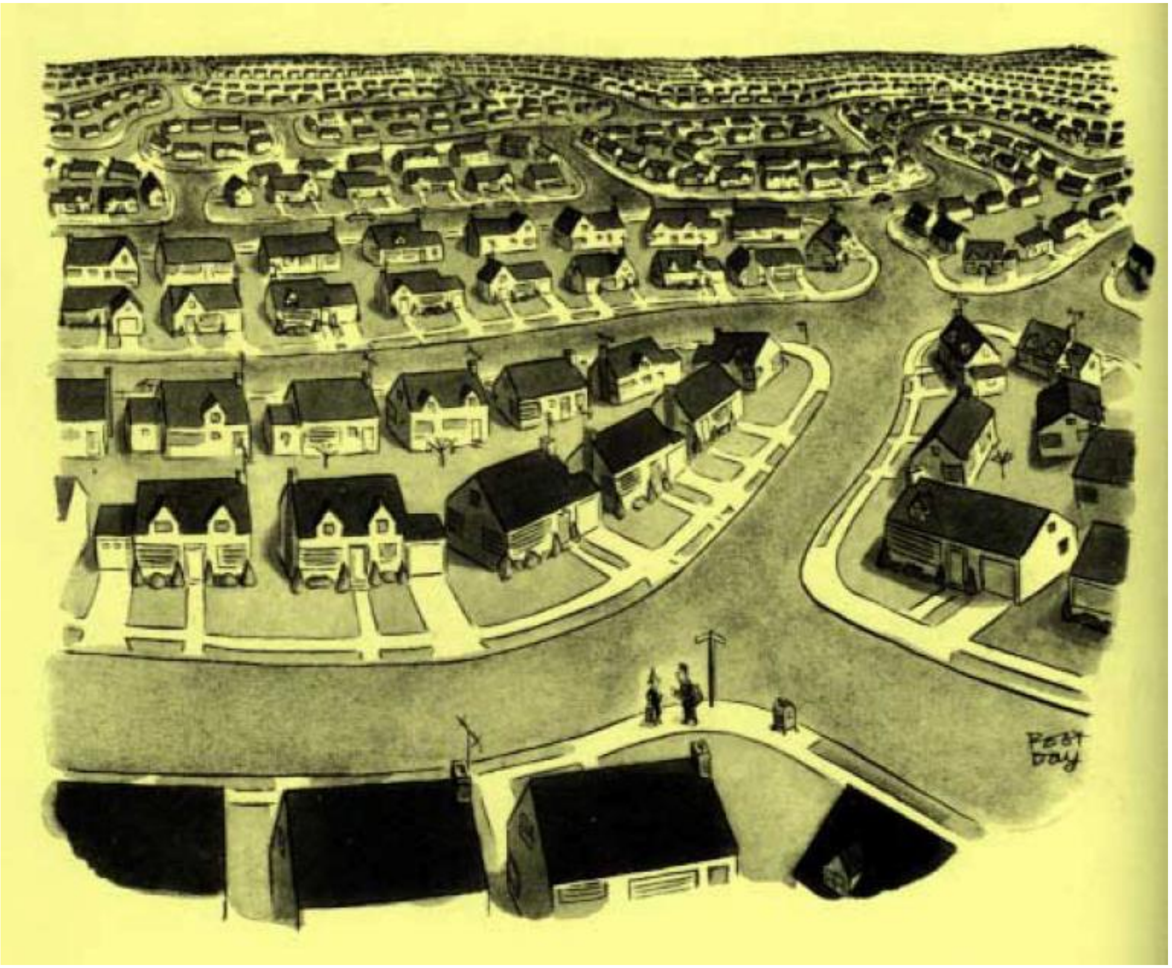


1. Case Study:
Human & Art - Recovery of 3D Structure



1. Case Study:
Human & Art - Recovery of 3D Structure



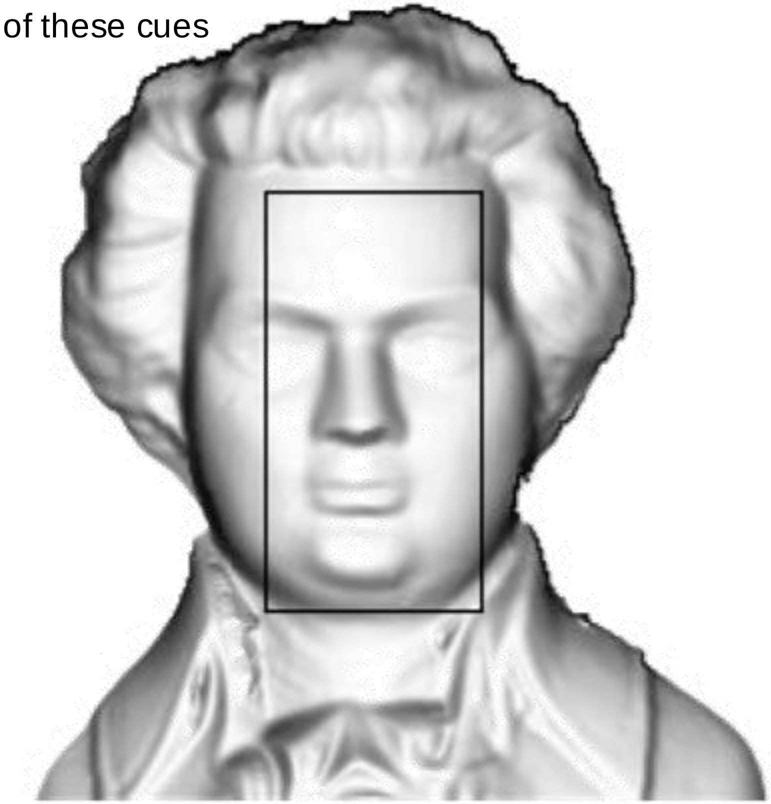


SAPIENZA
UNIVERSITÀ DI ROMA

1. Case Study
Computer Vision - Recovery of 3D Structure

- take all the cues of artists and ‘turn them around’
  - exploit these cues to **infer** the structure of the world
  - need **mathematical** and **computational models** of these cues

- sometimes called ‘inverse graphics’



http://www.vrvis.at/ar2/adm/shading/

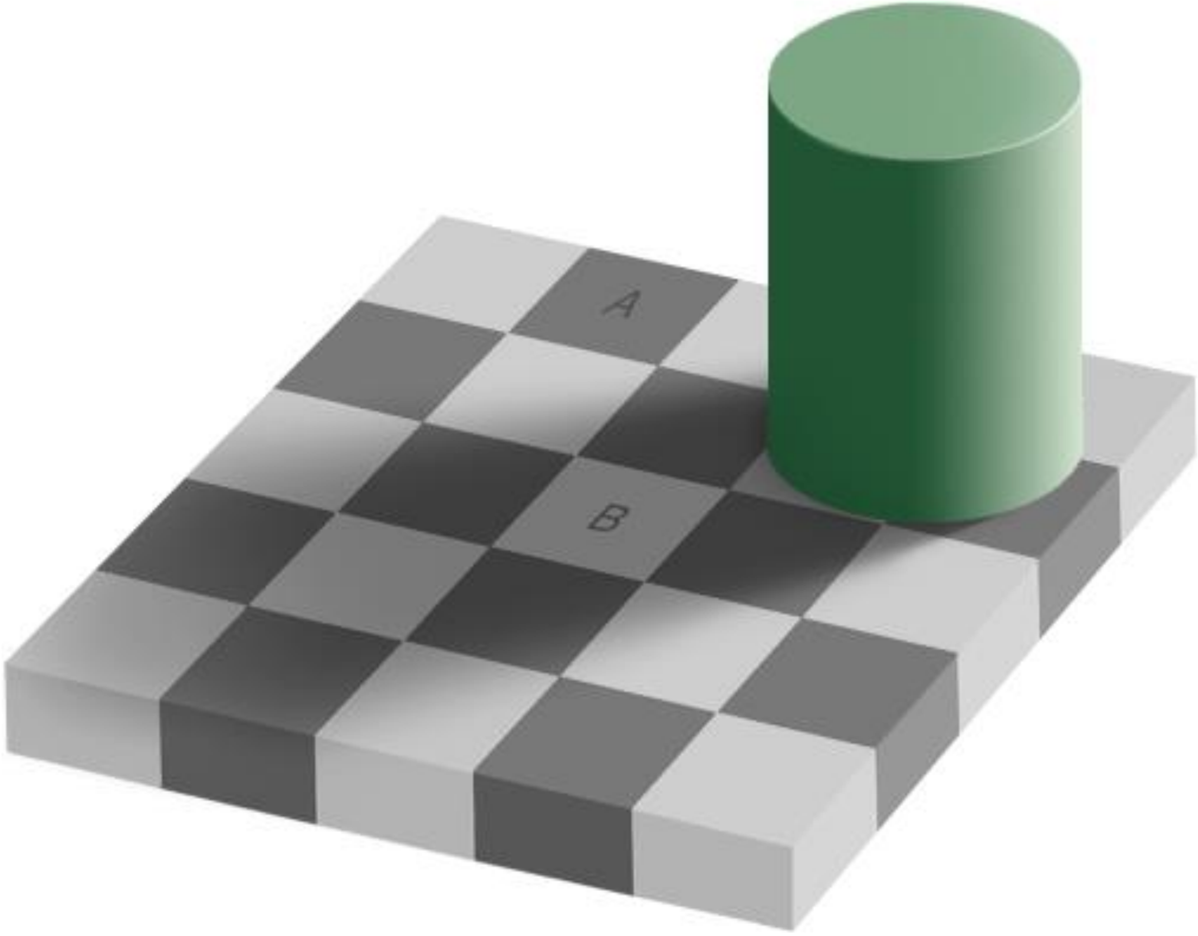
A ‘trompe l’oeil’

- depth-perception
  - movement of ball stays the same
  - location/trace of shadow changes

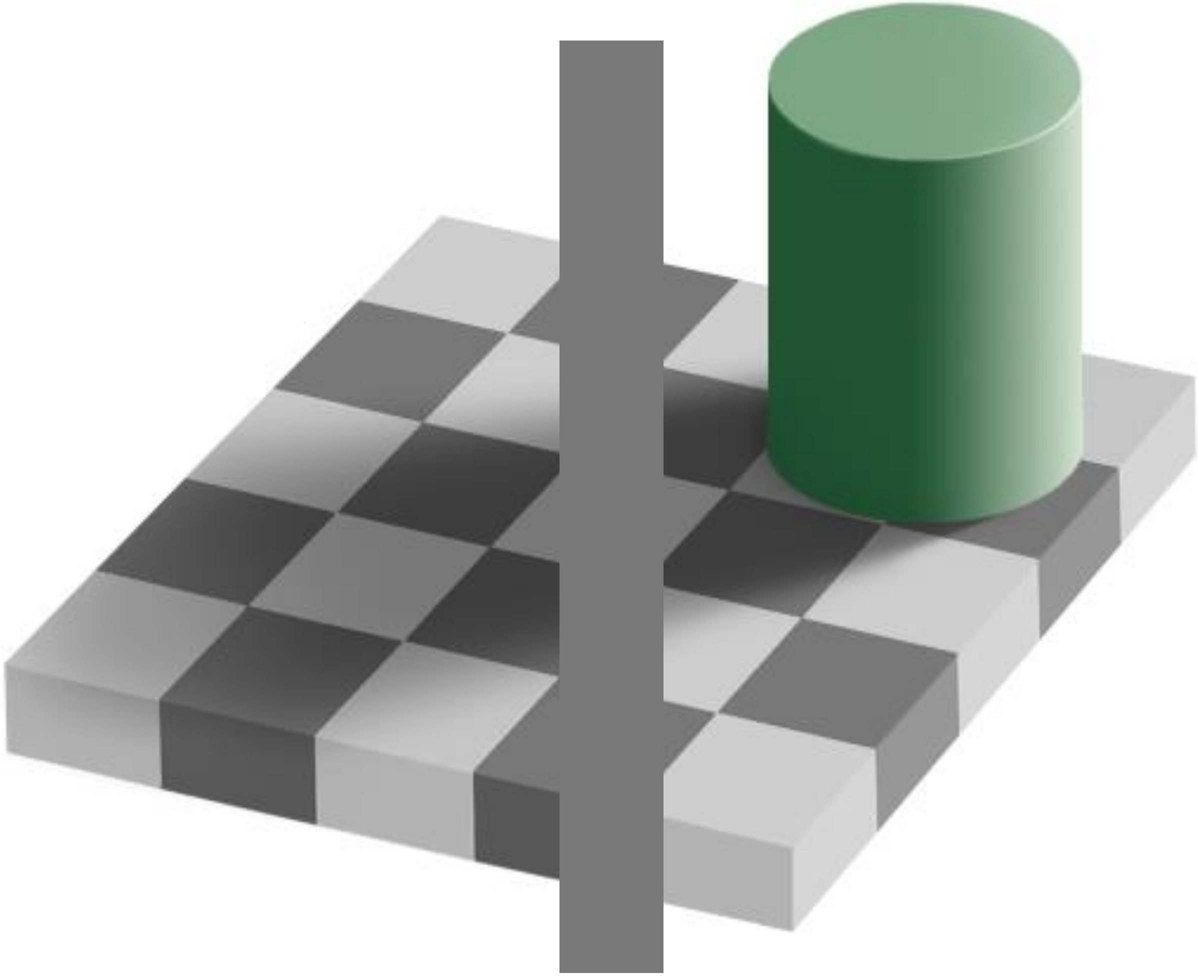
Another ‘trompe l’oeil’

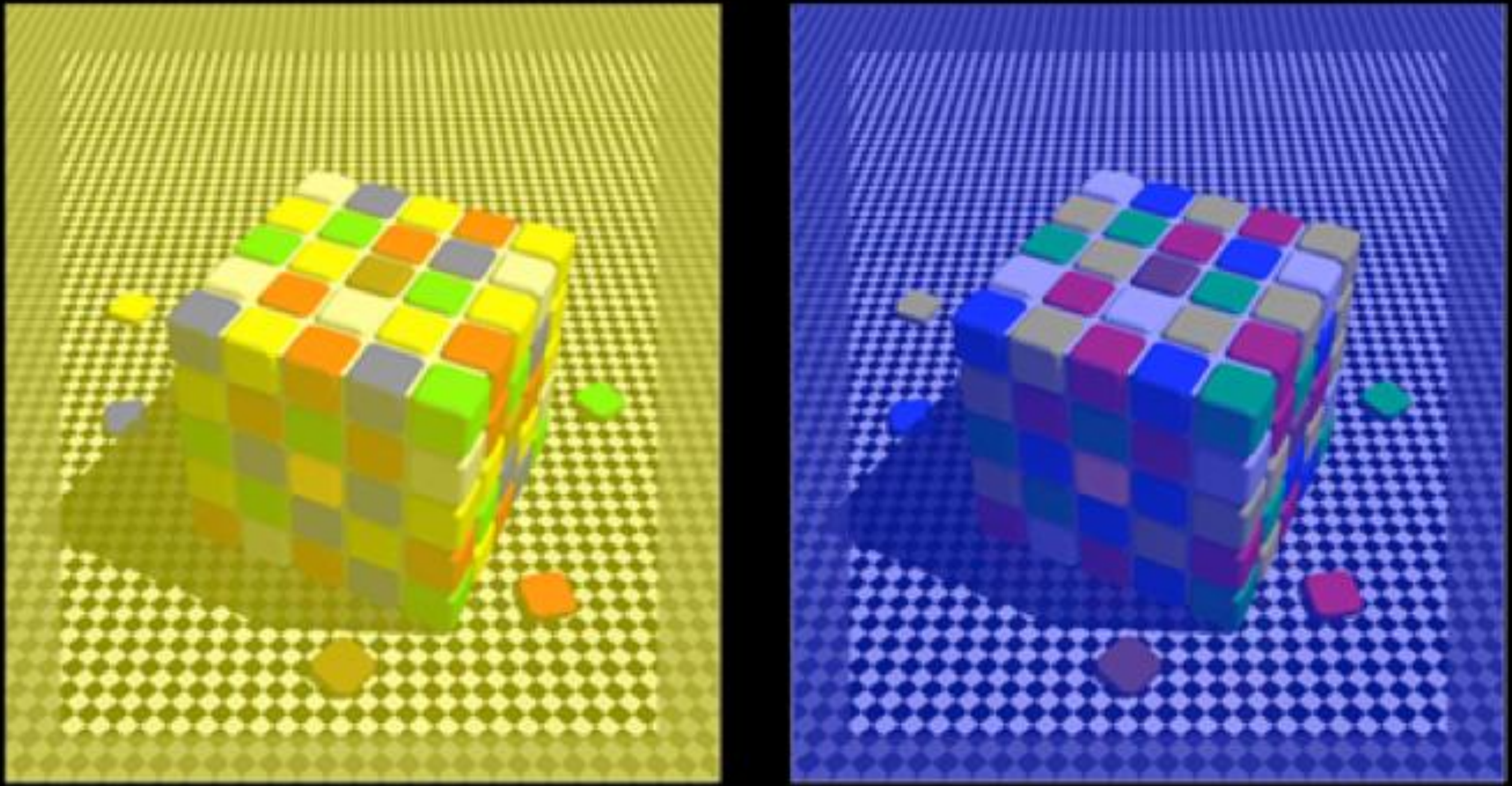
- illusory motion
  - only shadows changes
  - square is stationary

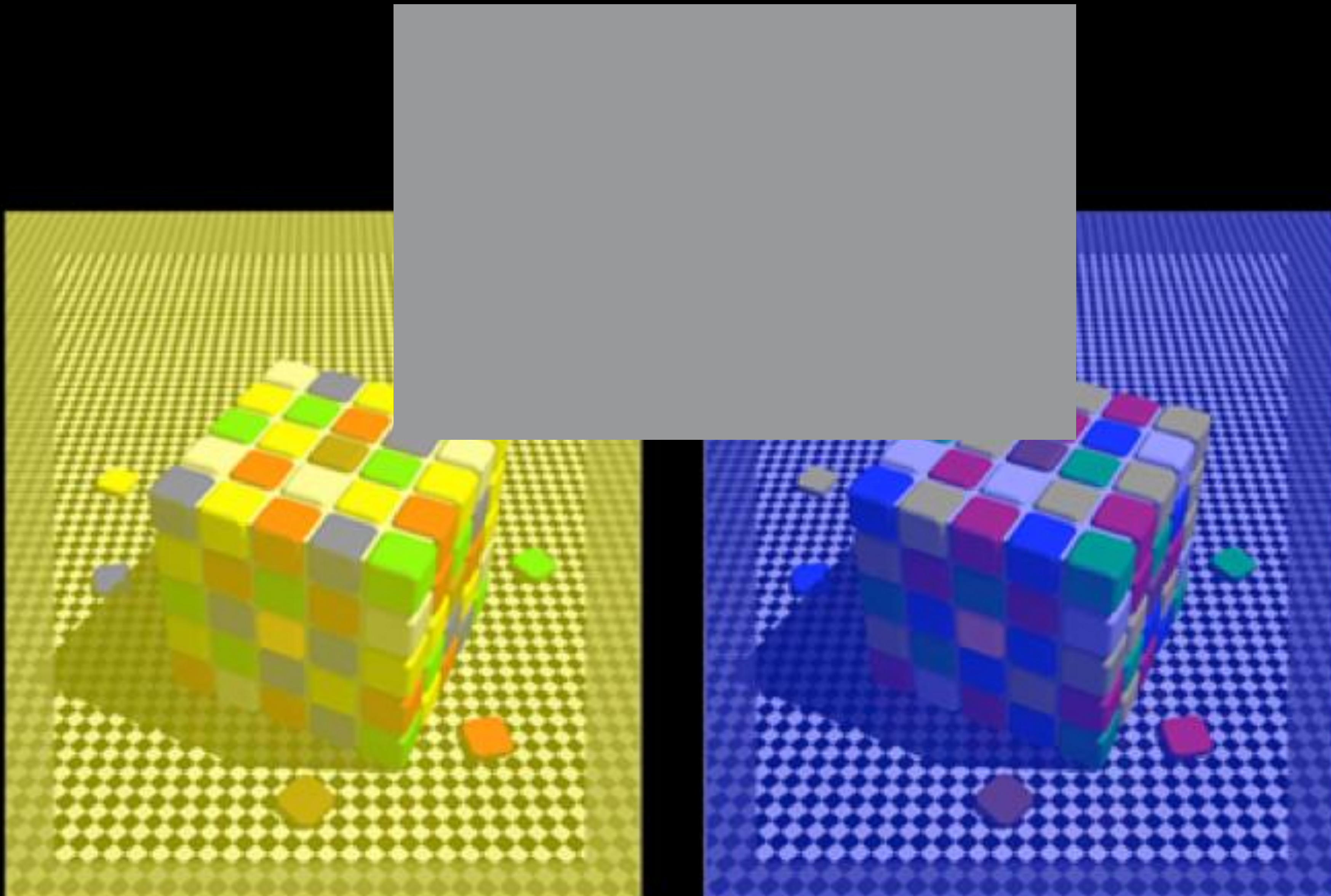
Color & Shading

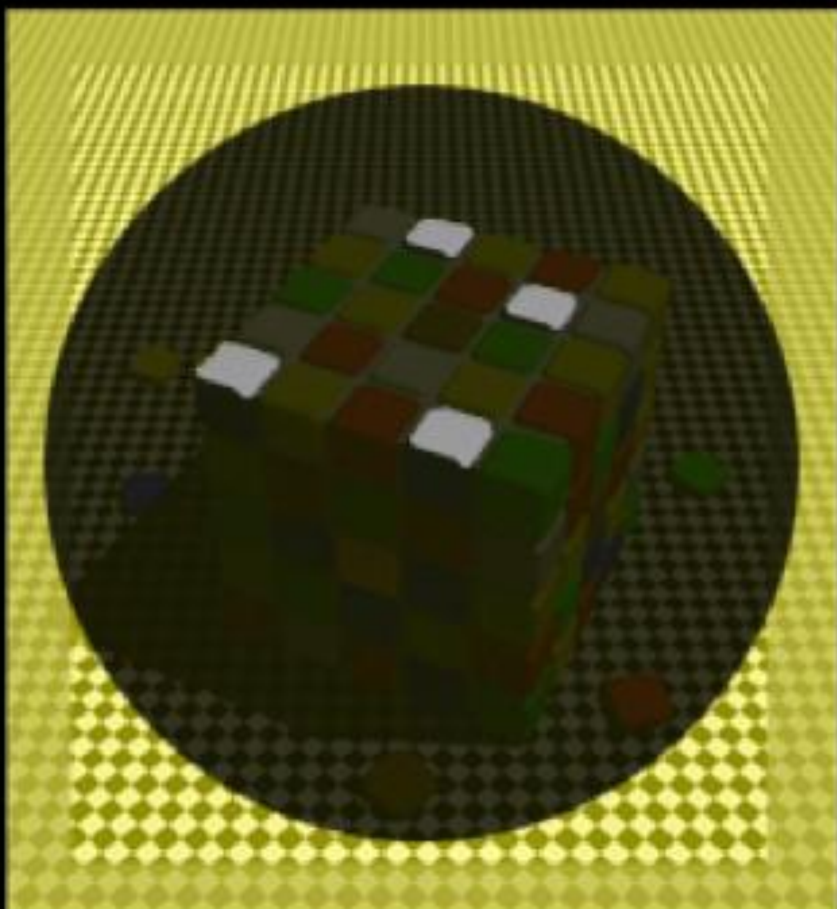
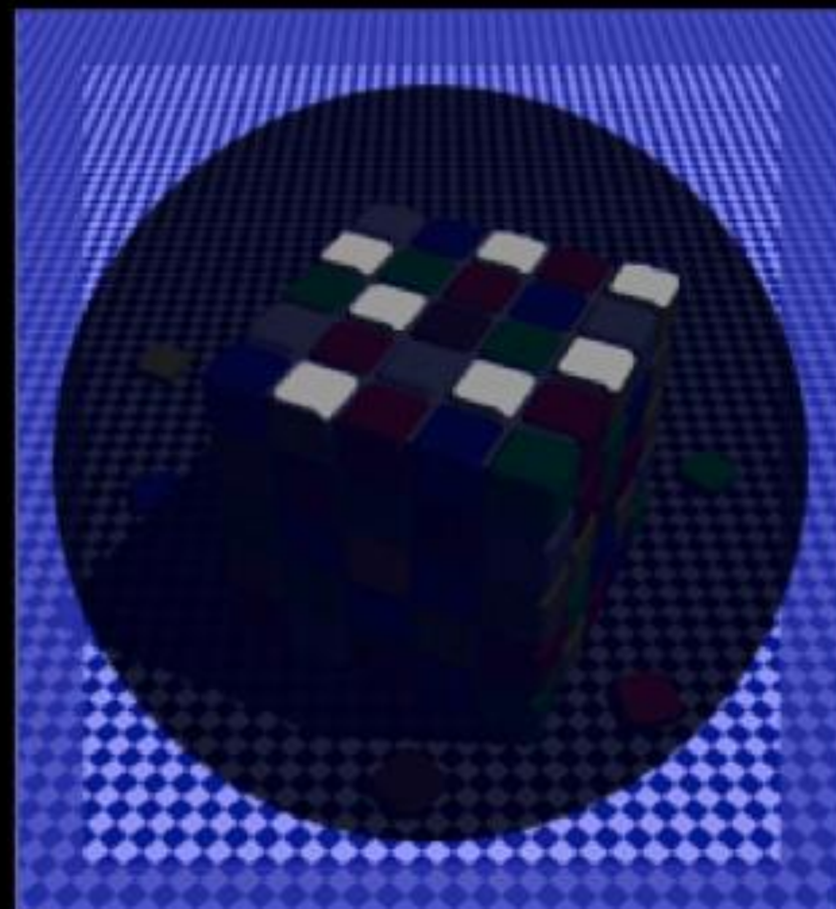


Color & Shading







Do you still believe what you see?

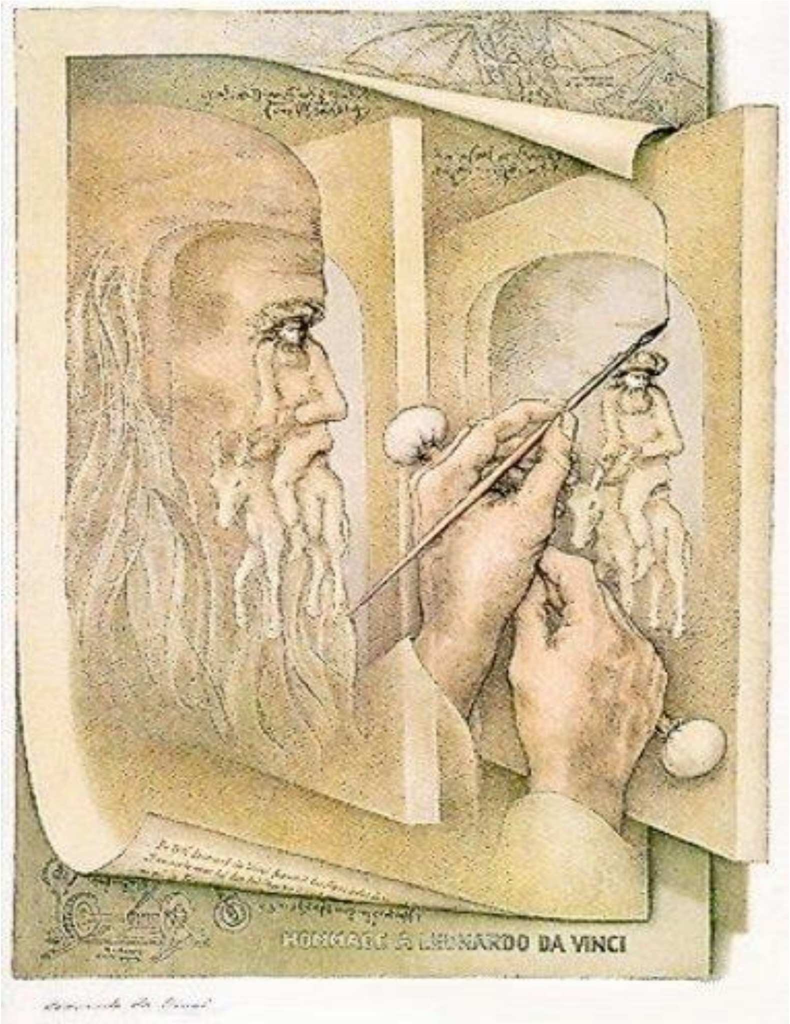
- Experiment
  - carefully point flash light into your eye from one corner
  - don’t hurt yourself!

- Observation
  - you’ll see your own blood vessels
  - they are actually in front of the retina
  - we’ve adapted to their usual shadow

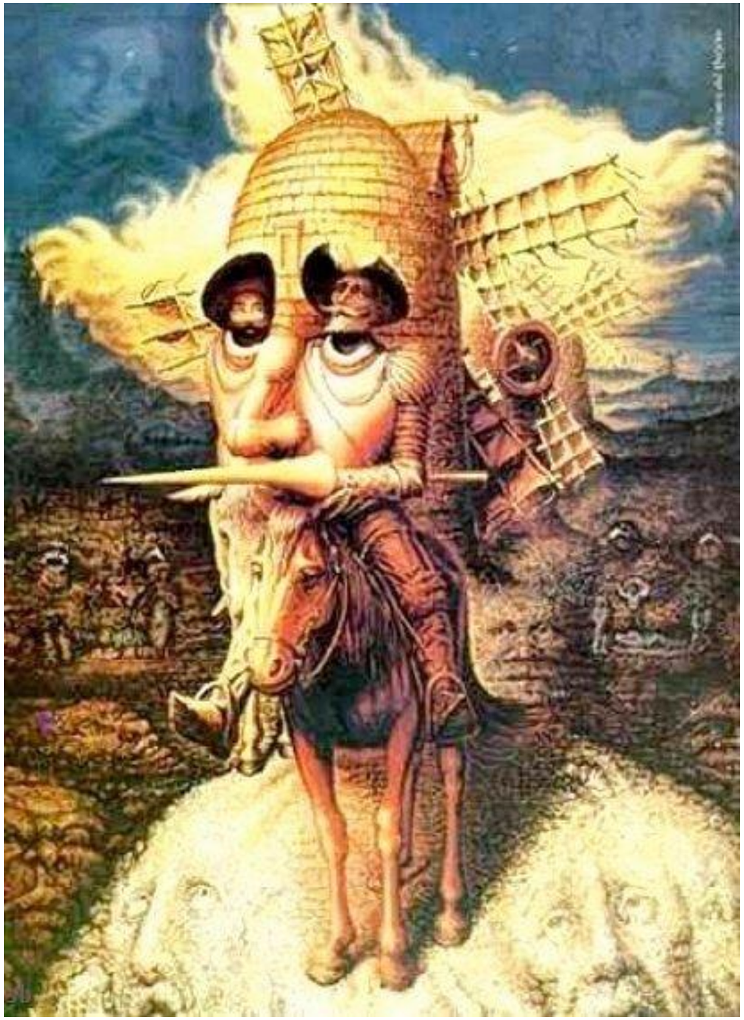
2. Case Study:
Computer Vision & Object Recognition

- is it more than inverse graphics?
- how do you recognize
  - the banana?
  - the glass?
  - the towel?
- how can we make computers to do this?
- ill posed problem:
  - missing data
  - ambiguities
  - multiple possible explanations

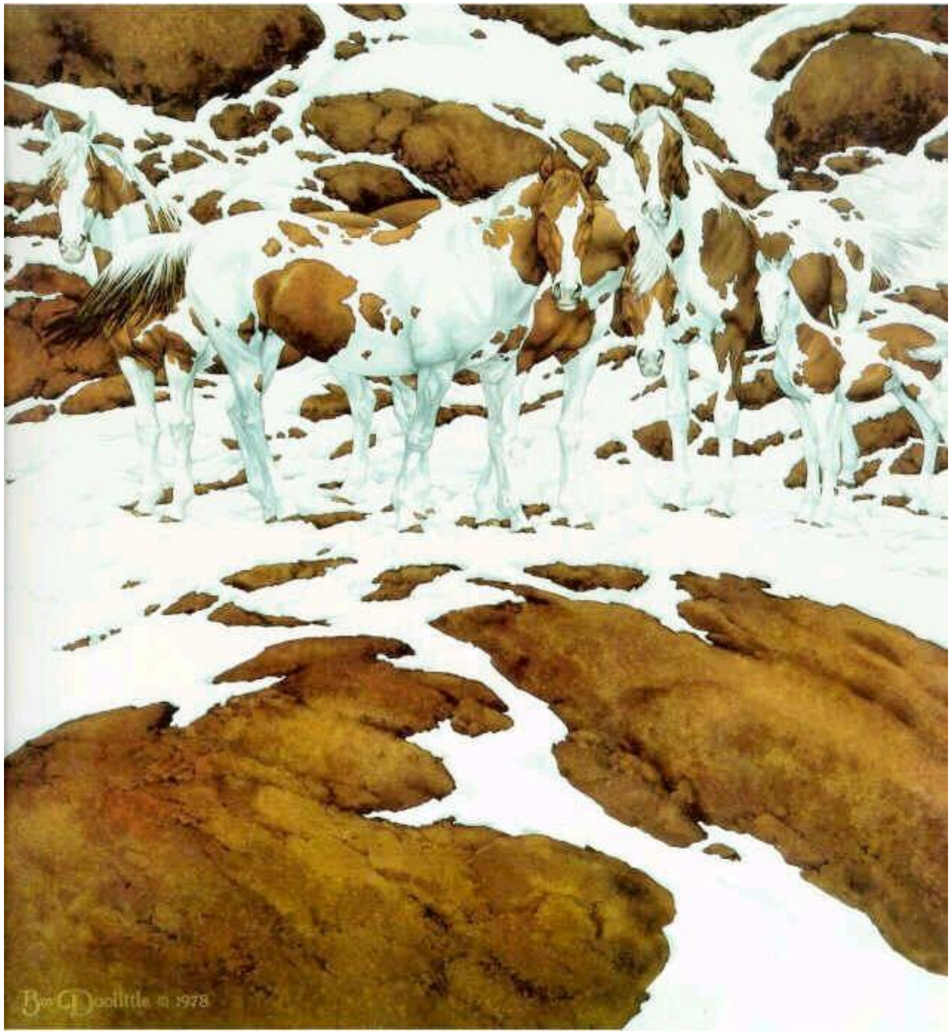
Complexity of Recognition



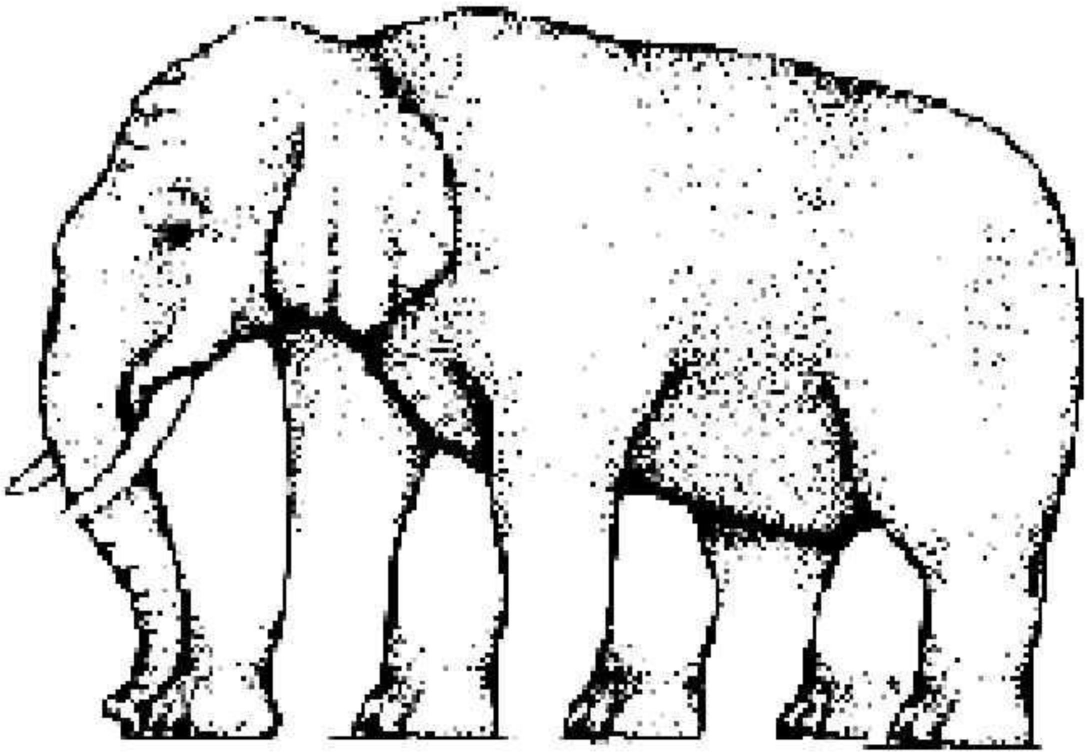
Complexity of Recognition



Complexity of Recognition



Complexity of Recognition



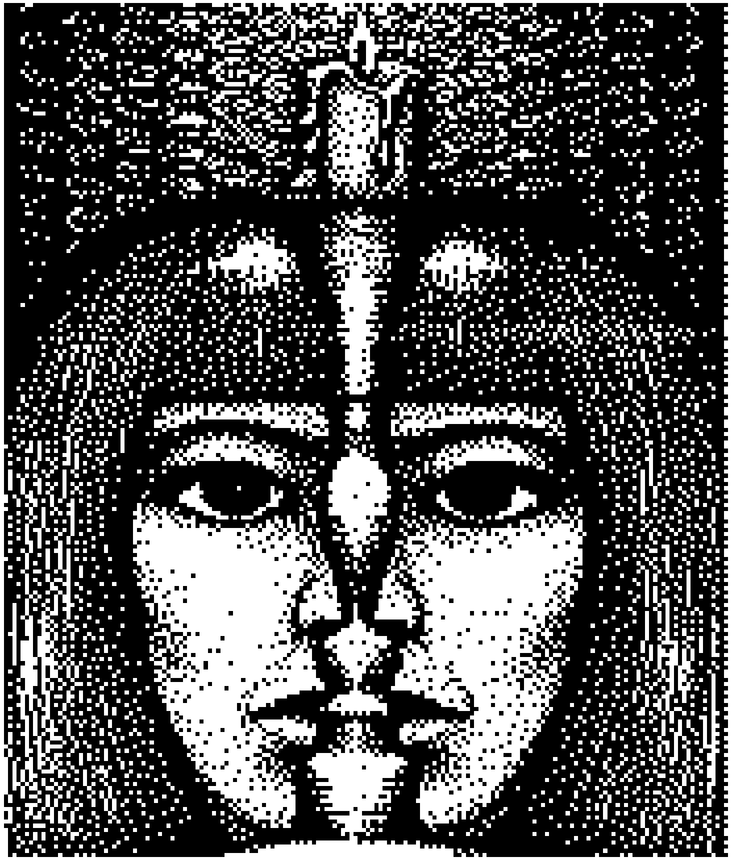
Recognition: the Role of Context

- Antonio Torralba

Recognition: the role of Prior Expectation

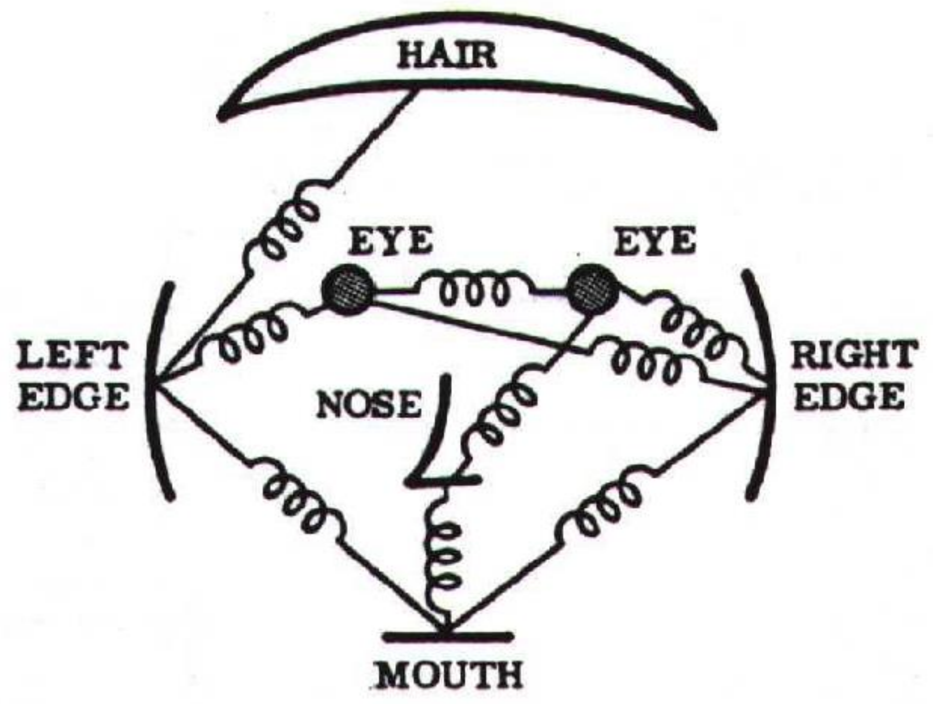
- Giuseppe Arcimboldo

One or Two Faces ?

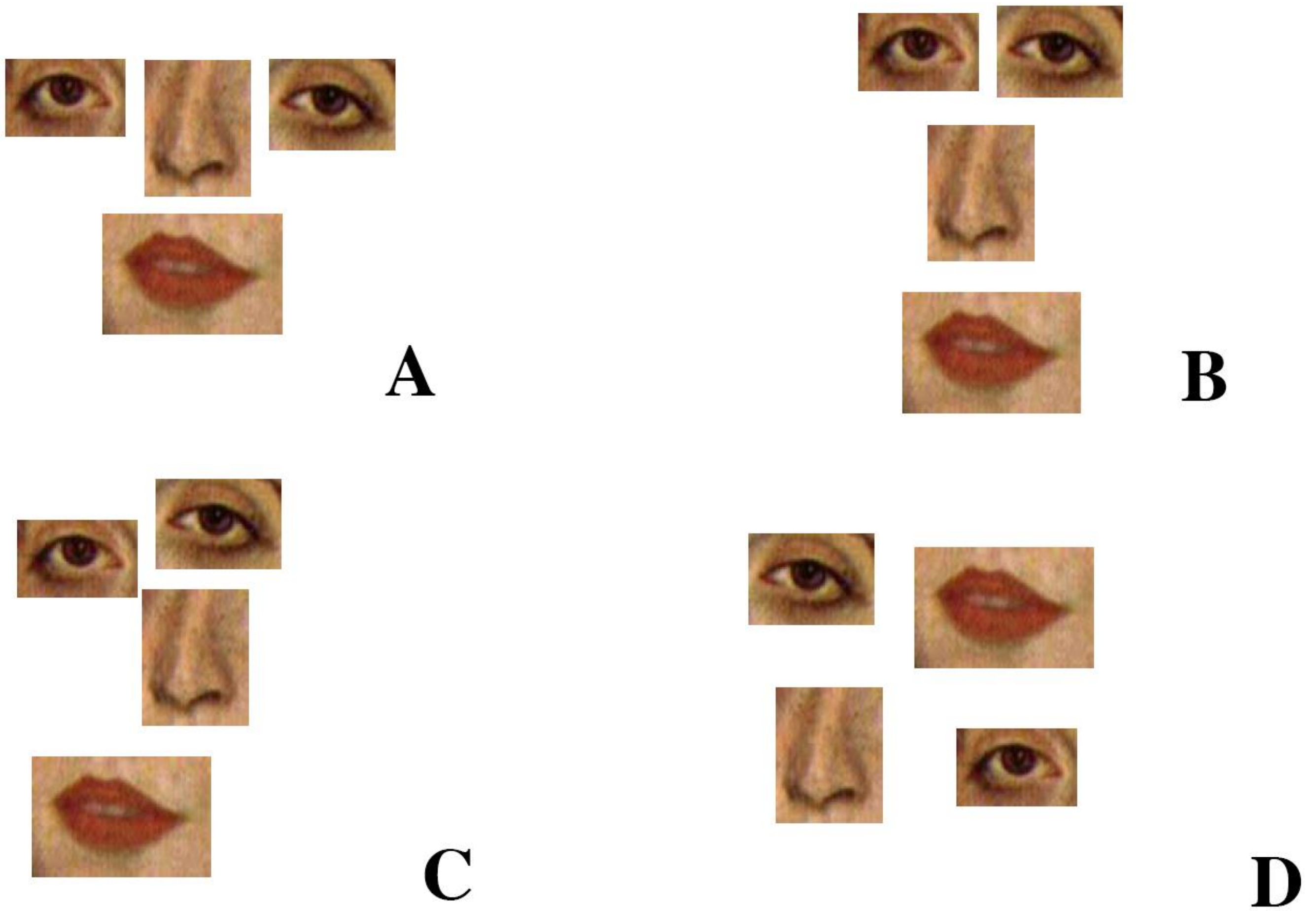


Class of Models: Pictorial Structure

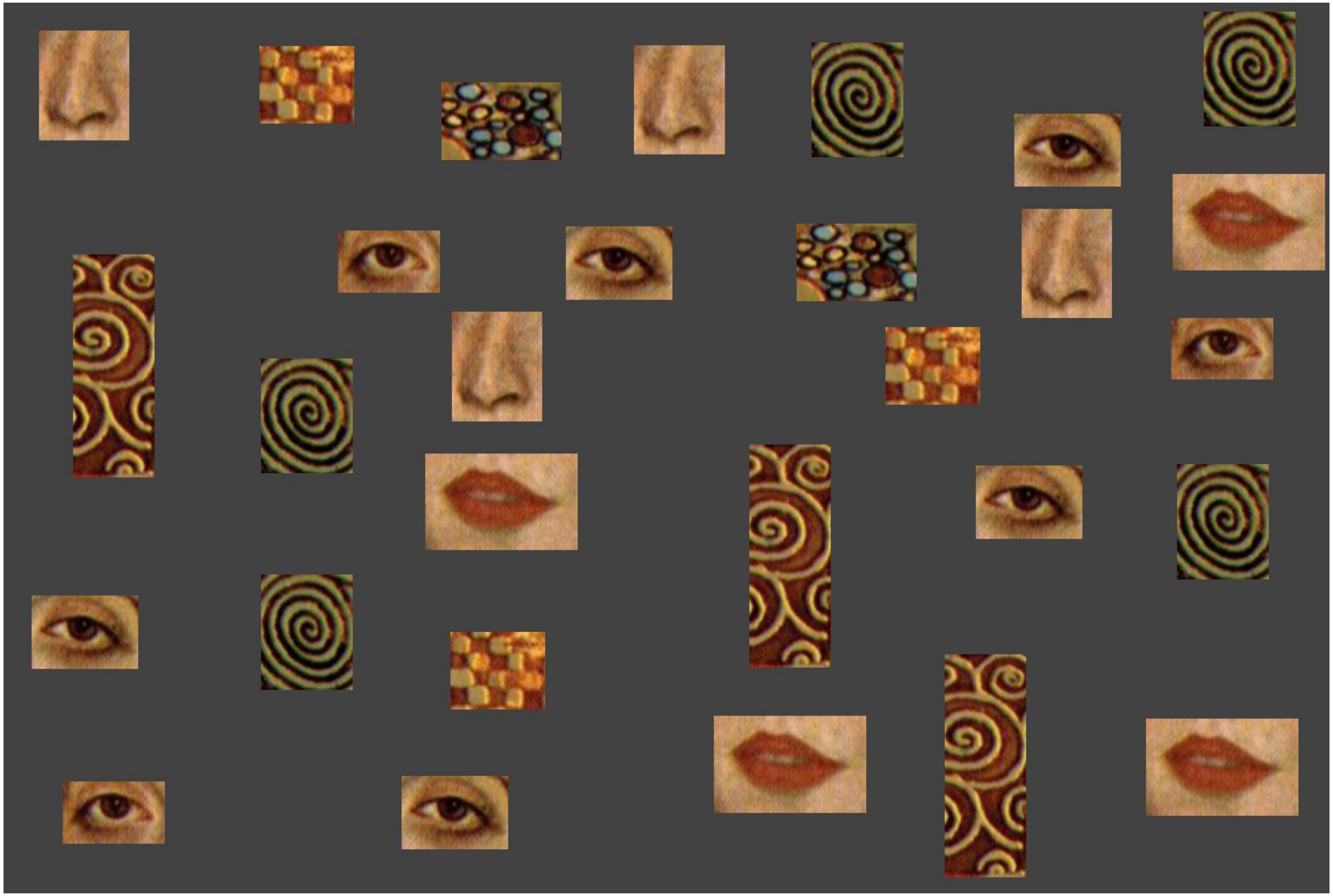
- Fischler & Elschlager 1973
- Model has two components
  - parts
    (2D image fragments)
  - structure
    (configuration of parts)



Deformations



Clutter



SAPIENZA
UNIVERSITÀ DI ROMA

Example

Recognition, Localization, and Segmentation

- a few terms
- … let’s briefly define what we mean by that

Object Recognition

- Different Types of Recognition Problems:
  - Object **Identification**
    - recognize your apple, your cup, your dog
  - Object **Classification**
    - recognize any apple, any cup, any dog
    - also called:
      generic object recognition,
      object categorization, …
    - typical definition: ‘basic level category’

- Recognition and
  - **Segmentation**: separate pixels belonging to the foreground (object) and the background
  - **Localization/Detection**: position of the object in the scene, pose estimate (orientation, size/scale, 3D position)

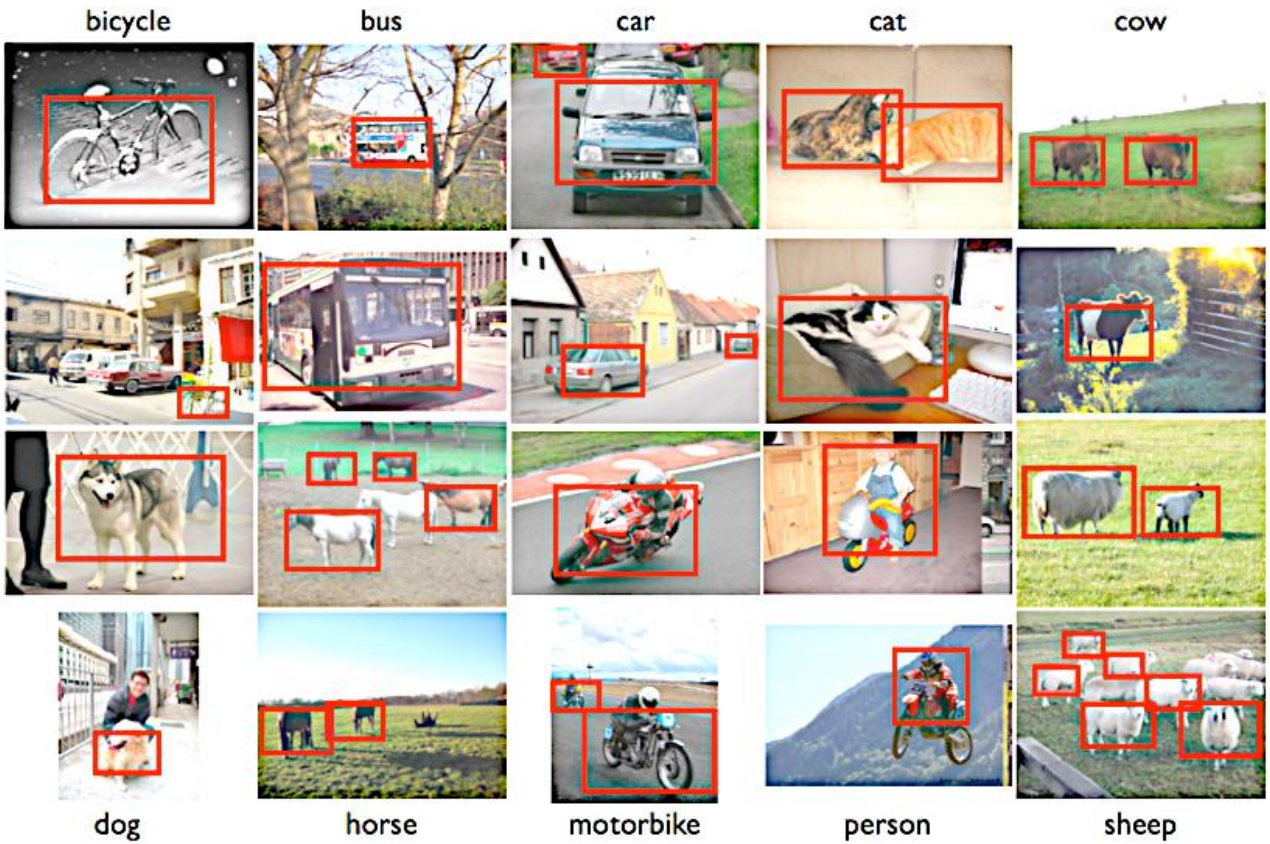
Object Recognition

- Different Types of Recognition Problems:
  - Object **Identification**
    - recognize your apple, your cup, your dog
  - Object **Classification**
    - recognize any apple, any cup, any dog
    - also called:
      generic object recognition,
      object categorization, …
    - typical definition: 'basic level category'

Which Level is right for Object Classes?

- Basic-Level Categories
  - the highest level at which category members have **similar perceived shape**
  - the highest level at which a **single mental image** can reflect the entire category
  - the highest level at which a person uses similar **motor actions** to interact with category members
  - the level at which human subjects are usually **fastest** at identifying category members
  - the first level named and understood by **children**
  - (while the definition of basic-level categories depends on culture there exist a remarkable consistency across cultures...)
- Most recent work in object recognition has focused on this problem
  - Most mature algorithms are in this field

Detection & Recognition of Visual Categories

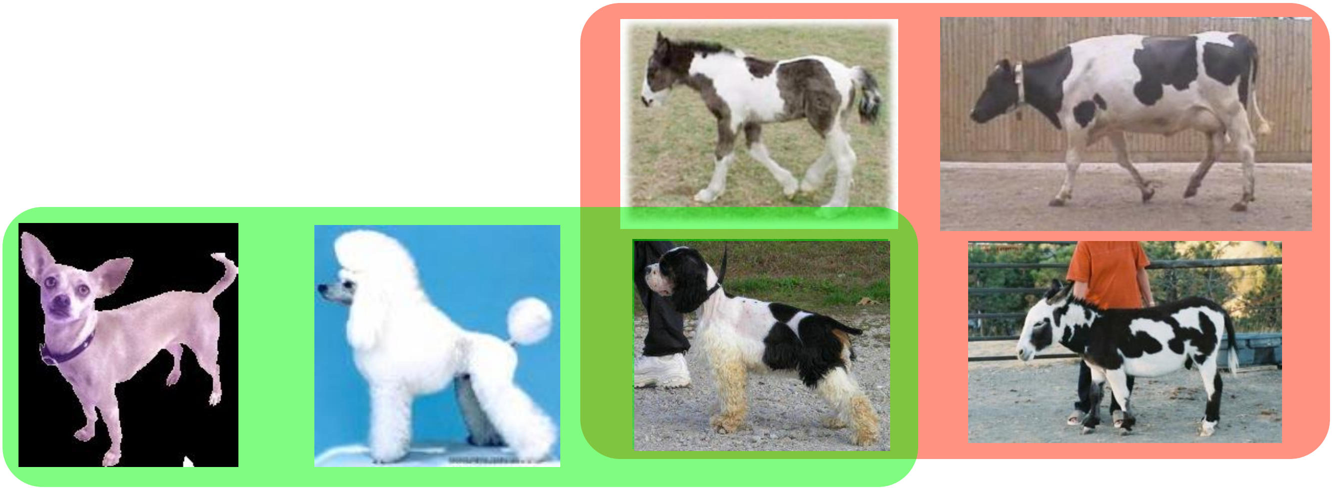


Challenges:

- multi-scale
- multi-view
- multi-class
- varying illumination
- occlusion
- cluttered background
- articulation
- high intraclass variance
- low interclass variance

Challenges of Visual Categorization

- low inter-class variation



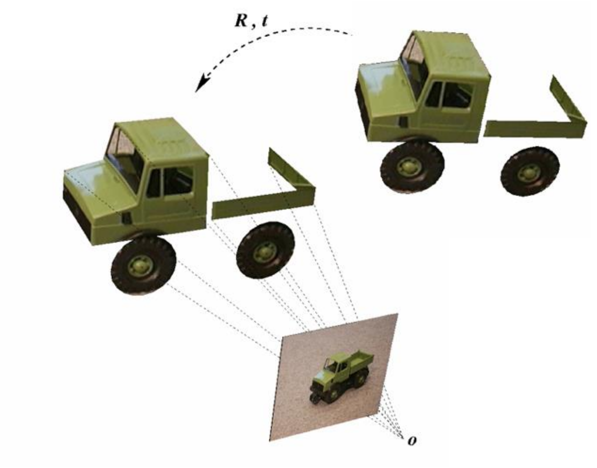
- large intra-class variation

More than Object Recognition

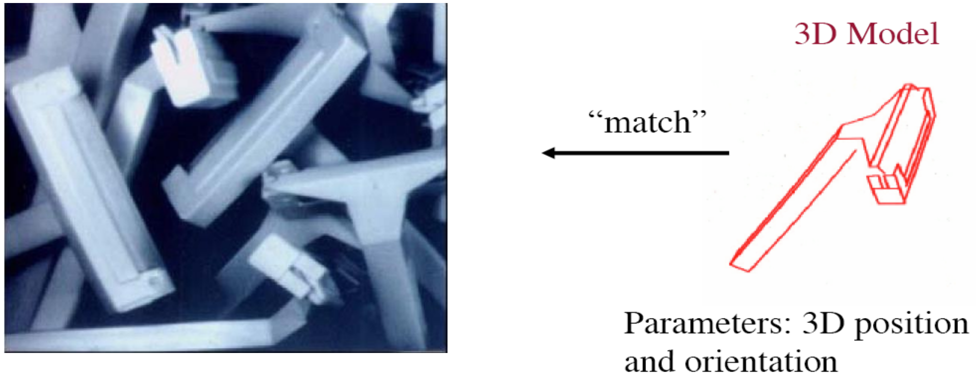
- Recognition and
  - **Segmentation**: separate pixels belonging to the foreground (object) and the background

More than Object Recognition

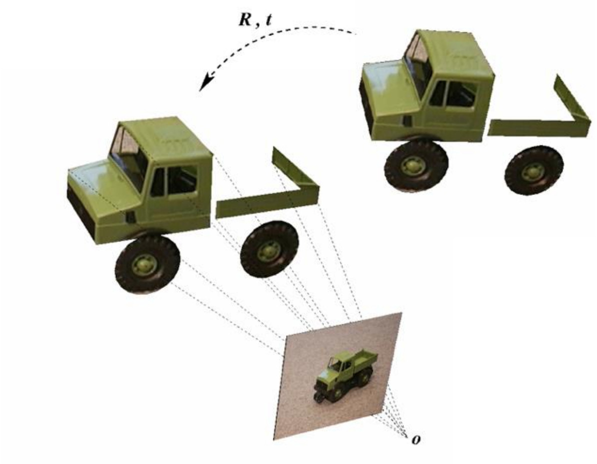
- Recognition and
  - **Localization**: to position the object in the scene, estimate the object's pose (orientation, size/scale, 3D position)

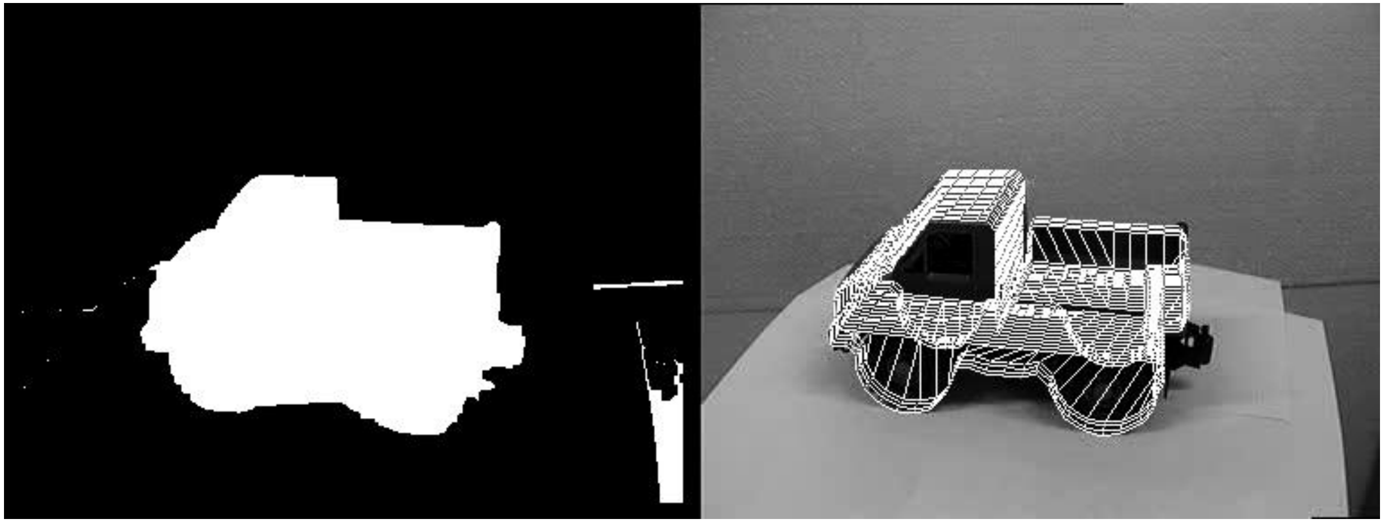


  - Example from David Lowe:



Localization: Example Video 1





Localization: Example Video 2

Object Recognition

- Different Types of Recognition Problems:
  - Object **Identification**
    - recognize your apple, your cup, your dog
  - Object **Classification**
    - recognize any apple, any cup, any dog
    - also called:
      generic object recognition,
      object categorization, …
    - typical definition: 'basic level category'

- Recognition and
  - **Segmentation**: separate pixels belonging to the foreground (object) and the background
  - **Localization/Detection**: position of the object in the scene, pose estimate (orientation, size/scale, 3D position)

Basics of Digital Image Filtering

Basics of Digital Image Filtering

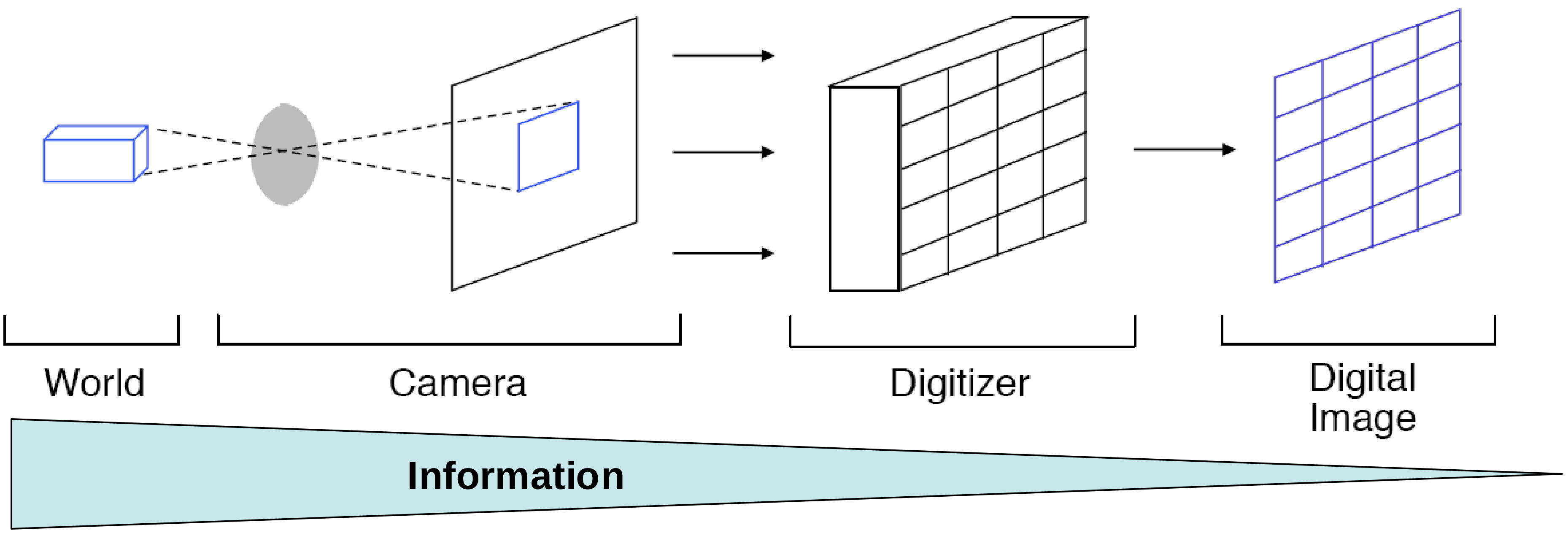
- Linear Filtering
  - Gaussian Filtering
- Multi Scale Image Representation
  - Gaussian Pyramid
- Edge Detection
  - ‘Recognition using Line Drawings’
  - Image derivatives (1st and 2nd order)
- Object Instance Identification using Color Histograms
- Performance evaluation

Basics of Digital Image Filtering

- Linear Filtering
  - Gaussian Filtering
- Multi Scale Image Representation
  - Gaussian Pyramid
- Edge Detection
  - ‘Recognition using Line Drawings’
  - Image derivatives (1st and 2nd order)
- Object Instance Identification using Color Histograms
- Performance evaluation

Computer Vision and its Components

- computer vision: ‘reverse’ the imaging process
  - **2D (2-dimensional) digital image processing**
  - ‘pattern recognition’ / 3D image analysis
  - image understanding



Digital Image Processing

- Image Filtering
  - take some local image patch (e.g. 3x3 block)
  - image filtering: apply some function to local image patch

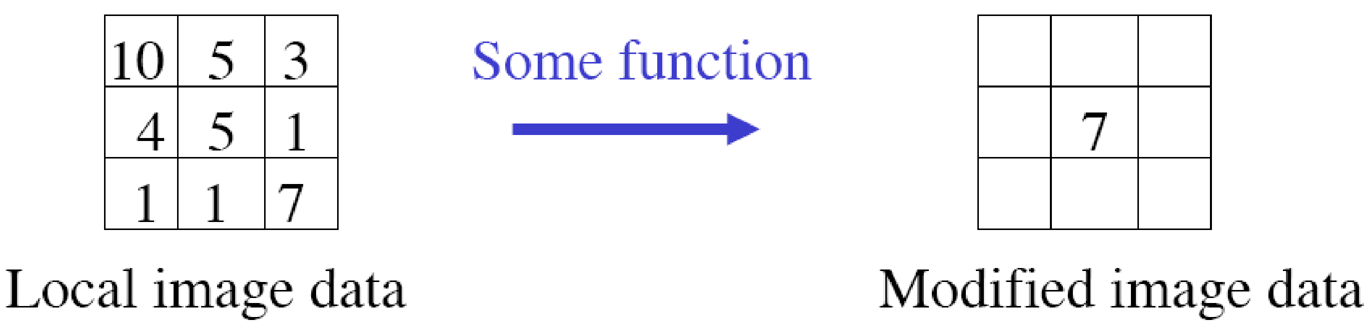


Image Filtering

- Some Examples:
  - what assumptions are you making to infer the center value?

3	3	3
3	?	3
3	3	3

3

3	4	3
2	?	5
5	4	2

3 or 4

Image Filtering: 2D Signals and Convolution

- Image Filtering
  - to reduce noise,
  - to fill-in missing values/information
  - to extract image features (e.g. edges/corners), etc.

$$\begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 20 & 2 \\ \hline 3 & 2 & 3 \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 3 & 2 \\ \hline 3 & 2 & 3 \\ \hline \end{array}$$

- Simplest case:
  - linear filtering: replace each pixel by a linear combination of its neighbors

- 2D convolution (discrete): $f[m,n] = I \otimes g = \sum_{k,l} I[m-k, n-l]g[k,l]$
  - discrete Image: *I[m,n]*
  - filter 'kernel': *g[k,l]*
  - 'filtered' image: *f[m,n]*

$$\underset{f[m,n]}{\begin{array}{|c|c|c|}\hline & & \\ \hline & 18 & \\ \hline & & \\ \hline \end{array}} = \underset{I[k,l]}{\begin{array}{|c|c|c|}\hline 8 & 5 & 2 \\ \hline 7 & 5 & 3 \\ \hline 9 & 4 & 1 \\ \hline \end{array}} \otimes \underset{g[k,l]}{\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}}$$

can be expressed as matrix multiplication!

Image Filtering: 2D Signals and Convolution

- 2D convolution (discrete): $f[m,n] = I \otimes g = \sum_{k,l} I[m-k, n-l]g[k,l]$
  - discrete Image: *I[m,n]*
  - filter 'kernel': *g[k,l]*
  - 'filtered' image: *f[m,n]*

$$
\underset{f[m,n]}{\begin{array}{|c|c|c|}\hline \quad & \quad & \quad \\ \hline & 18 & \\ \hline & & \\ \hline \end{array}}
=
\underset{I[k,l]}{\begin{array}{|c|c|c|}\hline 8 & 5 & 2 \\ \hline 7 & 5 & 3 \\ \hline 9 & 4 & 1 \\ \hline \end{array}}
\otimes
\underset{g[k,l]}{\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}}
$$

(m,n) l k (m,n) l k

$$= \sum_{\substack{-1<k<+1 \\ -1<l<+1}} I[m-k, n-l]g[k,l] \quad (m,n)$$

- mirror the filter (k and l)
- swipe it across the image
- multiply and sum

$$
\begin{aligned}
&= I[m+1, n+1]g[-1,-1] && (k=-1, l=-1) \\
&\quad + I[m+1, n]g[-1,0] && (k=-1, l=0) \\
&\quad + I[m+1, n-1]g[-1,+1] && (k=-1, l=+1) \\
&\quad + \ldots
\end{aligned}
$$

Image Filtering: 2D Signals and Convolution

- 2D convolution (discrete): $f[m,n] = I \otimes g = \sum_{k,l} I[m-k, n-l]g[k,l]$
  - discrete Image: *I[m,n]*
  - filter 'kernel': *g[k,l]*
  - 'filtered' image: *f[m,n]*

$$\underset{f[m,n]}{\begin{array}{|c|c|c|}\hline & & \\ \hline & 18 & \\ \hline & & \\ \hline\end{array}} = \underset{I[k,l]}{\begin{array}{|c|c|c|}\hline 8 & 5 & 2 \\ \hline 7 & 5 & 3 \\ \hline 9 & 4 & 1 \\ \hline\end{array}} \otimes \underset{g[k,l]}{\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline\end{array}}$$

- special case:
  - convolution (discrete) of a 2D-image with a 1D-filter

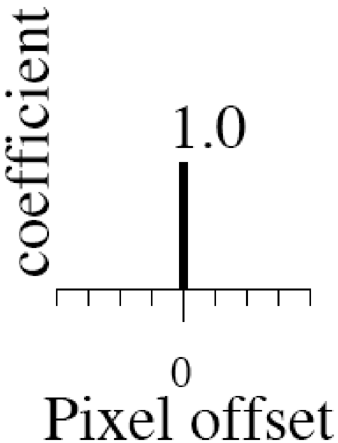
$$f[m,n] = I \otimes g = \sum_{k} I[m-k, n]g[k]$$

$$\underset{g[k]}{\begin{array}{|c|}\hline -1 \\ \hline 0 \\ \hline 1 \\ \hline\end{array}}$$

Linear Filtering (warm-up slide)

$$f[m,n] = I \otimes g = \sum_k I[m-k,n]g[k]$$

original



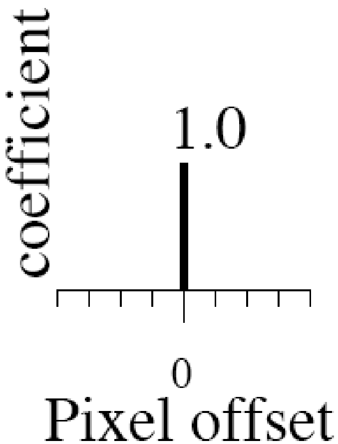
$= ?$

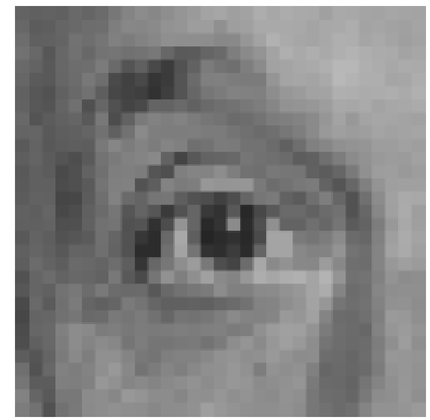
$I \quad \otimes \quad g \quad = f$

Linear Filtering (warm-up slide)

$$f[m,n] = I \otimes g = \sum_k I[m-k,n]g[k]$$

original





Filtered
(no change)

$$I \quad \otimes \quad g \quad = f$$

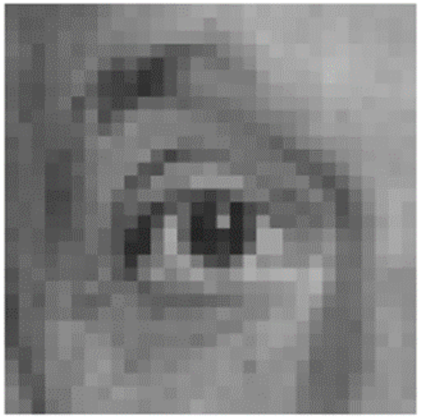
Try it out in GIMP

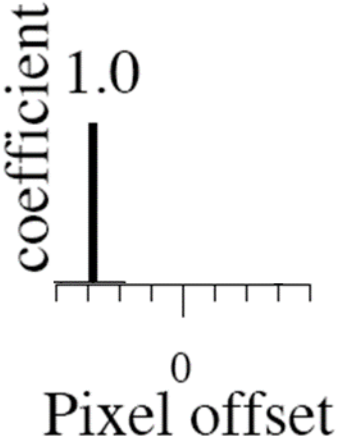
- You can try out linear filter kernels in the free image manipulation tool GIMP - availble at gimp.org

- open image
- from the menu pick:
  - Filters
    - Generic
      - Convolution Matrix ...
- enter filter kernel in “Matrix”
- press “ok” to apply

Linear Filtering

$$f[m,n] = I \otimes g = \sum_k I[m-k,n]g[k]$$





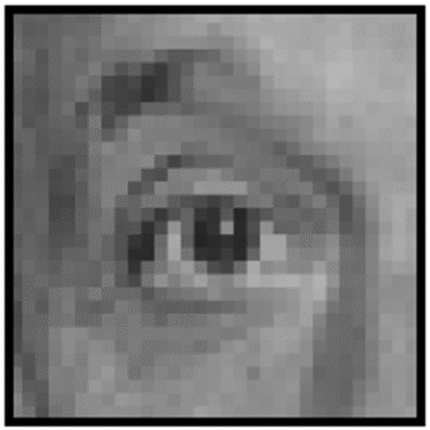
?

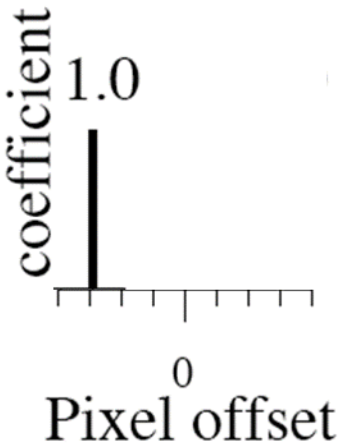
original

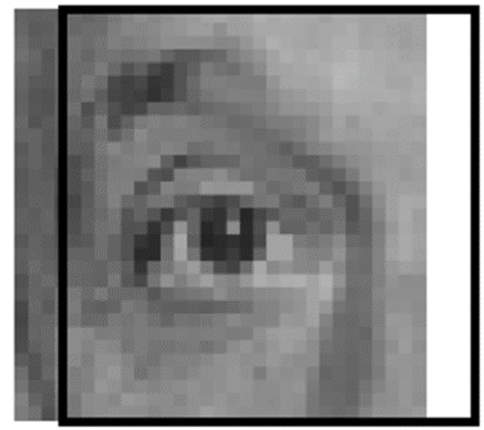
$$I \qquad \otimes \qquad g \qquad = f$$

Linear Filtering

$$f[m,n] = I \otimes g = \sum_k I[m-k,n]g[k]$$







original

$I \qquad \otimes \qquad g$

shifted

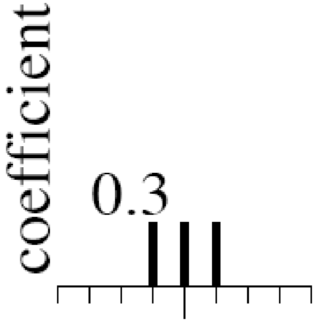
$= f$

Sapienza Università di Roma

Linear Filtering

$$f[m,n] = I \otimes g = \sum_k I[m-k,n]g[k]$$

original



?

$I \quad \otimes \quad g \quad = f$

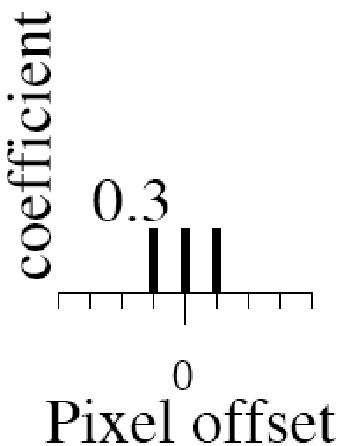
Blurring

$$f[m,n] = I \otimes g = \sum_k I[m-k,n]g[k]$$

original

I

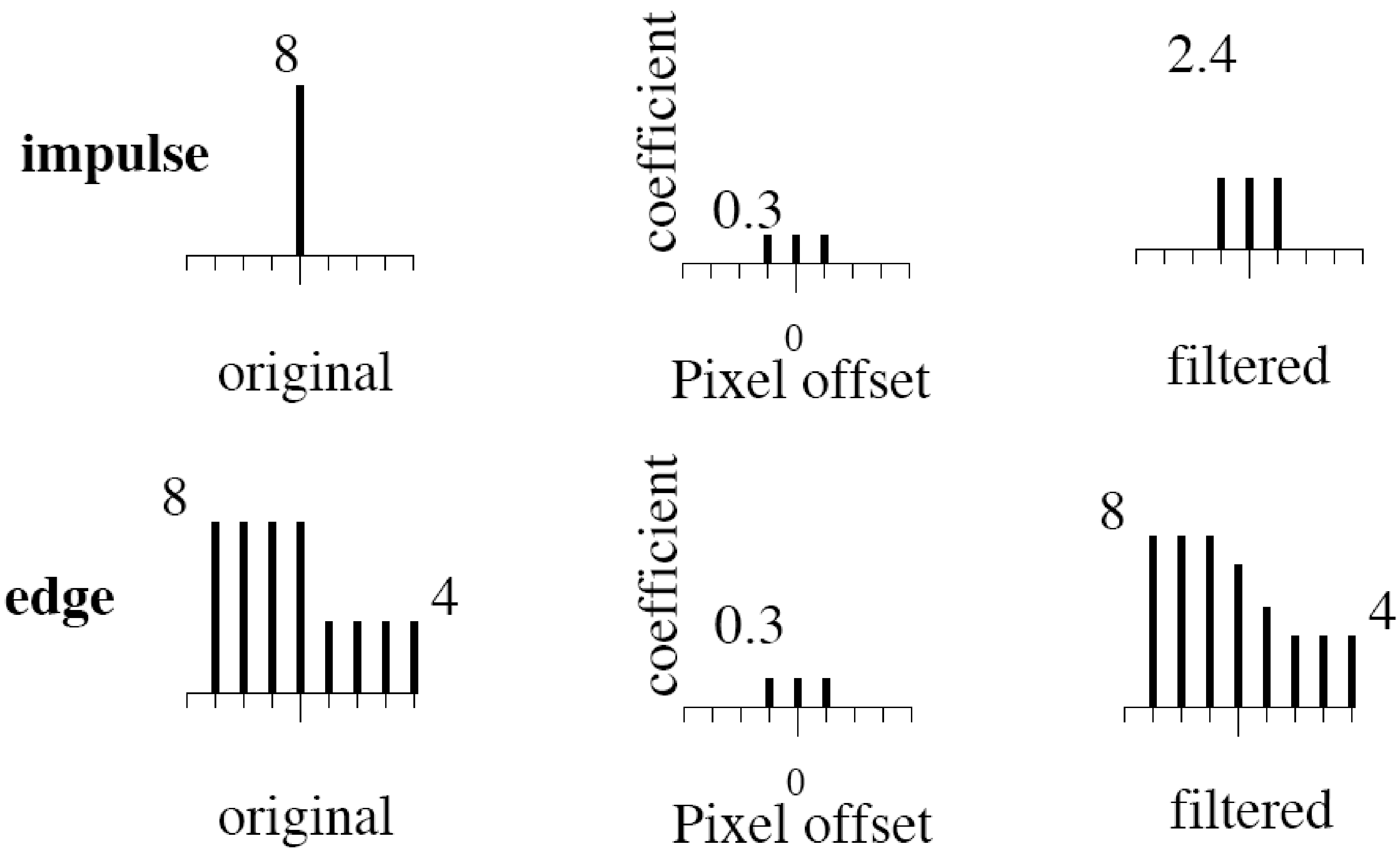
$\otimes$



g

Blurred (filter applied in both dimensions).

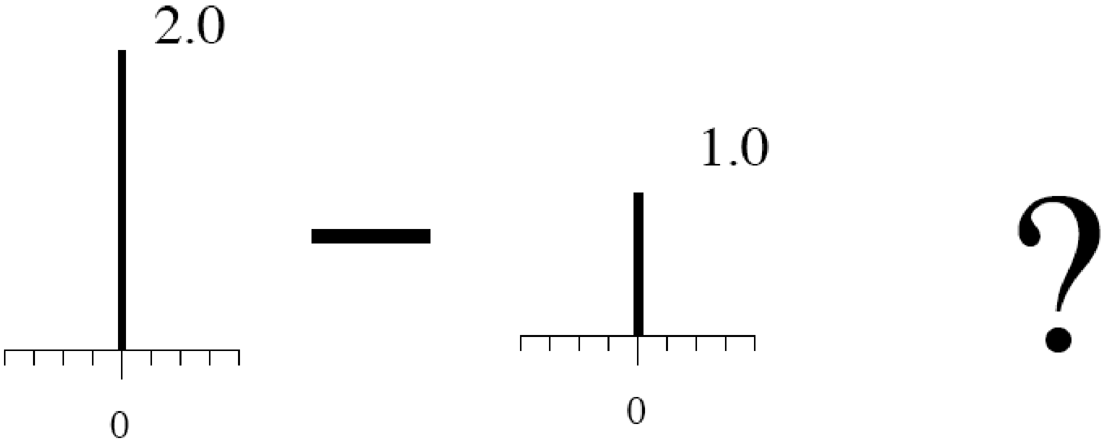
Blurring Examples



Linear Filtering (warm-up slide)

$$f[m,n] = I \otimes g_1 - I \otimes g_2 = I \otimes (g_1 - g_2)$$

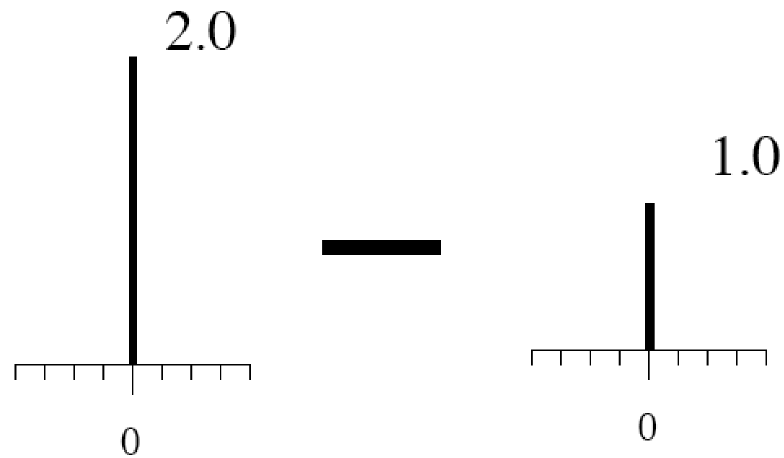
original



Linear Filtering (warm-up slide)

$$f[m,n] = I \otimes g_1 - I \otimes g_2 = I \otimes (g_1 - g_2)$$

original

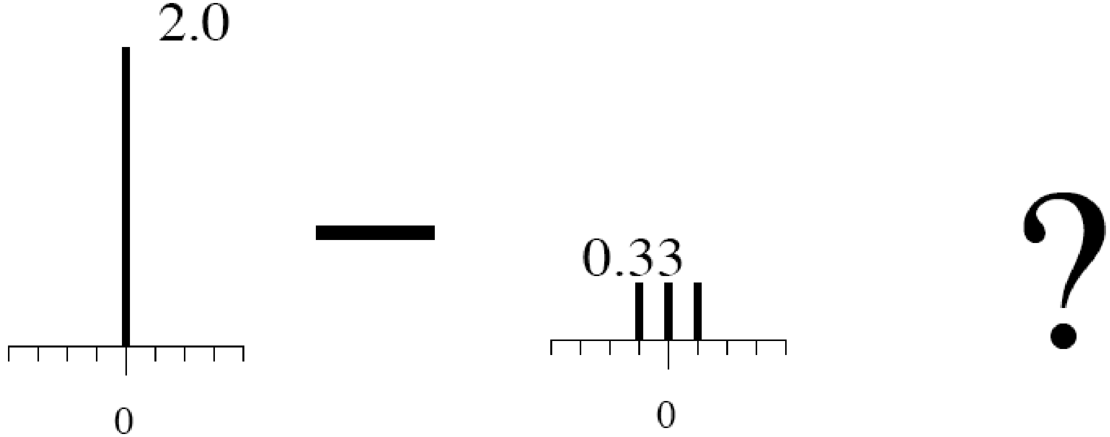


Filtered
(no change)

Linear Filtering

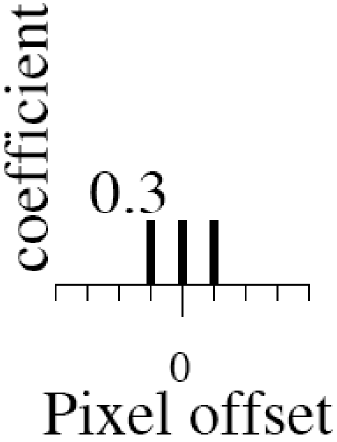
$$f[m,n] = I \otimes g_1 - I \otimes g_2 = I \otimes (g_1 - g_2)$$

original



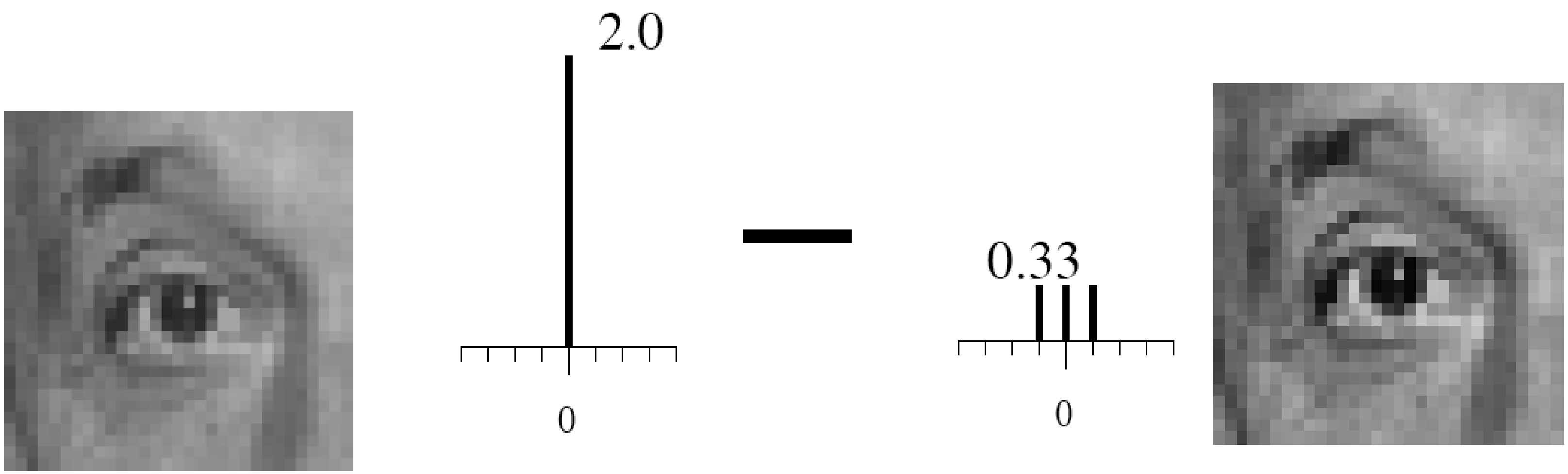
(remember blurring)

original



Blurred (filter applied in both dimensions).

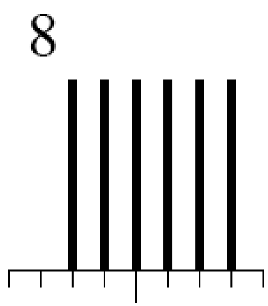
Sharpening

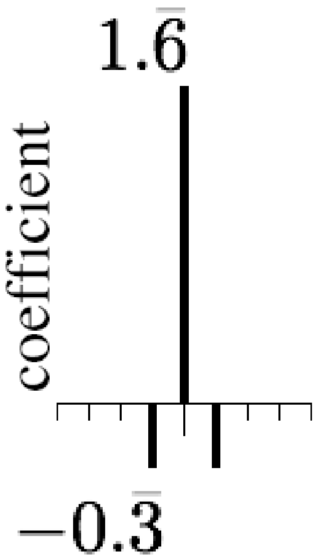


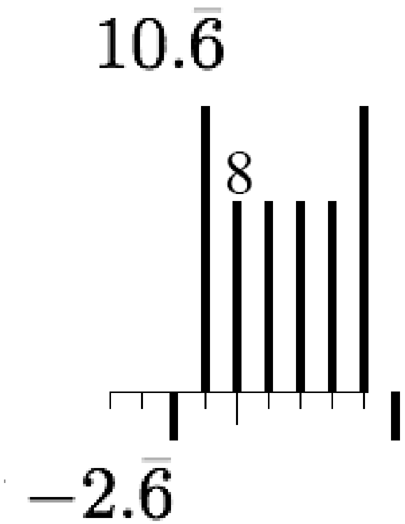
original

Sharpened original

Sharpening Example

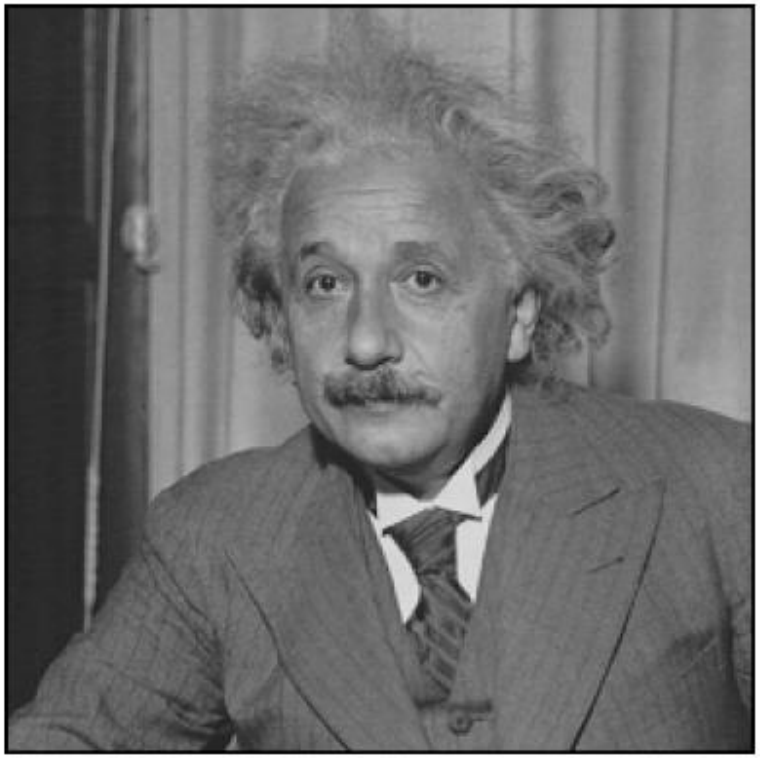




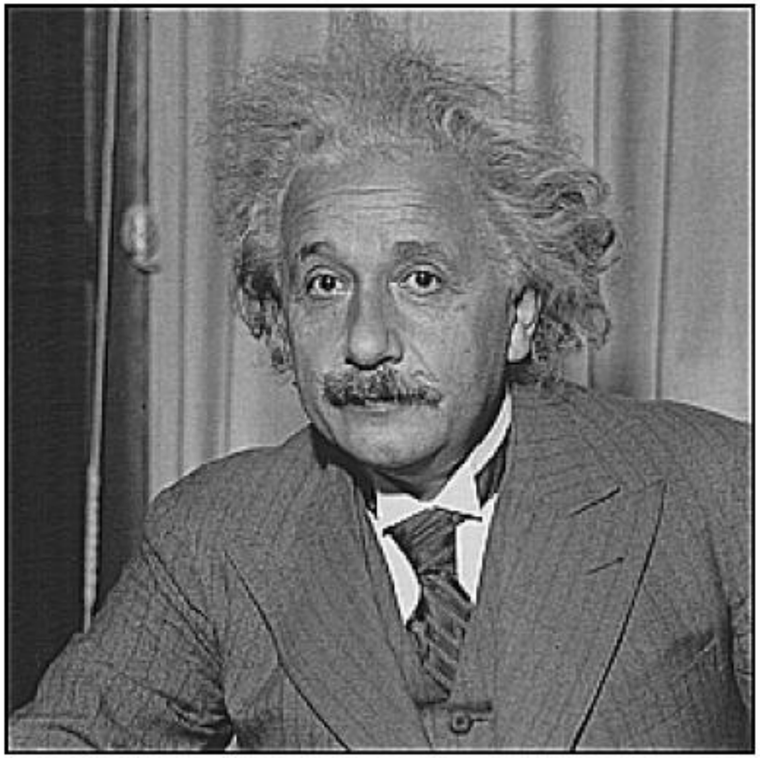


Sharpened
(differences are accentuated; constant areas are left untouched).

Sharpening



before



after

Image Filtering
Interim summary

- Images may need low-level adjustment such as filtering, in order to enhance image quality (e.g. denoising) or extract useful information (e.g. edges)
- Filtering for enhancement → improve contrast
- Filtering for smoothing → removes noise
- Filtering for template matching → detect known patterns

Image Filtering: 2D Signals and Convolution

- 2D convolution (discrete): $f[m,n] = I \otimes g = \sum_{k,l} I[m-k, n-l]g[k,l]$
  - discrete Image: *I[m,n]*
  - filter 'kernel': *g[k,l]*
  - 'filtered' image: *f[m,n]*

$$\underset{f[m,n]}{\begin{array}{|c|c|c|}\hline & & \\ \hline & 18 & \\ \hline & & \\ \hline\end{array}} = \underset{I[k,l]}{\begin{array}{|c|c|c|}\hline 8 & 5 & 2 \\ \hline 7 & 5 & 3 \\ \hline 9 & 4 & 1 \\ \hline\end{array}} \otimes \underset{g[k,l]}{\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline\end{array}}$$

- special case:
  - convolution (discrete) of a 2D-image with a 1D-filter

$$f[m,n] = I \otimes g = \sum_{k} I[m-k, n]g[k]$$

$g[k]$

-1
0
1

Linear Systems

- Basic Properties:
  - homogeneity $T[a\ X] = a\ T[X]$
  - additivity $T[X_1 + X_2] = T[X_1] + T[X_2]$
  - superposition $T[aX_1 + bX_2] = a\ T[X_1] + b\ T[X_2]$
  - linear systems $\Leftrightarrow$ superposition

- examples:
  - matrix operations (additions, multiplication)
  - convolutions

Filtering to Reduce Noise

- "Noise" is what we're not interested in
  - low-level noise: light fluctuations, sensor noise, quantization effects, finite precision, …
  - complex noise (not today): shadows, extraneous objects.

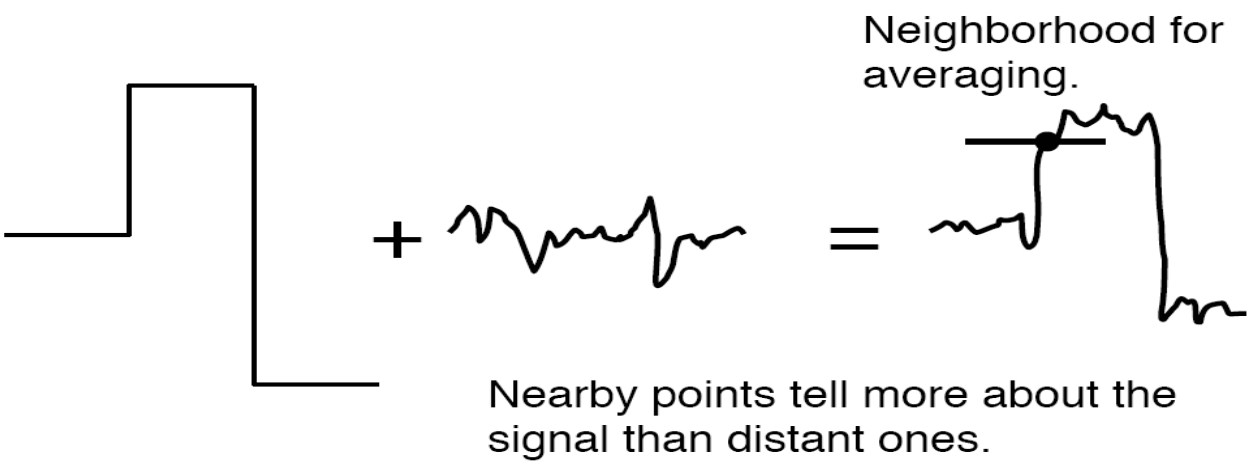
- Assumption:
  - the pixel's neighborhood contains information about its intensity

$$\begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 20 & 2 \\ \hline 3 & 2 & 3 \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|c|}\hline 2 & 3 & 3 \\ \hline 3 & 3 & 2 \\ \hline 3 & 2 & 3 \\ \hline \end{array}$$

Model: Additive Noise

- Image I = Signal S + Noise N:

$$S \qquad + \qquad N \qquad\qquad = \qquad I$$



Model: Additive Noise

- Image I = Signal S + Noise N
  - i.e. noise does not depend on the signal

- we consider:
  - I_i : intensity of i'th pixel
  - $I_i = s_i + n_i$ with $E(n_i) = 0$
    - s_i deterministic
    - n_i, n_j independent for $i \neq j$
    - n_i, n_j i.i.d. (independent, identically distributed)

- therefore:
  - intuition: averaging noise reduces its effect
  - better: smoothing as inference about the signal

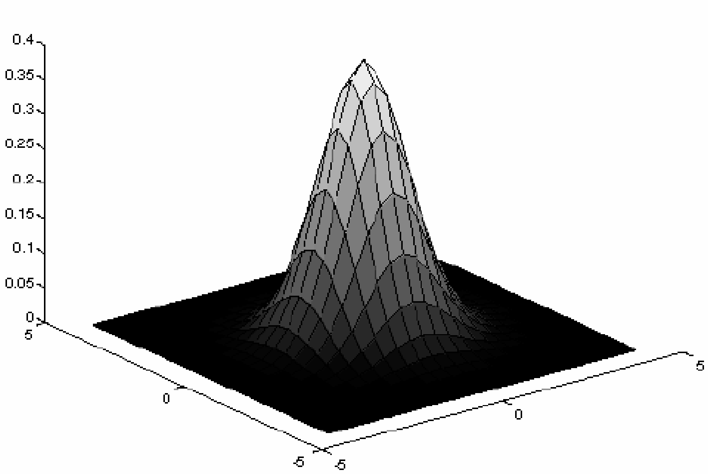
Average Filter

- Average Filter
  - replaces each pixel with an average of its neighborhood
  - Mask with positive entries that sum to 1
- if all weights are equal, it is called a BOX filter

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

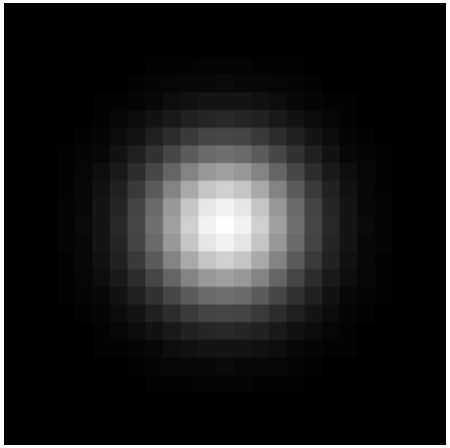
Gaussian Averaging (An Isotropic Gaussian)

- Rotationally symmetric
- Weights nearby pixels more than distant ones
  - this makes sense as 'probabilistic' inference



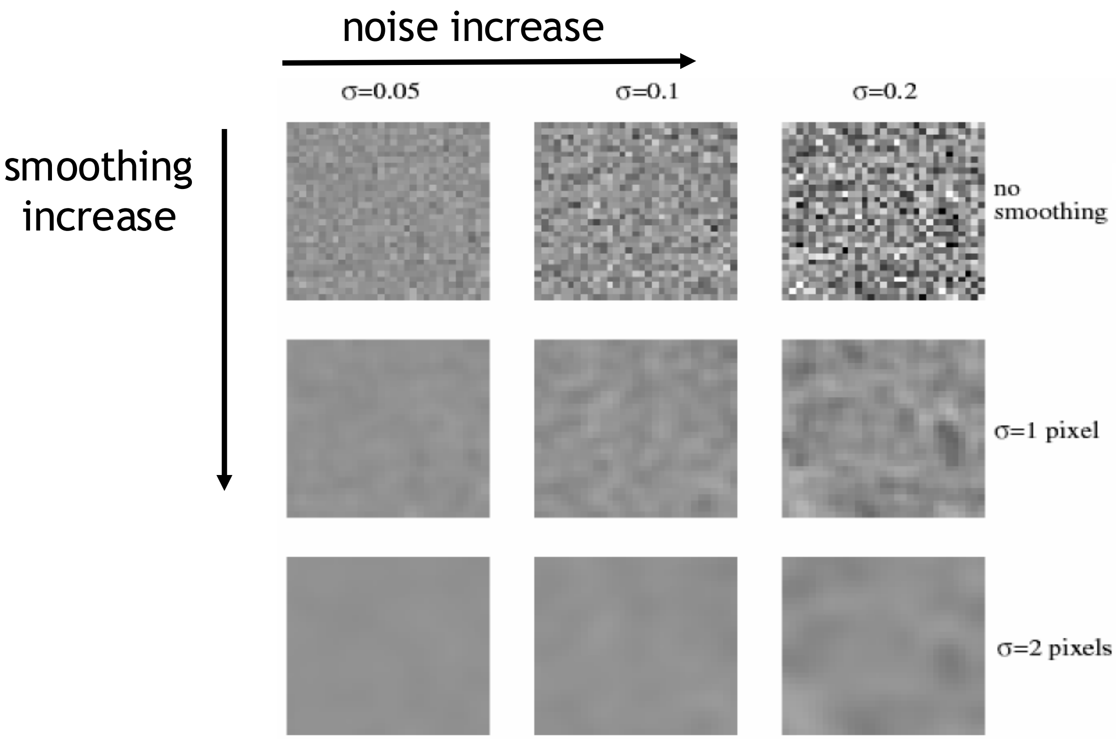
- the pictures show a smoothing kernel proportional to

$$g(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$



Smoothing with a Gaussian

- Effects of smoothing:
  - each column shows realizations of an image of Gaussian noise
  - each row shows smoothing with Gaussians of different width



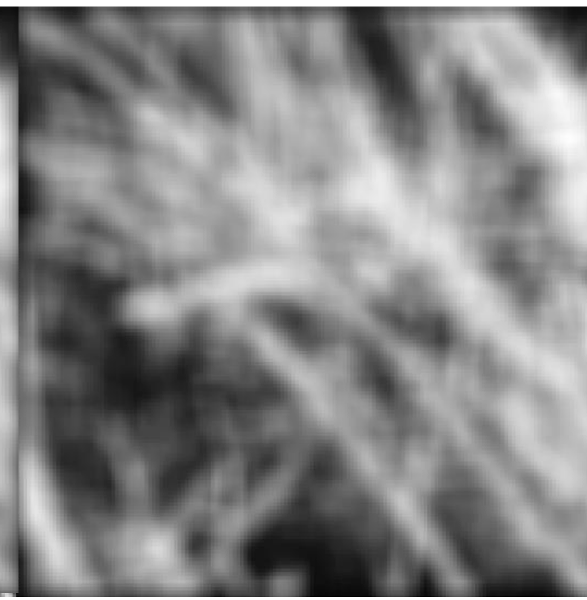
Smoothing with a Gaussian

- Example:

Original Image

Gaussian-filtered

Box-filtered



Efficient Implementation

- Both, the BOX filter and the Gaussian filter are separable:
  - first convolve each row with a 1D filter
  - then convolve each column with a 1D filter

$$(f_x \otimes f_y) \otimes I = f_x \otimes (f_y \otimes I)$$

  - remember:
    - convolution is linear - associative and commutative
- Example: separable BOX filter

$$\underbrace{\begin{array}{|c|c|c|}\hline \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \hline \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \hline \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \hline \end{array}}_{f_x \otimes f_y} = \underbrace{\begin{array}{|c|c|c|}\hline \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \end{array}}_{f_x} \otimes \underbrace{\begin{array}{|c|}\hline \frac{1}{3} \\ \hline \frac{1}{3} \\ \hline \frac{1}{3} \\ \hline \end{array}}_{f_y}$$

Example: Separable Gaussian

- Gaussian in x-direction

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

- Gaussian in y-direction

$$g(y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{y^2}{2\sigma^2}\right)$$

- Gaussian in both directions

$$g(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

Separable Gaussian

- Gaussian separability:
  - an n dimensional Gaussian convolution is equivalent to n 1-D Gaussian convolutions

$$h(i,j) = f(i,j) * g(i,j) =$$

$$= \sum_{k=1}^{m} \sum_{l=1}^{n} g(k,l) f(i-k, j-l) =$$

$$= \sum_{k=1}^{m} \sum_{l=1}^{n} e^{-\frac{(k^2+l^2)}{2\sigma^2}} f(i-k, j-l) =$$

$$= \sum_{k=1}^{m} e^{-\frac{k^2}{2\sigma^2}} \left[\underbrace{\sum_{l=1}^{n} e^{-\frac{l^2}{2\sigma^2}} f(i-k, j-l)}_{h'} \right] =$$

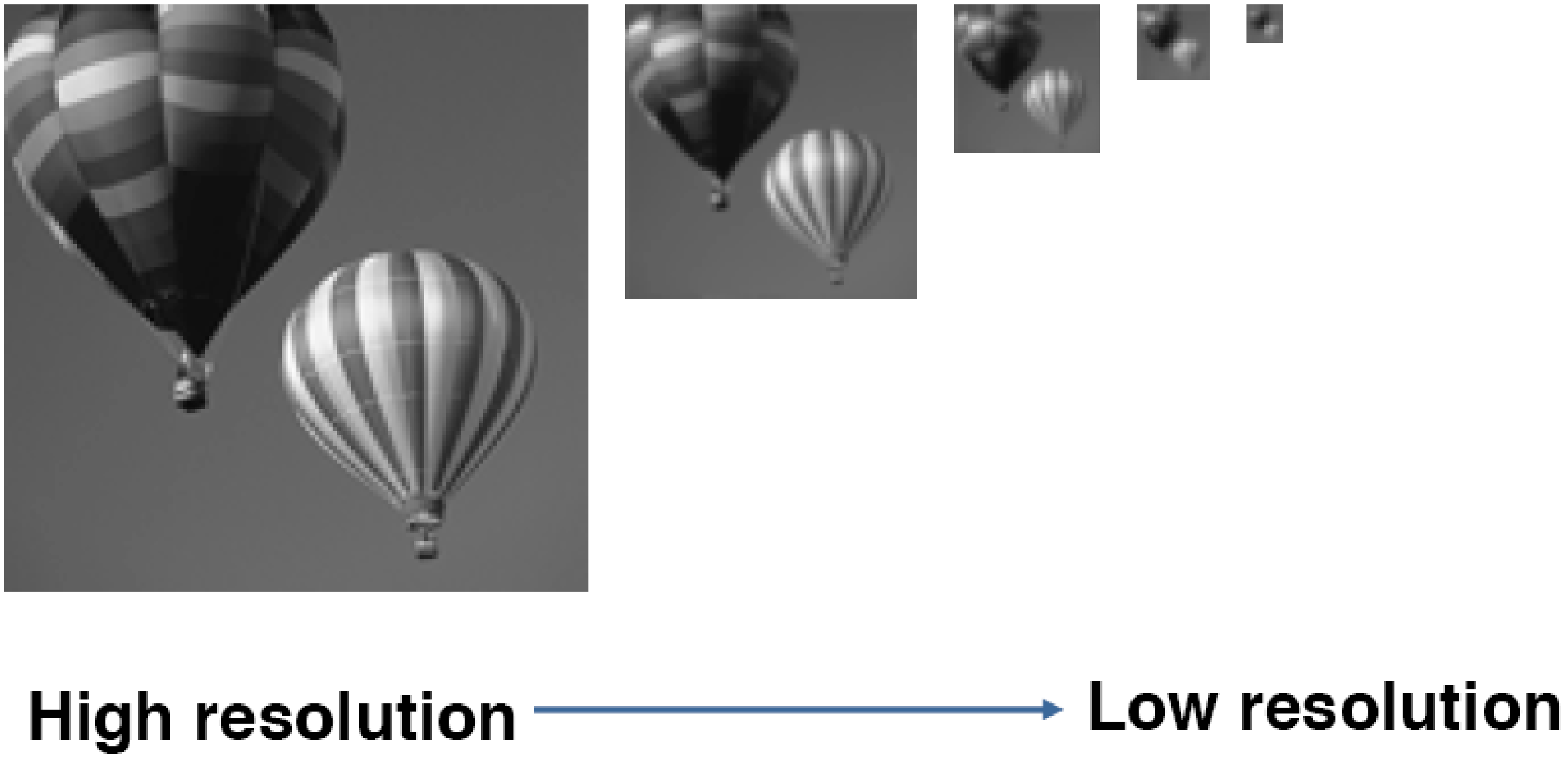
$$= \sum_{k=1}^{m} e^{-\frac{k^2}{2\sigma^2}} h'(i-k, j)$$

1-D Gaussian horizontally

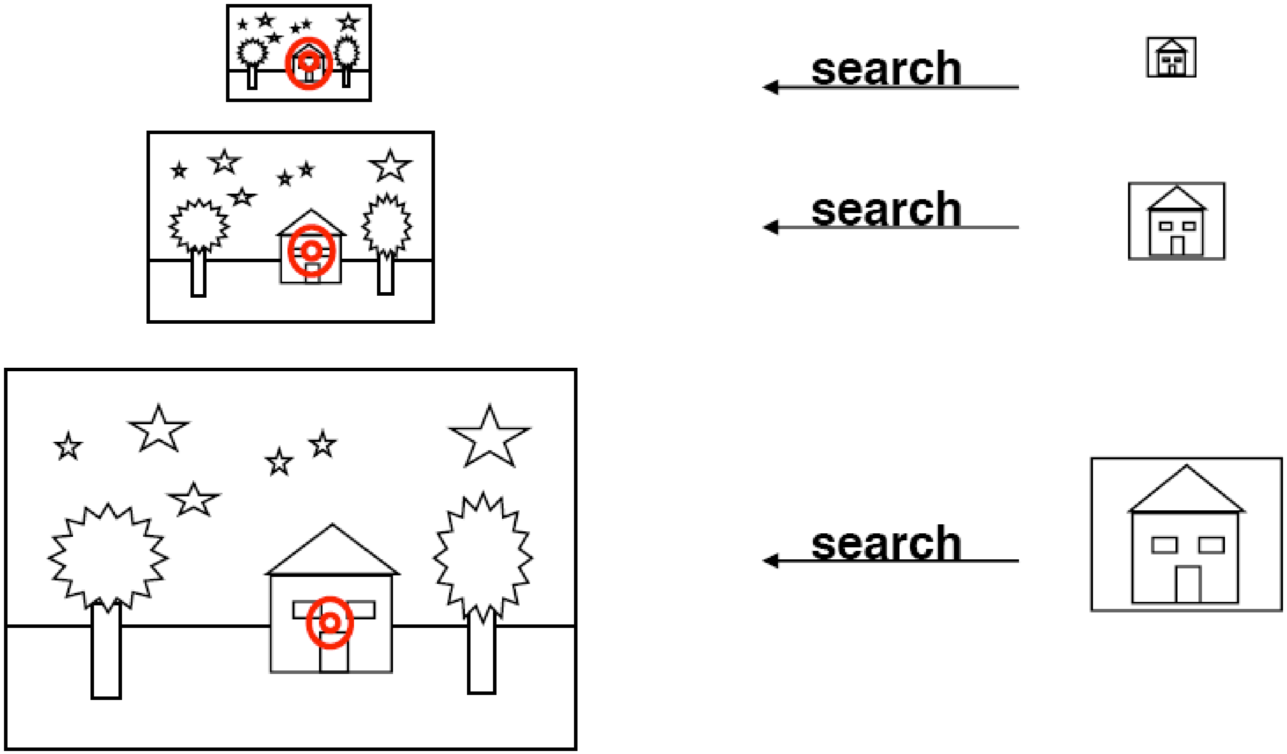
1-D Gaussian vertically

Multi-Scale Image Representation

- Gaussian Pyramids
- Example of a Gaussian Pyramid

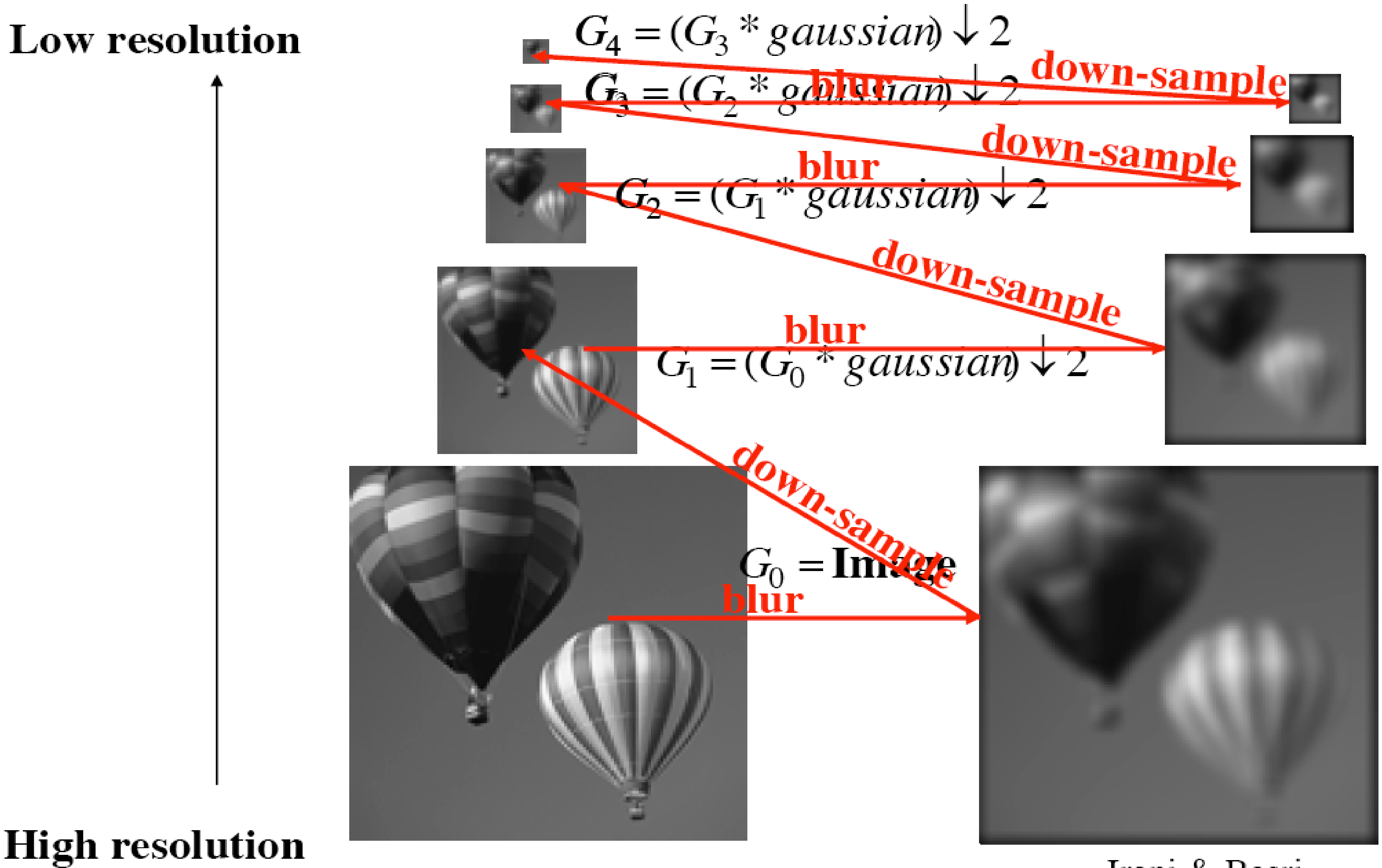


Motivation: Search across Scales



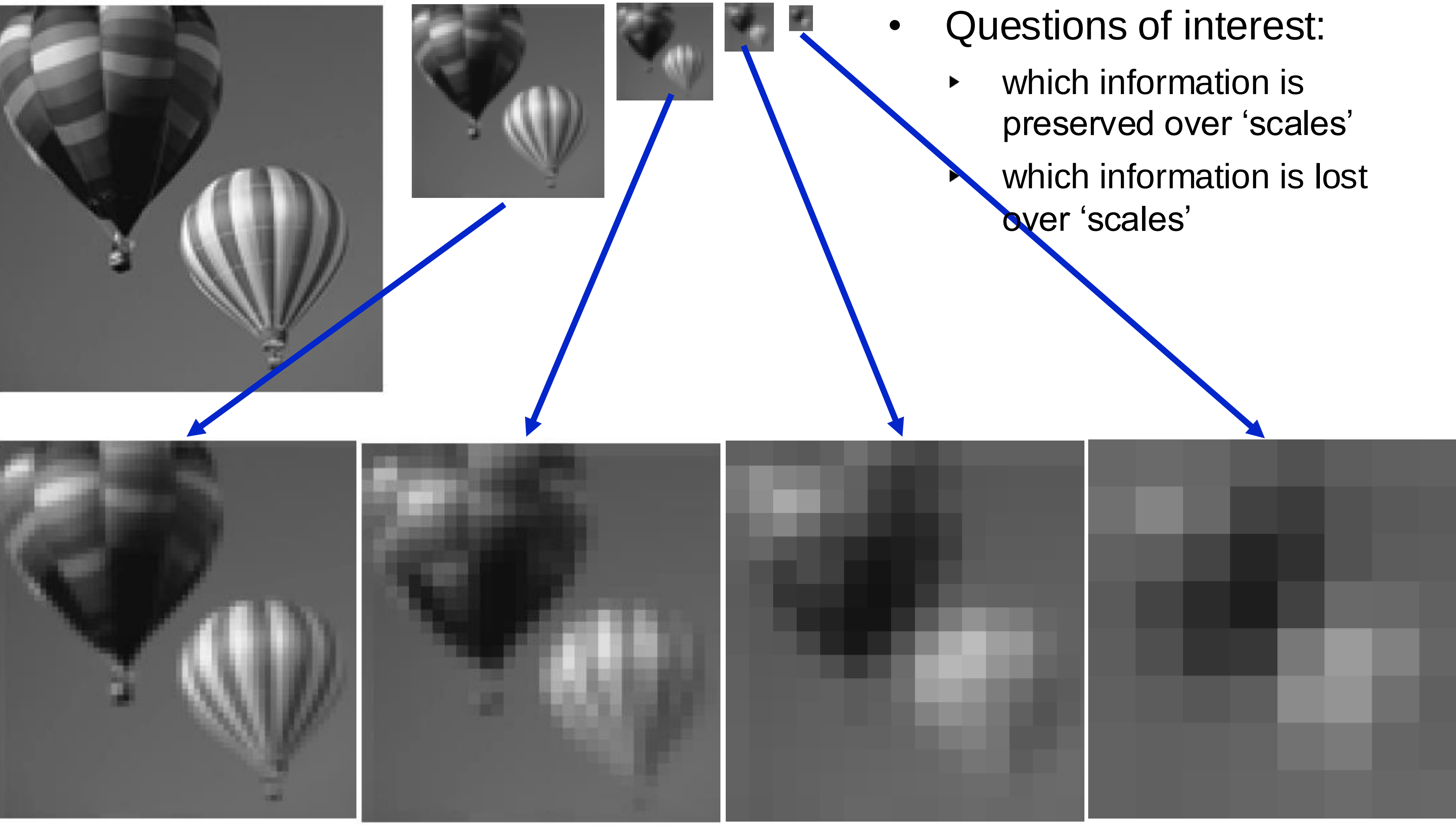
Irani & Basri

Computation of Gaussian Pyramid



Irani & Basri

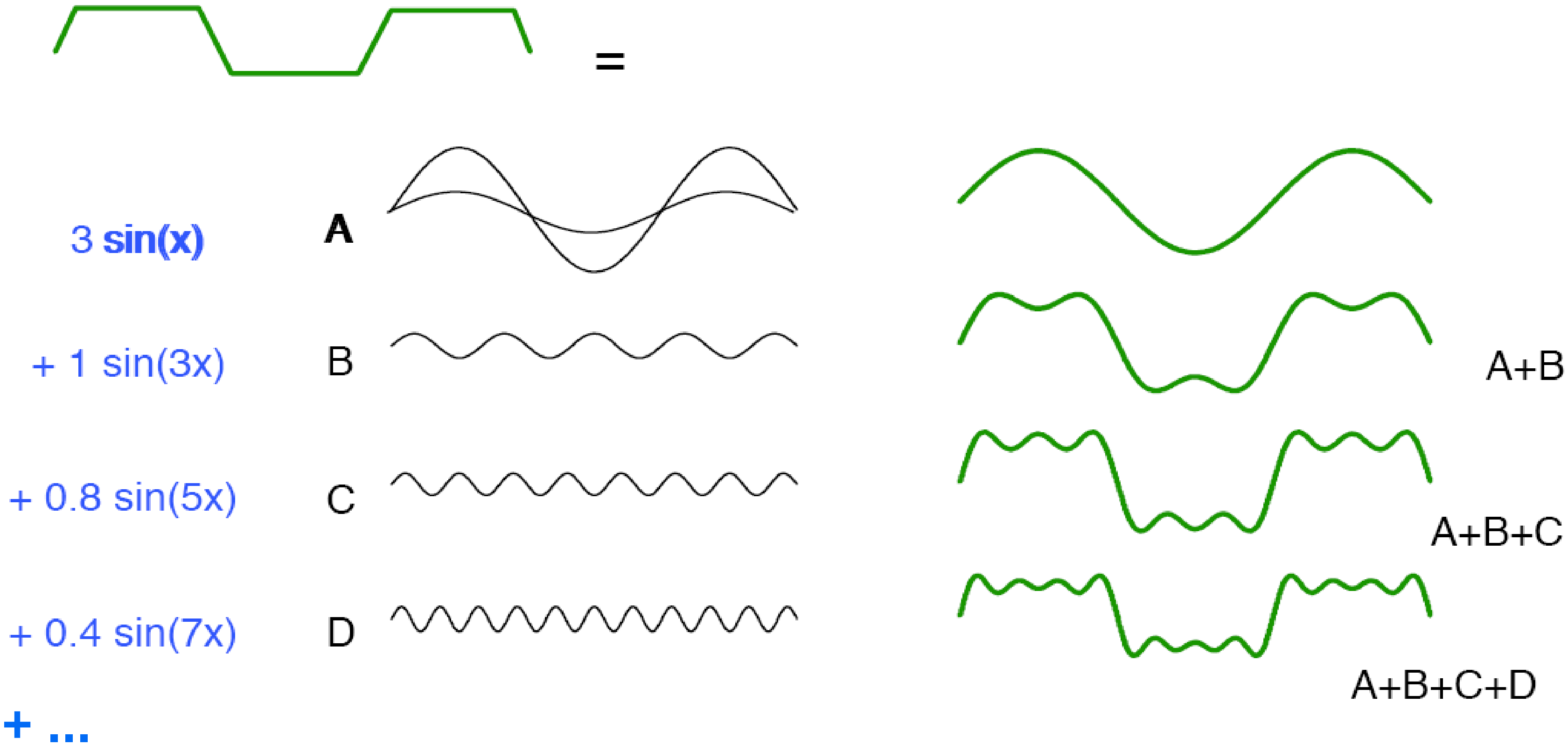
Gaussian Pyramid



- Questions of interest:
  - which information is preserved over ‘scales’
  - which information is lost over ‘scales’

Fourier Transform in Pictures

- a *very* little introduction on Fourier transforms to talk about spatial frequencies…



Subsampling without Average Filtering

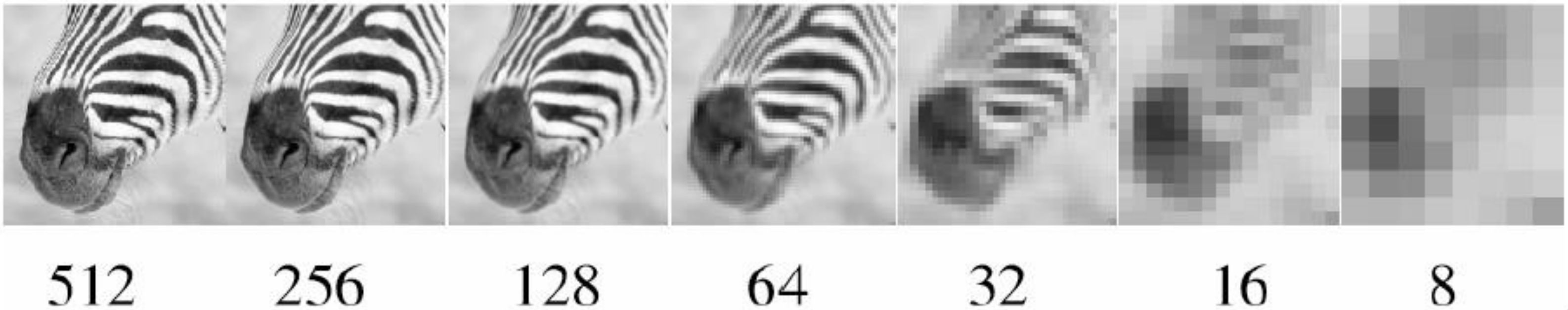
- Subsampling without average filtering leads to aliasing

Original image

Image with spatial aliasing

image source: https://en.wikipedia.org/wiki/Aliasing

Another Example



- a bar
  - in the big images is a hair (on the zebra's nose)
  - in smaller images, a stripe
  - in the smallest image, the animal's nose

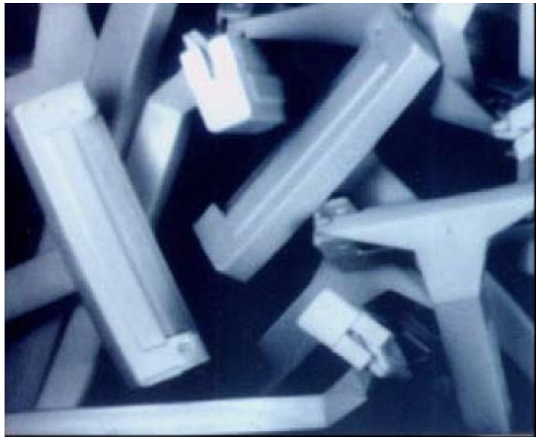
Basics of Digital Image Filtering

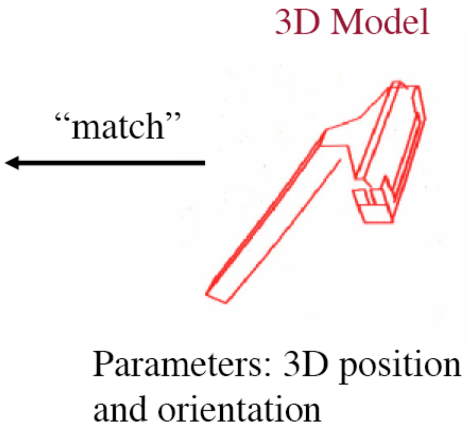
- Linear Filtering
  - Gaussian Filtering
- Multi Scale Image Representation
  - Gaussian Pyramid
- Edge Detection
  - ‘Recognition using Line Drawings’
  - Image derivatives (1st and 2nd order)
- Object Instance Identification using Color Histograms
- Performance evaluation

Line Drawings:
Good Starting Point for Recognition?

Example of Recognition & Localization

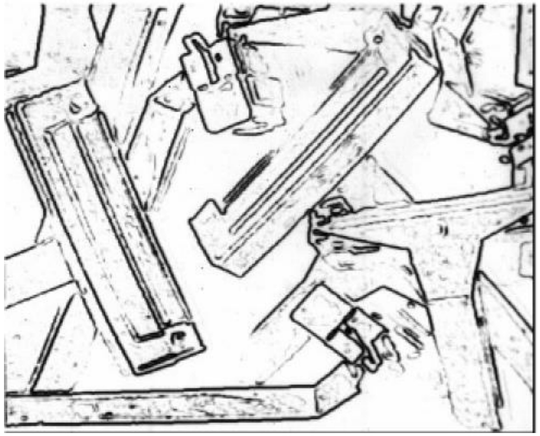
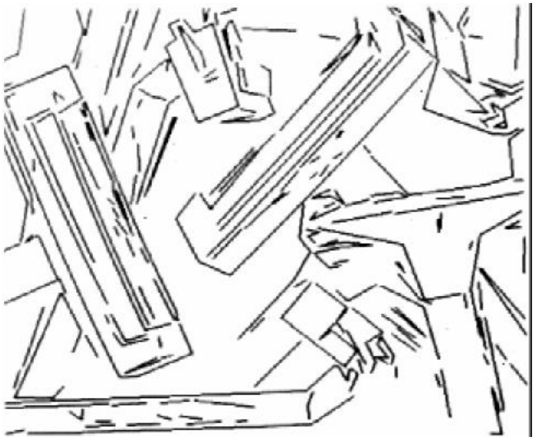
- David Lowe





Example of Recognition & Localization

- David Lowe
  - 1. ‘filter’ image to **find brightness changes**
  - 2. **‘fit’ lines** to the raw measurements

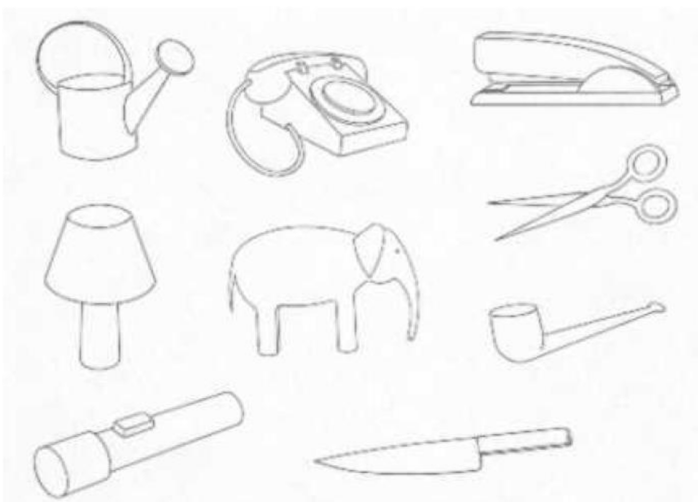




Example of Recognition & Localization

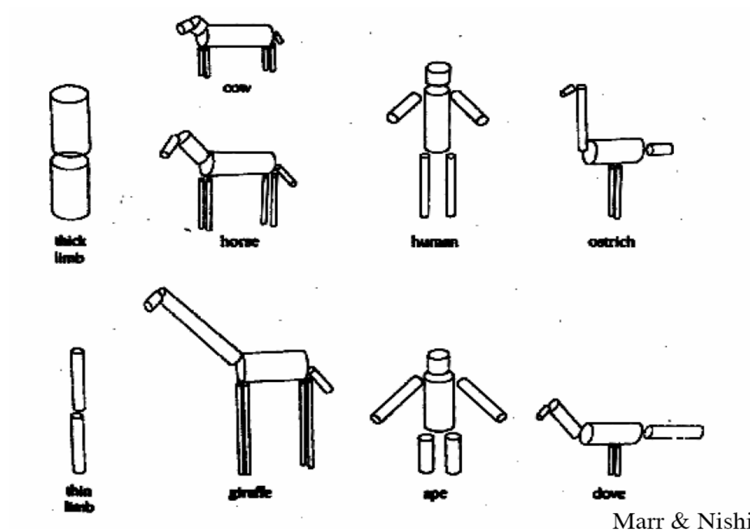
- David Lowe
  - 3. ‘project’ model into the image and **‘match’ to lines** (solving for 3D pose)

Class of Models

- Common Idea & Approach (in the 1980's)
  - matching of models (wire-frame/geons/generalized cylinders...) to edges and lines



Biederman's Geons



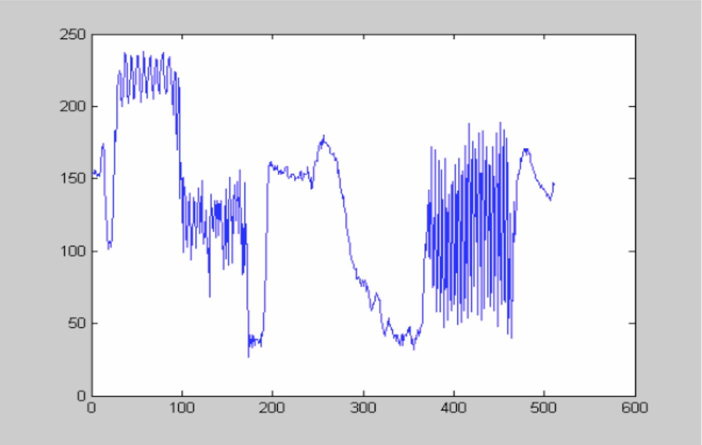
Marr & Nishihara

- so the 'only' remaining problem to solve is:
  - **reliably extract lines & edges** that can be matched to these models...

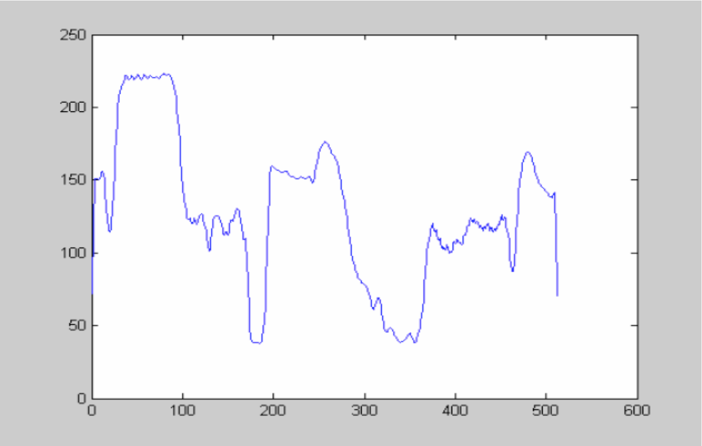
Actual 1D profile

- Barbara Image:
  - entire image

- line 250

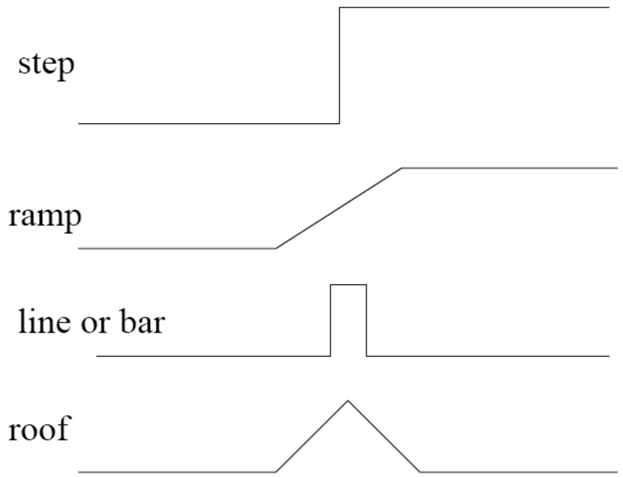


- line 250 smoothed with a Gaussian

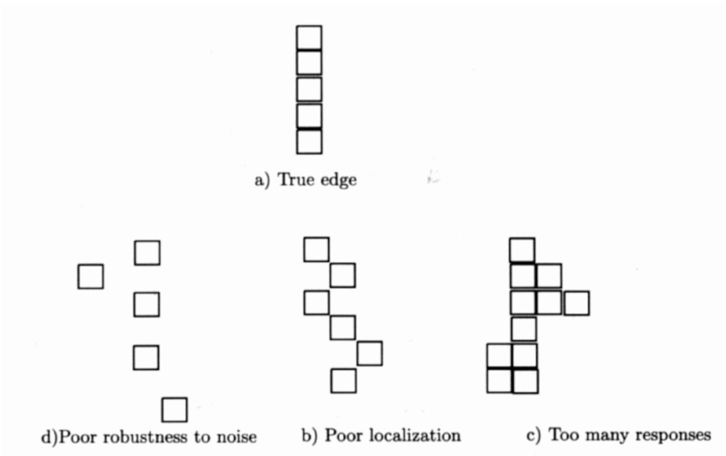


What are ‘edges’ (1D)

- Idealized Edge Types:



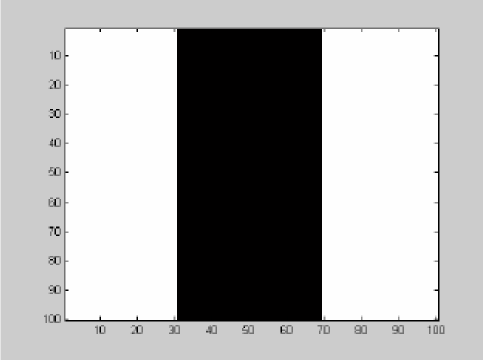
- Goals of Edge Detection:
  - **good detection**: filter responds to edge, not to noise
  - **good localization**: detected edge near true edge
  - **single response**: one per edge

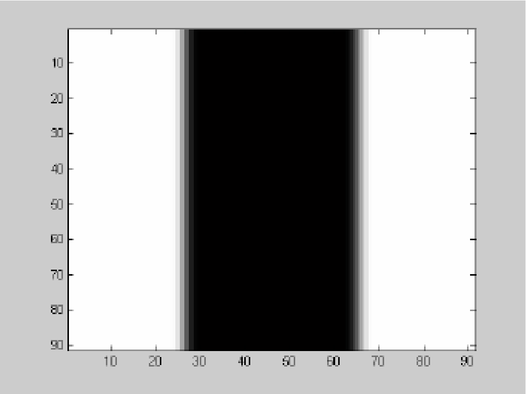


Edges

- Edges:
  - correspond to fast changes
  - where the magnitude of the derivative is large

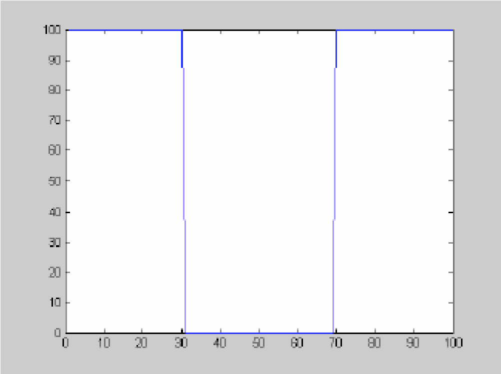
“image” of 2 step-edges

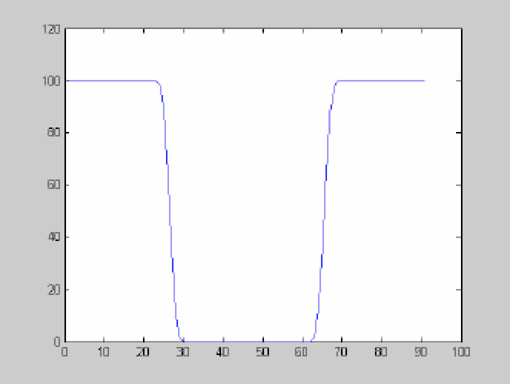




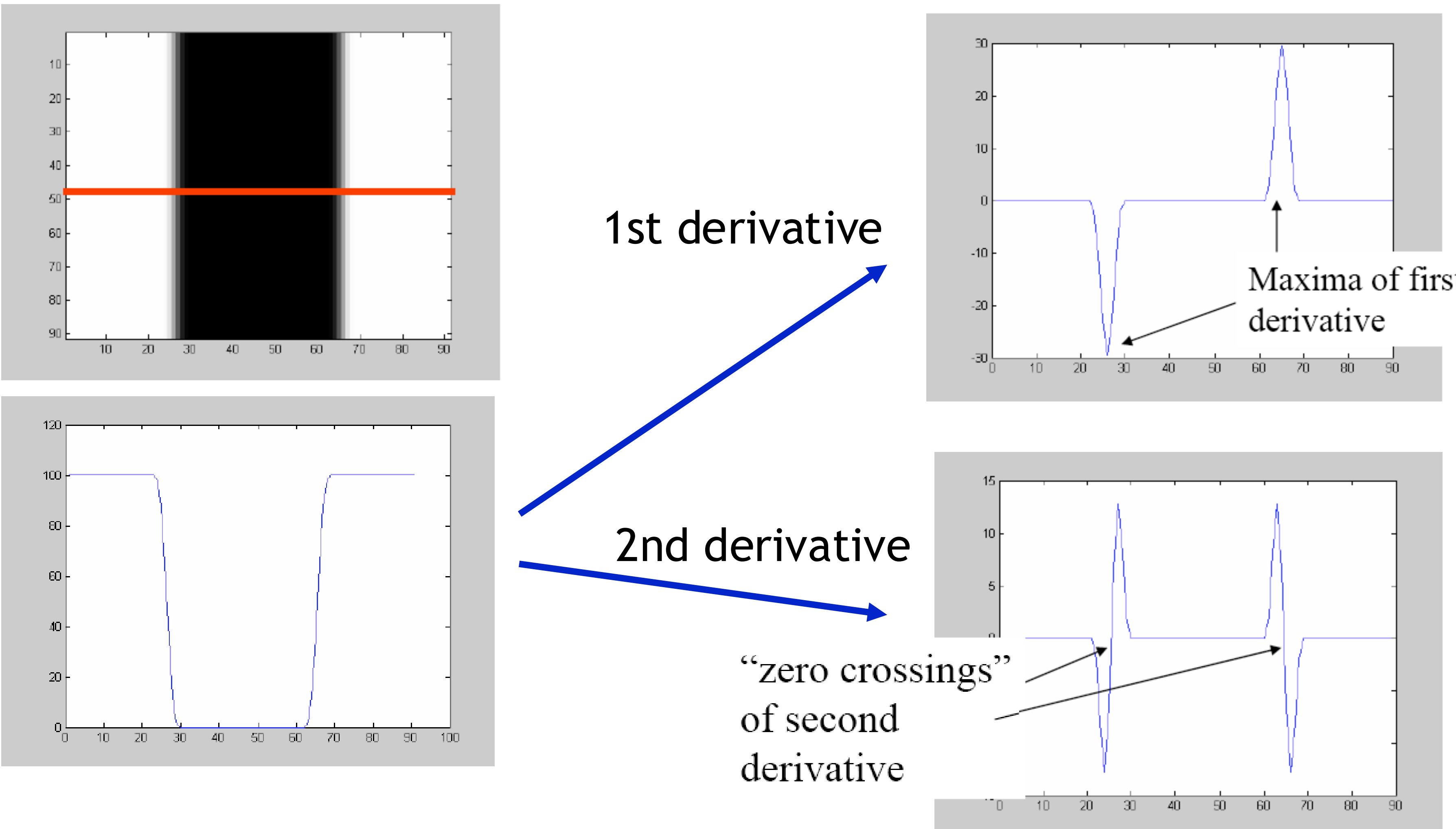
smoothing

single line of “image”





Edges & Derivatives…



Compute Derivatives

$$\frac{d}{dx}f(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h} \approx f(x+1)-f(x)$$

- we can implement this as a linear filter:
  - direct:

-1	1

  - or symmetric:

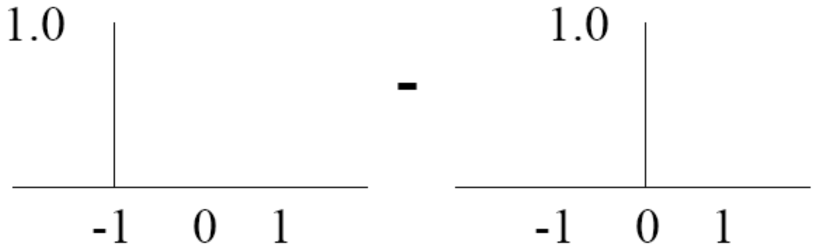
-1	0	1

Compute Derivatives

$$\frac{d}{dx}f(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h} \approx f(x+1)-f(x)$$

- we can implement this as a linear filter:
  - direct:

-1	1



  - or symmetric:

-1	0	1

Edge-Detection

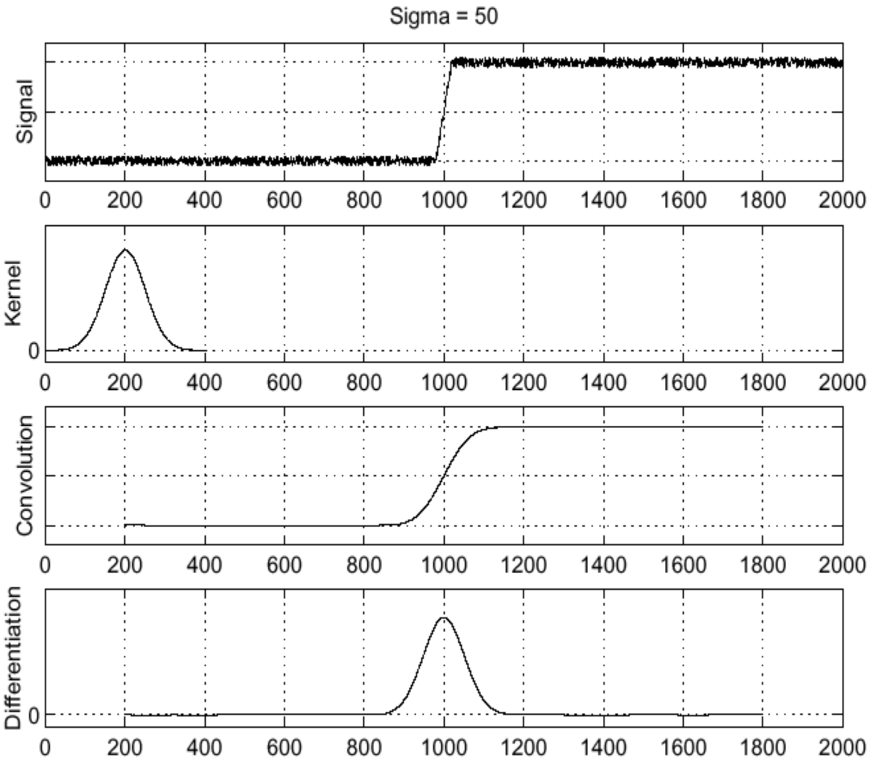
- based on 1st derivative:
  - smooth with Gaussian
  - calculate derivative
  - finds its maxima

f

g

$g \otimes f$

$$\frac{d}{dx}(g \otimes f)$$



Edge-Detection

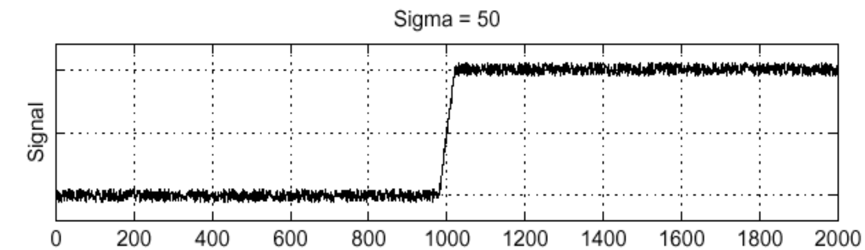
- Simplification:

$$\frac{d}{dx}(g \otimes f) = \left(\frac{d}{dx}g\right) \otimes f$$

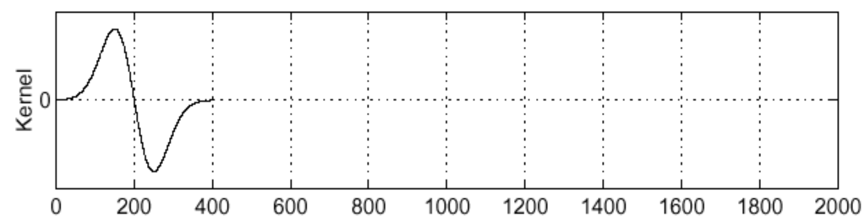
  - remember:
    derivative as well as convolution are linear operations

  - saves one operation

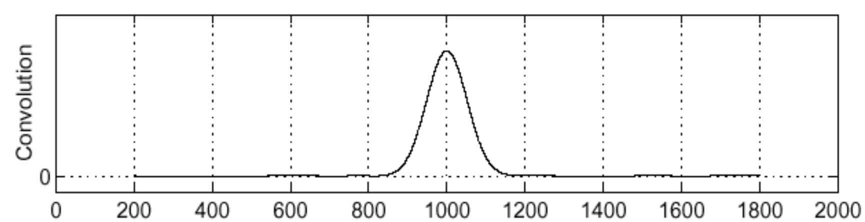
$$f$$



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



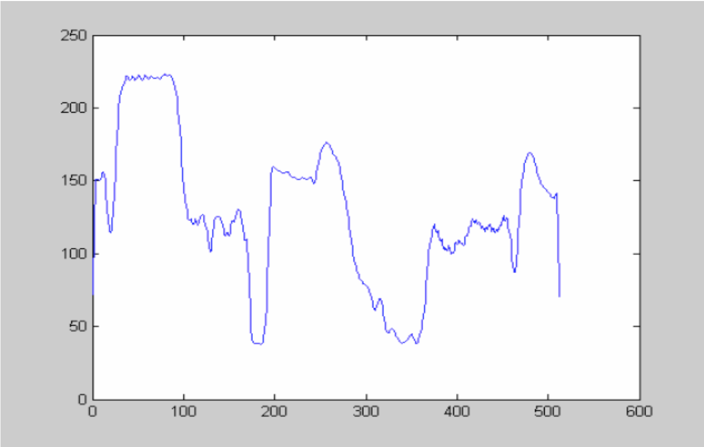
$$\left(\frac{d}{dx}g\right) \otimes f$$



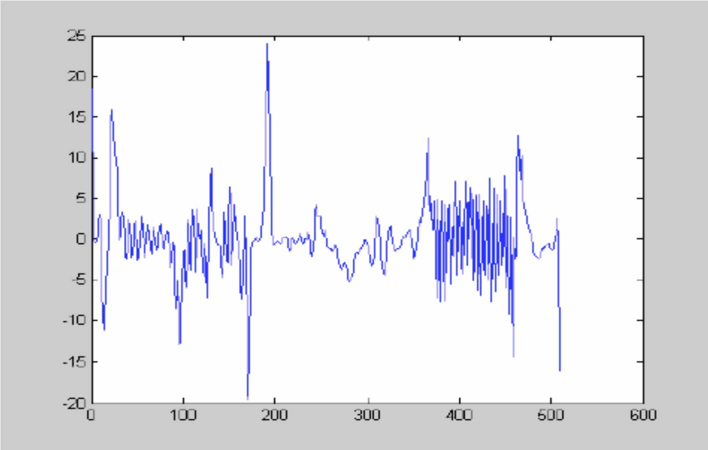
1D Barbara signal

- Barbara Image:
  - entire image

- line 250 (smoothed)



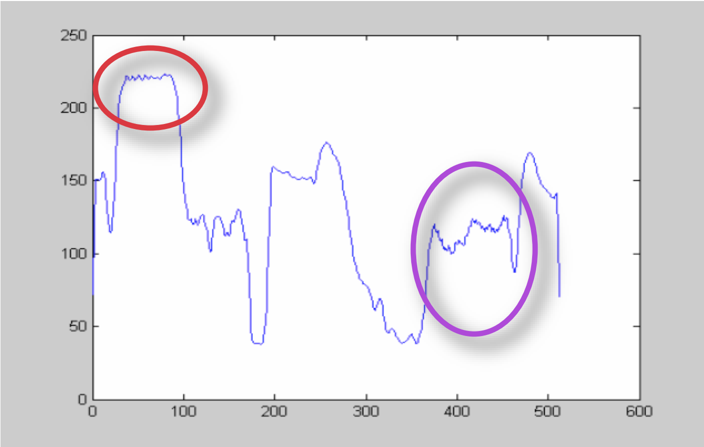
- 1st derivative



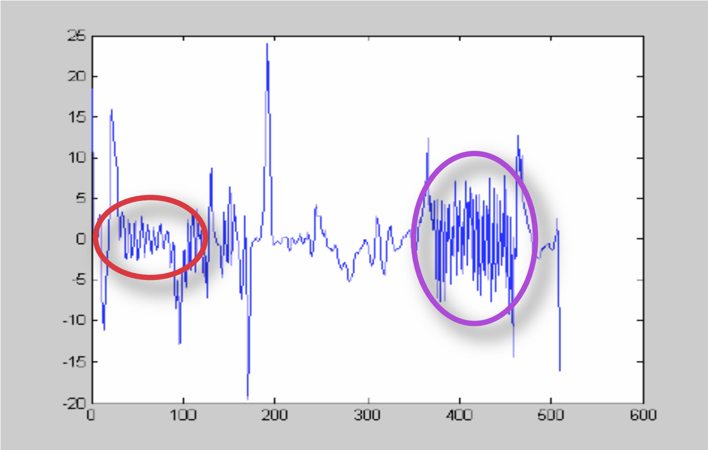
1D Barbara signal:
note the amplification of small variations

- Barbara Image:
  - entire image

- line 250 (smoothed)

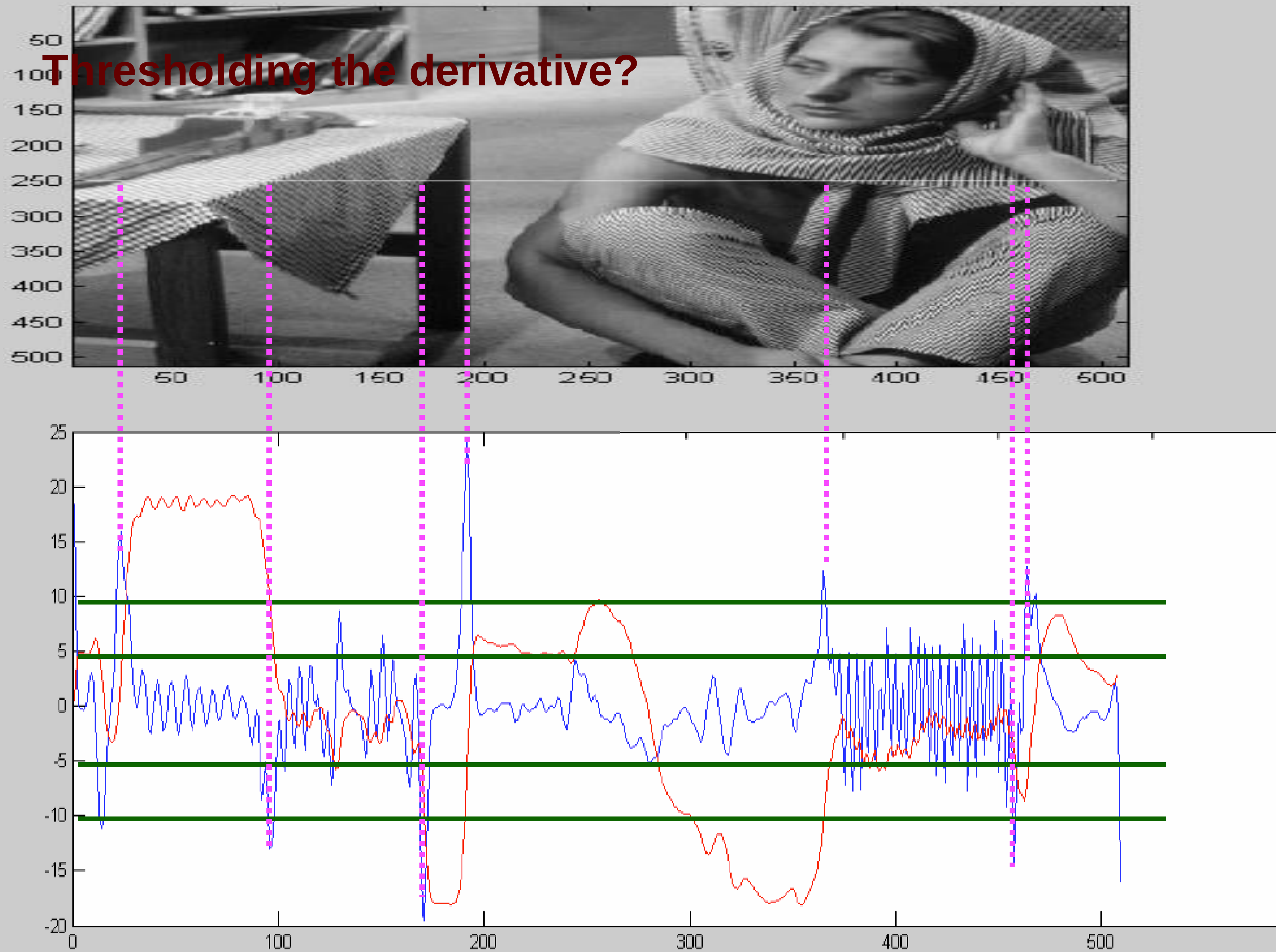


- 1st derivative



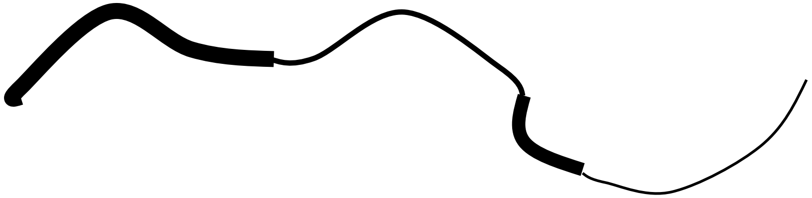
SAPIENZA
UNIVERSITÀ DI ROMA

Thresholding the derivative?



Implementing 1D edge detection

- algorithmically:
  - find peak in the 1st derivative
  - but
    - should be a local maxima
    - should be 'sufficiently' large
  - hysteresis: use 2 thresholds
    - high threshold to start edge curve (maximum value of gradient should be sufficiently large)
    - low threshold to continue them (in order to bridge "gaps" with lower magnitude)
    - (really only makes sense in 2D...)



Extension to 2D Edge Detection: Partial Derivatives

- partial derivatives
  - in x direction:

$$\frac{d}{dx} I(x, y) = I_x \approx I \otimes D_x$$

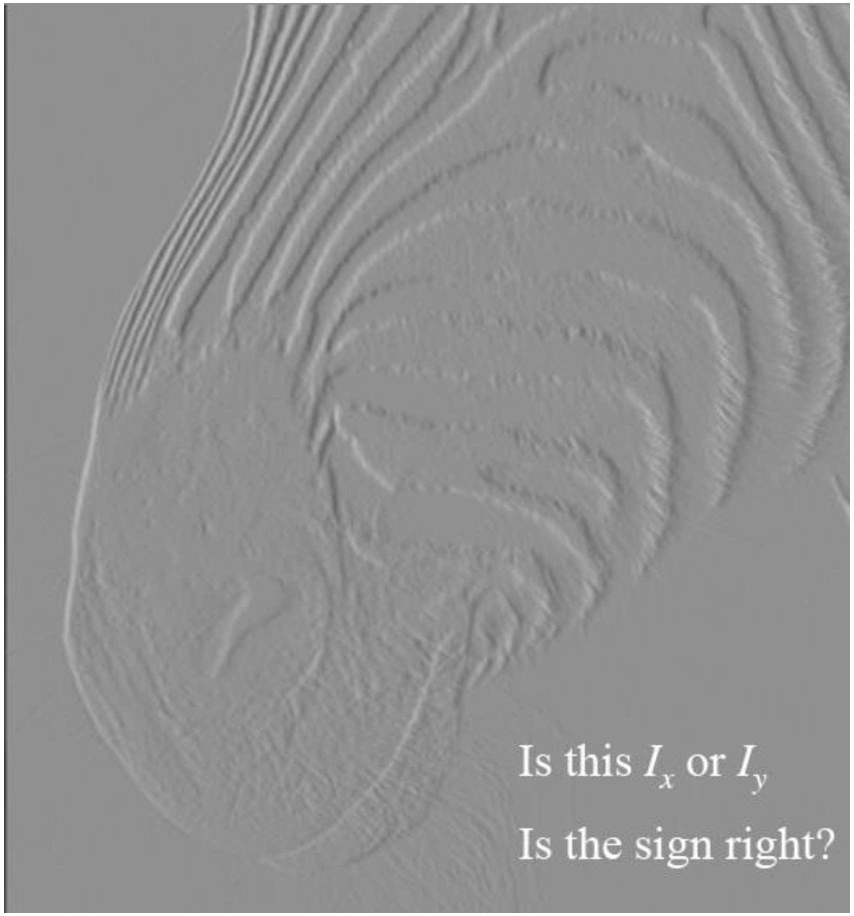
  - in y direction:

$$\frac{d}{dy} I(x, y) = I_y \approx I \otimes D_y$$

  - often approximated with simple filters (finite differences):

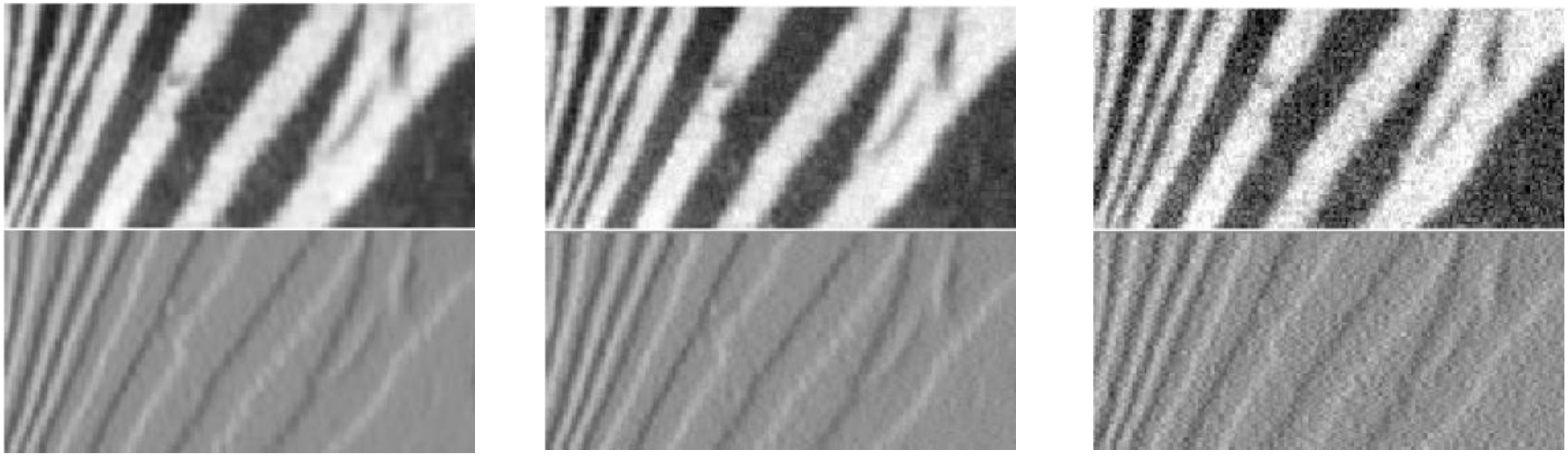
$$D_x = \frac{1}{3} \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad D_y = \frac{1}{3} \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Finite Differences



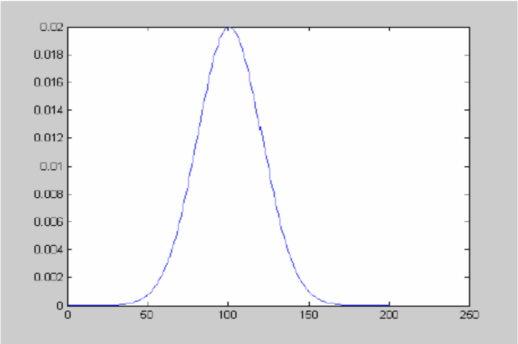
Finite Differences responding to noise

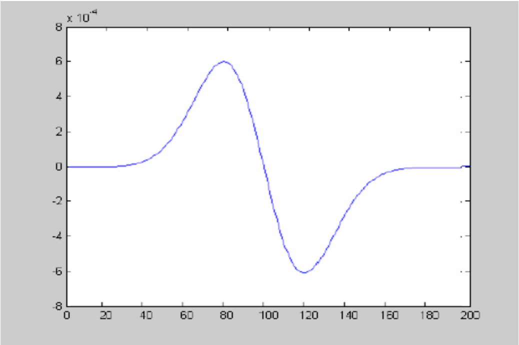
- increasing noise level (from left to right)
  - noise: zero mean additive Gaussian noise



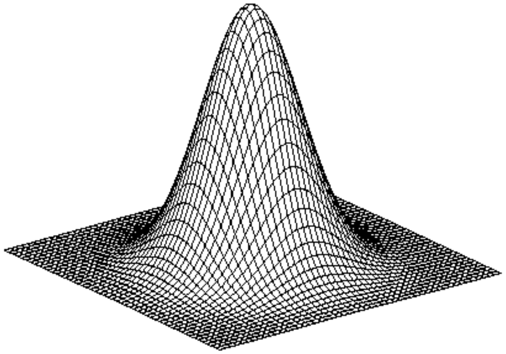
Again: Derivatives and Smoothing

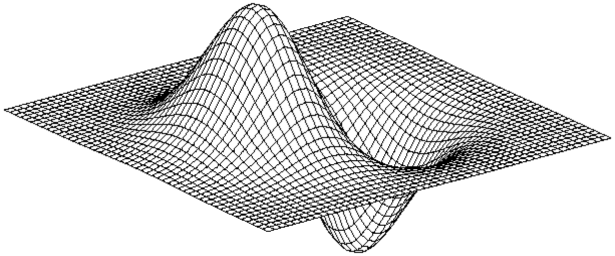
- derivative in x-direction: $$D_x \otimes (G \otimes I) = (D_x \otimes G) \otimes I$$
  - in 1D:





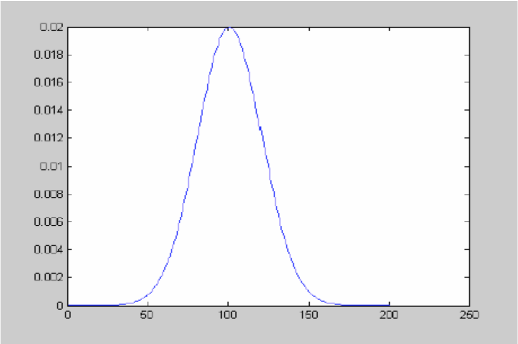
  - in 2D:

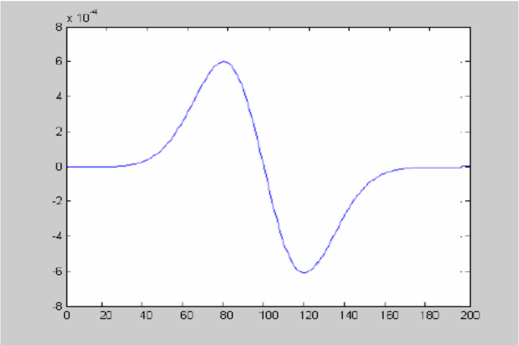




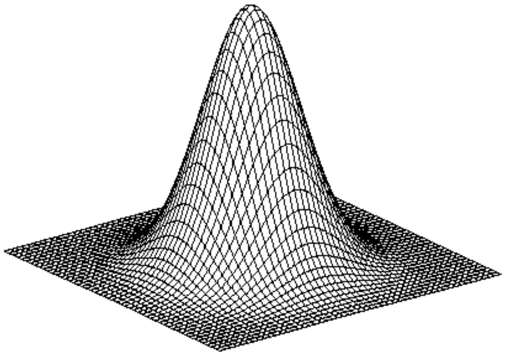
Again: Derivatives and Smoothing

- derivative in x-direction: $D_x \otimes (G \otimes I) = (D_x \otimes G) \otimes I$
  - in 1D:

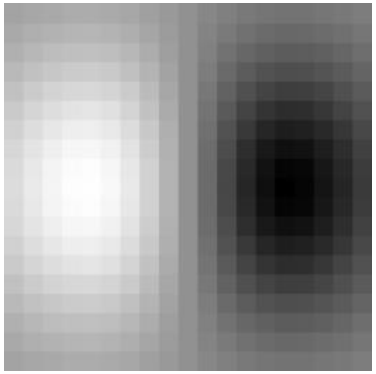




  - in 2D:



Dx



Dy

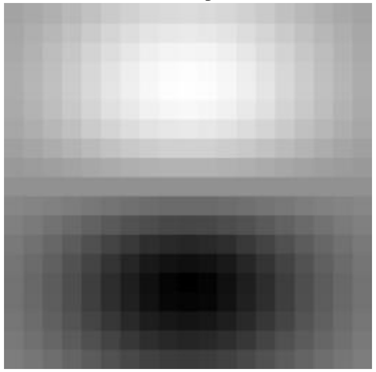
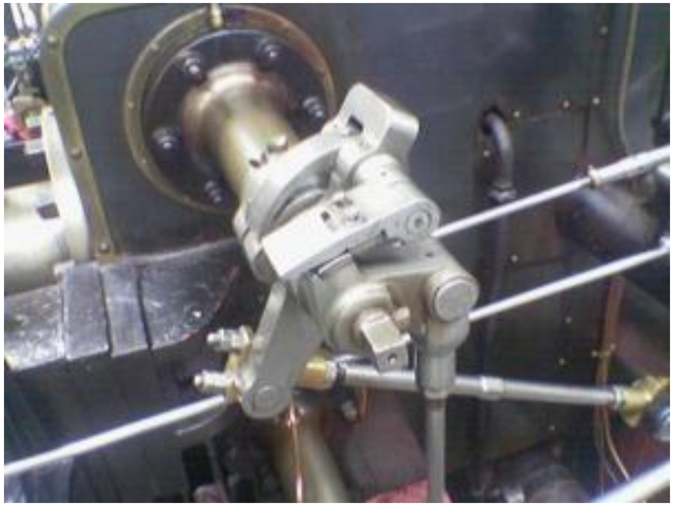
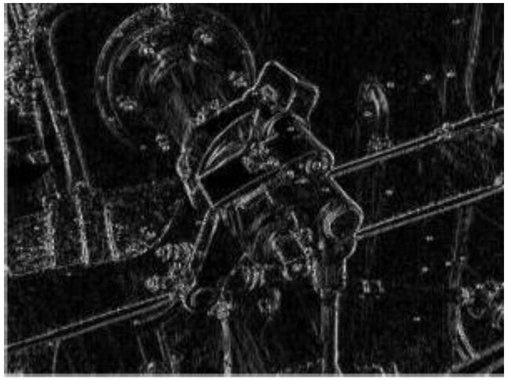


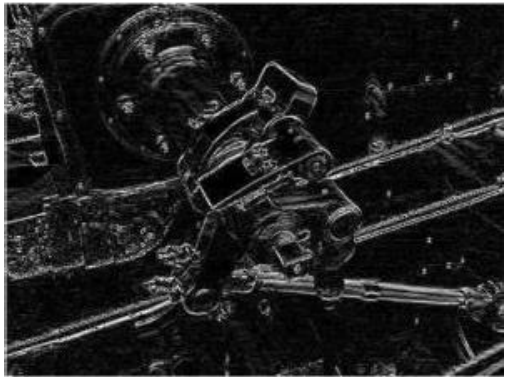
Image Filtering

- Edge detection using derivative of Gaussian filter:





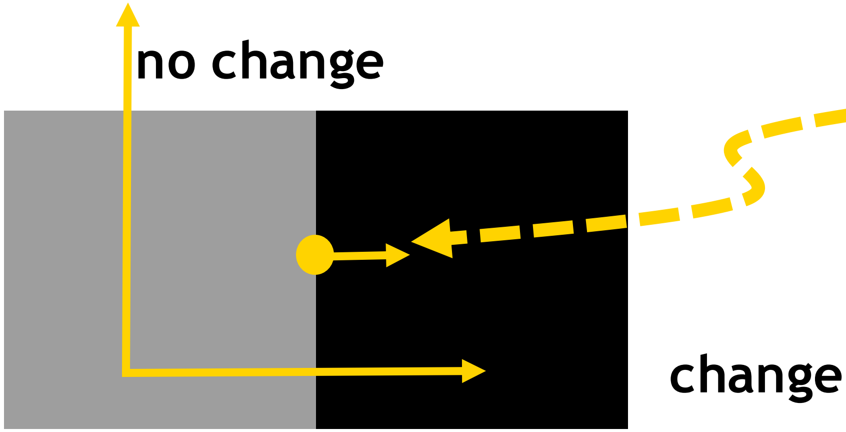
Edges along the x axis



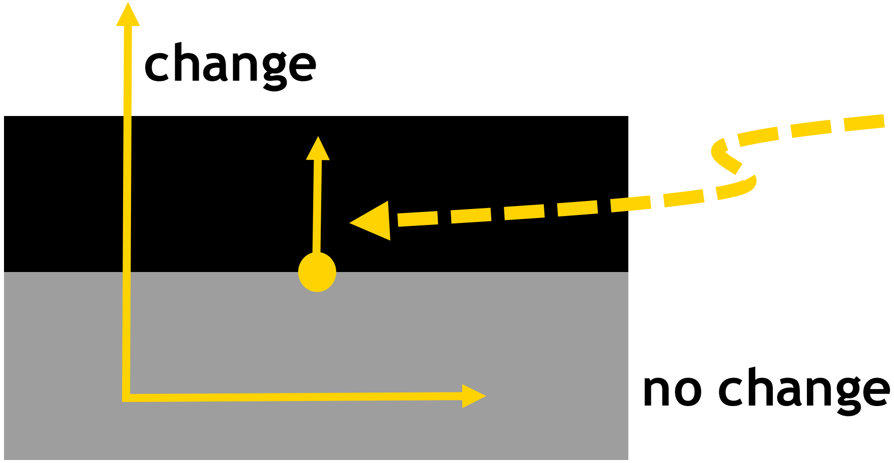
Edges along the y axis

SAPIENZA
UNIVERSITÀ DI ROMA

What is the gradient ?

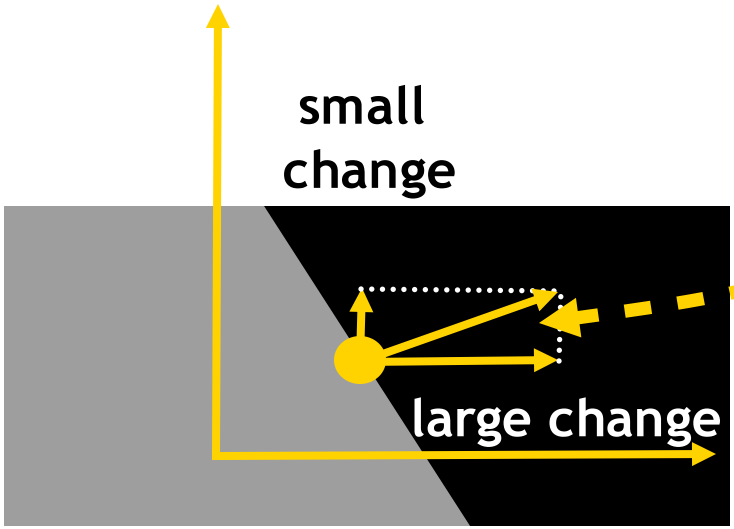


$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) = (k, 0)$$



$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) = (0, k)$$

What is the gradient ?



$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) = \left(k_x, k_y\right)$$

- gradient direction is perpendicular to edge
- gradient magnitude measures edge strength

2D Edge Detection

- calculate derivative
  - use the **magnitude** of the gradient
  - the gradient is:

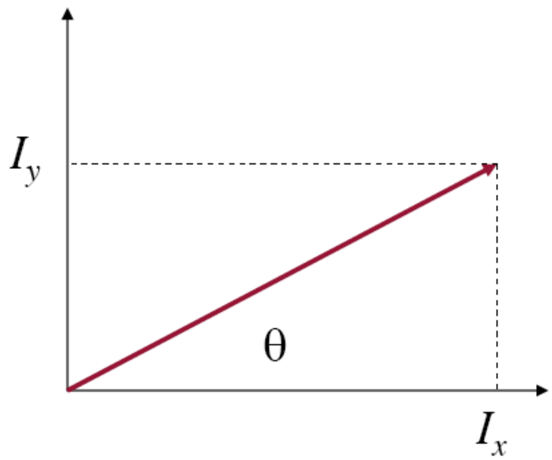
$$\nabla I = \left(I_x, I_y\right) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$$

  - the magnitude of the gradient is:

$$\|\nabla I\| = \sqrt{I_x^{\,2} + I_y^{\,2}}$$

  - the direction of the gradient is:

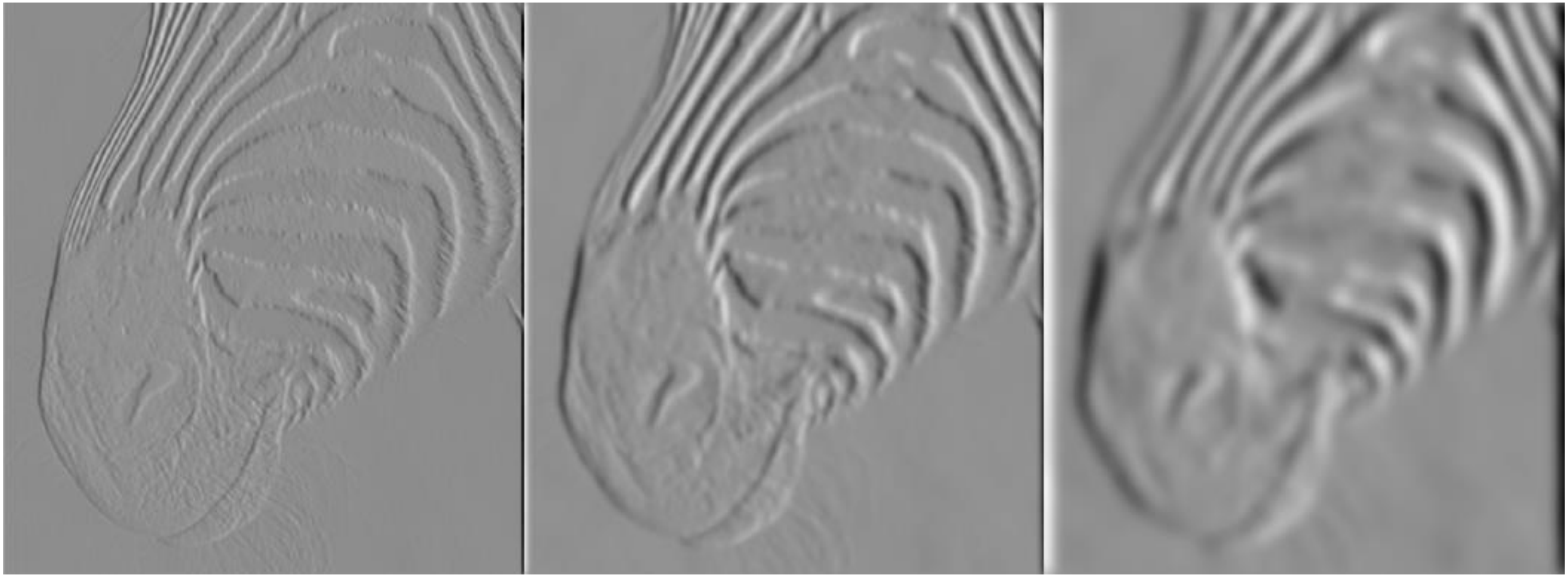
$$\theta = \arctan\left(I_y, I_x\right)$$



2D Edge Detection

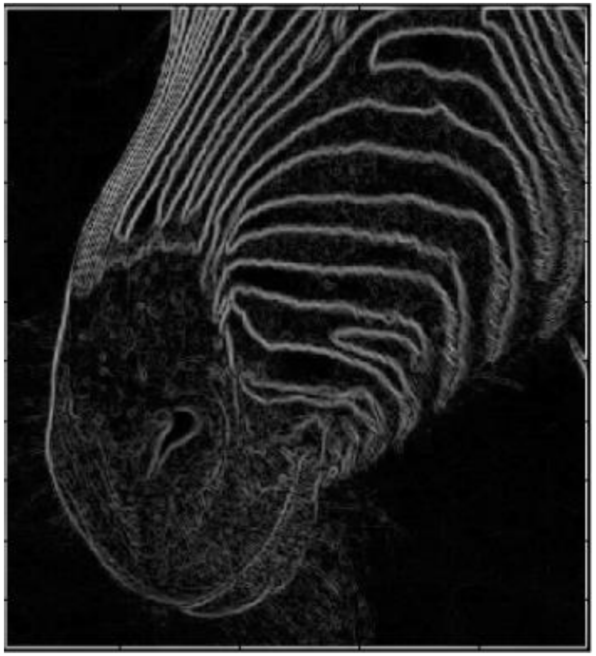
- the scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered
  - note: strong edges persist across scales

1 pixel **3 pixels** **7 pixels**



2D Edge Detection

- there are 3 major issues:
  - the gradient magnitude at different scales is different; which to choose?
  - the gradient magnitude is large along a thick trail; how to identify the significant points?
  - how to link the relevant points up into curves?



‘Optimal’ Edge Detection: Canny

- Assume:
  - linear filtering
  - additive i.i.d. Gaussian noise
- Edge Detection should have:
  - **good detection**: filter response to edge, not noise
  - **good localization**: detected edge near true edge
  - **single response**: one per edge
- then: optimal detector is approximately derivative of Gaussian

- detection/localization tradeoff:
  - more smoothing improves detection
  - and hurts localization

The Canny edge detector

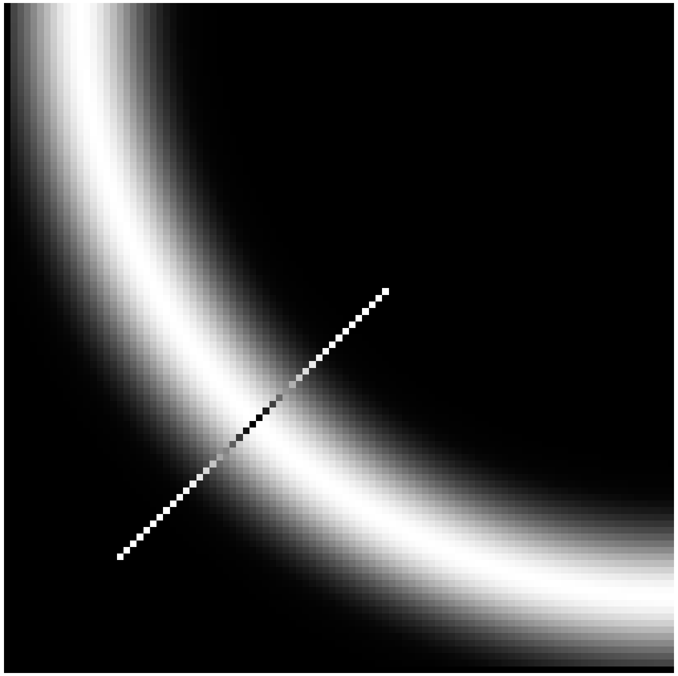
original image (Lena)

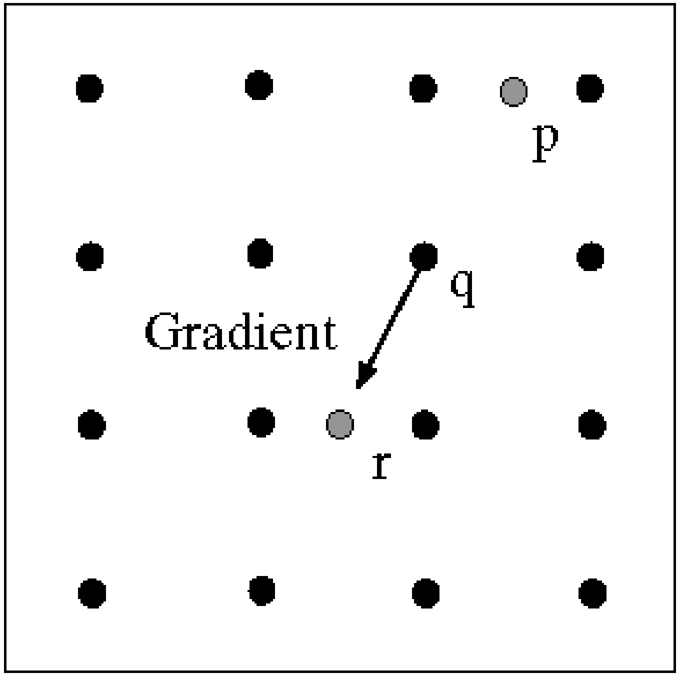
norm (=magnitude) of the gradient

thinning (non-maximum suppression)

thresholding

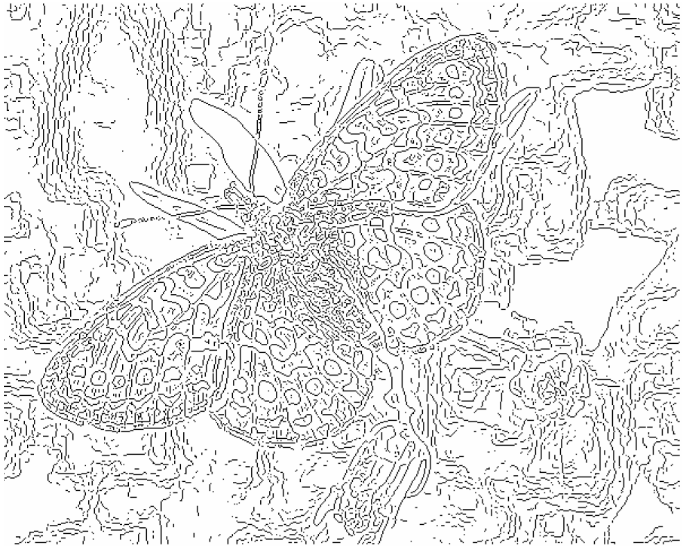
Non-maximum suppression





- Check if pixel is local maximum along gradient direction
  - choose the largest gradient magnitude along the gradient direction
  - requires checking interpolated pixels p and r

Butterfly Example (Ponce & Forsyth)

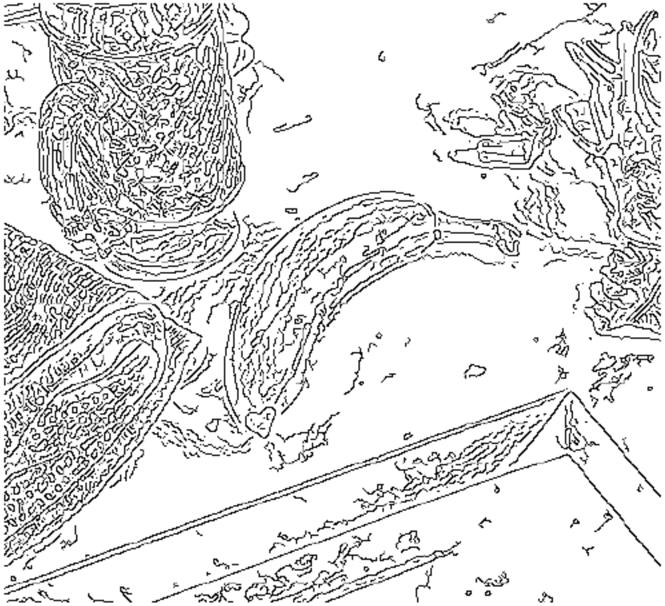


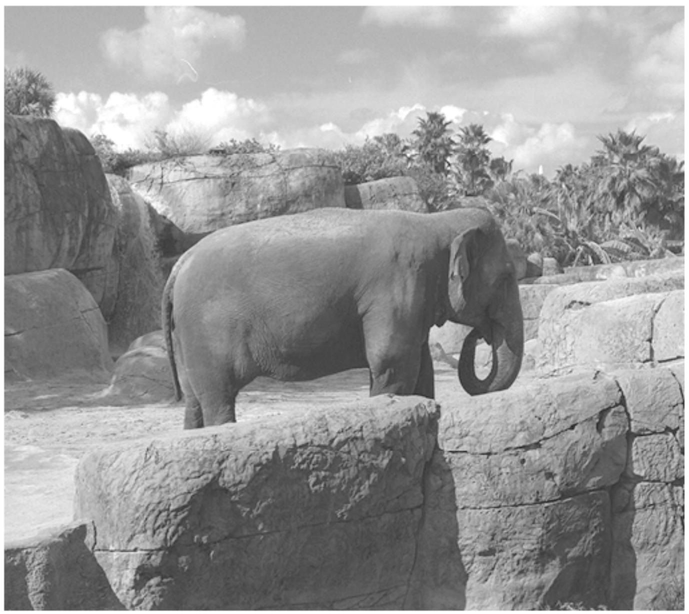
fine scale
high
threshold

coarse
scale
low
threshold

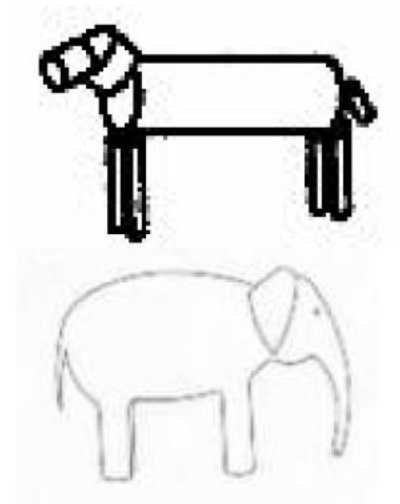
coarse
scale,
high
threshold

line drawing vs. edge detection





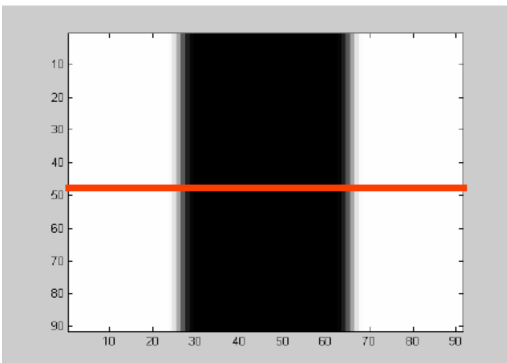
University of South Florida

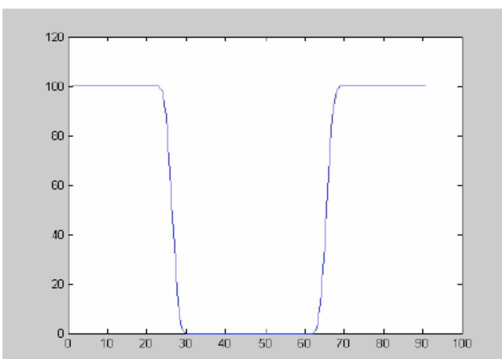


Match “model” to measurements?

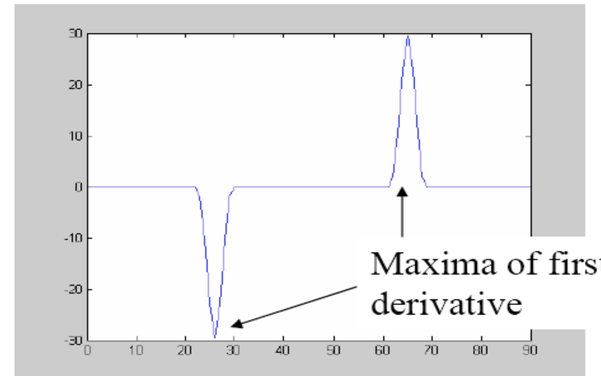
Edges & Derivatives…

- recall:
  - the zero-crossings of the second derivative tell us the location of edges

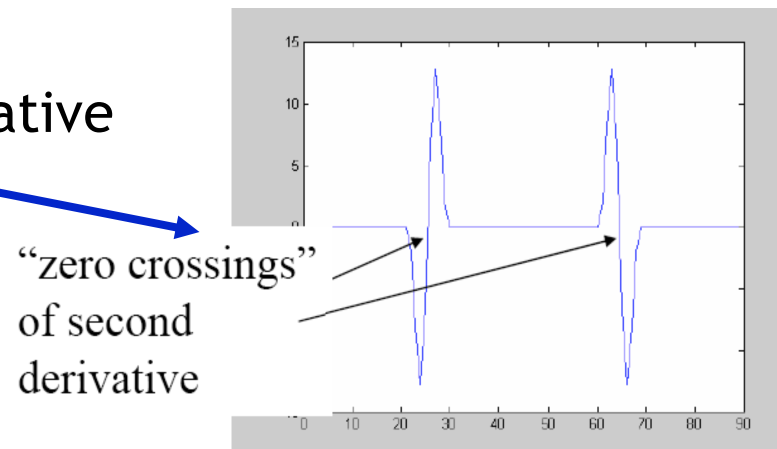




1st derivative



2nd derivative



Compute 2nd order derivatives

- 1st derivative:

$$\frac{d}{dx}f(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \approx f(x+1) - f(x)$$

- 2nd derivative:

$$\frac{d^2}{dx^2}f(x) = \lim_{h \to 0} \frac{\frac{d}{dx}f(x+h) - \frac{d}{dx}f(x)}{h} \approx \frac{d}{dx}f(x+1) - \frac{d}{dx}f(x)$$

$$\approx f(x+2) - 2f(x+1) + f(x)$$

- mask for
  - 1st derivative: 2nd derivative:

-1	1

1	-2	1

The Laplacian

- The Laplacian:

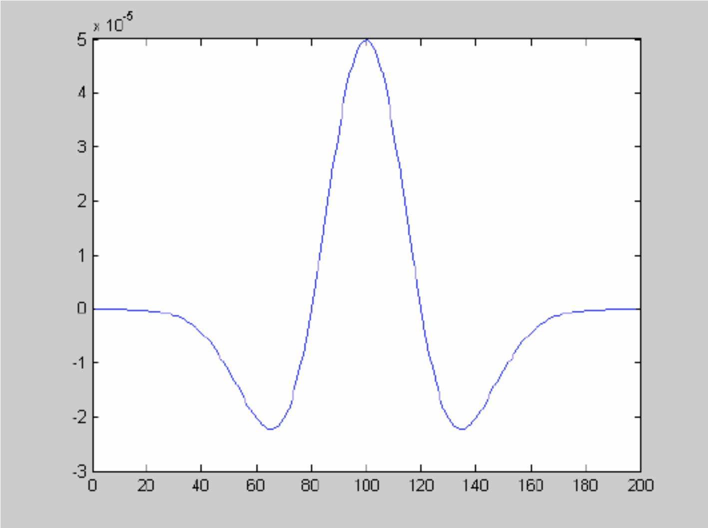
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

  - just another linear filter:

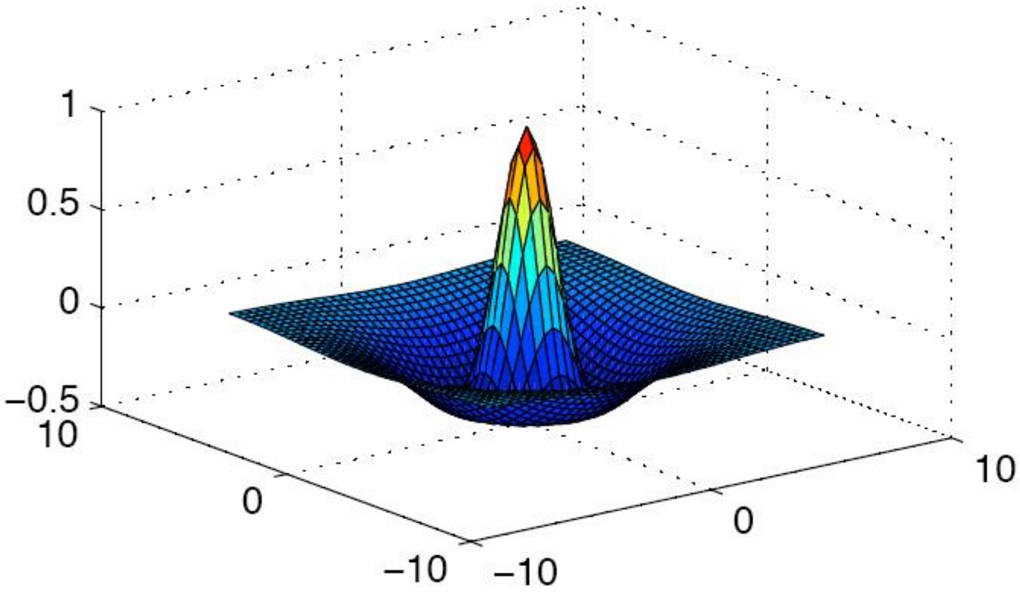
$$\nabla^2 \left(G \otimes f \right) = \nabla^2 G \otimes f$$

Second Derivative of Gaussian

- in 1D:



- in 2D ('mexican hat'):



1D edge detection

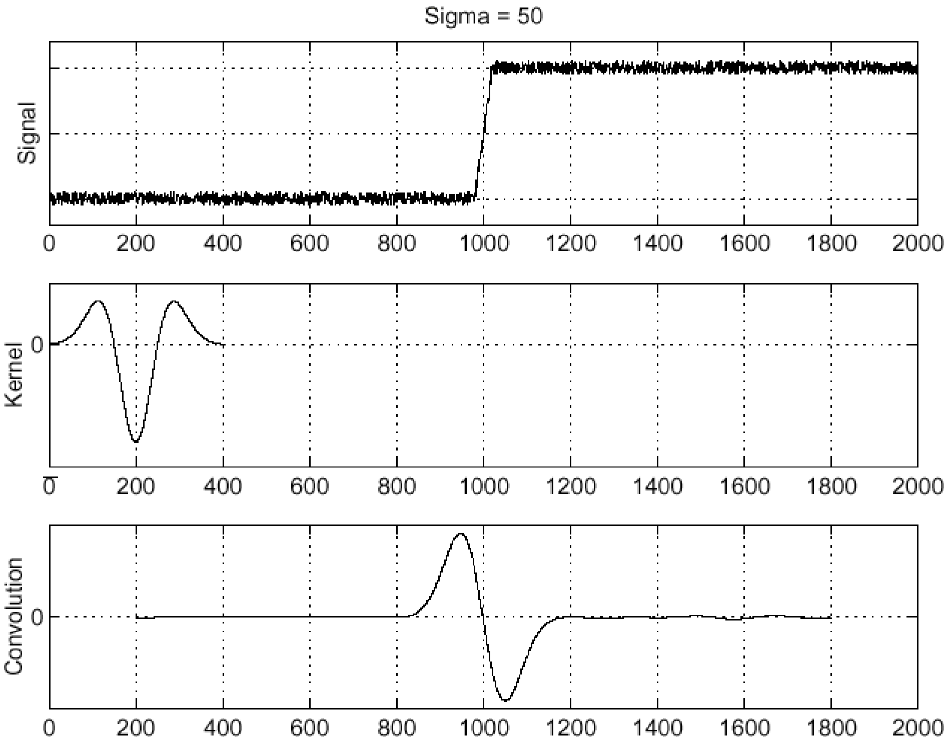
- using Laplacian

$$f$$

Laplacian of Gaussian operator

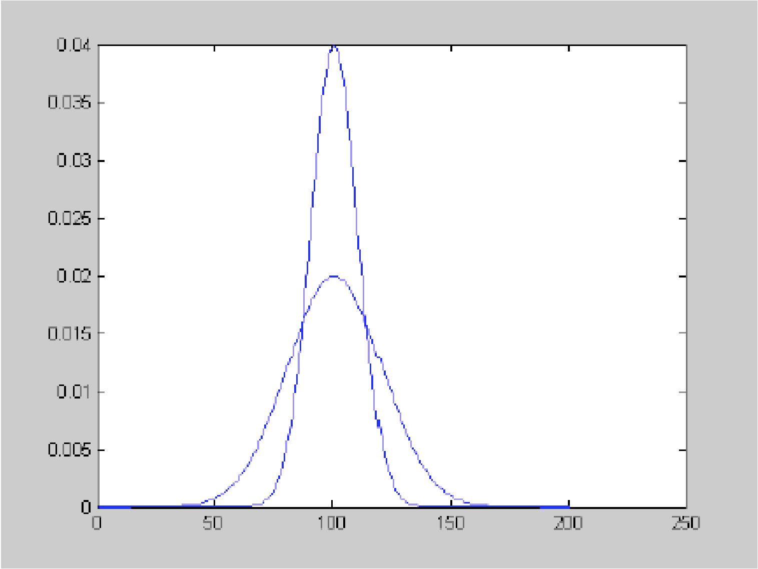
$$\frac{d^2}{dx^2}g$$

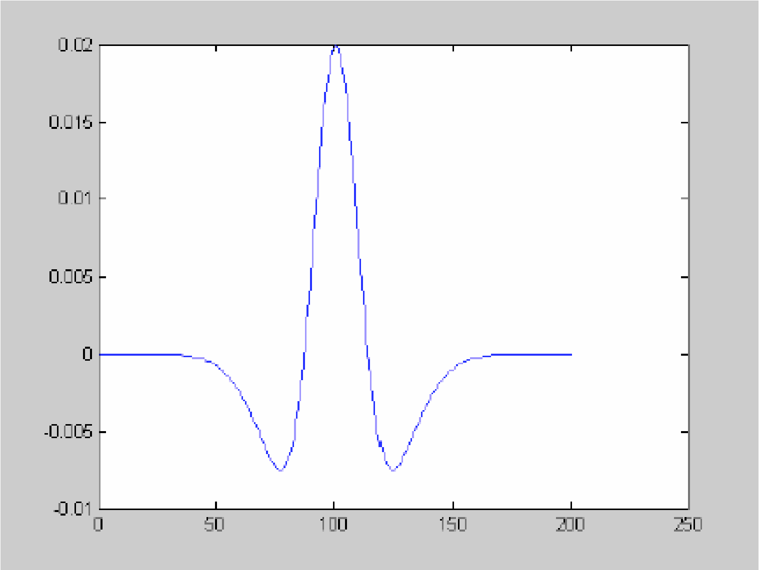
$$\left(\frac{d^2}{dx^2}g\right) \otimes f$$



Approximating the Laplacian

- Difference of Gaussians (DoG) at different scales:



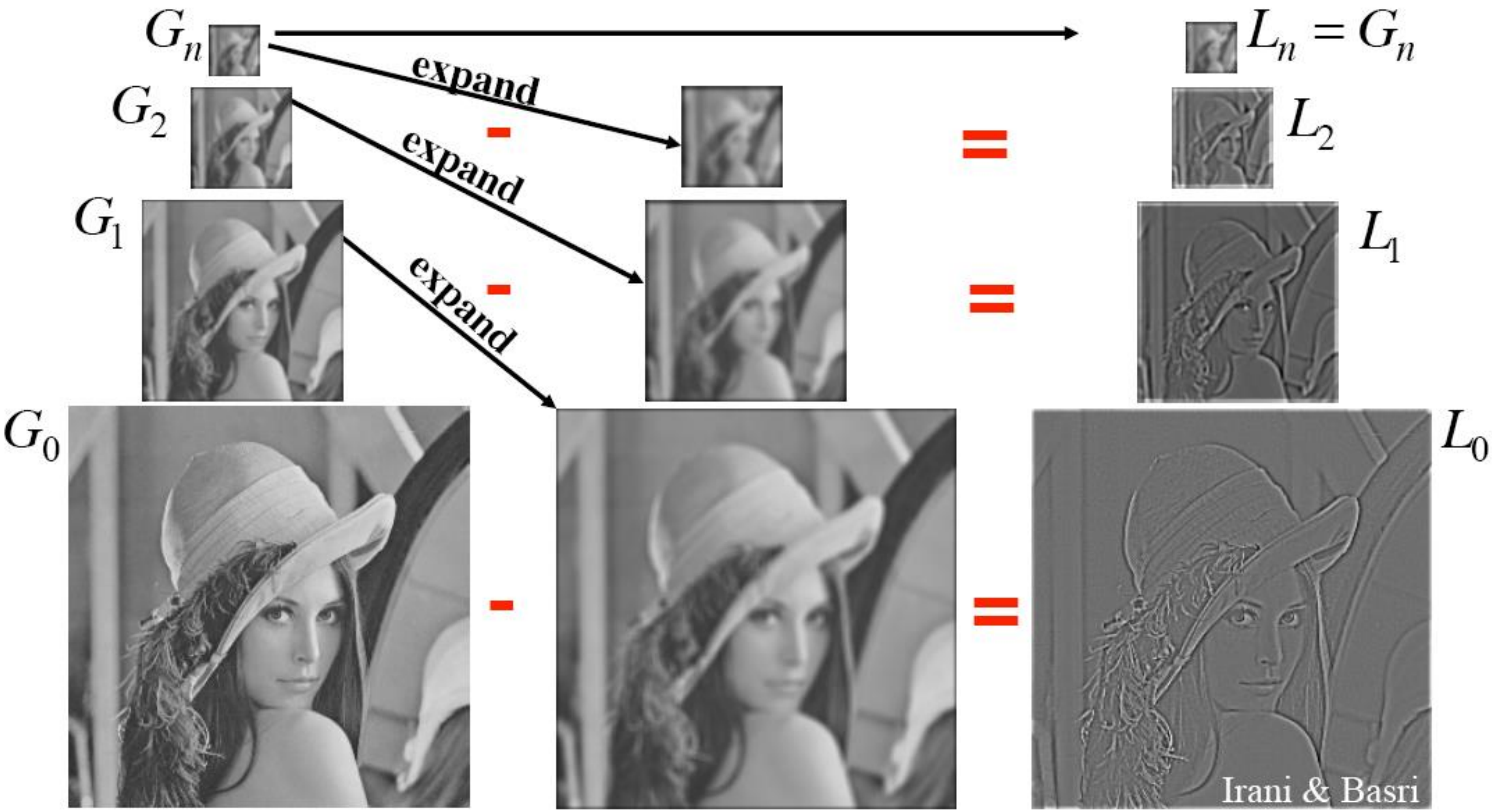


The Laplacian Pyramid

$$L_i = G_i - \text{expand}(G_{i+1})$$

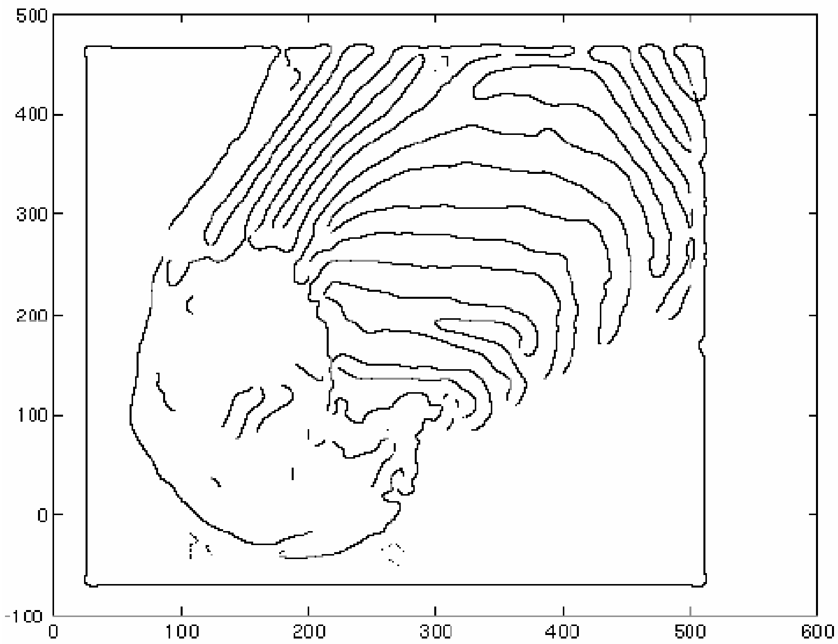
$$G_i = L_i + \text{expand}(G_{i+1})$$

Gaussian Pyramid **Laplacian Pyramid**

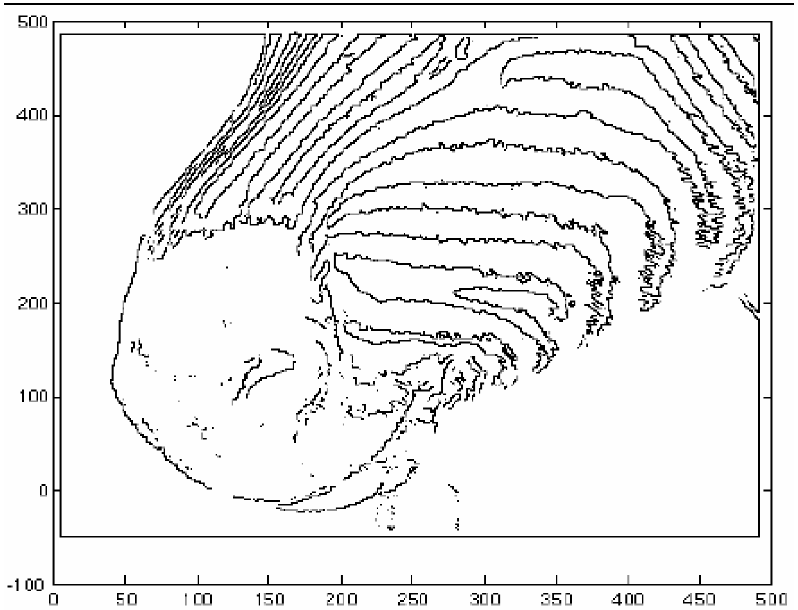


Edge Detection with Laplacian

- sigma = 4



- sigma = 2



Basics of Digital Image Filtering

- Linear Filtering
  - Gaussian Filtering
- Multi Scale Image Representation
  - Gaussian Pyramid
- Edge Detection
  - ‘Recognition using Line Drawings’
  - Image derivatives (1st and 2nd order)
- Object Instance Identification using Color Histograms
- Performance evaluation

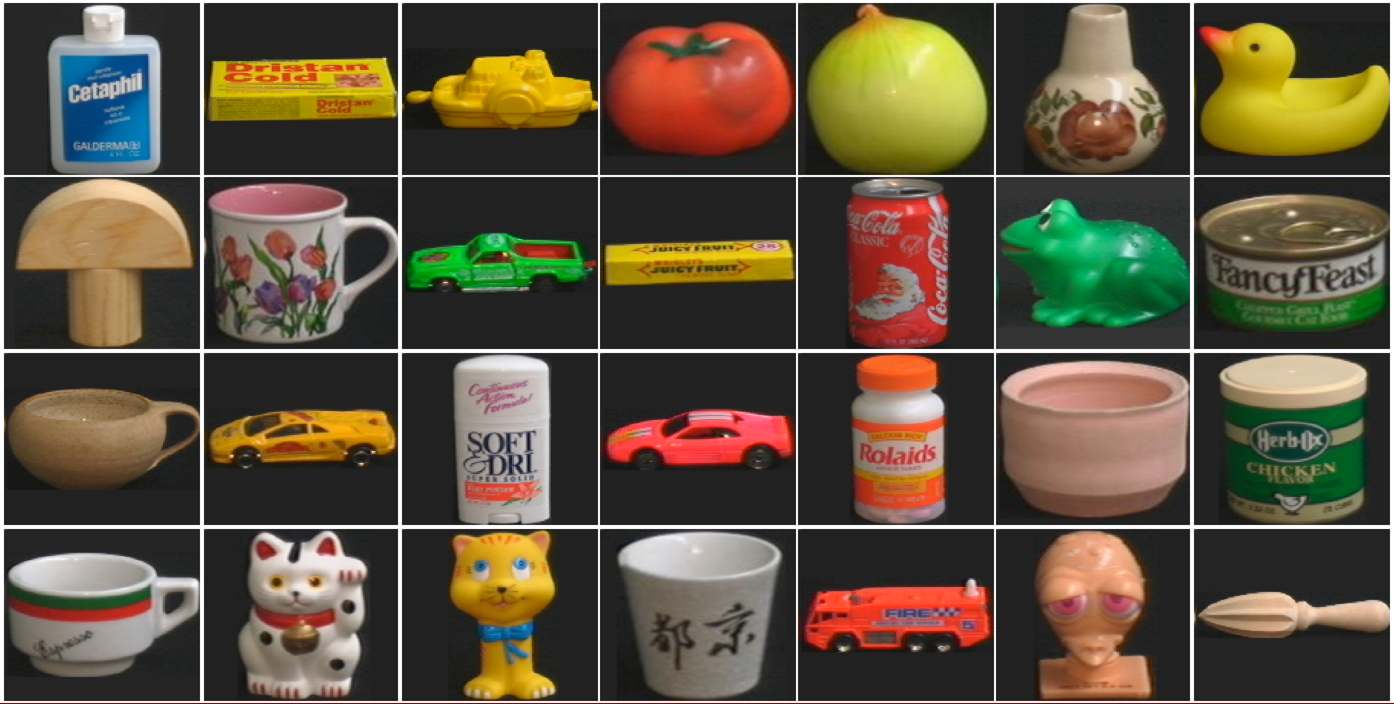
Object Instance Identification using Color Histograms

Object Recognition (reminder)

- Different Types of Recognition Problems:
  - Object **Identification**
    - recognize your apple, your cup, your dog
    - sometimes called: "instance recognition"
  - Object **Classification**
    - recognize any apple, any cup, any dog
    - also called: generic object recognition, object categorization, …
    - typical definition: 'basic level category'

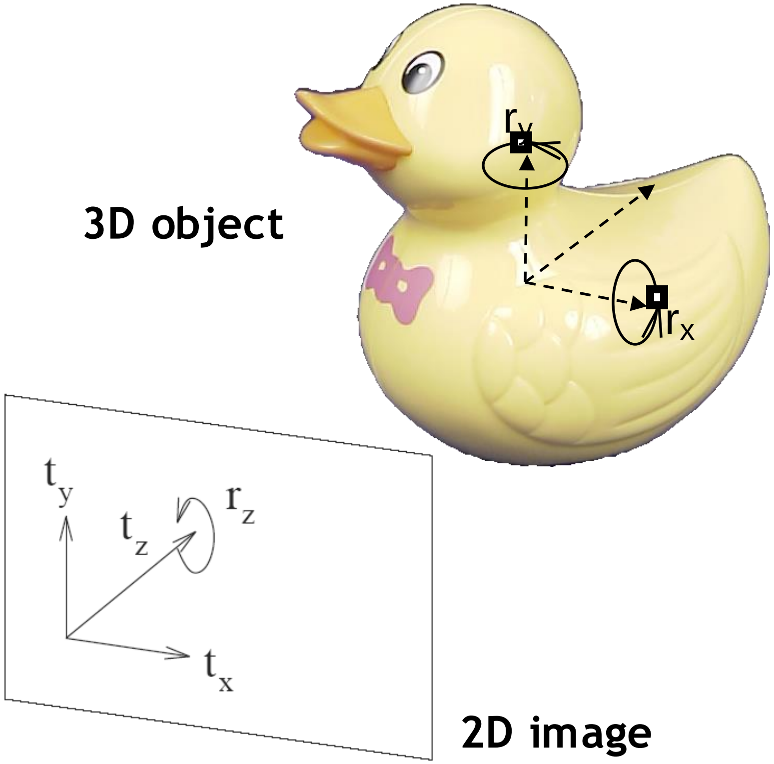
Object Identification

- Example Database for Object Identification:
  - COIL-100 - Columbia Object Image Library
  - contains 100 different objects, some form the same object class (e.g. cars,cups)



Challenges = Modes of Variation

- Viewpoint changes
  - Translation
  - Image-plane rotation
  - Scale changes
  - Out-of-plane rotation

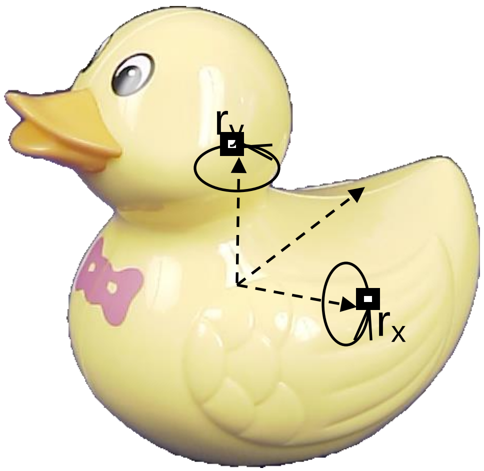


- Illumination
- Clutter
- Occlusion
- Noise

Appearance-Based Identification / Recognition

- Basic assumption
  - Objects can be represented by a collection of images (“appearances”).
  - For recognition, it is sufficient to just compare the 2D appearances.
  - No 3D model is needed.

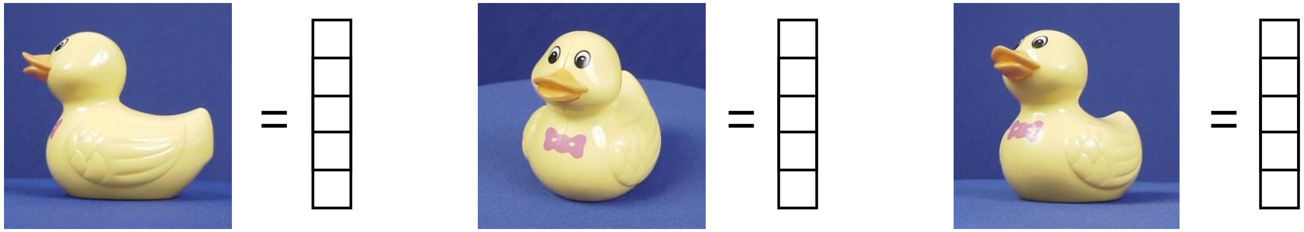
3D object



- ⇒ Fundamental paradigm shift in the 90’s

Global Representation

- Idea
  - Represent each view (of an object) by a global descriptor.



  - For recognizing objects, just match the (global) descriptors.
  - Modes of variation can be taken care of by:
    - built into the descriptor
      - e.g. a descriptor can be made invariant to image-plane rotations, translation
    - incorporate in the training data or the recognition process.
      - e.g. viewpoint changes, scale changes, out-of-plane rotation
    - robustness of descriptor or recognition process (descriptor matching)
      - e.g. illumination, noise, clutter, partial occlusion

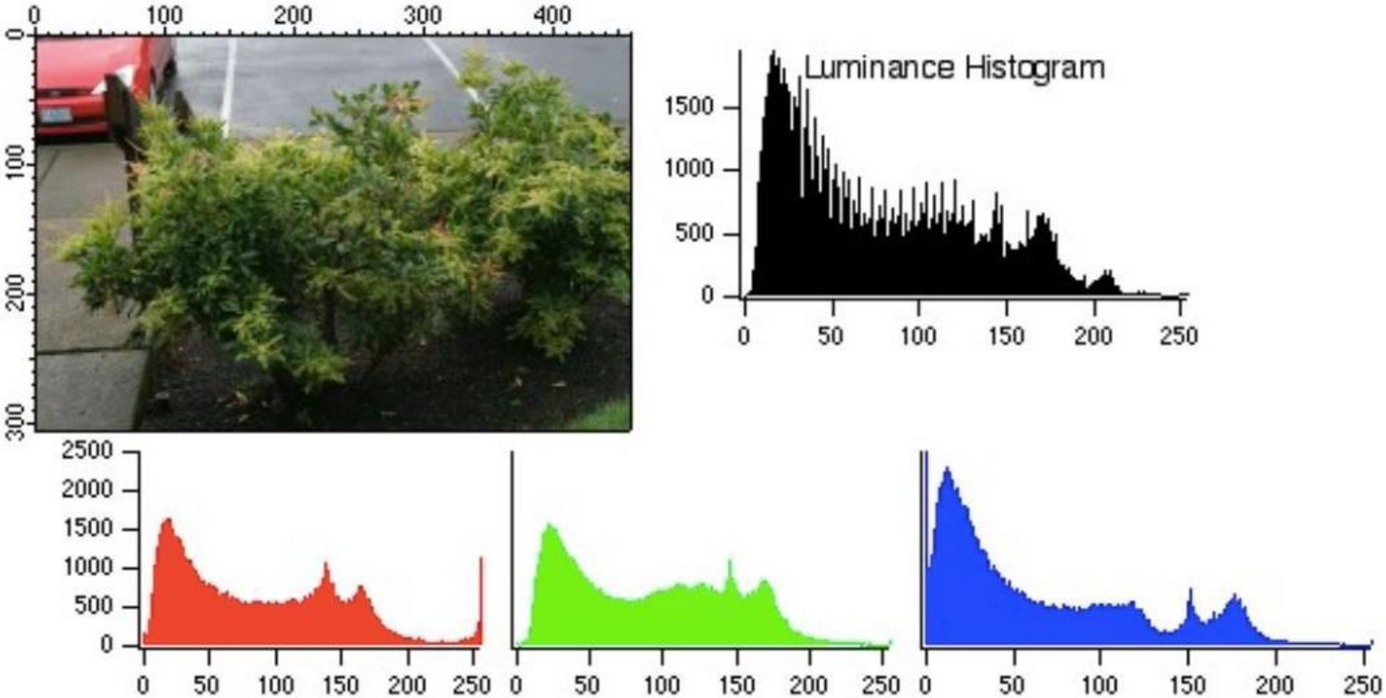
Case Study:
Use Color for Recognition

- Color:
  - Color stays constant under geometric transformations
  - Local feature
    - Color is defined for each pixel
    - Robust to partial occlusion

- Idea
  - Directly use object colors for identification / recognition
  - Better: use statistics of object colors

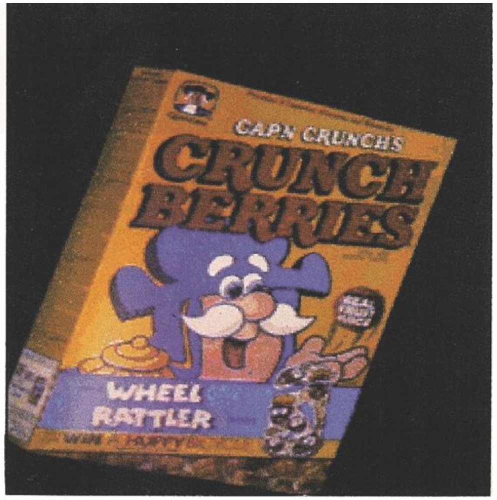
Color Histograms

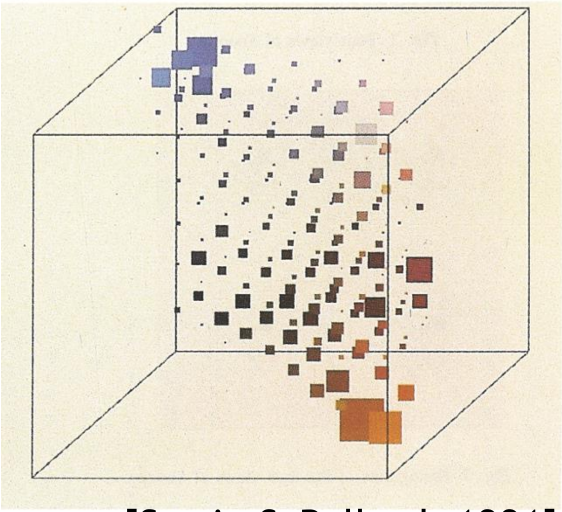
- Color statistics
  - Given: R,G,B for each pixel
  - Compute 1D histograms for the R, G and B, as well as for the luminance
    - E.g. Hist(R) = #(pixels with color R)



3D (Joint) Color Histograms

- Color statistics
  - Given: tri-stimulus R,G,B for each pixel
  - Compute 3D histogram
    - H(R,G,B) = #(pixels with color (R,G,B))



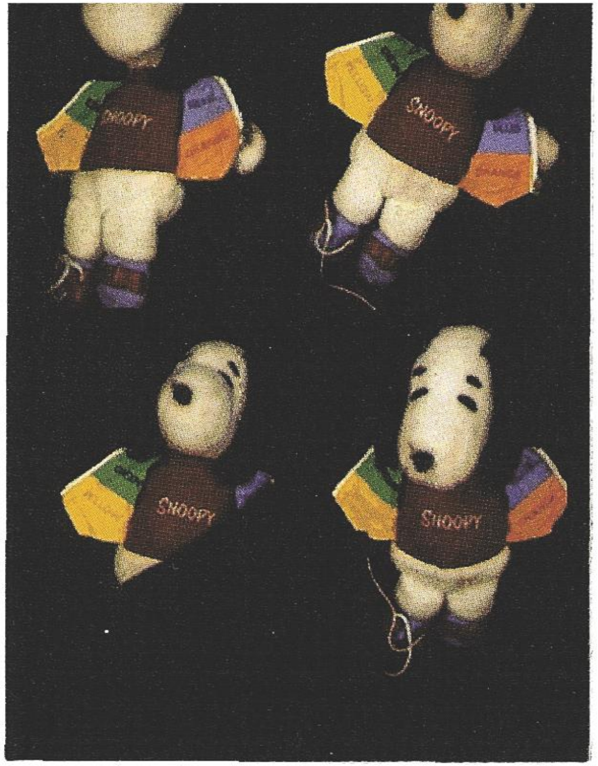


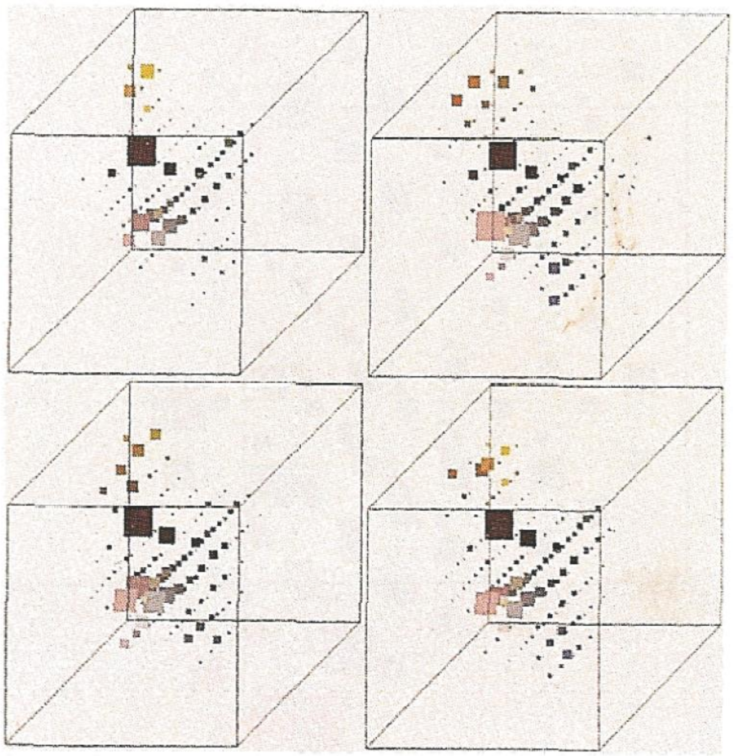
[Swain & Ballard, 1991]

- Embed the image into a "more meaningful" space endowed with some notion of "closeness"

Color Histograms

- Robust representation
  - presence of occlusion, rotation





[Swain & Ballard, 1991]

Color

- One component of the 3D color space is intensity
  - If a color vector is multiplied by a scalar, the intensity changes, but not the color itself.
  - This means colors can be normalized by the intensity.
    - Intensity is given by: I = R + G + B:
  - „Chromatic representation“

$$r = \frac{R}{R+G+B}$$

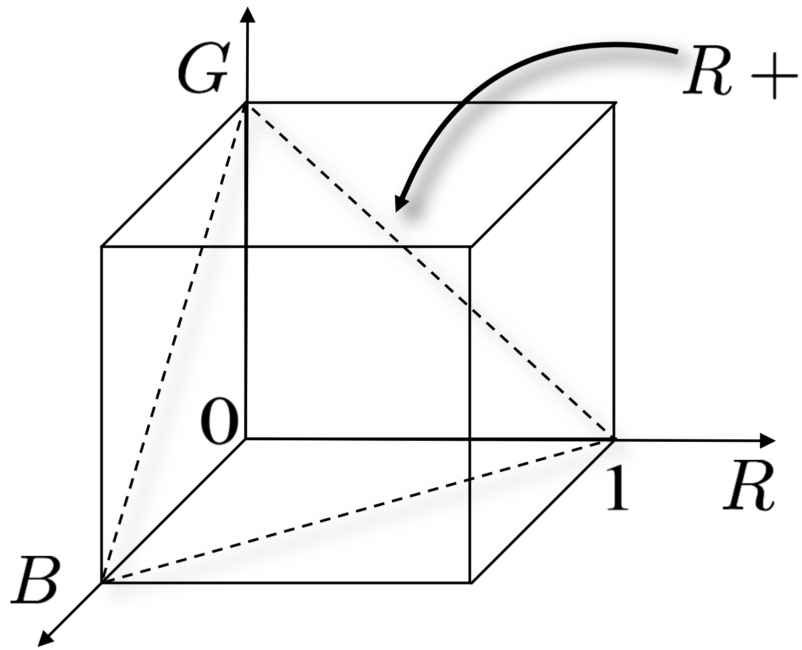
$$g = \frac{G}{R+G+B}$$

$$b = \frac{B}{R+G+B}$$

Color

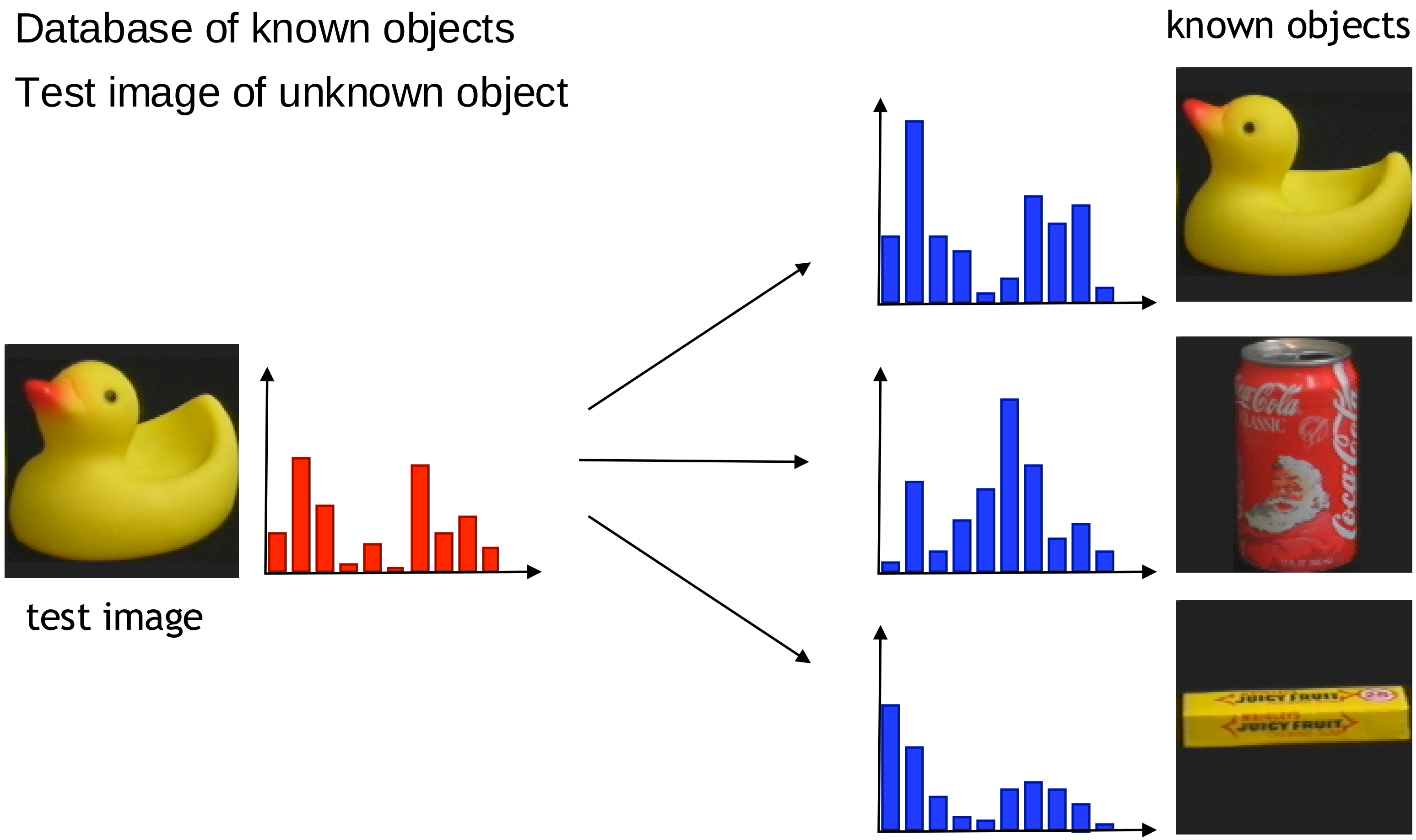
- Observation:
  - Since r + g + b = 1, only 2 parameters are necessary
  - E.g. one can use r and g
  - and obtains b = 1 - r - g

$$r + g + b = 1$$
$$\Rightarrow b = 1 - r - g$$



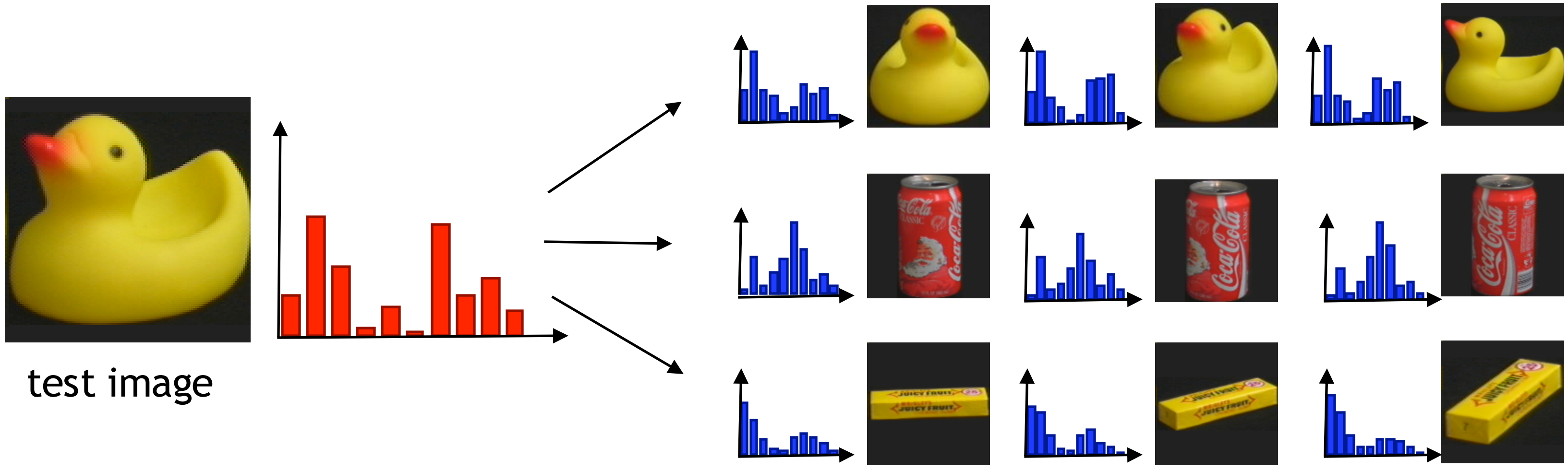
Recognition using Histograms

- Histogram comparison
  - Database of known objects
  - Test image of unknown object



Recognition using Histograms

- Database with multiple training views per object

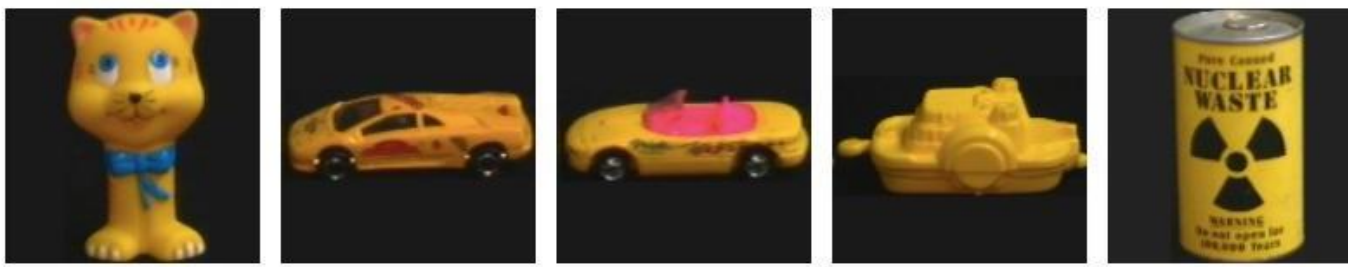


Recognition using Histograms

- Retrieved object instances given the query-image color histogram

Query

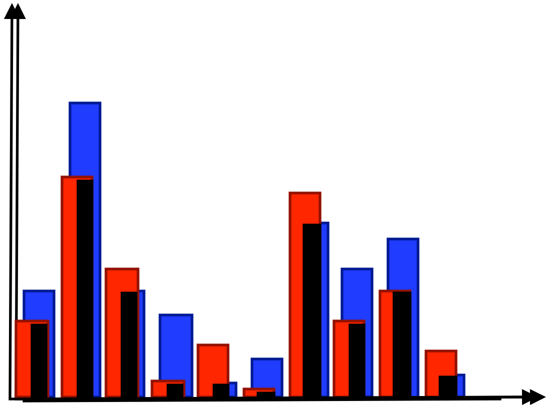
Retrieved objects



Histogram Comparison

- Comparison measures
  - Intersection

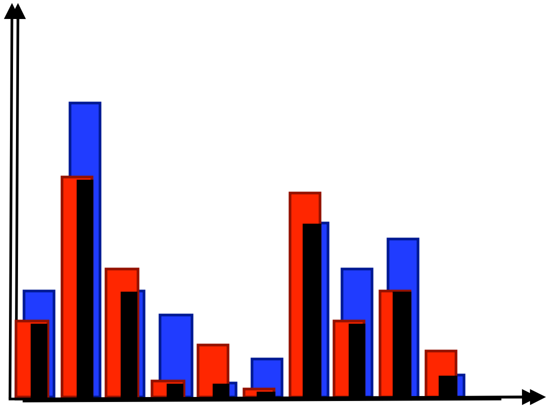
$$\cap(Q,V) = \sum_i \min(q_i, v_i)$$



Histogram Comparison

- Comparison measures
  - Intersection

$$\cap(Q,V) = \sum_i \min(q_i, v_i)$$



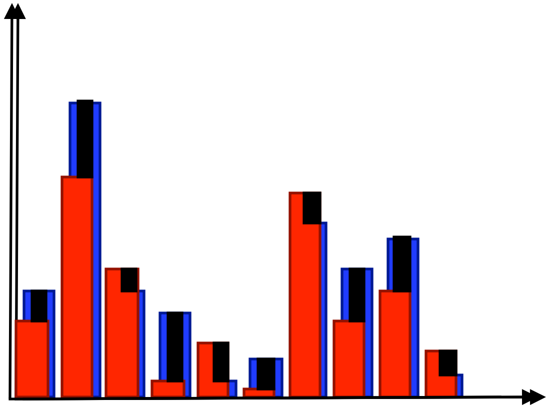
- Motivation
  - Measures the common part of both histograms
  - Range: [0,1]
  - For unnormalized histograms, use the following formula

$$\cap(Q,V) = \frac{1}{2}\left(\frac{\sum_i \min(q_i, v_i)}{\sum_i q_i} + \frac{\sum_i \min(q_i, v_i)}{\sum_i v_i}\right)$$

Histogram Comparison

- Comparison Measures
  - Euclidean Distance

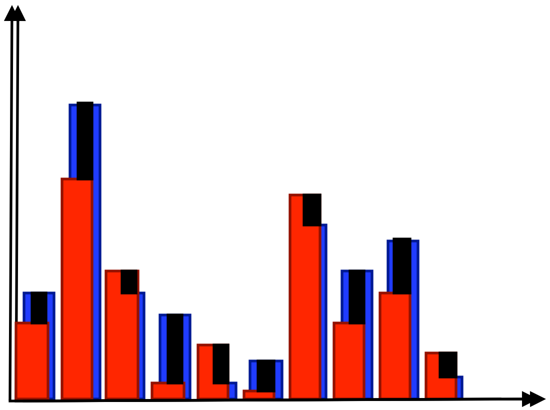
$$d(Q,V) = \sum_i (q_i - v_i)^2$$



Histogram Comparison

- Comparison Measures
  - Euclidean Distance

$$d(Q,V) = \sum_i (q_i - v_i)^2$$

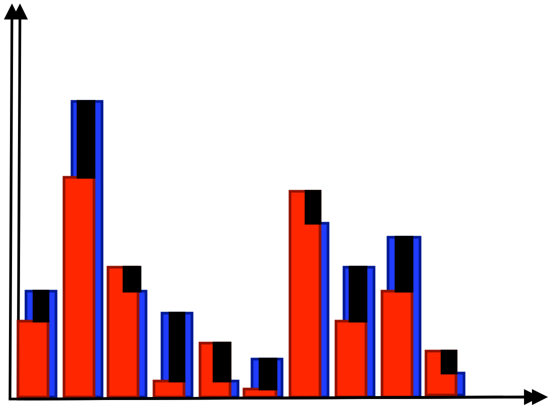


- Motivation
  - Focuses on the differences between the histograms
  - Range: [0,∞]
  - All cells are weighted equally.
  - Not very discriminant

Histogram Comparison

- Comparison Measures
  - Chi-square

$$\chi^2(Q,V) = \sum_i \frac{(q_i - v_i)^2}{q_i + v_i}$$



- Motivation
  - Statistical background:
    - Test if two distributions are different
    - Possible to compute a significance score
  - Range: $[0,\infty]$
  - Cells are not weighted equally!
    - therefore more discriminant
    - may have problems with outliers (therefore assume that each cell contains at least a minimum of samples)

Histogram Comparison

- Which measure is best?
  - Depends on the application…
  - Both Intersection and χ^2 give good performance.
    - Intersection is a bit more robust.
    - χ^2 is a bit more discriminative.
    - Euclidean distance is not robust enough.
  - There exist many other measures
    - e.g. statistical tests: Kolmogorov-Smirnov
    - e.g. information theoretic: Kullback-Leibler divergence, Jeffrey divergence, ...

Recognition using Histograms

- Simple algorithm
  1. Build a set of histograms $H = \{M_1, M_2, M_3, ...\}$ for each known object
     - more exactly, for each view of each object
  2. Build a histogram T for the test image.
  3. Compare T to each $M_k \in H$
     - using a suitable comparison measure
  4. Select the object with the best matching score
     - or reject the test image if no object is similar enough (distance above a threshold *t*)

"Nearest-Neighbor" strategy

Color Histograms

- Recognition (here object identification)
  - Works surprisingly well
  - In the first paper (1991), 66 objects could be recognized almost without errors

[Swain & Ballard, 1991]

Discussion: Color Histograms

- Advantages
  - Invariant to object translations
  - Invariant to image rotations
  - Slowly changing for out-of-plane rotations
  - No perfect segmentation necessary
  - Histograms change gradually when part of the object is occluded
  - Possible to recognize deformable objects
    - e.g. pullover
- Problems
  - The pixel colors change with the illumination („color constancy problem“)
    - Intensity
    - Spectral composition (illumination color)
  - Not all objects can be identified by their color distribution.

Basics of Digital Image Filtering

- Linear Filtering
  - Gaussian Filtering
- Multi Scale Image Representation
  - Gaussian Pyramid
- Edge Detection
  - ‘Recognition using Line Drawings’
  - Image derivatives (1st and 2nd order)
- Object Instance Identification using Color Histograms
- Performance evaluation

Performance evaluation

Performance Evaluation

- How can we say if method A is better than method B for the same task?

1. Compare a single number - e.g. accuracy (recognition rate), top-k accuracy
2. Compare curves - e.g. precision-recall curve, ROC curve

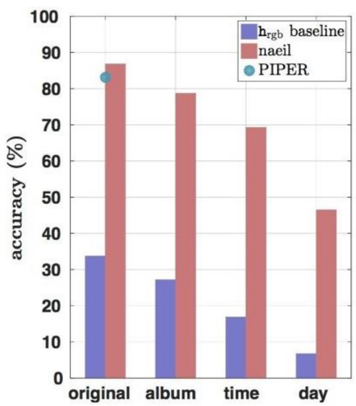
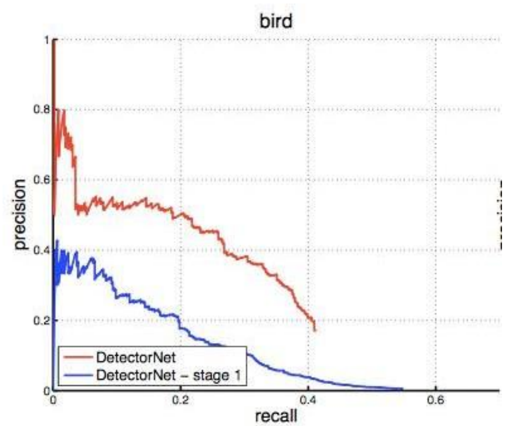
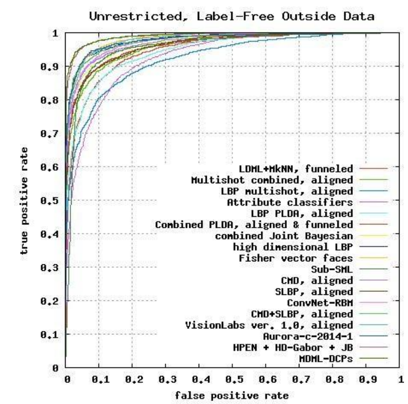


Figure 4. Recognition accuracy across different experimental setups on the test data.

Accuracy
(Oh, ICCV'15)



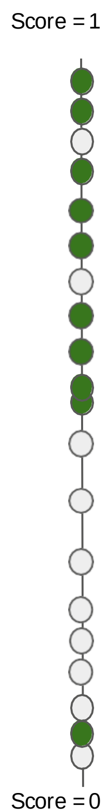
Precision-Recall
(Szegedy, NIPS'13)



ROC
(LFW Face verification)

Score-based evaluation

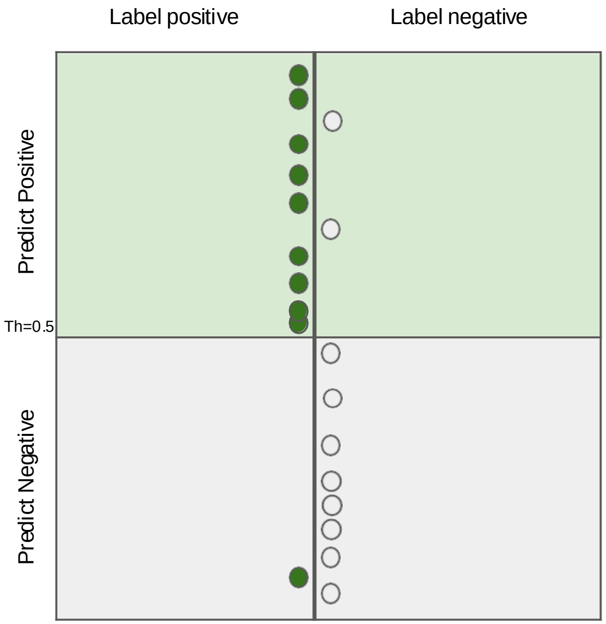
- The recognition algorithm identifies (classifies) the *query* object as matching the *training* image if their *similarity* is above a threshold *t*



	Positive example
	Negative example

Threshold -> Classifier -> Point Metrics

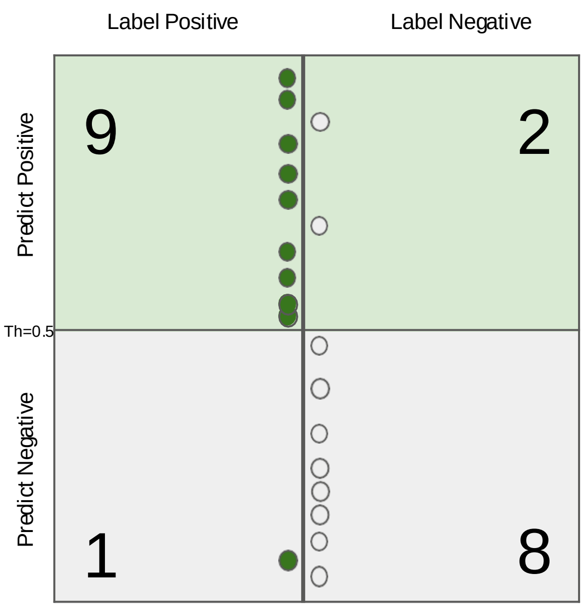
- The recognition algorithm identifies (classifies) the *query* object as matching the *training* image if their *similarity* is above a threshold *t*



Th
0.5

Point metrics: Confusion Matrix

- The recognition algorithm identifies (classifies) the *query* object as matching the *training* image if their *similarity* is above a threshold *t*

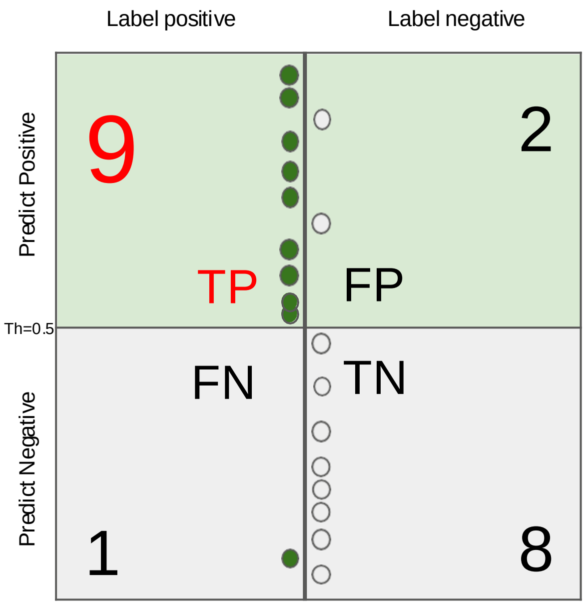


Th
0.5

Properties:

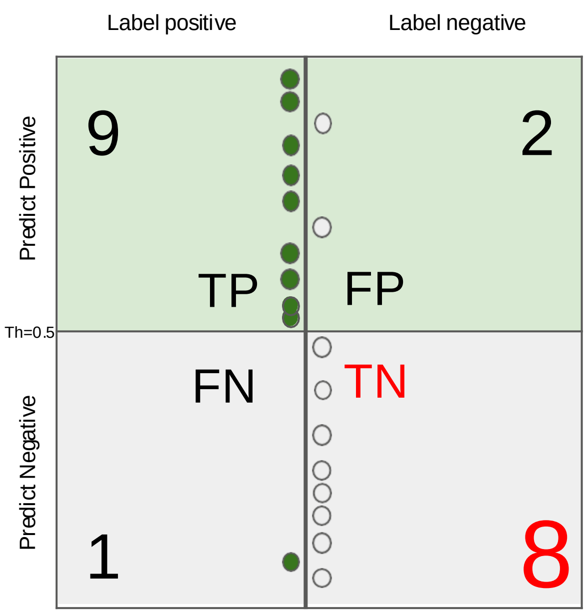
- Quality of model & threshold decide how columns are split into rows.
- We want diagonals to be “heavy”, off diagonals to be “light”.

Point metrics: True Positives



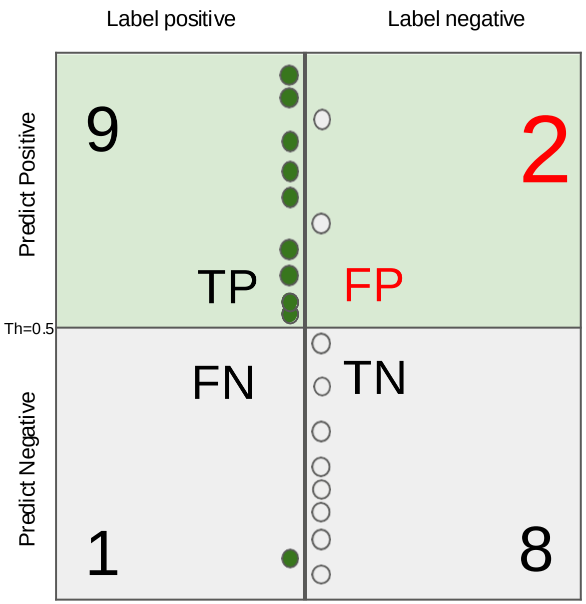
Th	TP
0.5	9

Point metrics: True Negatives



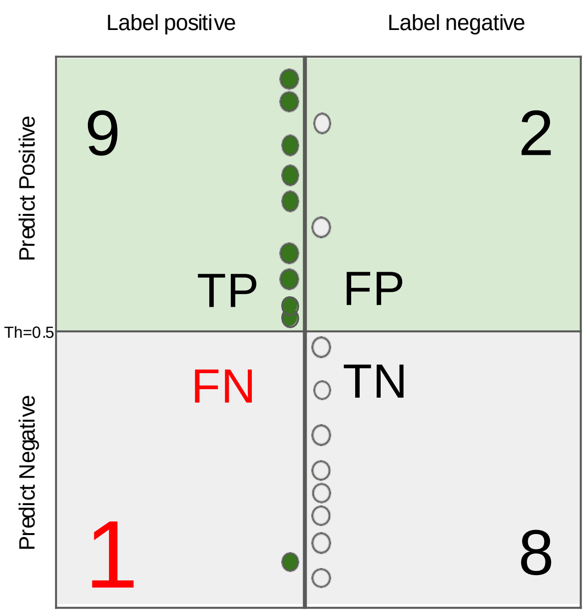
Th	TP	TN
0.5	9	8

Point metrics: False Positives



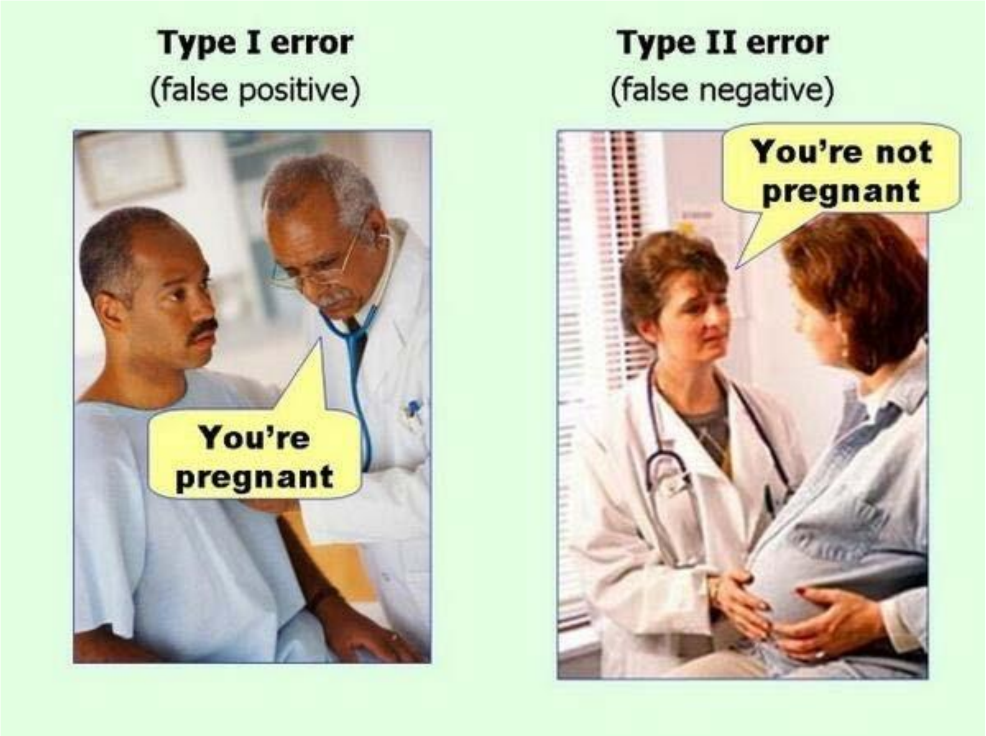
Th	TP	TN	FP
0.5	9	8	2

Point metrics: False Negatives

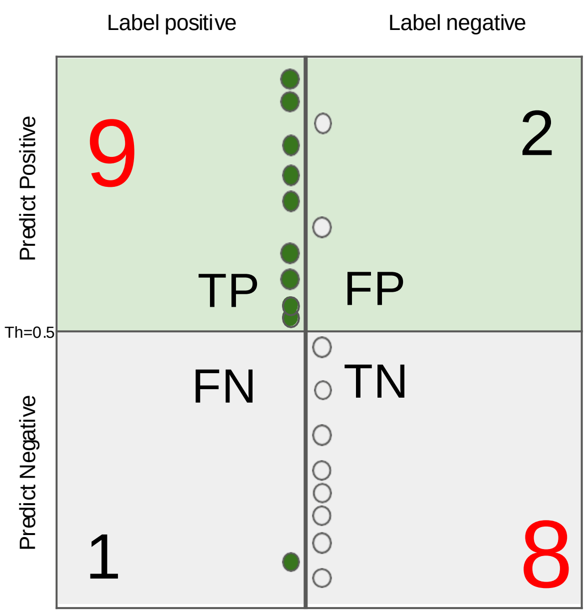


Th	TP	TN	FP	FN
0.5	9	8	2	1

FP and FN also called Type-1 and Type-2 errors



Point metrics: Accuracy

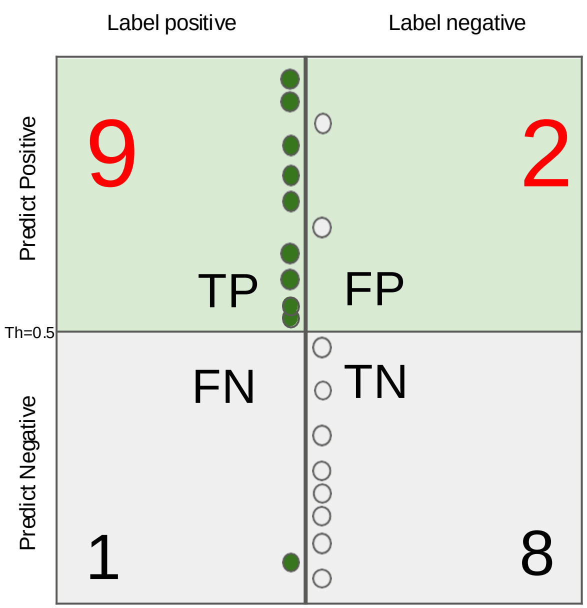


Th	TP	TN	FP	FN	Acc
0.5	9	8	2	1	.85

$$\text{Overall accuracy} = (TN + TP)/N$$

Equivalent to 0-1 Loss!

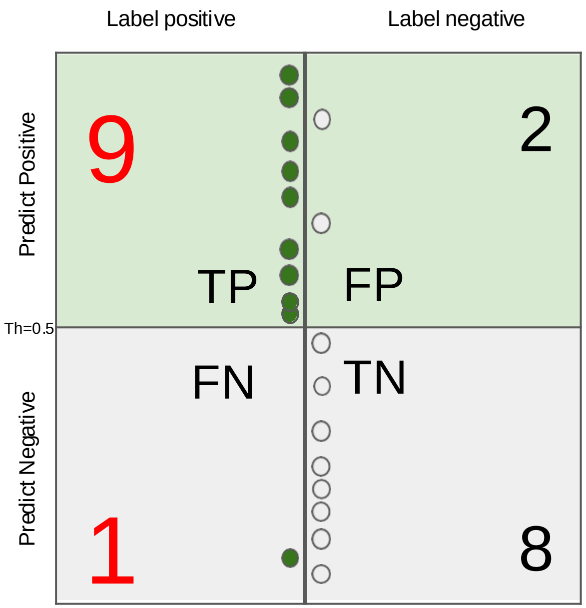
Point metrics: Precision



Th	TP	TN	FP	FN	Acc	Pr
0.5	9	8	2	1	.85	.81

$$\text{Precision} = \frac{TP}{TP + FP}$$

Point metrics:
Positive Recall, True Positive Rate, Sensitivity



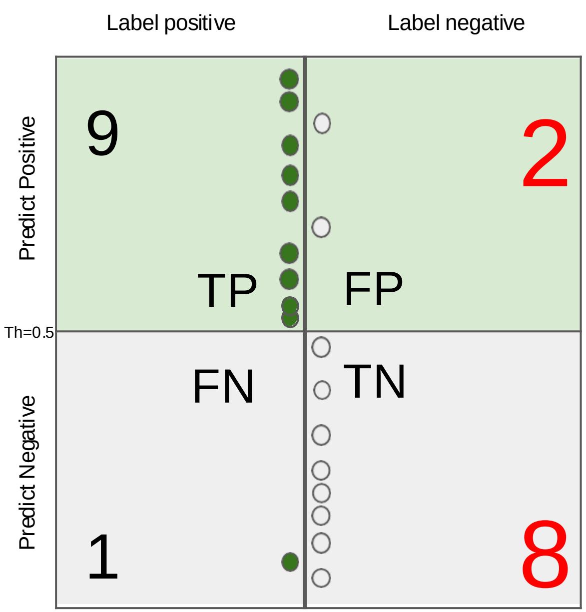
Th	TP	TN	FP	FN	Acc	Pr	Recall
0.5	9	8	2	1	.85	.81	.9

$$\text{Recall} = \text{True positive rate} = \frac{TP}{TP + FN} = \text{Sensitivity}$$

Trivial 100% recall = pull everybody above the threshold.
Trivial 100% precision = push everybody below the threshold except 1 green on top.
(Hopefully no gray above it!)

Striving for good precision with 100% recall =
pulling up the lowest green as high as possible in the ranking.
Striving for good recall with 100% precision =
pushing down the top gray as low as possible in the ranking.

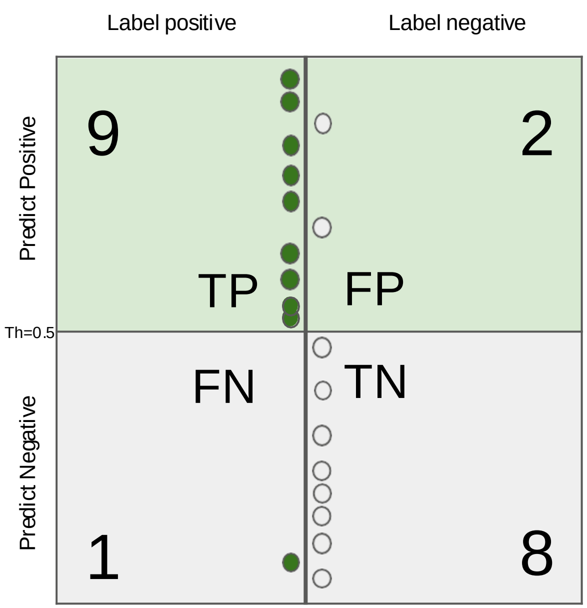
Point metrics:
Negative Recall, False Positive Rate, Specificity



Th	TP	TN	FP	FN	Acc	Pr	Recall	Spec
0.5	9	8	2	1	.85	.81	.9	0.8

$$\text{False positive rate} = \frac{FP}{TN + FP} = 1\text{-Specificity}$$

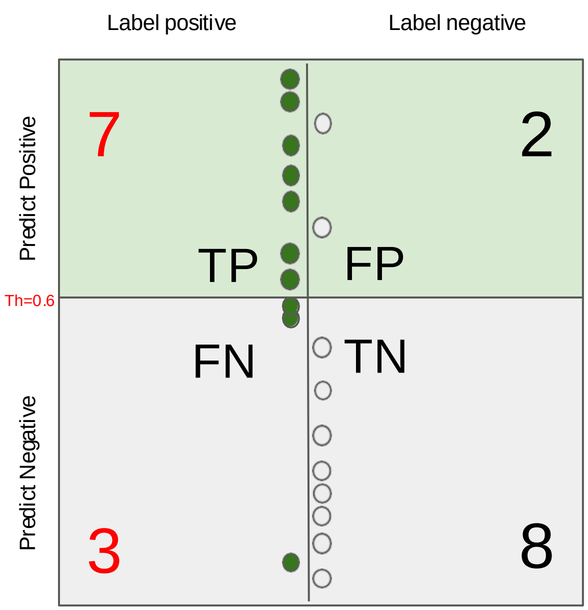
Point metrics: F1-score



Th	TP	TN	FP	FN	Acc	Pr	Recall	Spec	F1
0.5	9	8	2	1	.85	.81	.9	.8	.857

$$F_1 = \left(\frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}}\right) = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

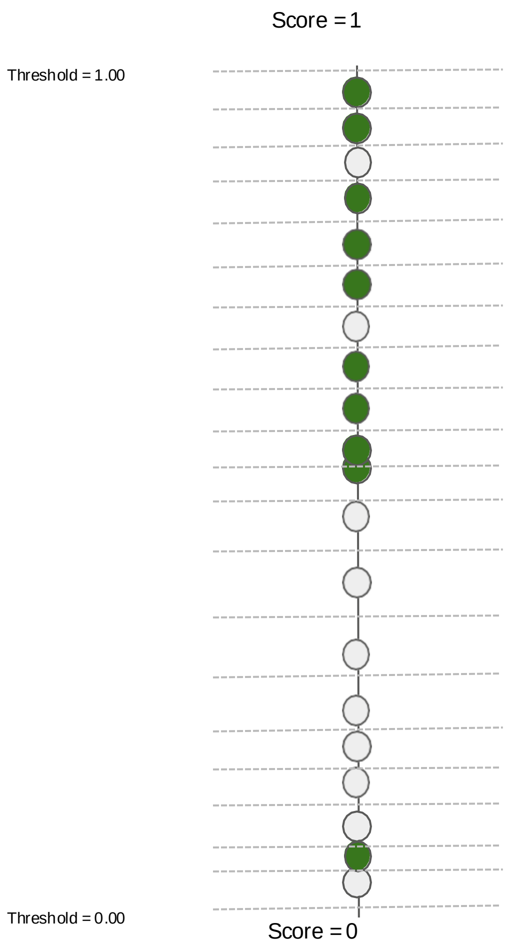
Point metrics: Changing threshold



Th	TP	TN	FP	FN	Acc	Pr	Recall	Spec	F1
0.6	7	8	2	3	.75	.77	.7	.8	.733

effective thresholds = # examples + 1

Threshold Scanning



Threshold	TP	TN	FP	FN	Accuracy	Precision	Recall	Specificity	F1
1.00	0	10	0	10	0.50	1	0	1	0
0.95	1	10	0	9	0.55	1	0.1	1	0.182
0.90	2	10	0	8	0.60	1	0.2	1	0.333
0.85	2	9	1	8	0.55	0.667	0.2	0.9	0.308
0.80	3	9	1	7	0.60	0.750	0.3	0.9	0.429
0.75	4	9	1	6	0.65	0.800	0.4	0.9	0.533
0.70	5	9	1	5	0.70	0.833	0.5	0.9	0.625
0.65	5	8	2	5	0.65	0.714	0.5	0.8	0.588
0.60	6	8	2	4	0.70	0.750	0.6	0.8	0.667
0.55	7	8	2	3	0.75	0.778	0.7	0.8	0.737
0.50	8	8	2	2	0.80	0.800	0.8	0.8	0.800
0.45	9	8	2	1	0.85	0.818	0.9	0.8	0.857
0.40	9	7	3	1	0.80	0.750	0.9	0.7	0.818
0.35	9	6	4	1	0.75	0.692	0.9	0.6	0.783
0.30	9	5	5	1	0.70	0.643	0.9	0.5	0.750
0.25	9	4	6	1	0.65	0.600	0.9	0.4	0.720
0.20	9	3	7	1	0.60	0.562	0.9	0.3	0.692
0.15	9	2	8	1	0.55	0.529	0.9	0.2	0.667
0.10	9	1	9	1	0.50	0.500	0.9	0.1	0.643
0.05	10	1	9	0	0.55	0.526	1	0.1	0.690
0.00	10	0	10	0	0.50	0.500	1	0	0.667

Recap

- The recognition algorithm identifies (classifies) the *query* object as matching the *training* image if their *similarity* is above a threshold *t*
- Compare actual outcomes to predicted outcomes using a *confusion matrix (classification matrix)*

	Predicted = 0	**Predicted = 1**
Actual = 0	True Negatives (TN)	False Positives (FP)
Actual = 1	False Negatives (FN)	True Positives (TP)

N = number of observations

Overall accuracy = $(TN + TP)/N$ Overall error rate = $(FP + FN)/N$

$$\text{False positive rate} = \frac{FP}{TN + FP} = 1\text{-Specificity}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{True positive rate} = \frac{TP}{TP + FN} = \text{Sensitivity} = \text{Recall}$$

Performance Evaluation
(Overall) Accuracy

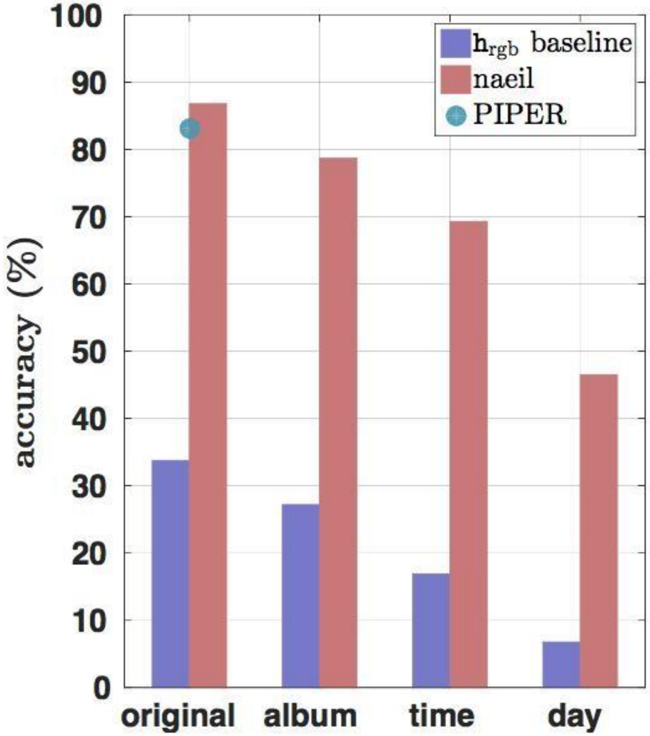


Figure 4. Recognition accuracy across different experimental setups on the test data.

Oh, ICCV'15

$$\frac{\#\text{Correct Predictions}}{\#\text{Total Examples}}$$

Threshold Value

- The recognition algorithm identifies (classifies) the *query* object as matching the *training* image if their *similarity* is above a threshold *t*

- The lower the *t* the more query images are classified as matching
  - More TP but also more FP
- The higher the *t* the less query images are classified as matching
  - More TN but also more FN

- What value should we pick for *t*?

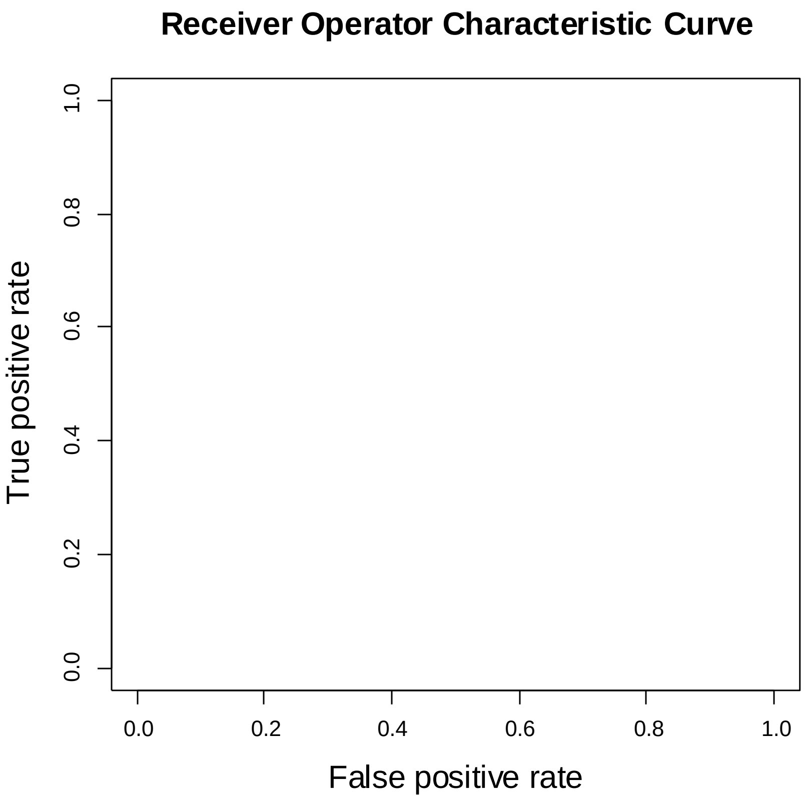
Receiver Operator Characteristic (ROC)

- True positive rate (TPR)
  - the larger the TPR the larger the recall of actual true matches (lower threshold *t*)

$$\text{True positive rate} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

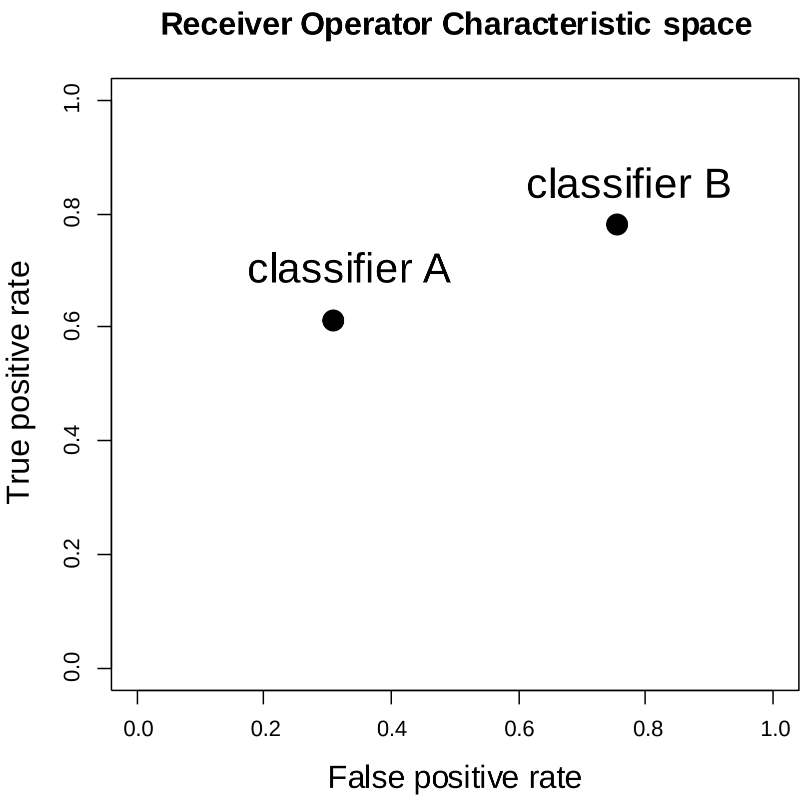
- False positive rate (FPR)
  - The larger the FPR the larger number of false alarms (lower threshold *t*)

$$\text{False positive rate} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$



Receiver Operator Characteristic (ROC) space

- True positive rate (TPR)
  - the larger the TPR
    the larger the recall
    of actual true matches
- False positive rate (FPR)
  - The larger the FPR
    the larger number
    of false alarms



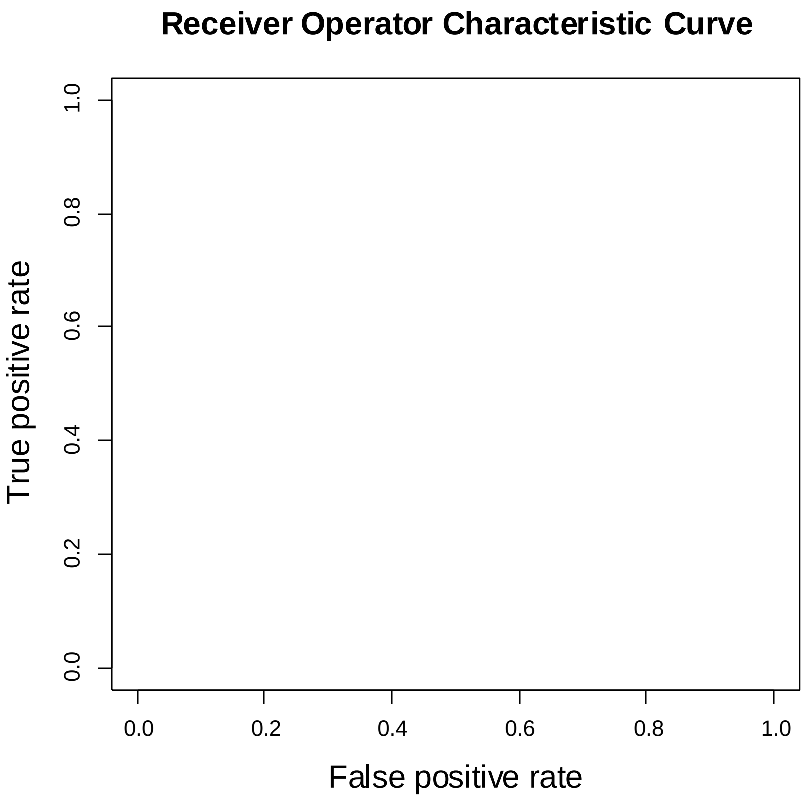
Selecting a Threshold using ROC

- Capture all thresholds simultaneously
- Low threshold t
  - Large TPR
  - Large FPR

$$\text{True positive rate} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- High threshold t
  - Small TPR
  - Small FPR

$$\text{False positive rate} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

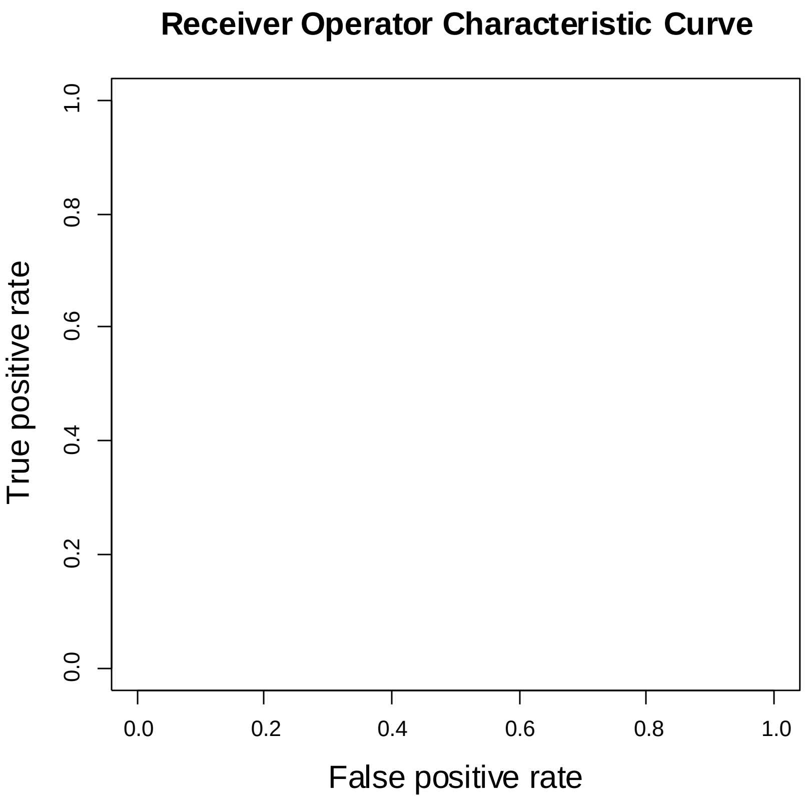


Selecting a Threshold using ROC

- Choose best threshold *t* for the best trade off
  - cost of failing to identify an object
  - cost of raising the false alarms

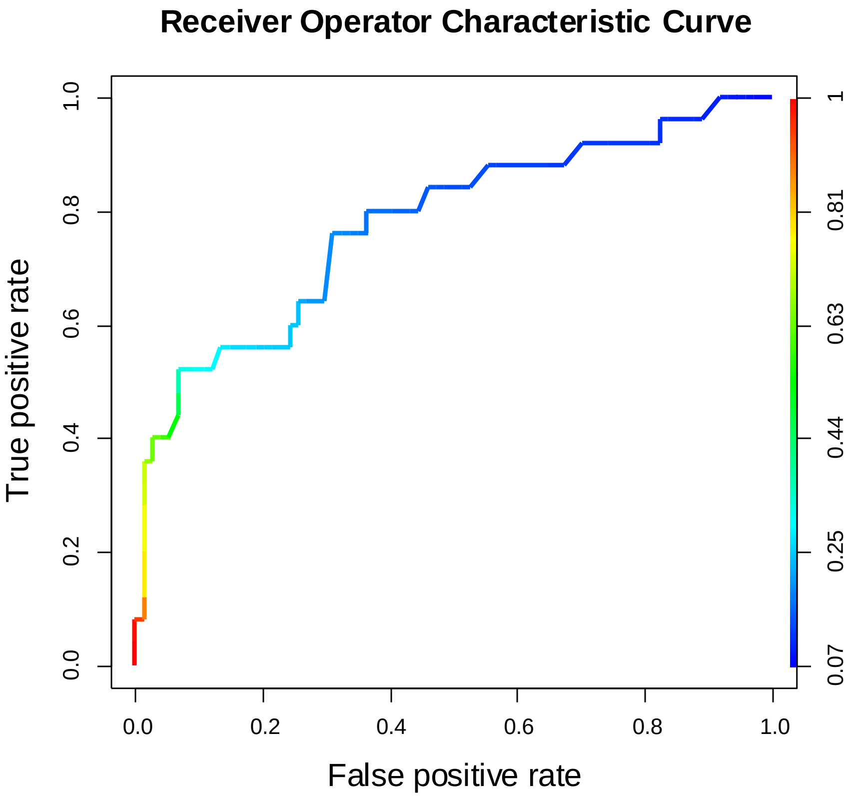
$$\text{True positive rate} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{False positive rate} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$



Selecting a Threshold using ROC

- Choose best threshold *t* for the best trade off
  - cost of failing to identify an object
  - cost of raising the false alarms



$$\text{True positive rate} = \frac{TP}{TP + FN}$$

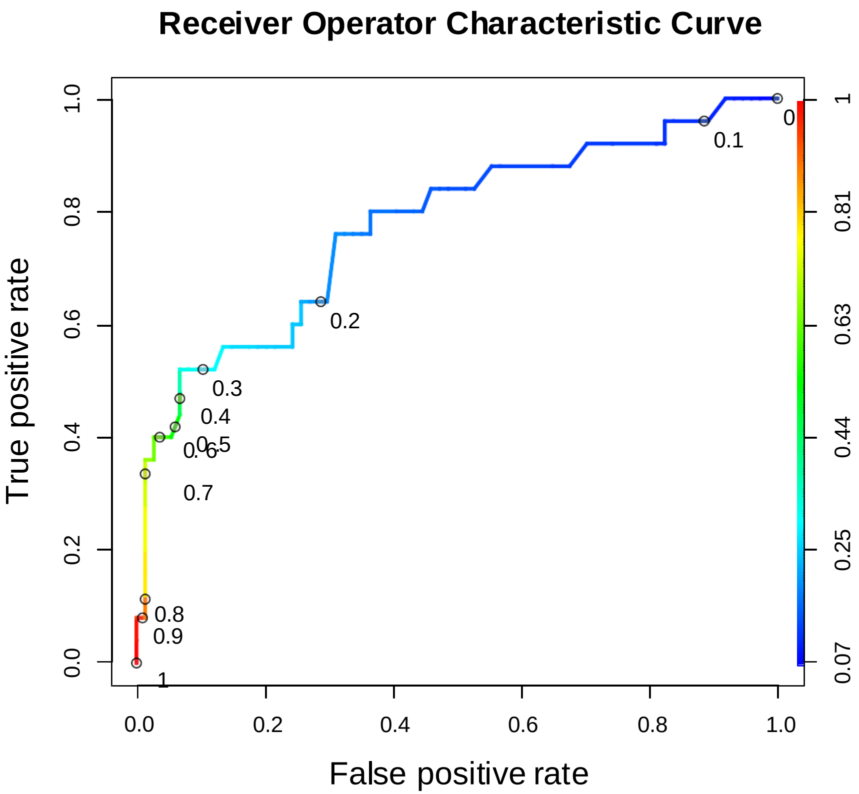
$$\text{False positive rate} = \frac{FP}{TN + FP}$$

Selecting a Threshold using ROC

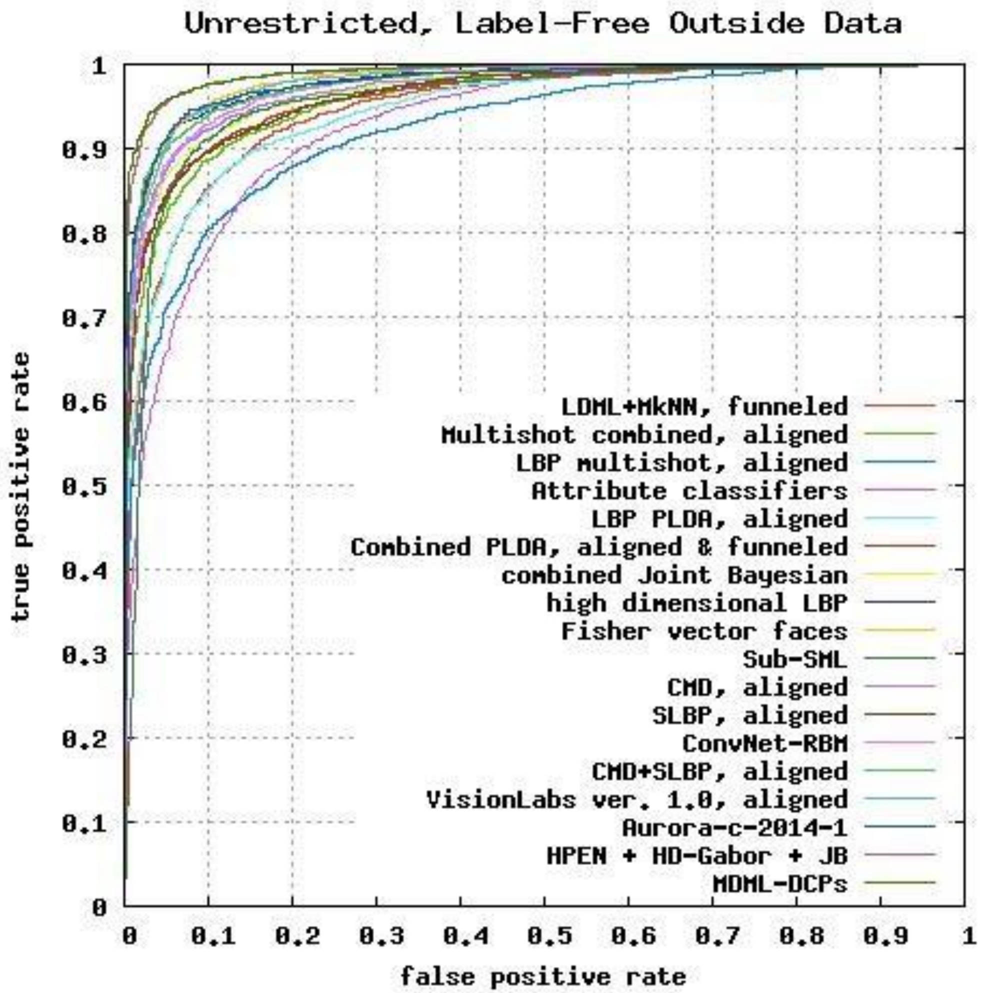
- Choose best threshold *t* for the best trade off
  - cost of failing to identify an object
  - cost of raising the false alarms

$$\text{True positive rate} = \frac{TP}{TP + FN}$$

$$\text{False positive rate} = \frac{FP}{TN + FP}$$



Performance Evaluation
ROC curve



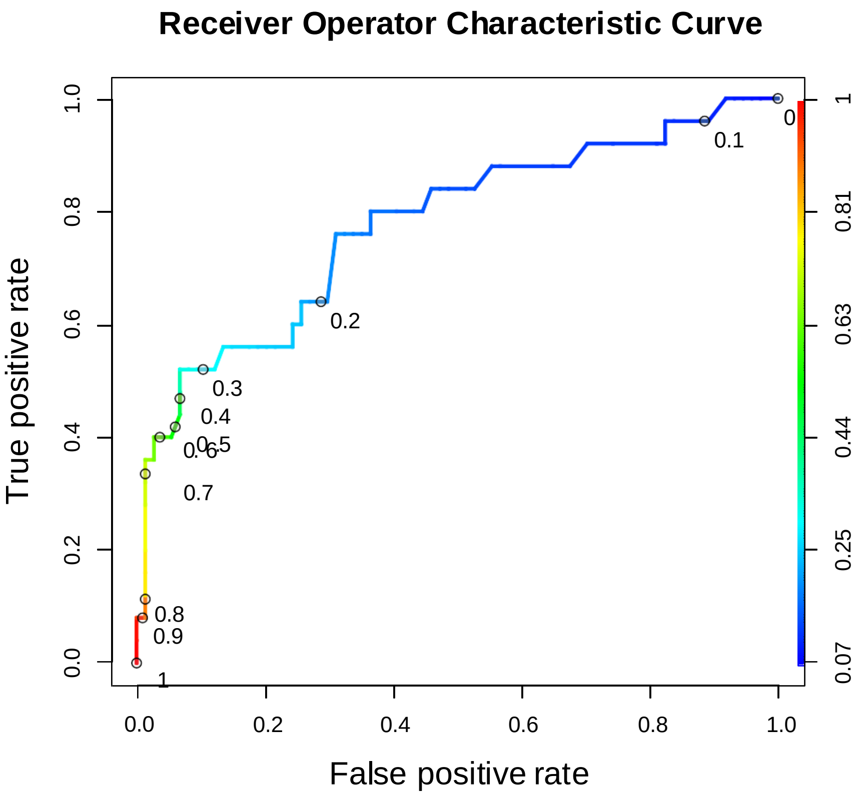
LFW Face verification

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{N} = \frac{FP}{TN + FP}$$

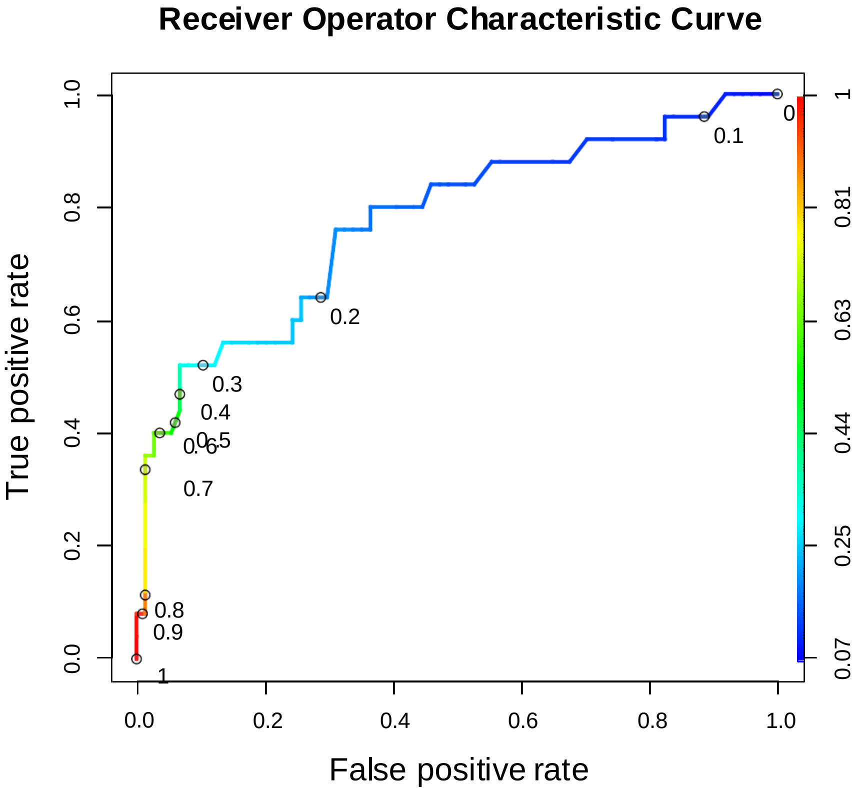
Performance across thresholds

- The **area under the ROC curve (AUROC)**
- Interpretation
  - Given a random positive and negative, proportion of the time you guess which is which correctly
- Less affected by sample balance than accuracy



Area Under the ROC Curve (AUROC)

- What is a good AUROC?
  - Maximum of 1
    (perfect prediction)
  - Minimum of 0.5
    (just guessing)

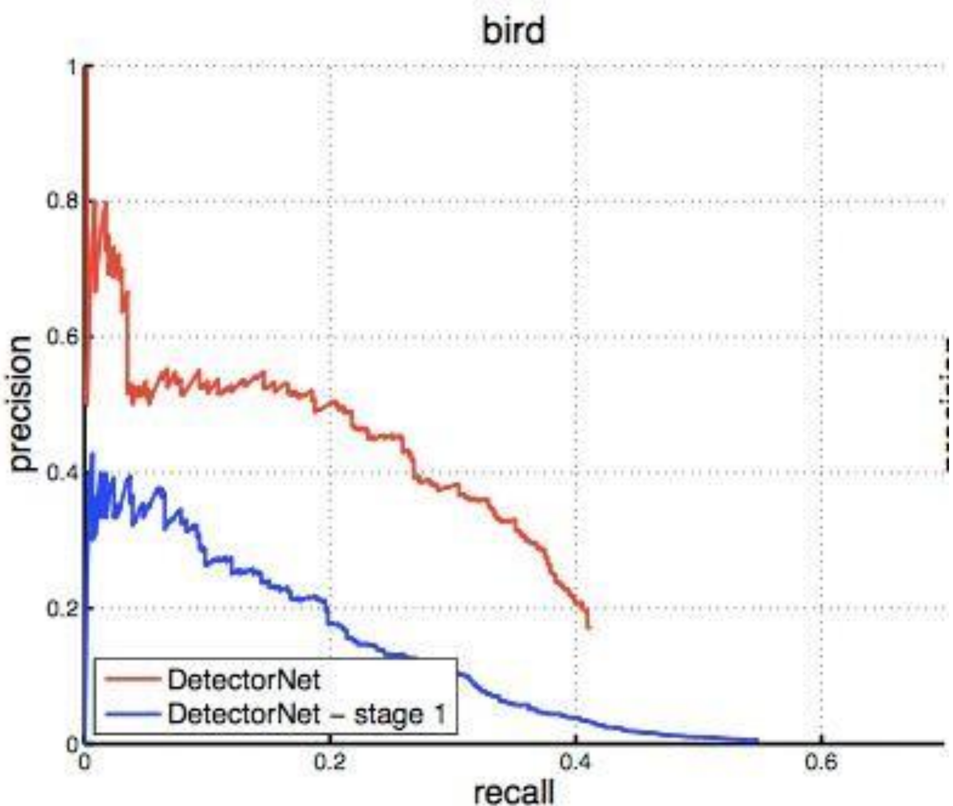


$$\text{True positive rate} = \frac{TP}{TP + FN}$$

$$\text{False positive rate} = \frac{FP}{TN + FP}$$

Performance Evaluation
Precision-recall curve

- Preferred for detection, where TN's are otherwise undefined



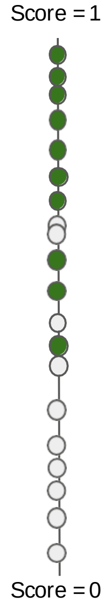
Szegedy, NIPS'13

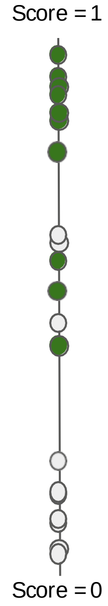
$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

Confidence

Model A





Model B

Two models scoring the same data set. Is one of them better than the other?

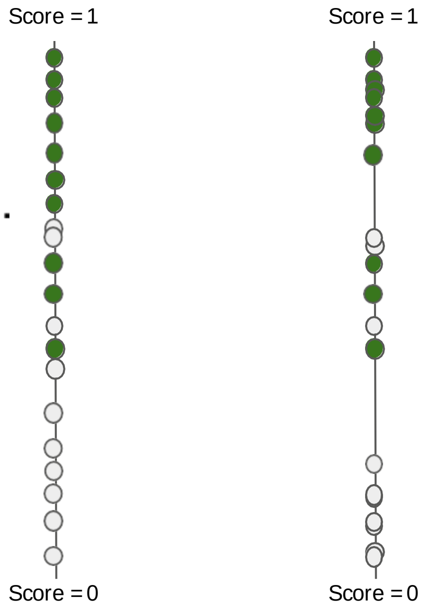
Log-Loss and Brier Score

- Same ranking, and therefore the same AUROC, AUPRC, accuracy!

$$\text{Log Loss} = \frac{1}{N}\sum_{i=1}^{N} -y_i \log \hat{y}_i - (1 - y_i) \log (1 - \hat{y}_i).$$

- Rewards confident correct answers, heavily penalizes confident wrong answers.
- One perfectly confident wrong prediction is fatal.
  -> Well-**calibrated** model
  - **Proper** scoring rule: Minimized at $\hat{y} = y$

$$\text{Brier Score} = \frac{1}{N}\sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$



Thank you

Acknowledges: some slides and material from Bernt Schiele, Mario Fritz, Michael Black, Bill Freeman, Fei-Fei, Justin Johnson, Serena Yeung, Yining Chen, Anand Avati, Andrew Ng

SAPIENZA
UNIVERSITÀ DI ROMA