Computer Vision

Spring 2006 15-385,-685

Instructor: S. Narasimhan

Wean 5403

T-R 3:00pm – 4:20pm

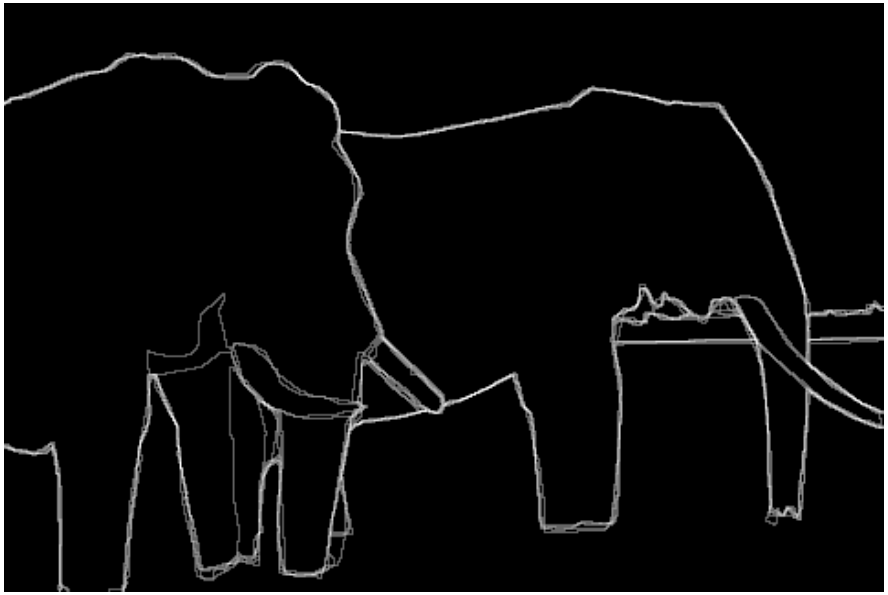
Boundary Detection: Hough Transform

Lecture #9

Reading: Computer Vision (Ballard and Brown): Chapter 4

"Use of the Hough Transform to detect lines and curves in pictures", Comm. ACM 15, 1, January 1972 (pgs 112-115)

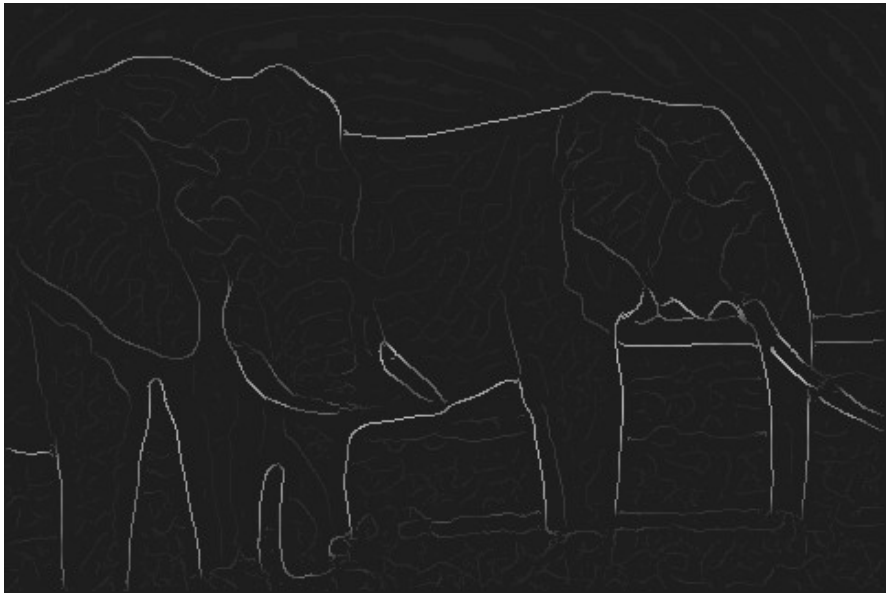
Boundaries of Objects



Marked by many users

http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/bench/html/images.html

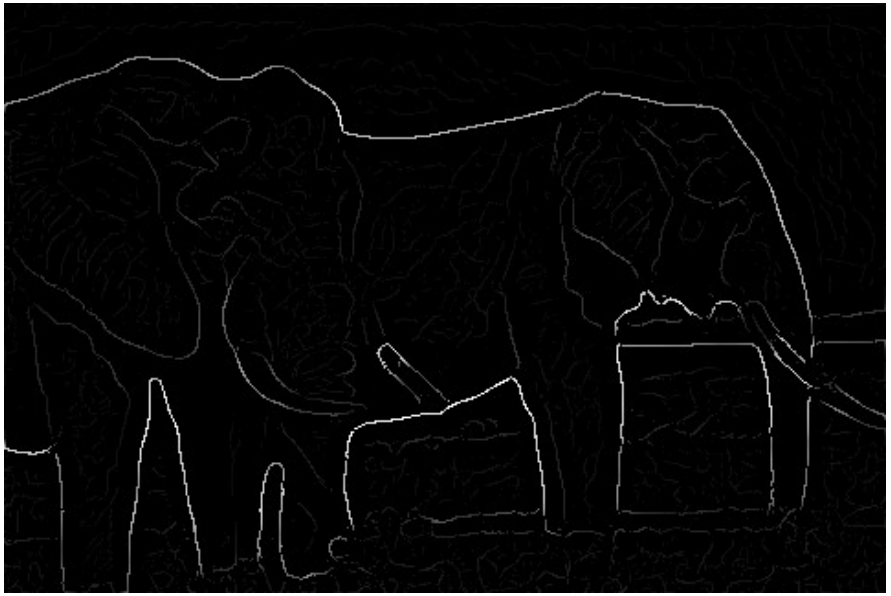
Boundaries of Objects from Edges



Brightness Gradient (Edge detection)

- Missing edge continuity, many spurious edges

Boundaries of Objects from Edges



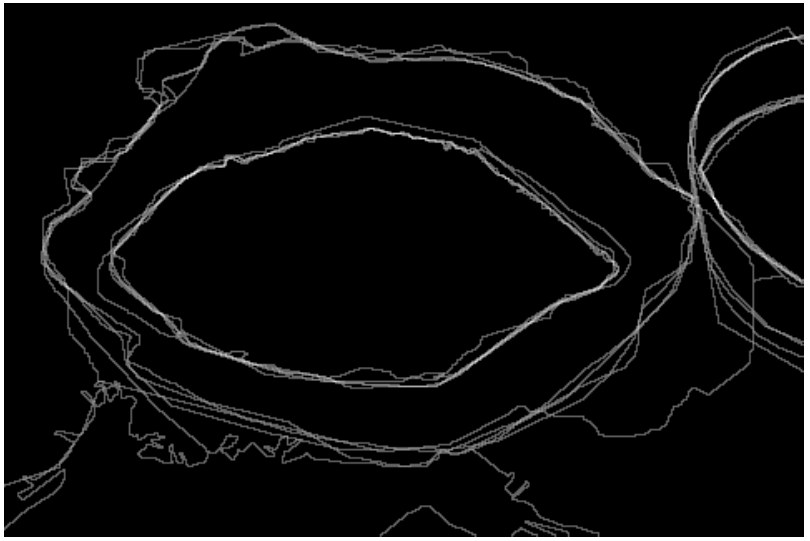
Multi-scale Brightness Gradient

- But, low strength edges may be very important

Boundaries of Objects from Edges

Image

Machine Edge Detection



Human Boundary Marking

Boundaries in Medical Imaging

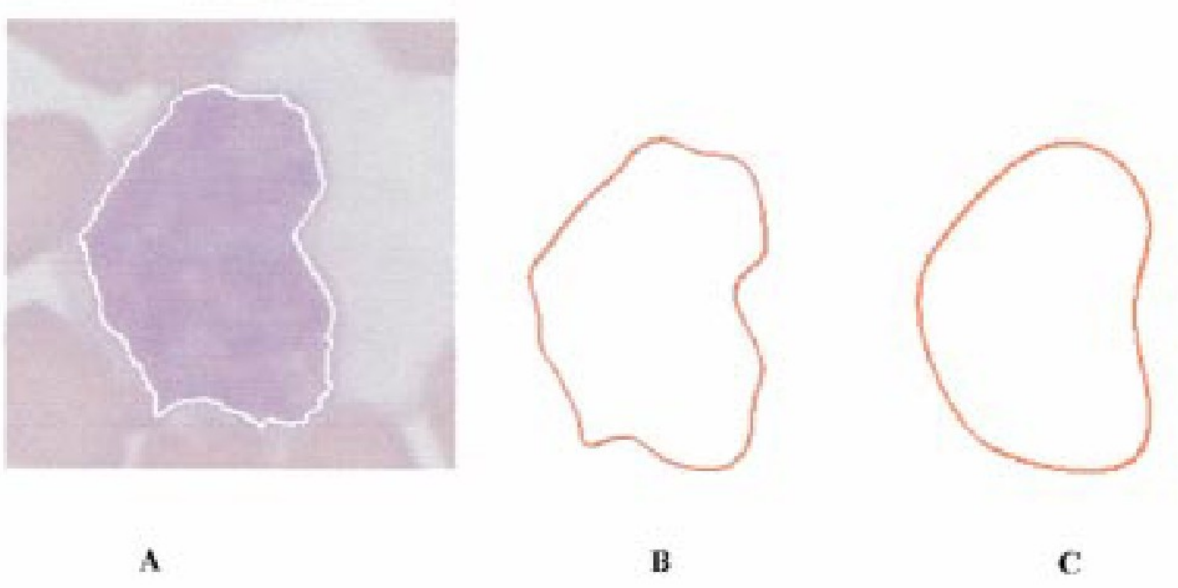
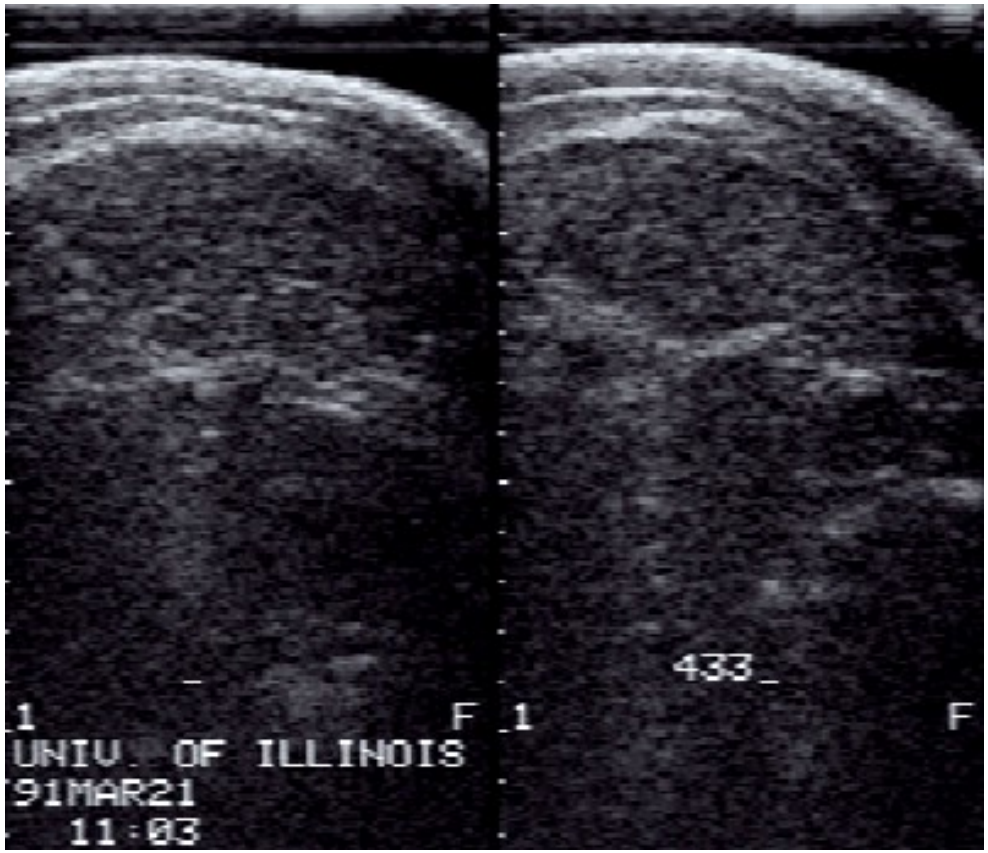


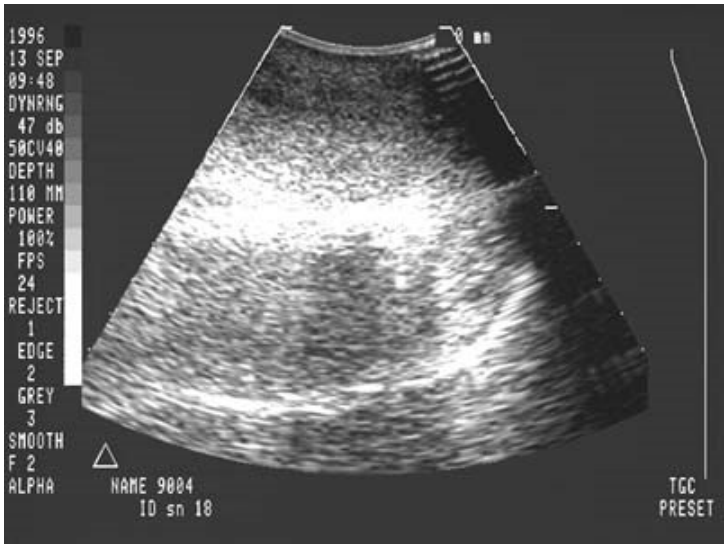
Fig. 2. Representation of a closed contour by elliptic Fourier descriptors. (a) Input. (b) Series truncated at 16 harmonics. (c) Series truncated to four harmonics.

Detection of cancerous regions.

[Foran, Comaniciu, Meer, Goodell, 00]

Boundaries in Ultrasound Images





Hard to detect in the presence of large amount of speckle noise

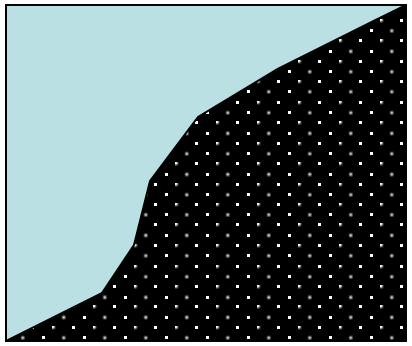
Boundaries of Objects

Sometimes hard even for humans!

Topics

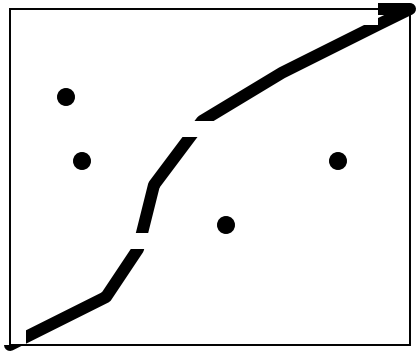
- Preprocessing Edge Images
- Edge Tracking Methods
- Fitting Lines and Curves to Edges
- The Hough Transform

Preprocessing Edge Images



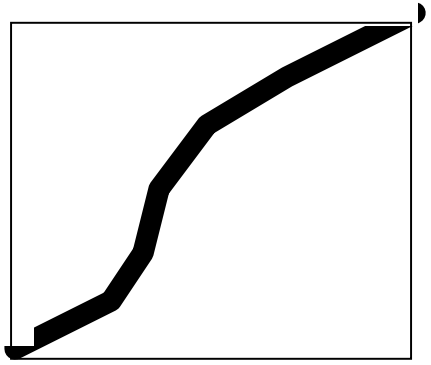
Image

Edge detection
and Thresholding

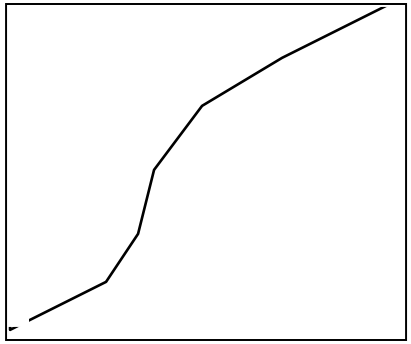


Noisy edge image
Incomplete boundaries

Shrink and Expand



Thinning



Edge Tracking Methods

Adjusting a priori Boundaries:

Given: Approximate Location of Boundary

Task: Find Accurate Location of Boundary

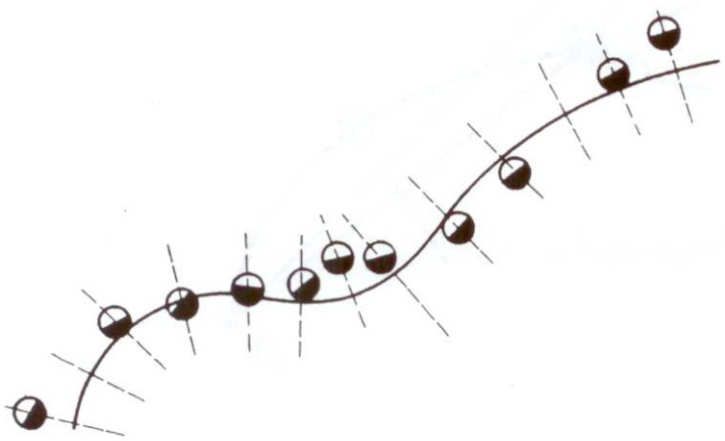


Fig. 4.2 Search orientations from an approximate boundary location.

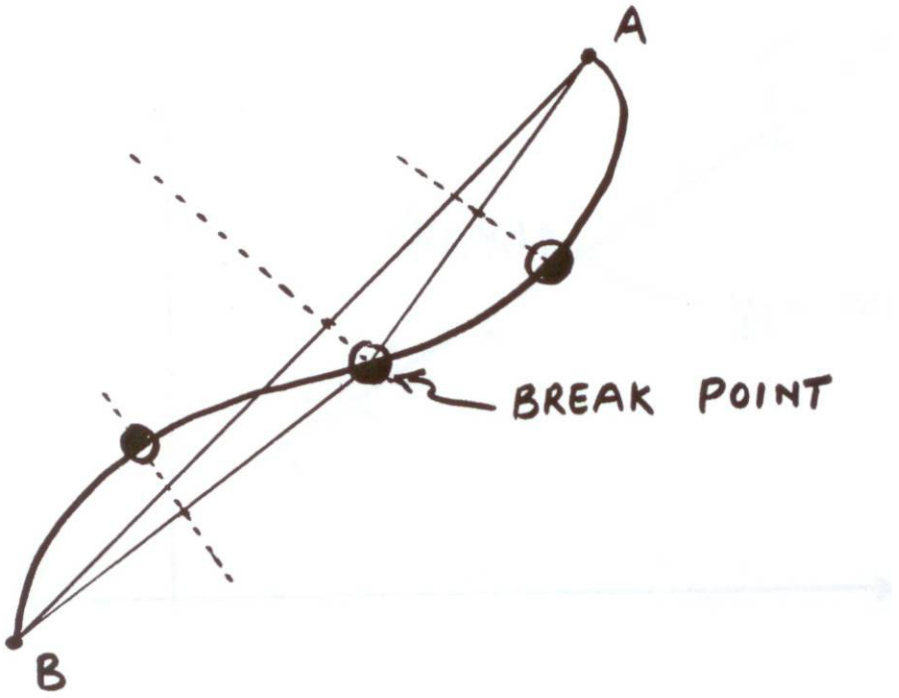
- Search for STRONG EDGES along normals to approximate boundary.
- Fit curve (eg., polynomials) to strong edges.

Edge Tracking Methods

Divide and Conquer:

Given: Boundary lies between points A and B

Task: Find Boundary

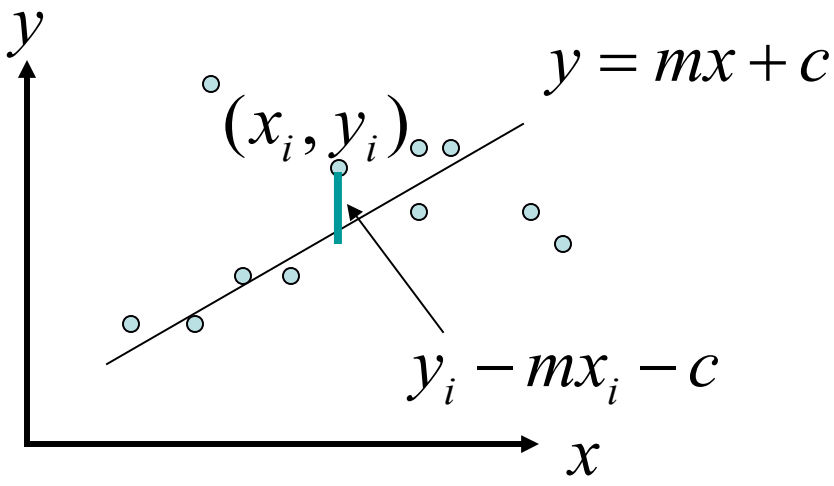


- Connect A and B with Line
- Find strongest edge along line bisec
- Use edge point as break point
- Repeat

Fitting Lines to Edges (Least Squares)

Given: Many (x_i, y_i) pairs

Find: Parameters (m, c)



Minimize: Average square distance:

$$E = \sum_i \frac{(y_i - mx_i - c)^2}{N}$$

Using:

$$\frac{\partial E}{\partial m} = 0 \quad \& \quad \frac{\partial E}{\partial c} = 0$$

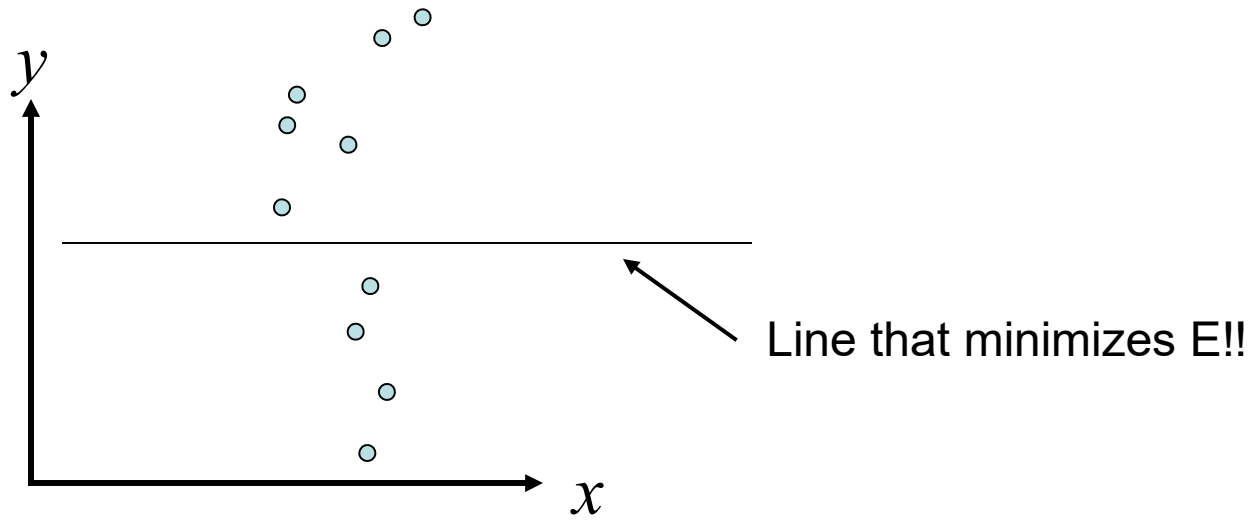
$$c = \bar{y} - m\,\bar{x}$$

$$m = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

Note:

$$\bar{y} = \frac{\sum_i y_i}{N} \qquad \bar{x} = \frac{\sum_i x_i}{N}$$

Problem with Parameterization

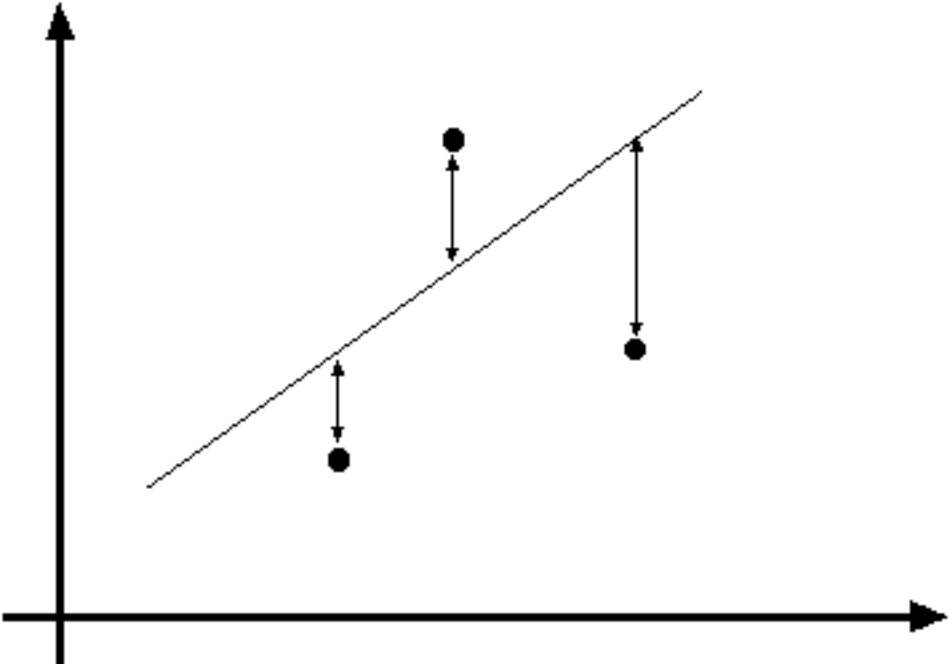


Solution: Use a different parameterization

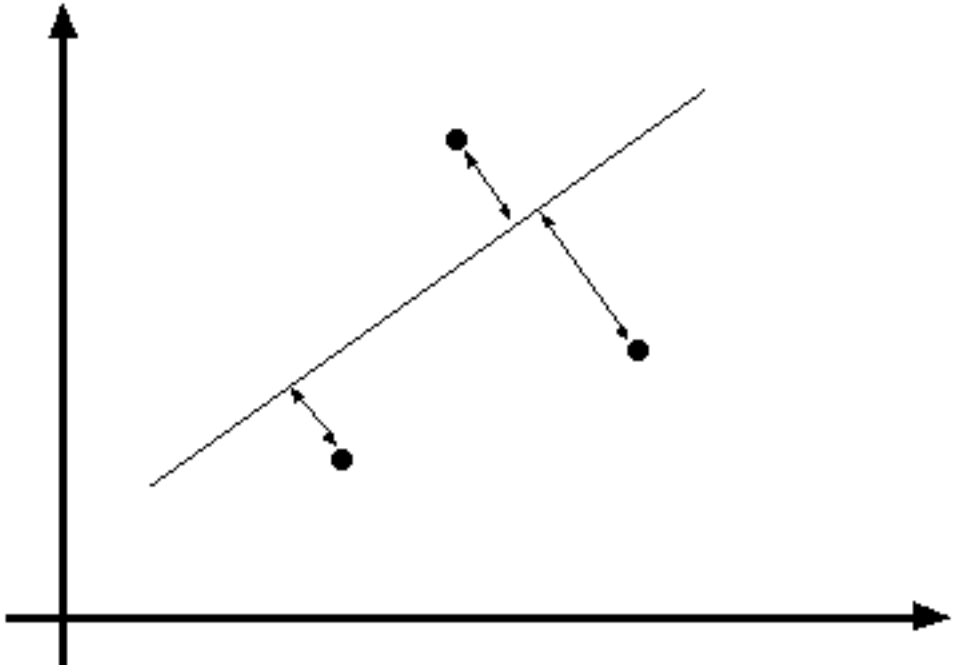
(same as the one we used in computing Minimum Moment of Inertia)

$$E = \frac{1}{N}\sum_i (\rho - x_i \cos\theta + y_i \sin\theta)^2$$

Note: Error E must be formulated carefully!



Line fitting can be max. likelihood - but choice of model is important

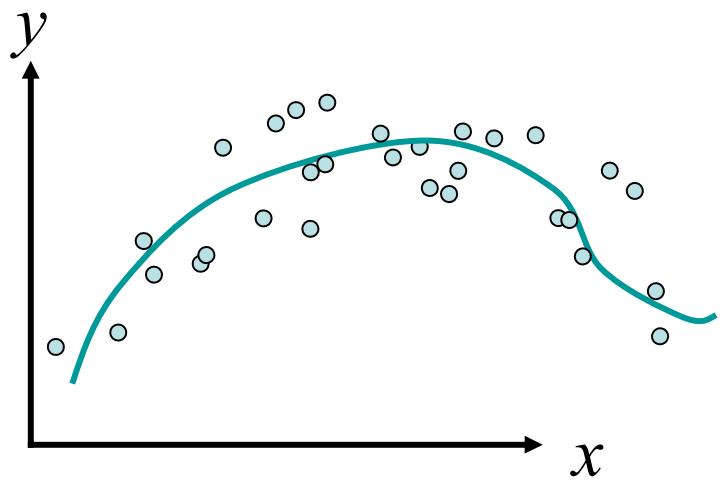


Curve Fitting

Find Polynomial:

$$y = f(x) = ax^3 + bx^2 + cx + d$$

that best fits the given points (x_i, y_i)



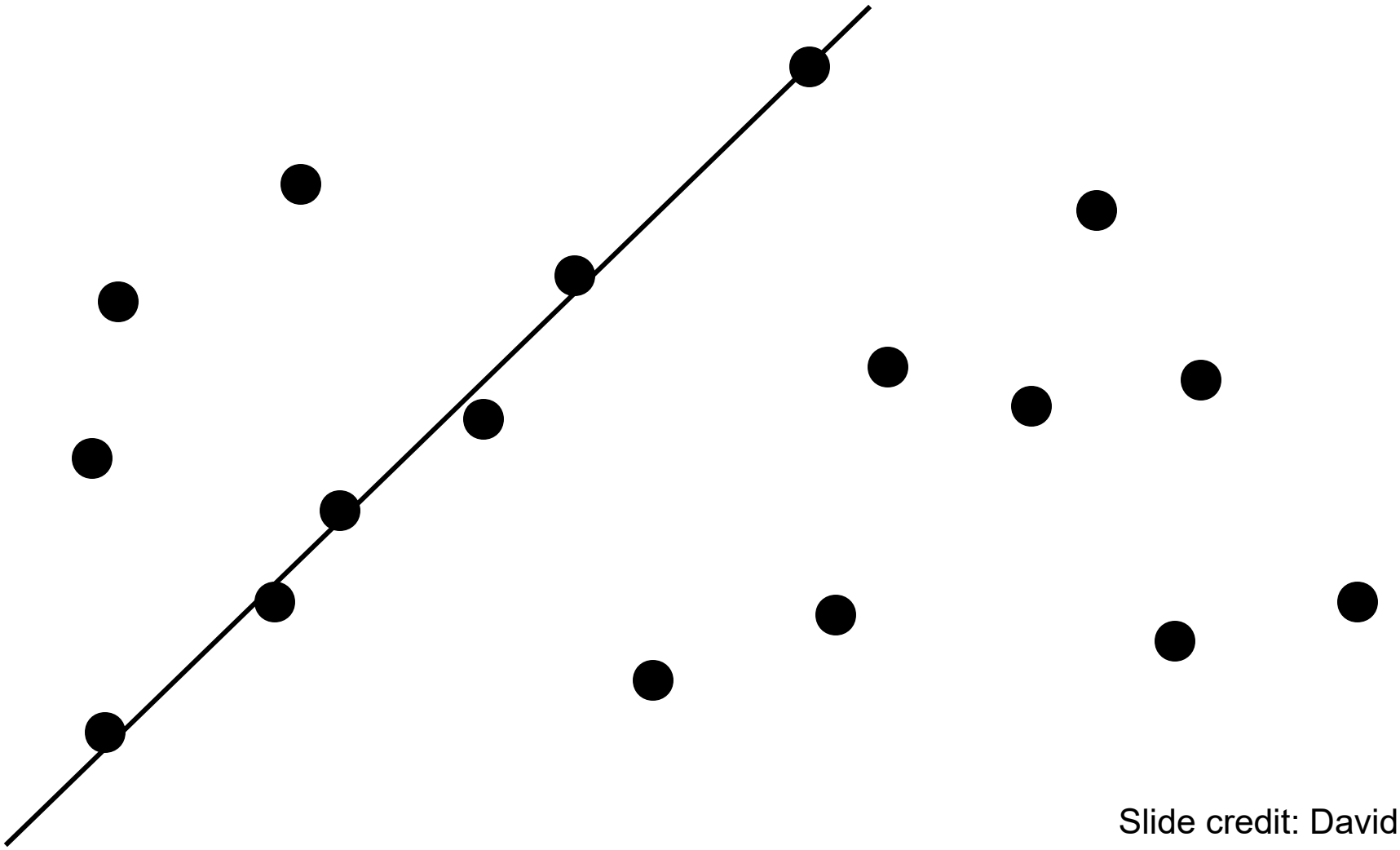
Minimize:

$$\frac{1}{N}\sum_i [y_i - (ax_i^3 + bx_i^2 + cx_i + d)]^2$$

Using: $\frac{\partial E}{\partial a} = 0$, $\frac{\partial E}{\partial b} = 0$, $\frac{\partial E}{\partial c} = 0$, $\frac{\partial E}{\partial d} = 0$

Note: $f(x)$ is LINEAR in the parameters (a, b, c, d)

Line Grouping Problem



Slide credit: David Jacobs

This is difficult because of:

- Extraneous data: clutter or multiple models
  - We do not know what is part of the model?
  - Can we pull out models with a few parts from much larger amounts of background clutter?
- Missing data: only some parts of model are present
- Noise

- **Cost:**
  - It is not feasible to check all combinations of features by fitting a model to each possible subset

Hough Transform

- Elegant method for direct object recognition
    - Edges need not be connected
    - Complete object need not be visible
    - Key Idea: Edges VOTE for the possible model

Image and Parameter Spaces

Equation of Line: $y = mx + c$

Find: (m, c)

Consider point: (x_i, y_i)

$$y_i = mx_i + c \quad or \quad c = -x_i m + y_i$$

Parameter space also called Hough Space

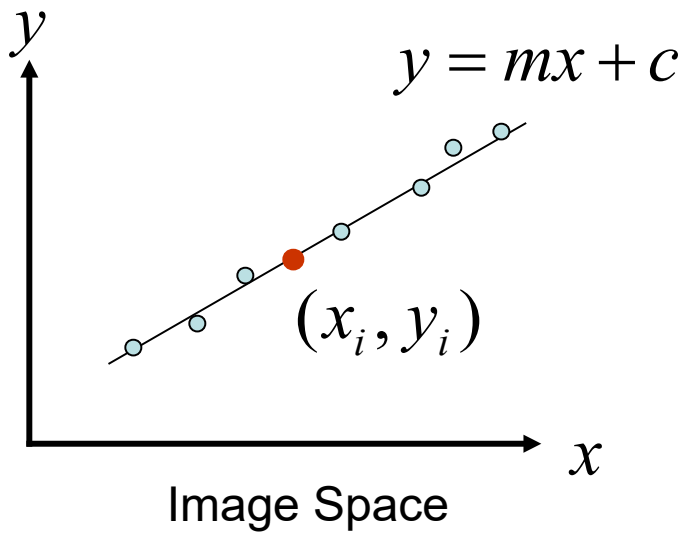
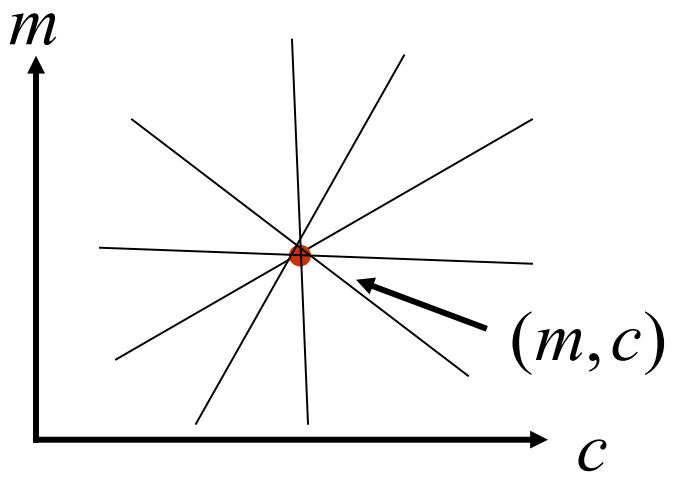


Image Space



Parameter Space

Line Detection by Hough Transform

Algorithm:

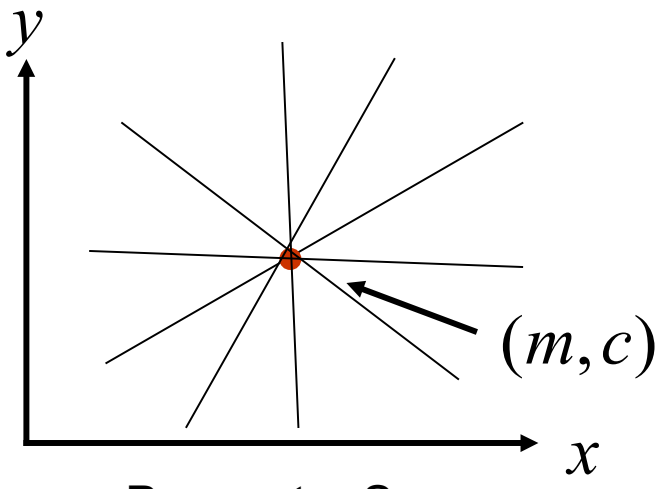
- Quantize Parameter Space (m,c)
- Create Accumulator Array $A(m,c)$
- Set $A(m,c)=0 \quad \forall m,c$
- For each image edge (x_i, y_i) increment:

$$A(m,c) = A(m,c)+1$$

- If (m,c) lies on the line:

$$c = -x_i m + y_i$$

- Find local maxima in $A(m,c)$



Parameter Space

$A(m,c)$

	1						1		
		1				1			
			1		1				
				2					
			1		1				
		1				1			
	1						1		

Better Parameterization

NOTE: $-\infty \le m \le \infty$

Large Accumulator

More memory and computations

Improvement: (Finite Accumulator Array Size)

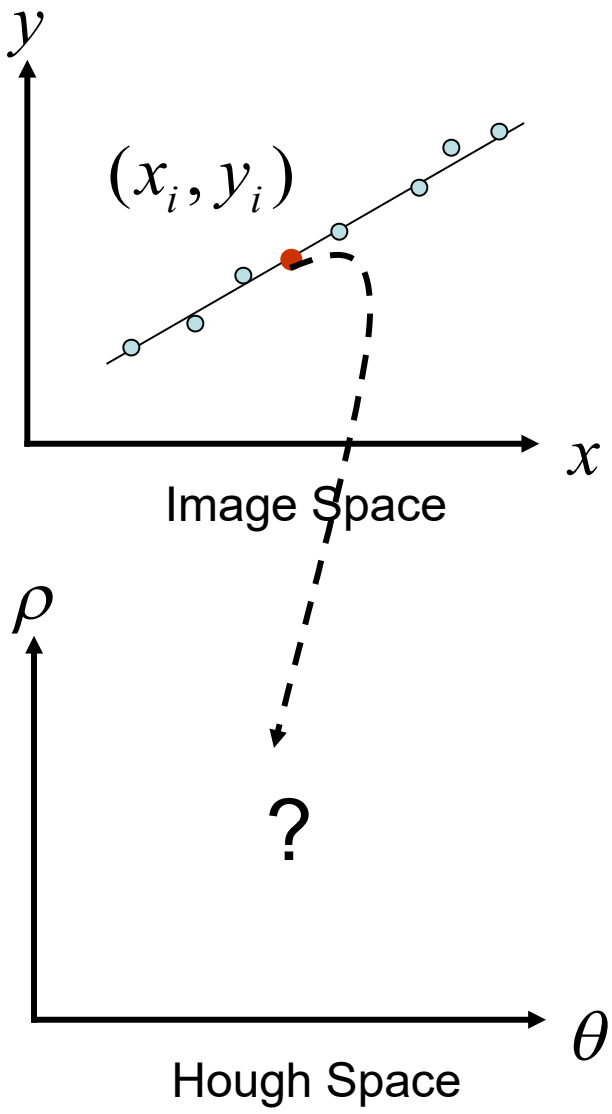
Line equation: $\rho = -x\cos\theta + y\sin\theta$

Here $0 \le \theta \le 2\pi$

$0 \le \rho \le \rho_{\max}$

Given points (x_i, y_i) find (ρ, θ)

Hough Space Sinusoid



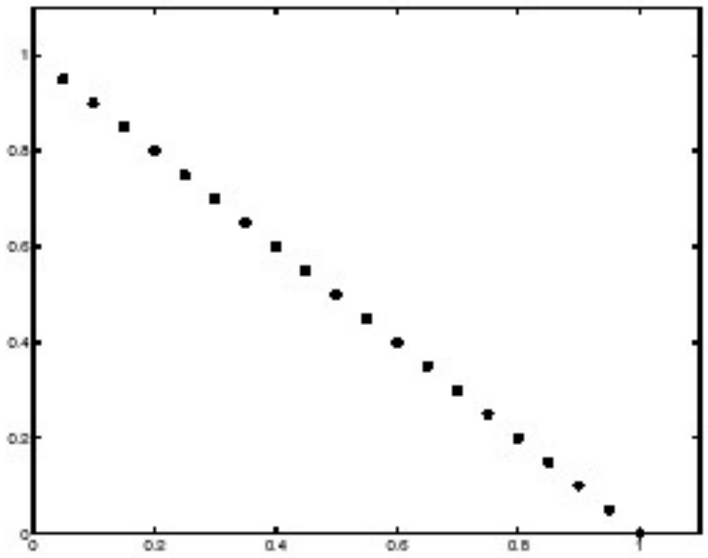
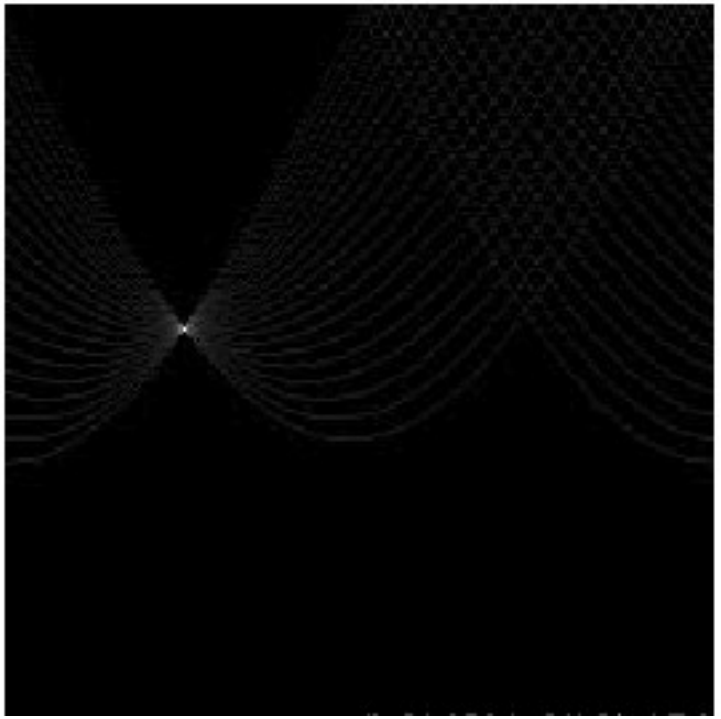
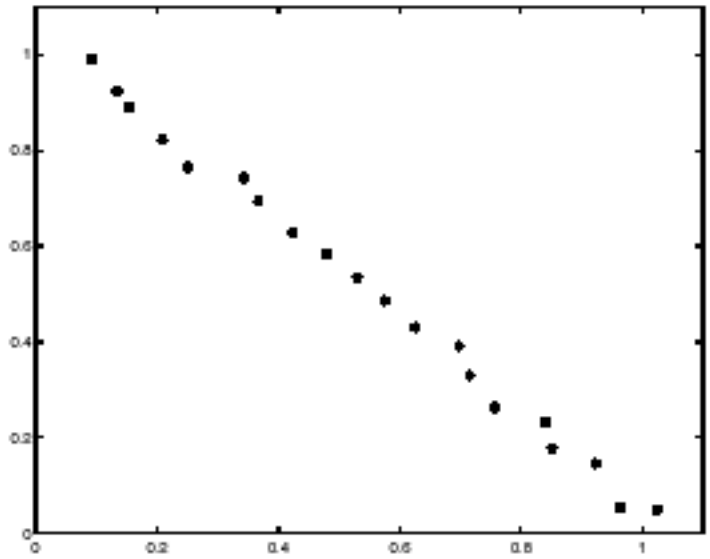


Image space



Votes

Horizontal axis is θ, vertical is rho.



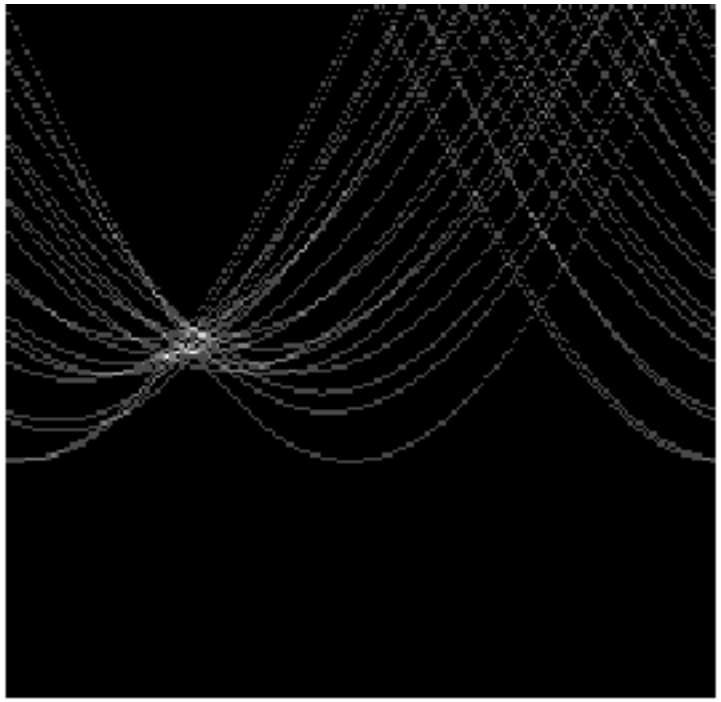
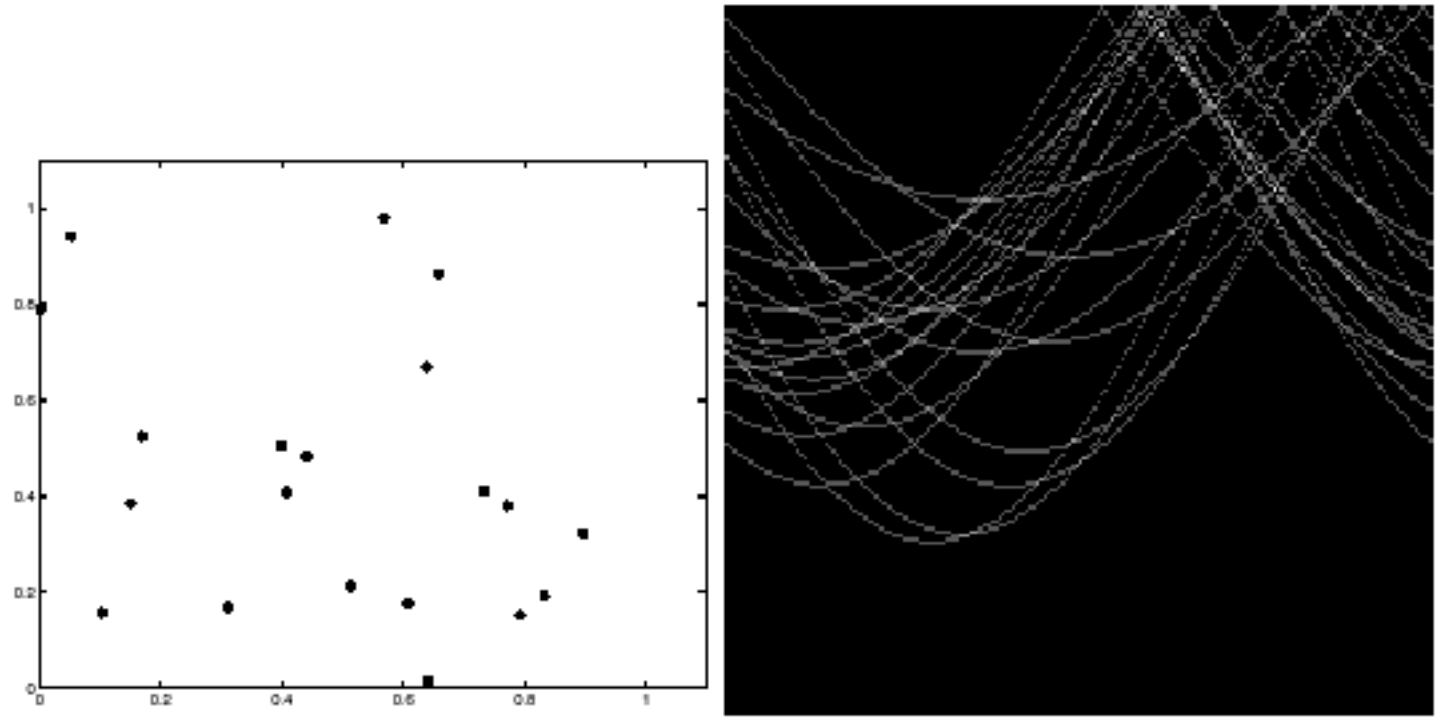


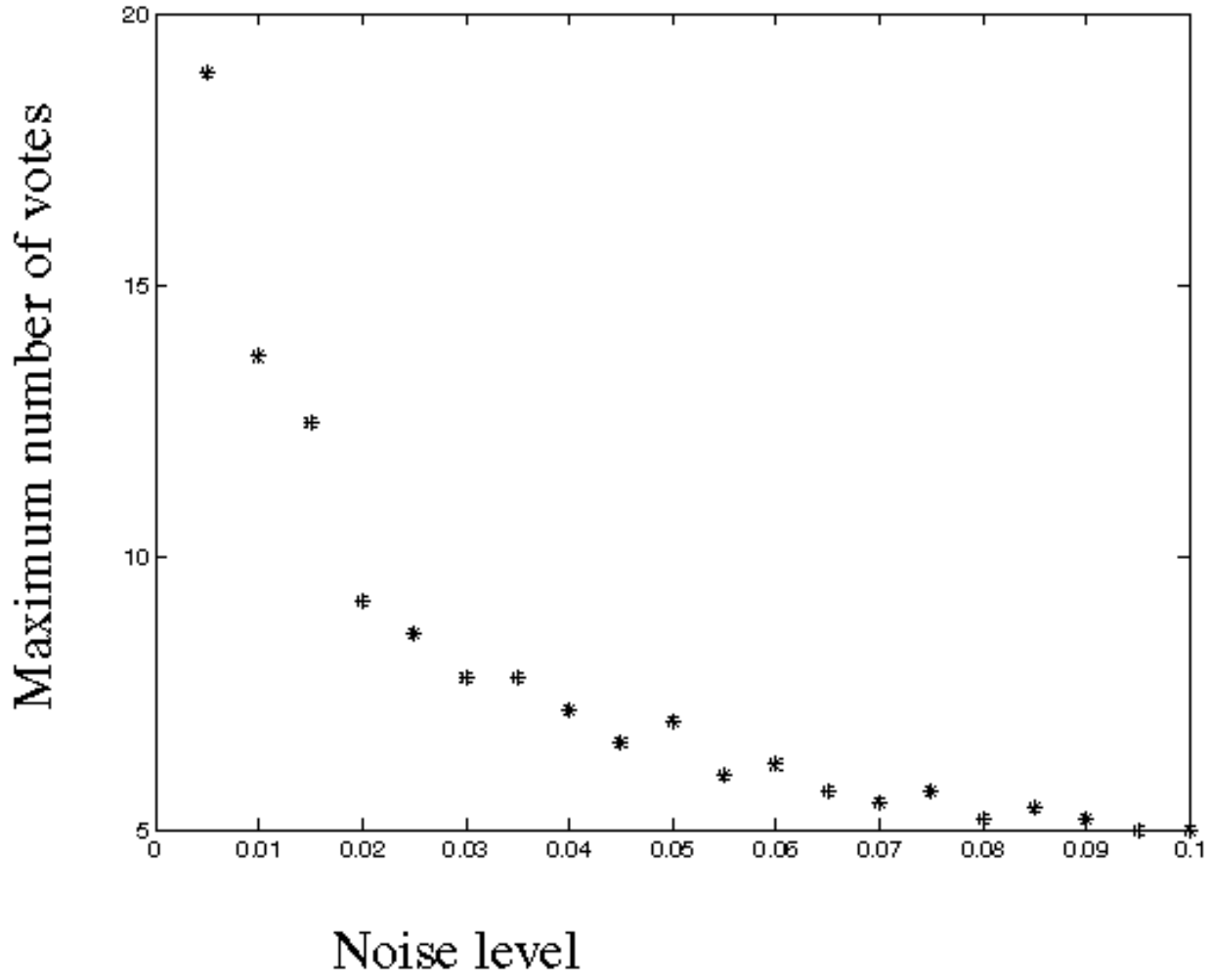
Image space

votes

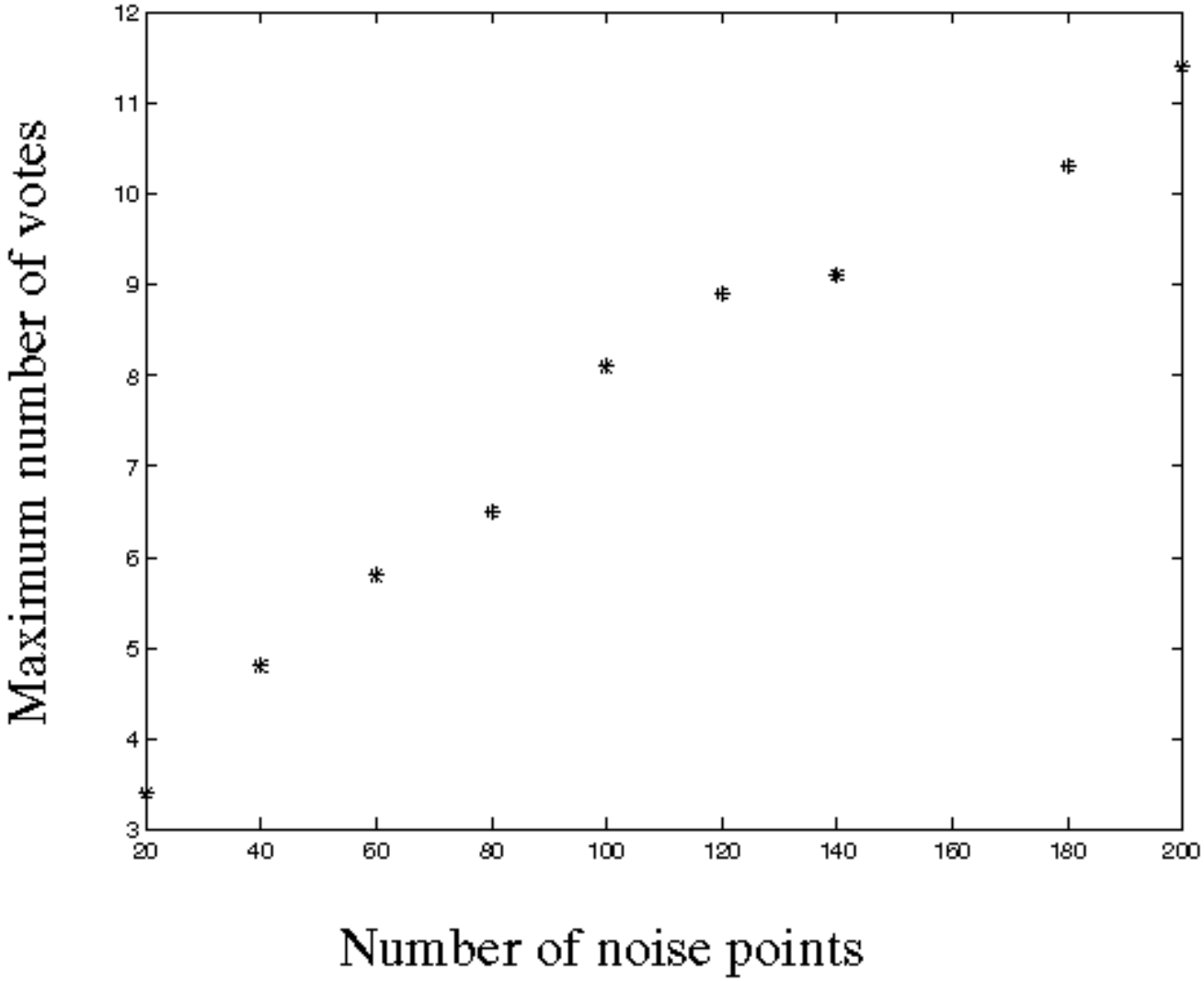

0
0.2
0.4
0.6
0.8
1
0
0.2
0.4
0.6
0.8
1

Mechanics of the Hough transform

- Difficulties
  - how big should the cells be? (too big, and we merge quite different lines; too small, and noise causes lines to be missed)
- How many lines?
  - Count the peaks in the Hough array
  - Treat adjacent peaks as a single peak
- Which points belong to each line?
  - Search for points close to the line
  - Solve again for line and iterate



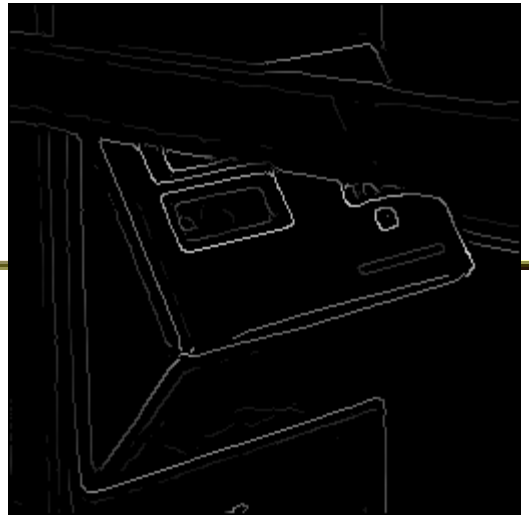
Fewer votes land in a single bin when noise increases.



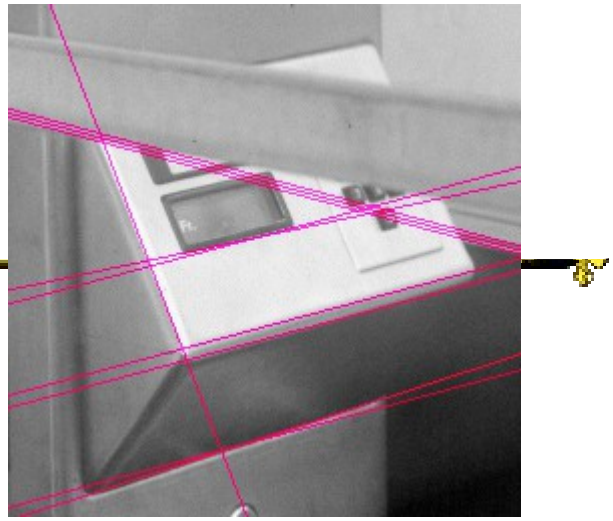
Adding more clutter increases number of bins with false peaks.

Real World Example

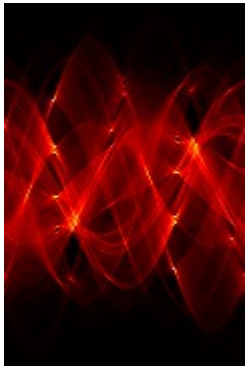
Original



Edge Detection



Found Lines



Parameter Space

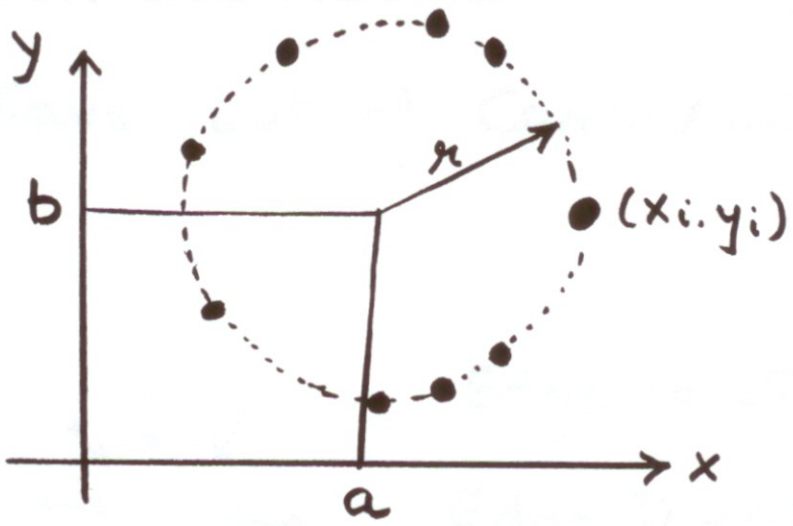
Finding Circles by Hough Transform

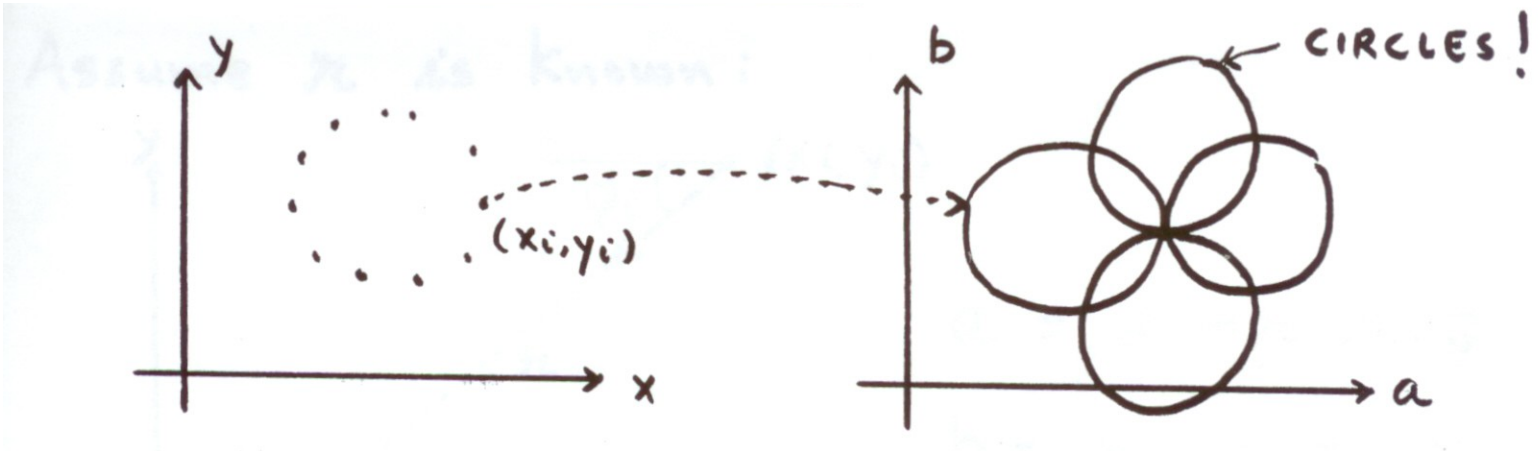
Equation of Circle:

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

If radius is known: (2D Hough Space)

Accumulator Array $A(a,b)$

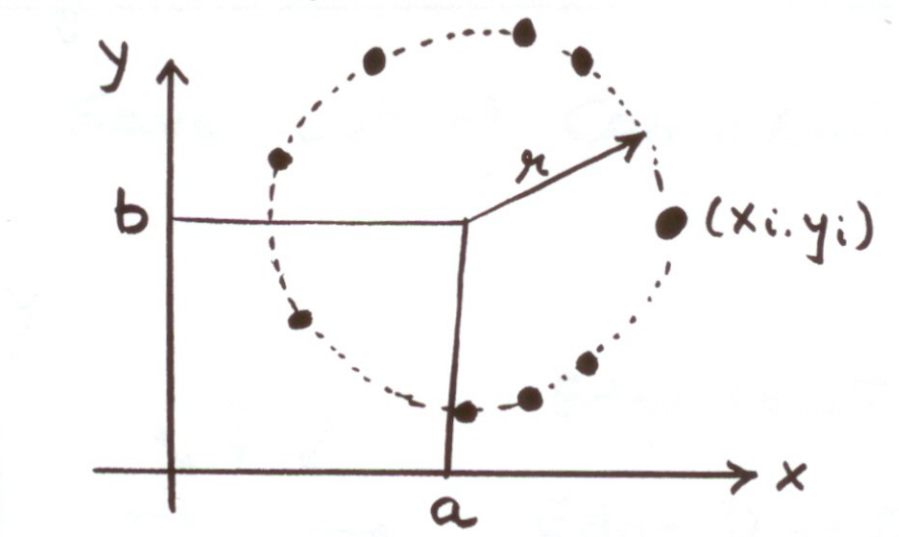




Finding Circles by Hough Transform

Equation of Circle:

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$



If radius is not known: 3D Hough Space!

Use Accumulator array $A(a, b, r)$

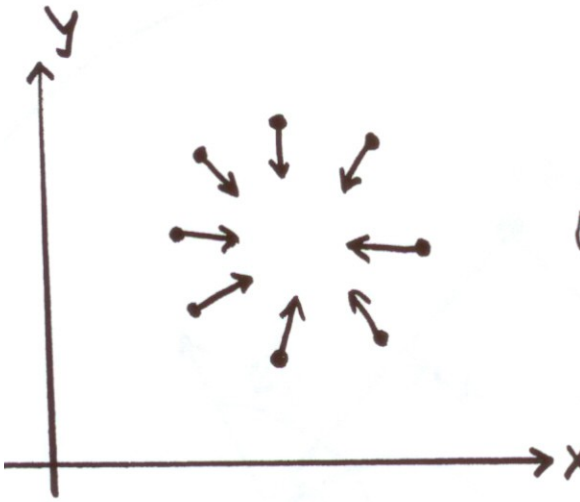
What is the surface in the hough space?

Using Gradient Information

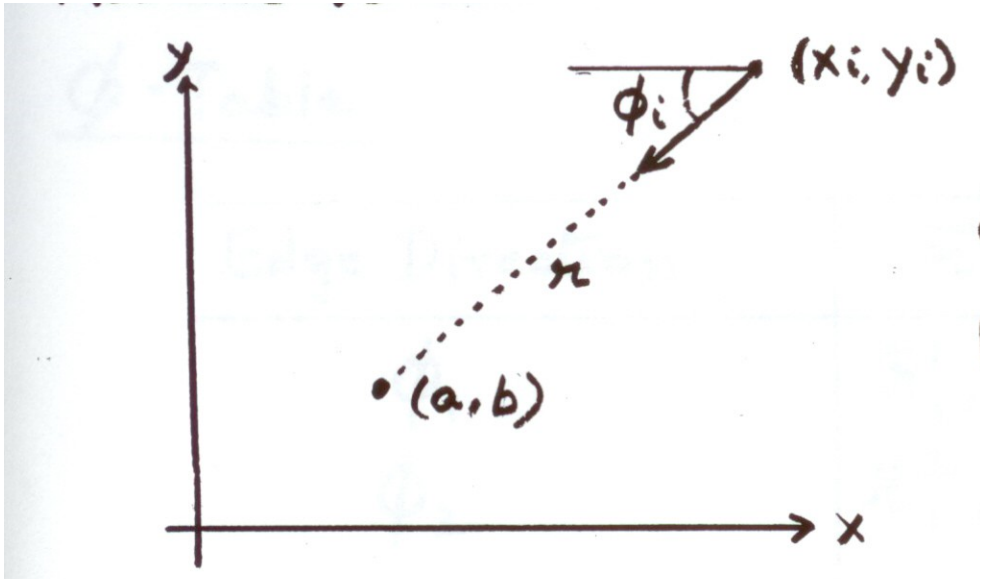
- Gradient information can save lot of computation:

Edge Location (x_i, y_i)

Edge Direction ϕ_i



Assume radius is known:



$$a = x - r\cos\phi$$

$$b = y - r\sin\phi$$

Need to increment only one point in Accumulator!!

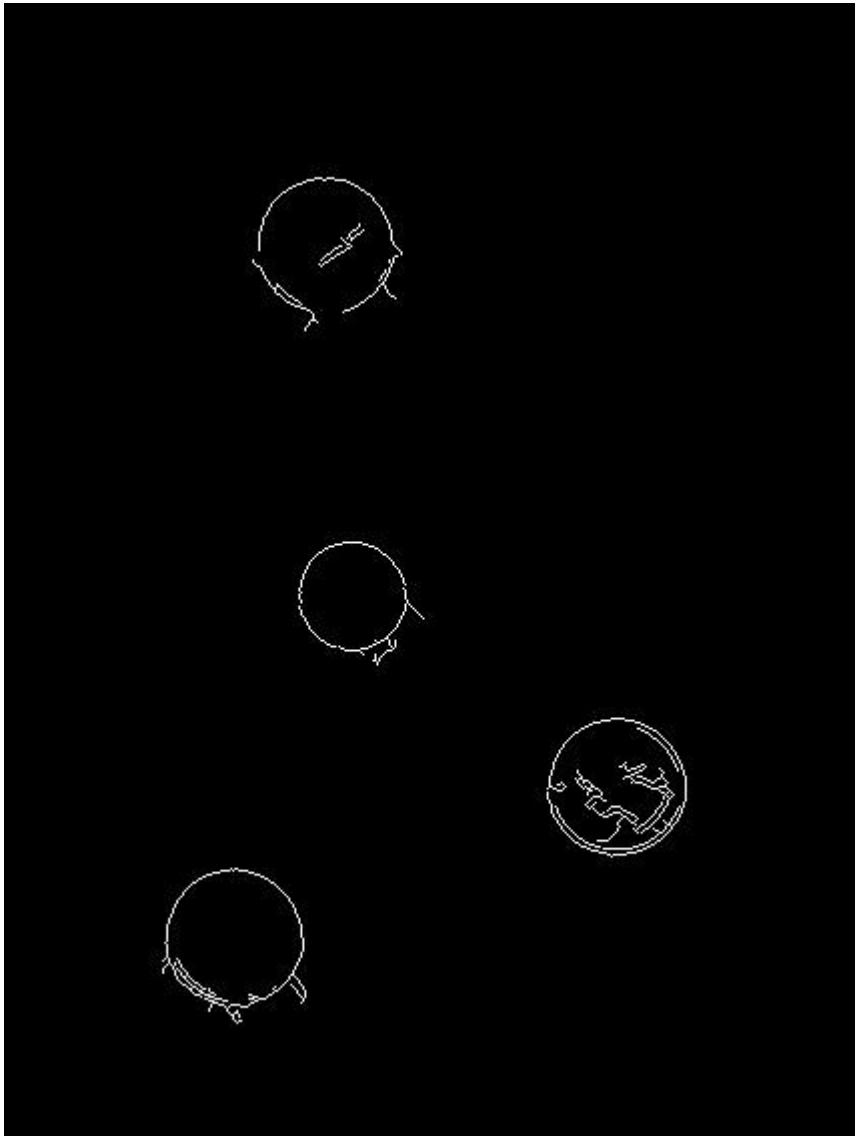
Real World Circle Examples

Crosshair indicates results of Hough transform, bounding box found via motion differencing.

Finding Coins

Original

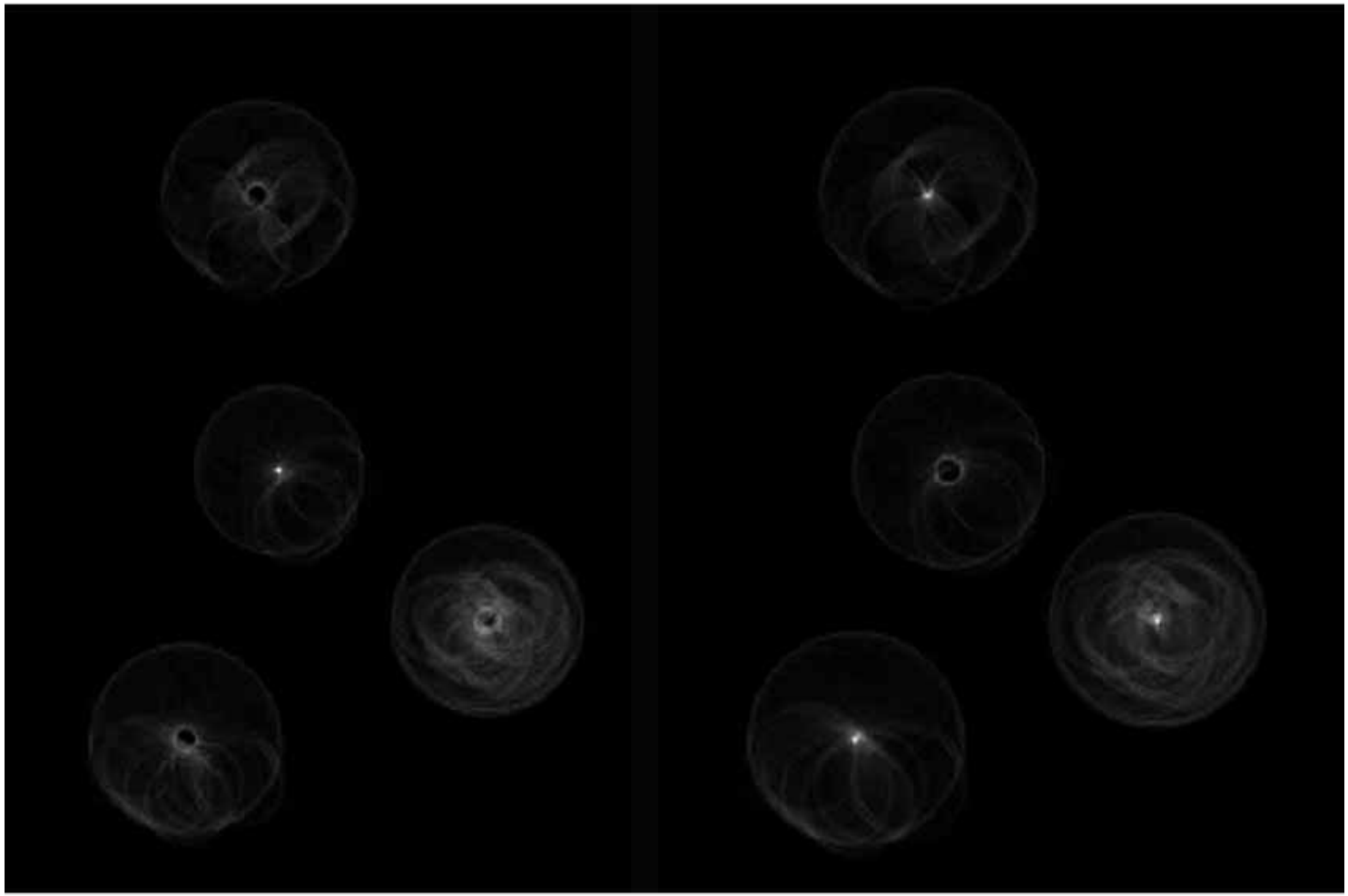
Edges (note noise)



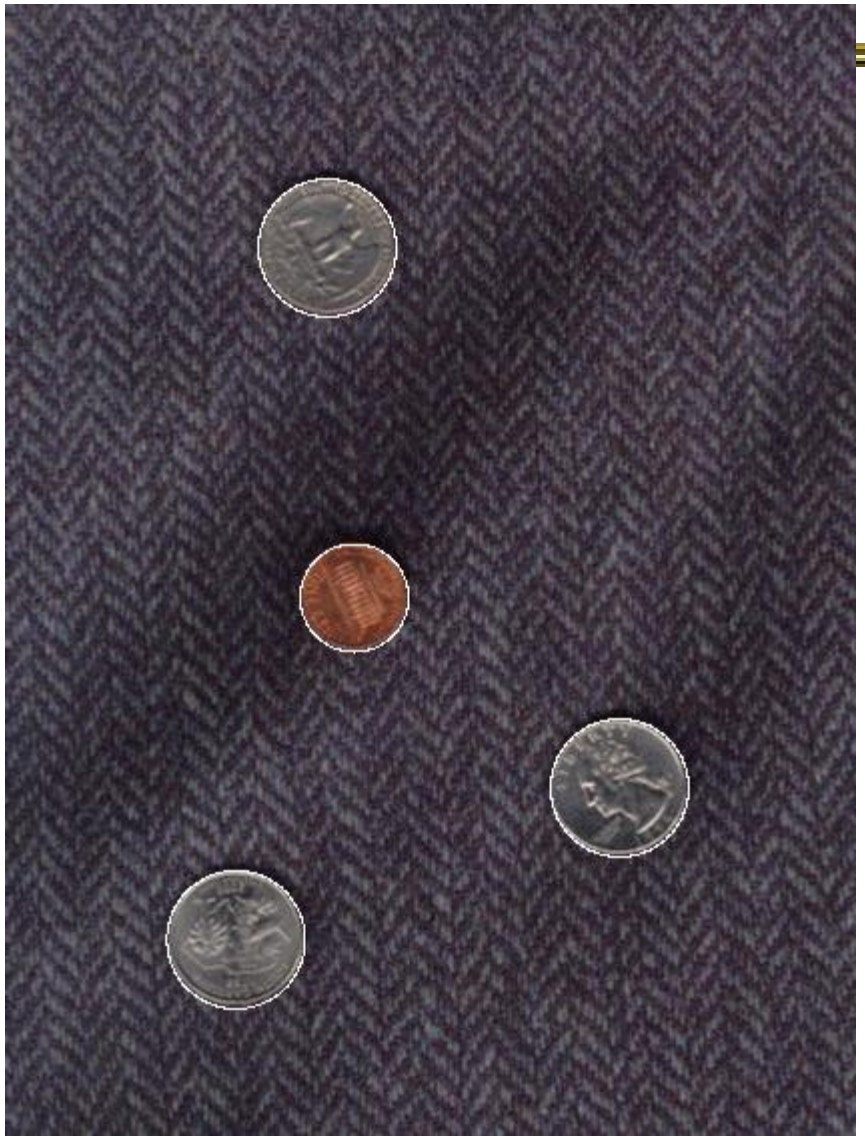
Finding Coins (Continued)

Penn

Quarters



Finding Coins (Continued)

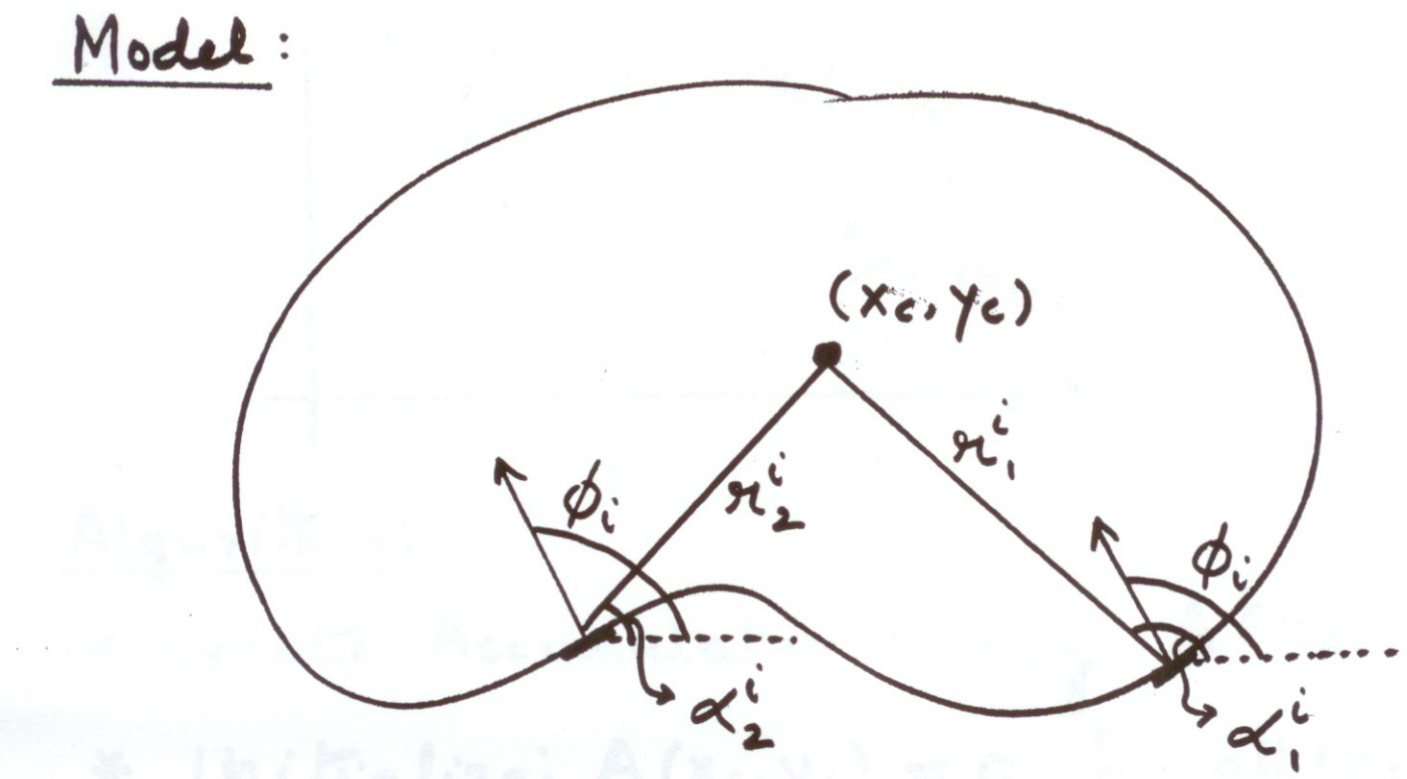


Note that because the quarters and penny are different sizes, a different Hough transform (with separate accumulators) was used for each circle size.

Coin finding sample images from: Vivek Kwatra

Generalized Hough Transform

- Model Shape NOT described by equation



Generalized Hough Transform

- Model Shape NOT described by equation

ϕ-Table

Edge Direction	$\bar{r} = (r, \alpha)$
ϕ_1	$\bar{r}_1^1, \bar{r}_2^1, \bar{r}_3^1$
ϕ_2	$\bar{r}_1^2, \bar{r}_2^2$
ϕ_i	$\bar{r}_1^i \vdots \bar{r}_2^i$
ϕ_n	$\bar{r}_1^n, \bar{r}_2^n$

Generalized Hough Transform

Find Object Center (x_c, y_c) given edges (x_i, y_i, ϕ_i)

Create Accumulator Array $A(x_c, y_c)$

Initialize: $A(x_c, y_c) = 0 \quad \forall (x_c, y_c)$

For each edge point (x_i, y_i, ϕ_i)

For each entry $\overline{r}_k^i$ in table, compute:

$$x_c = x_i + r_k^i \cos \alpha_k^i$$

$$y_c = y_i + r_k^i \sin \alpha_k^i$$

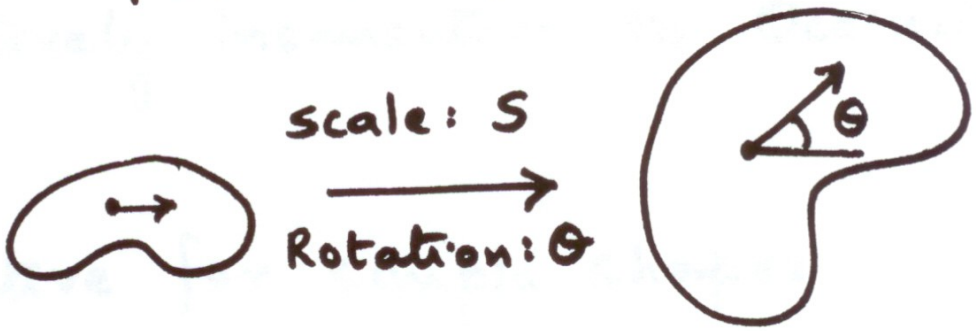
Increment Accumulator: $A(x_c, y_c) = A(x_c, y_c) + 1$

Find Local Maxima in $A(x_c, y_c)$

Scale & Rotation:

Use Accumulator Array:

$$A[x_c, y_c, S, \theta]$$



Use:

$$x_c = x_i + r^i_k S \cos(\alpha^i_k + \theta)$$

$$y_c = y_i + r^i_k S \sin(\alpha^i_k + \theta)$$

$$A(x_c, y_c, S, \theta) = A(x_c, y_c, S, \theta) + 1 .$$

Hough Transform: Comments

- Works on Disconnected Edges
- Relatively insensitive to occlusion
- Effective for simple shapes (lines, circles, etc)
- Trade-off between work in Image Space and Parameter Space
- Handling inaccurate edge locations:
    - Increment Patch in Accumulator rather than a single point

Next Class

- Lightness and Retinex.
- Reading: Horn, Chapter 9.
- Research webpages of Edward Adelson (MIT).
- Google for illusions.

Object recognition

Recognition

The “Margaret Thatcher Illusion”, by Peter Thompson

Recognition

The “Margaret Thatcher Illusion”, by Peter Thompson

Readings

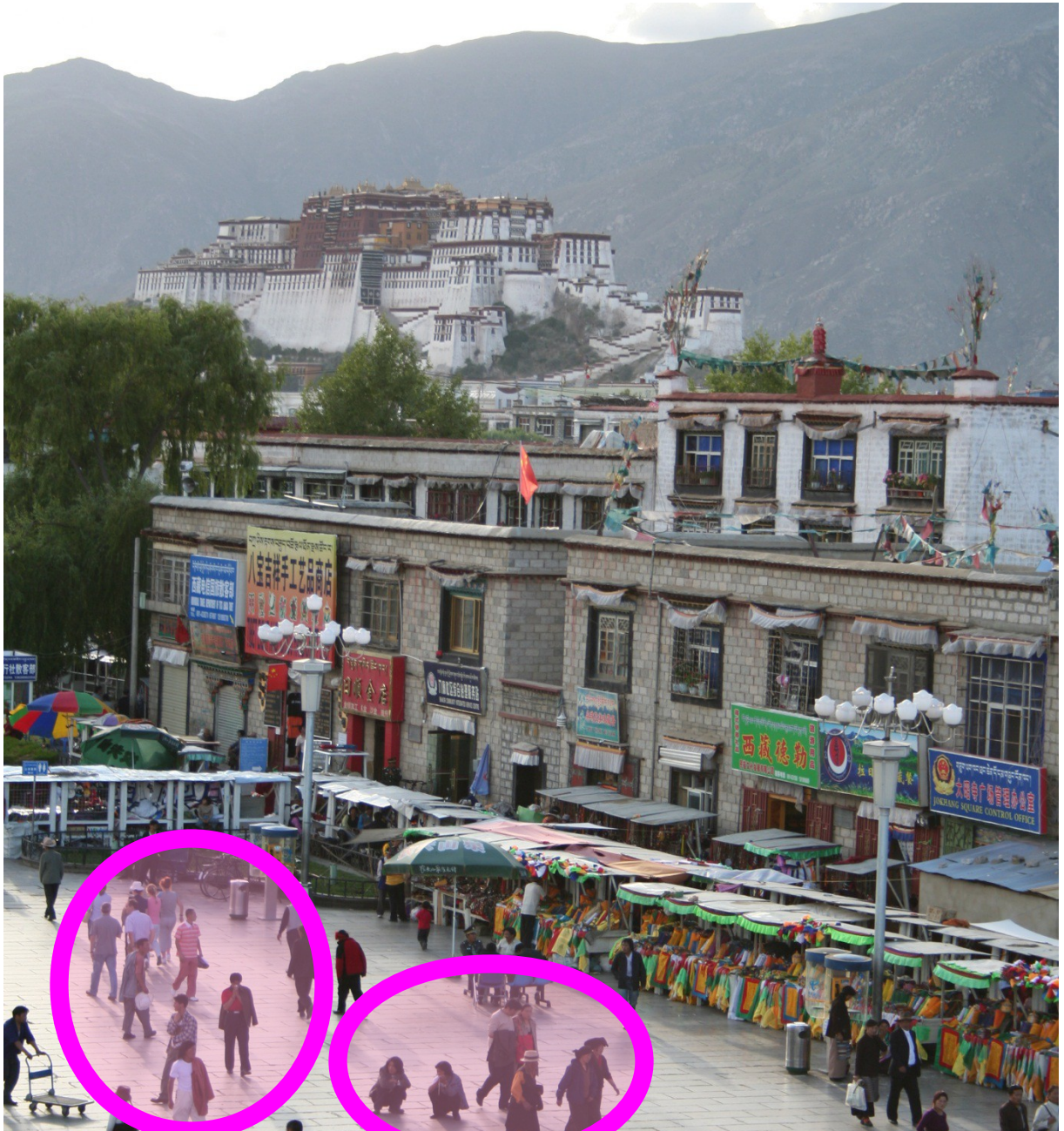
- Szeliski Chapter 14

What do we mean by “object recognition”?

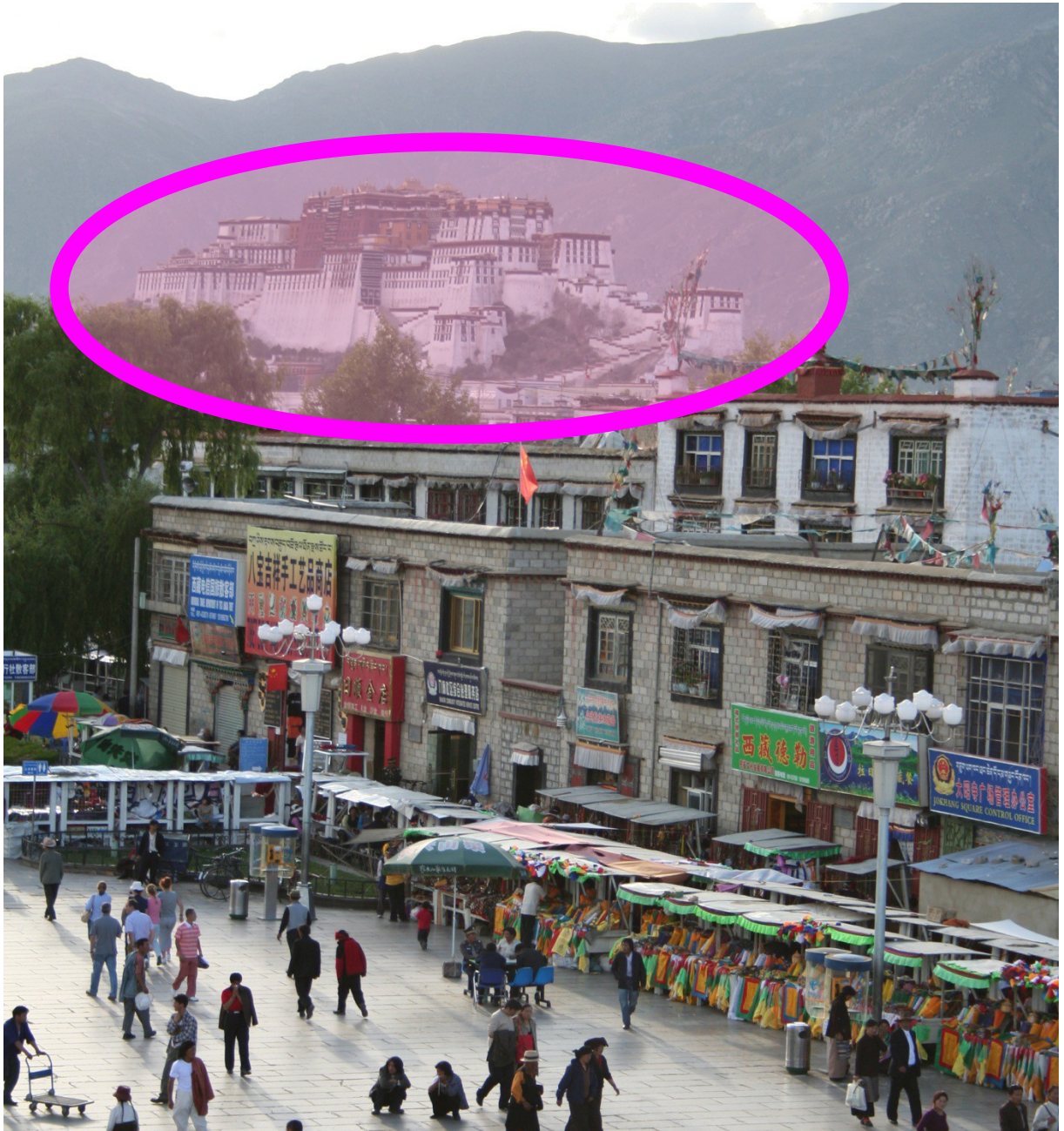
Next 15 slides adapted from Li, Fergus, & Torralba’s excellent short course on category and object recognition

Verification: is that a lamp?

Detection: are there people?

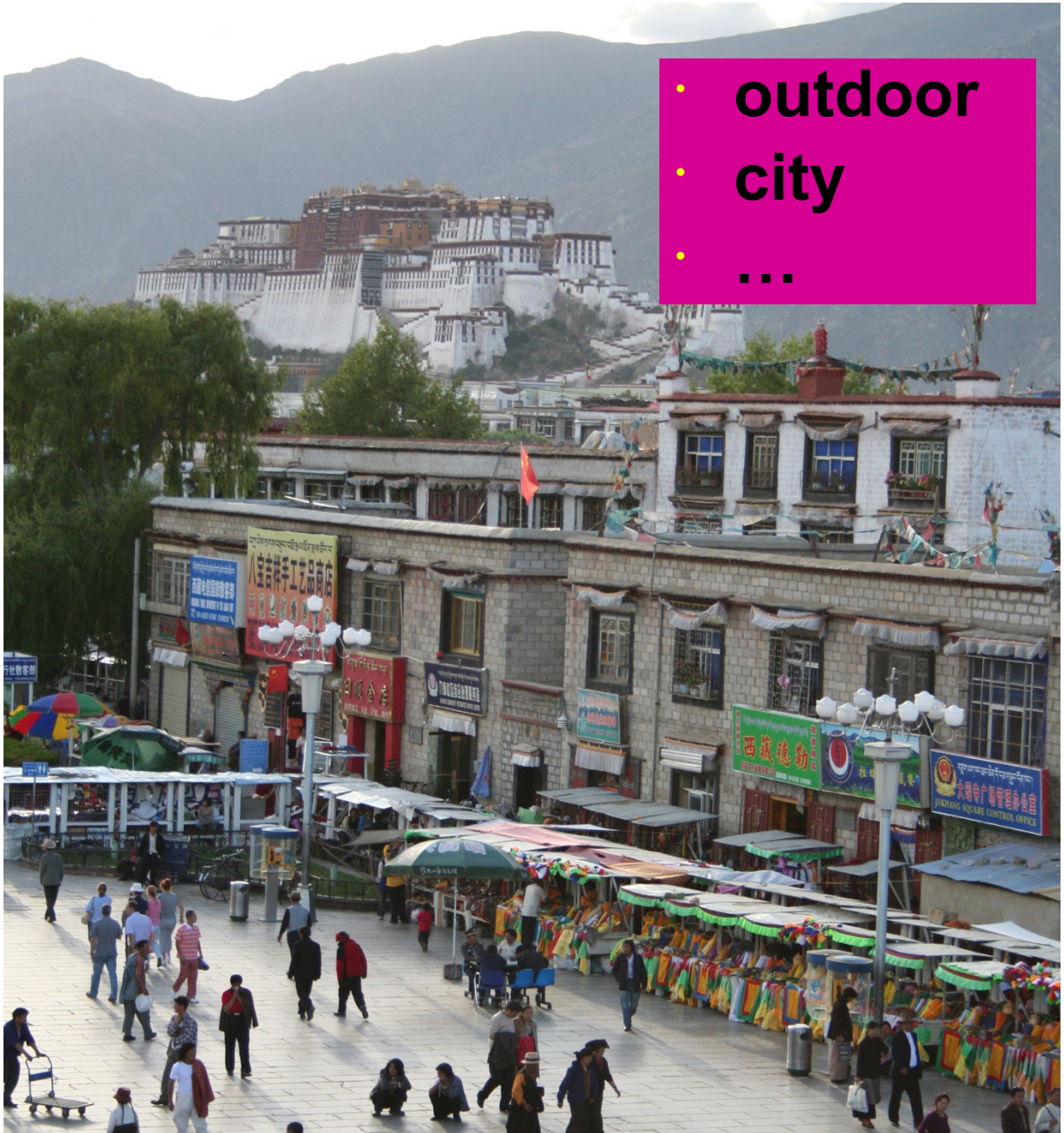


Identification: is that Potala Palace?



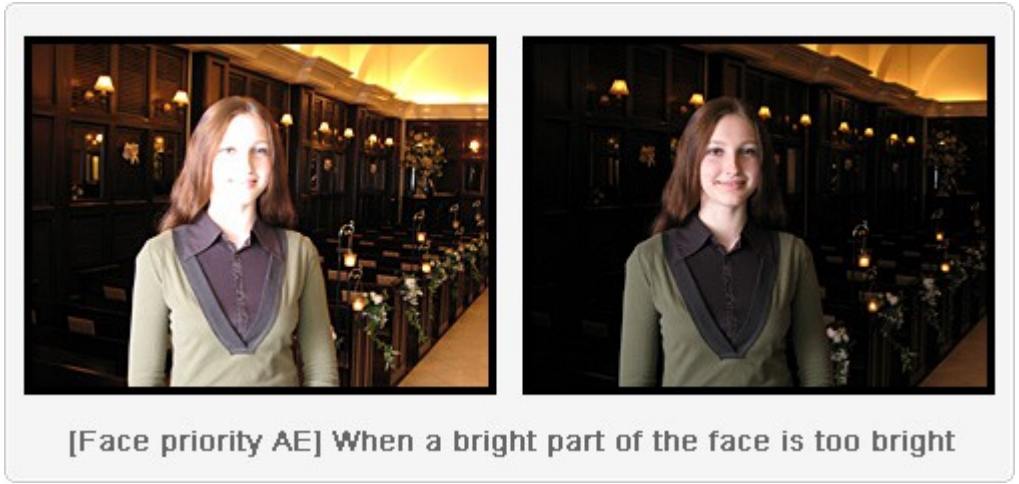
Object categorization

Scene and context categorization



- outdoor
- city
- …

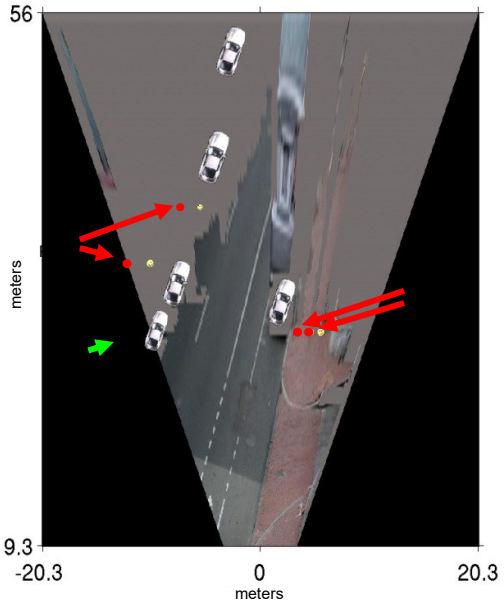
Applications: Computational photography

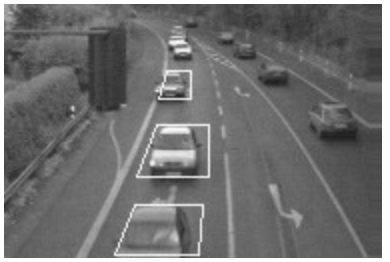


[Face priority AE] When a bright part of the face is too bright

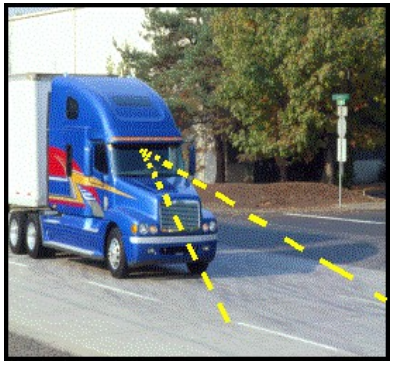
Applications: Assisted driving

Pedestrian and car detection





Lane detection



- Collision warning systems with adaptive cruise control,
- Lane departure warning systems,
- Rear object detection systems,

Applications: image search

Google Similar Images LABS

Search images

Places
London
New York
Egypt
Forbidden City

Celebrities
Michael Jordan
Angelina Jolie
Halle Berry
Seth Rogan
Rihanna

Art
impressionism
Keith Haring
cubism
Salvador Dalí
pointillism

Shopping
evening gown
necklace
shoes

Refine your image search with visual similarity

Similar Images allows you to search for images using pictures rather than words. Click the "Similar images" link under an image to find other images that look like it. Try a search of your own or click on an example below.

paris


Similar images


Similar images


Similar images


Similar images

temple


Similar images


Similar images


Similar images


Similar images

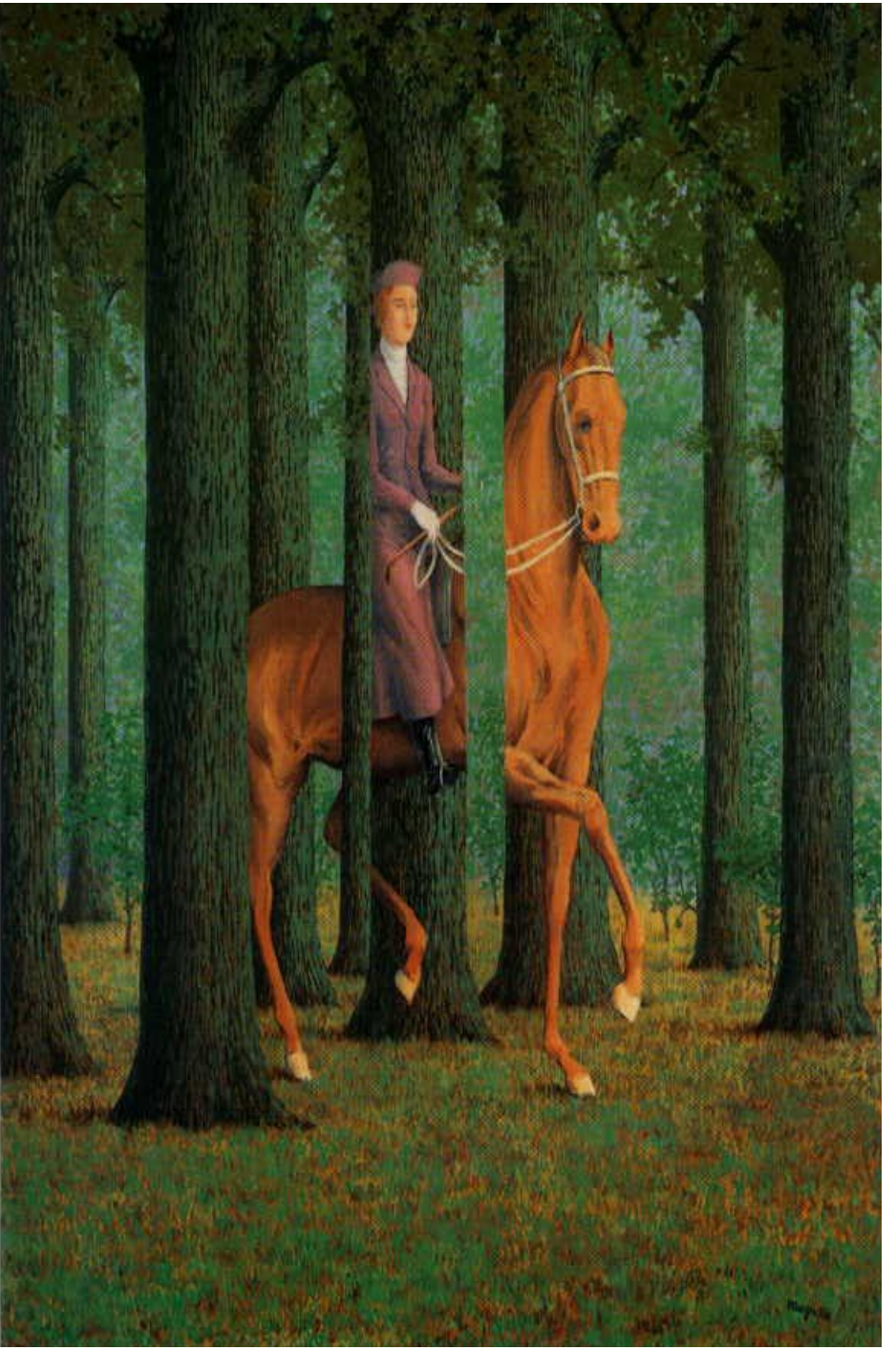
Challenges: viewpoint variation

Michelangelo 1475-1564

Challenges: illumination variation

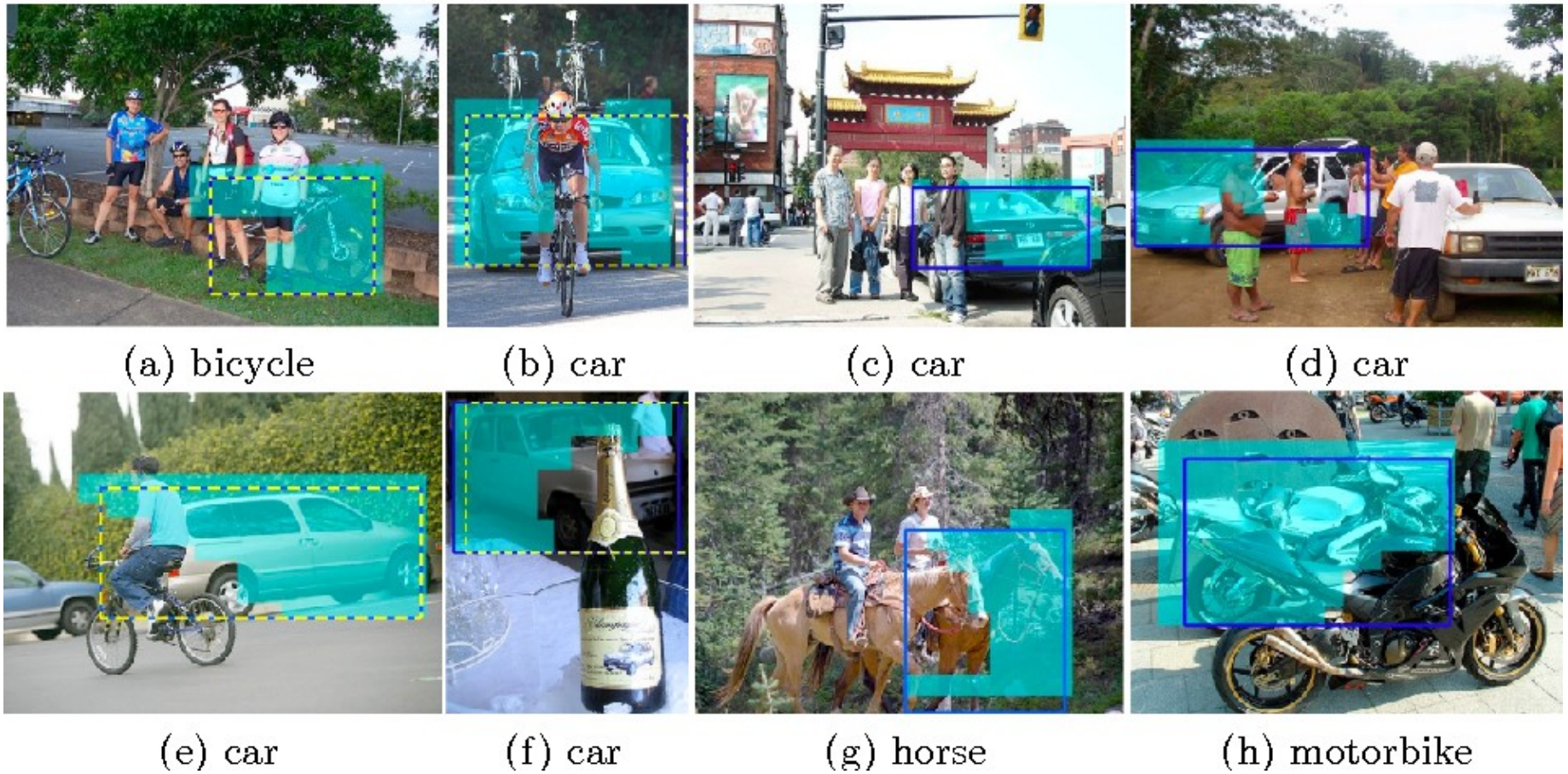
slide credit: S. Ullman

Challenges: occlusion



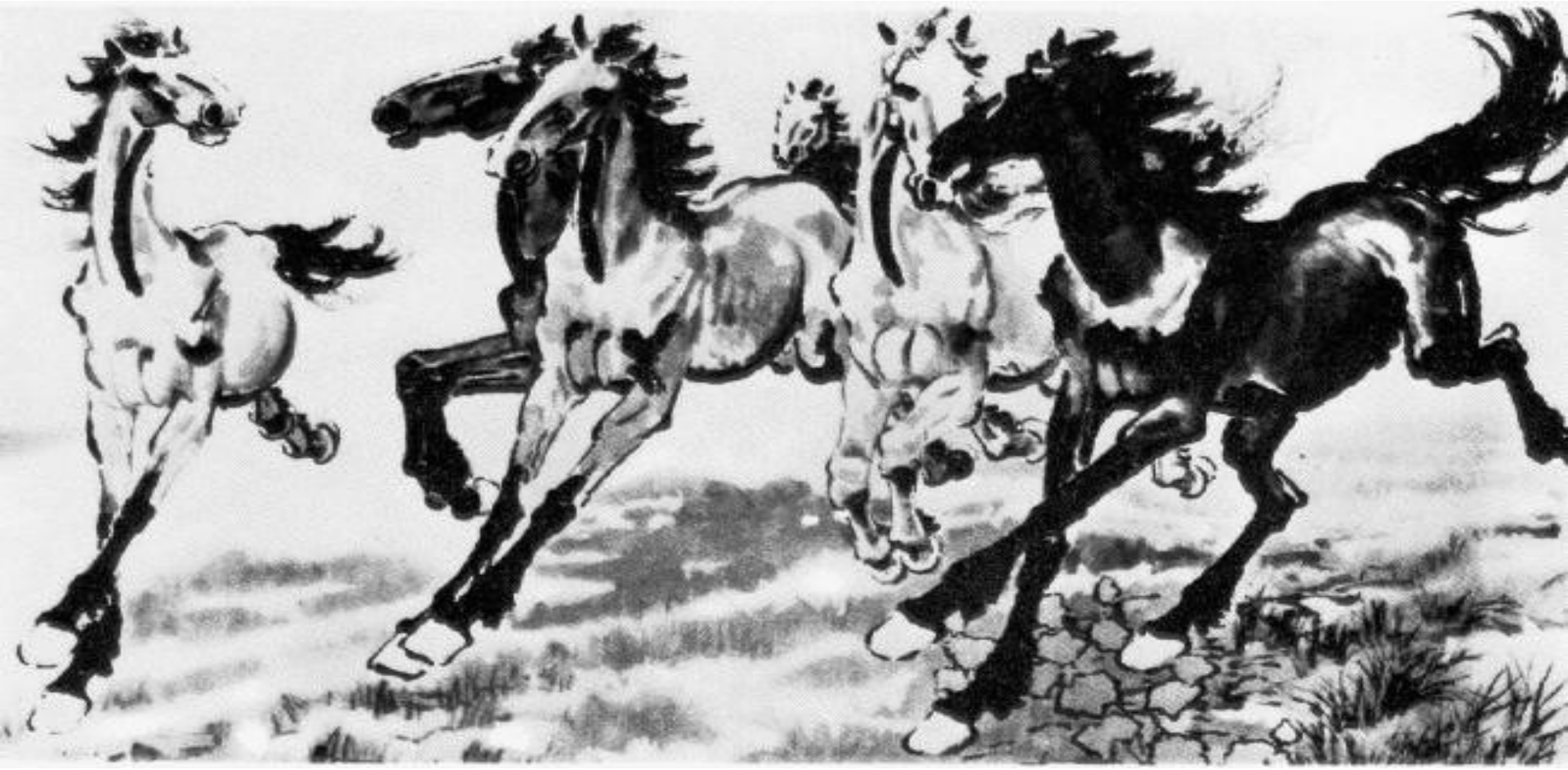
Magritte, 1957

Detection of Partially visible objects



Challenges: scale

Challenges: deformation



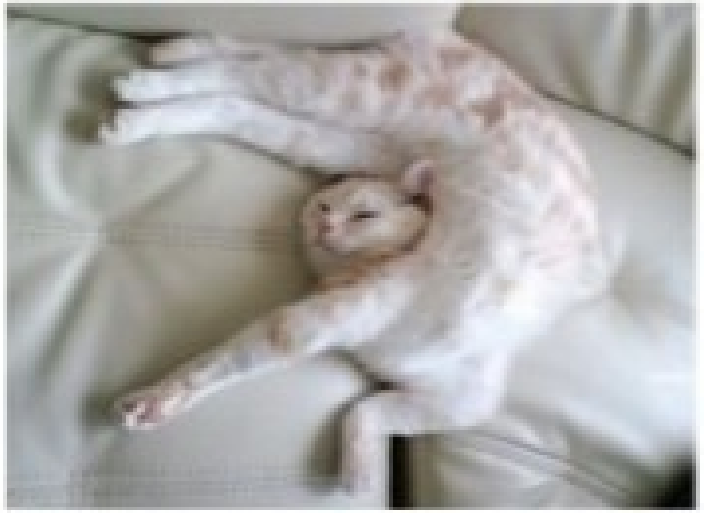
Xu, Beihong 1943

Challenges: background clutter

Klimt, 1913

Challenges: intra-class variation

Deformation



Occlusion

Background clutter

Intra-class variation

Face Detection

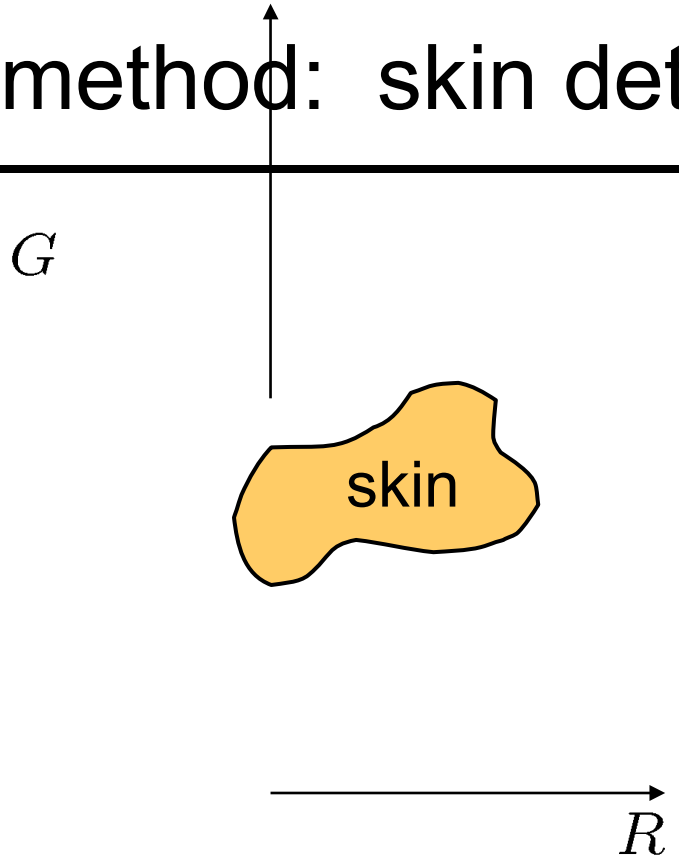
Today

- skin detection
- face detection with Viola Jones

Face detection

How to tell if a face is present?

One simple method: skin detection



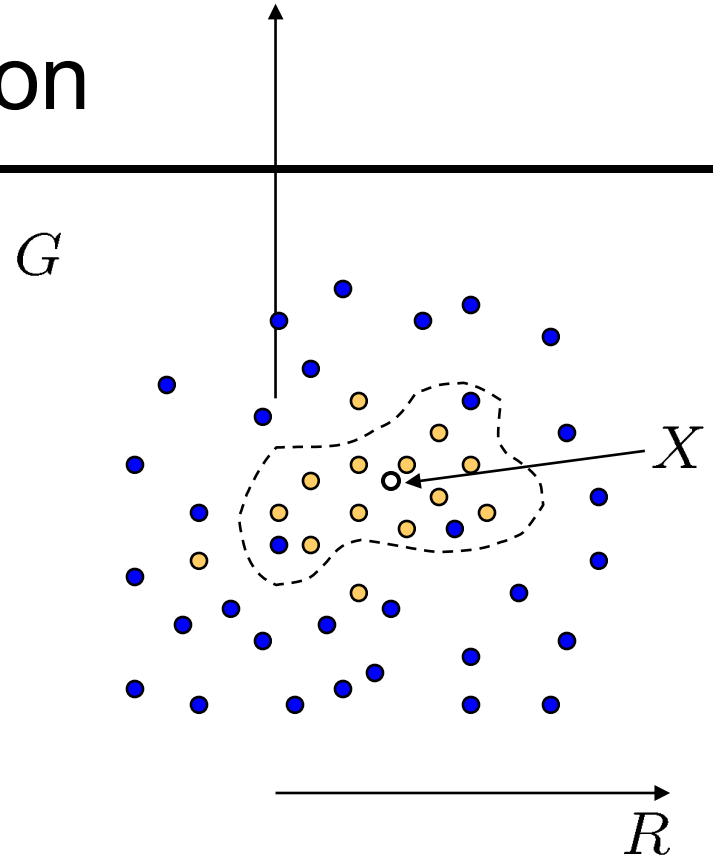
Skin pixels have a distinctive range of colors

- Corresponds to region(s) in RGB color space
  - for visualization, only R and G components are shown above

Skin classifier

- A pixel X = (R,G,B) is skin if it is in the skin region
- But how to find this region?

Skin detection



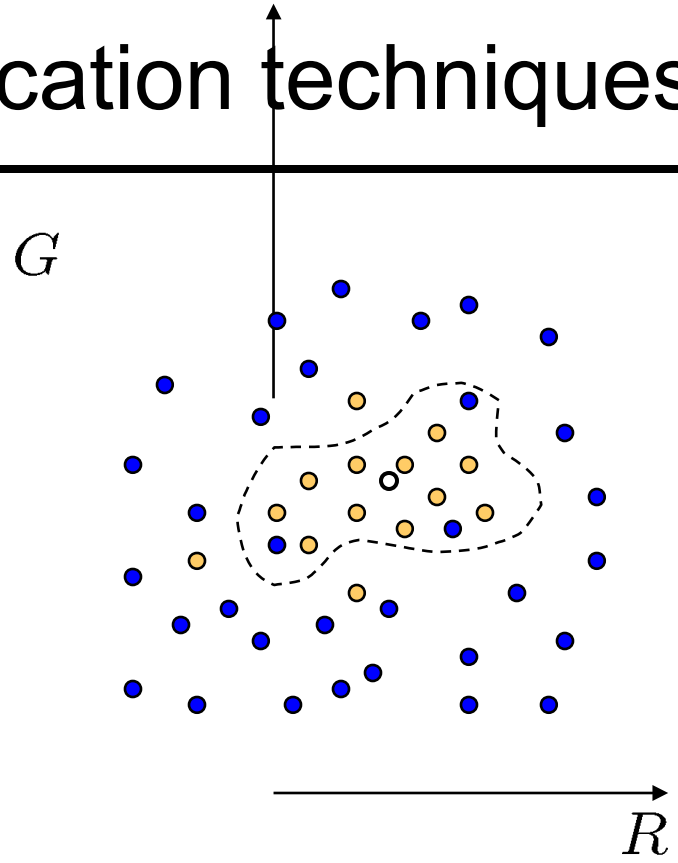
Learn the skin region from examples

- Manually label pixels in one or more “training images” as skin or not skin
- Plot the training data in RGB space
  - skin pixels shown in orange, non-skin pixels shown in blue
  - some skin pixels may be outside the region, non-skin pixels inside. Why?

Skin classifier

- Given X = (R,G,B): how to determine if it is skin or not?

Skin classification techniques



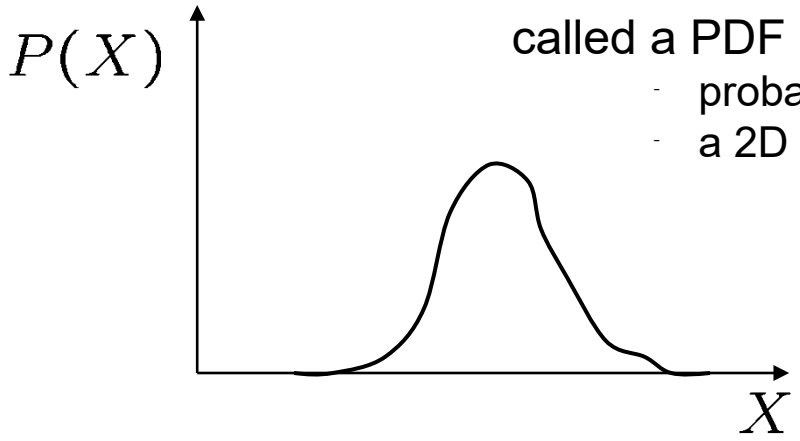
Skin classifier

- Given X = (R,G,B): how to determine if it is skin or not?
- Nearest neighbor
  - find labeled pixel closest to X
  - choose the label for that pixel
- Data modeling
  - fit a model (curve, surface, or volume) to each class
- Probabilistic data modeling
  - fit a probability model to each class

Probability

Basic probability

- X is a random variable
- P(X) is the probability that X achieves a certain value



called a PDF

- probability distribution/density function
- a 2D PDF is a surface, 3D PDF is a volume

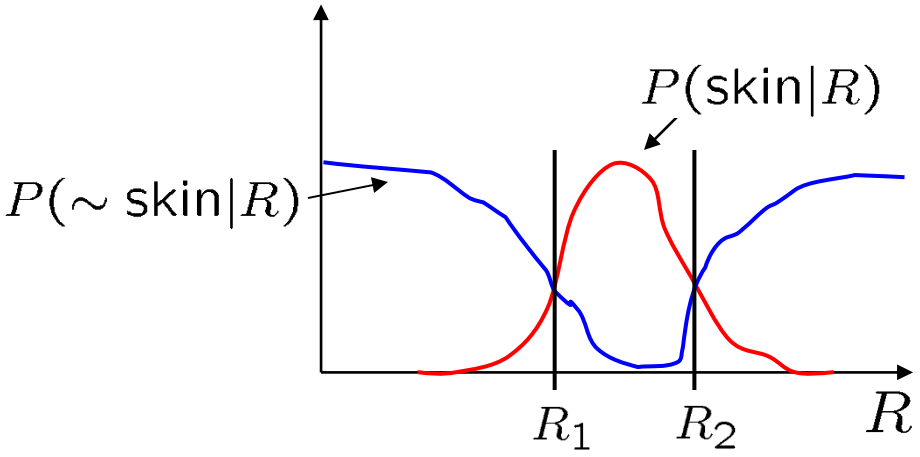
$$0 \leq P(X) \leq 1$$

- $\int_{-\infty}^{\infty} P(X)dX = 1$ continuous X

$\sum P(X) = 1$ discrete X

- or

Probabilistic skin classification



Now we can model uncertainty

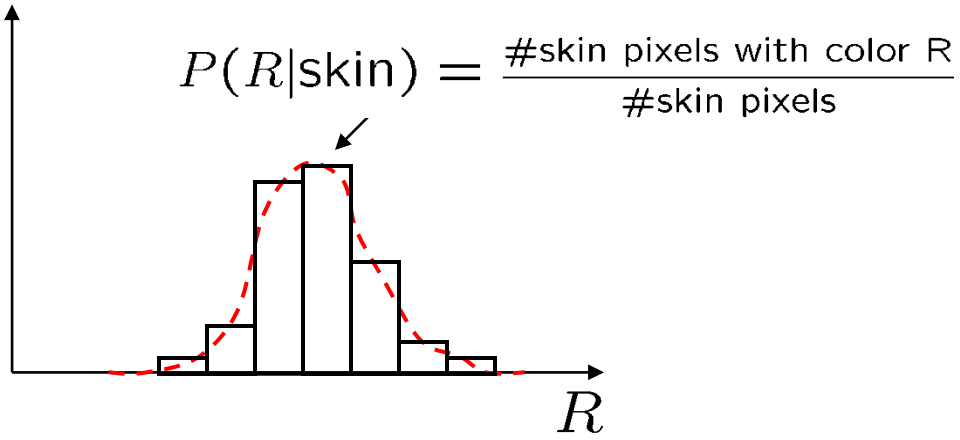
- Each pixel has a probability of being skin or not skin
  - $P(\sim \text{skin}|R) = 1 - P(\text{skin}|R)$

Skin classifier

- Given X = (R,G,B): how to determine if it is skin or not?
- Choose interpretation of highest probability
  - set X to be a skin pixel if and only if $R_1 < X \leq R_2$

Where do we get $P(\text{skin}|R)$ and $P(\sim \text{skin}|R)$?

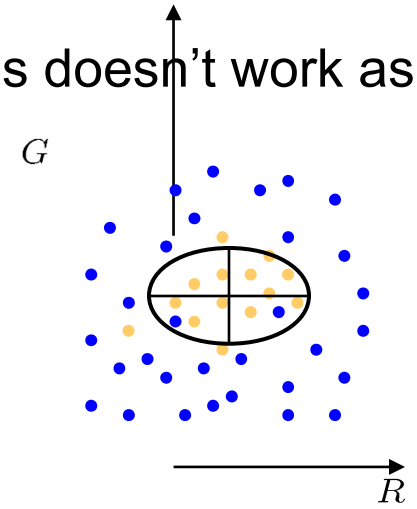
Learning conditional PDF's



We can calculate P(R | skin) from a set of training images

- It is simply a histogram over the pixels in the training images
  - each bin Ri contains the proportion of skin pixels with color Ri

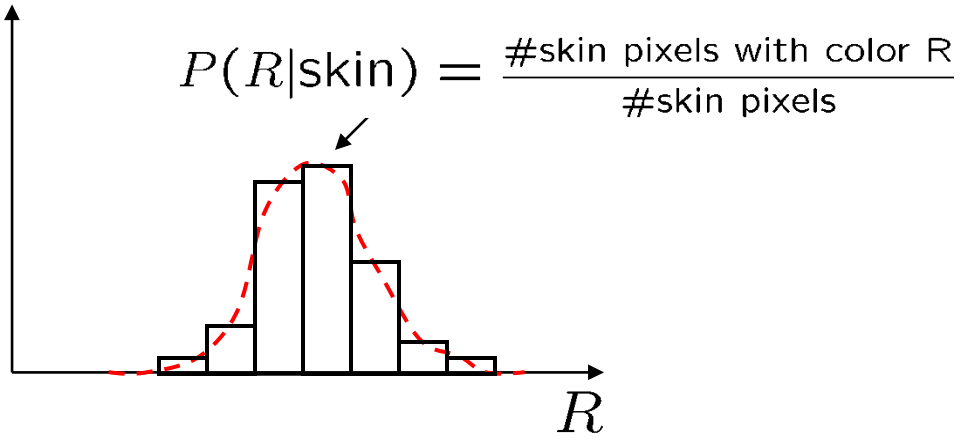
This doesn't work as well in higher-dimensional spaces. Why not?



Approach: fit parametric PDF functions

- common choice is rotated Gaussian
  - center $\mathbf{c} = \overline{X}$
  - covariance $\sum_X (X - \overline{X})(X - \overline{X})^T$
    - orientation, size defined by eigenvecs, eigenvals

Learning conditional PDF's



We can calculate P(R | skin) from a set of training images

- It is simply a histogram over the pixels in the training images
  - each bin Ri contains the proportion of skin pixels with color Ri

But this isn't quite what we want

- Why not? How to determine if a pixel is skin?
- We want P(skin | R) not P(R | skin)
- How can we get it?

Bayes rule

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

In terms of our problem:

what we measure (**likelihood**)

domain knowledge (**prior**)

$$P(\text{skin}|R) = \frac{P(R|\text{skin})\ P(\text{skin})}{P(R)}$$

what we want (**posterior**)

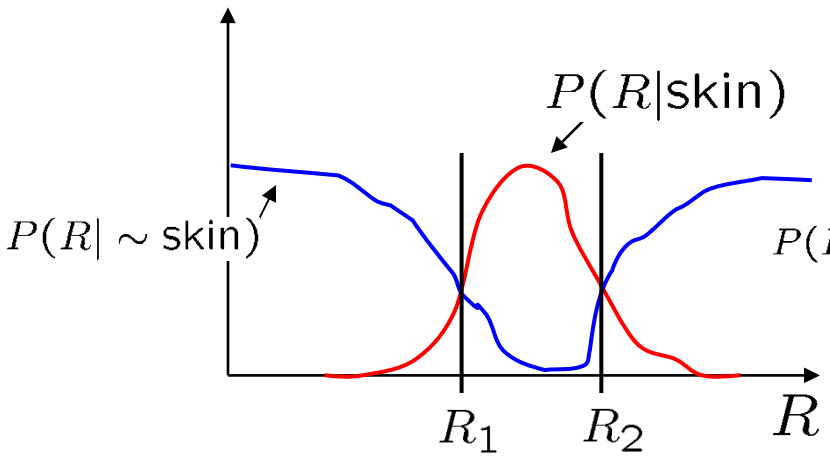
normalization term

$$P(R) = P(R|\text{skin})P(\text{skin}) + P(R|\sim\text{skin})P(\sim\text{skin})$$

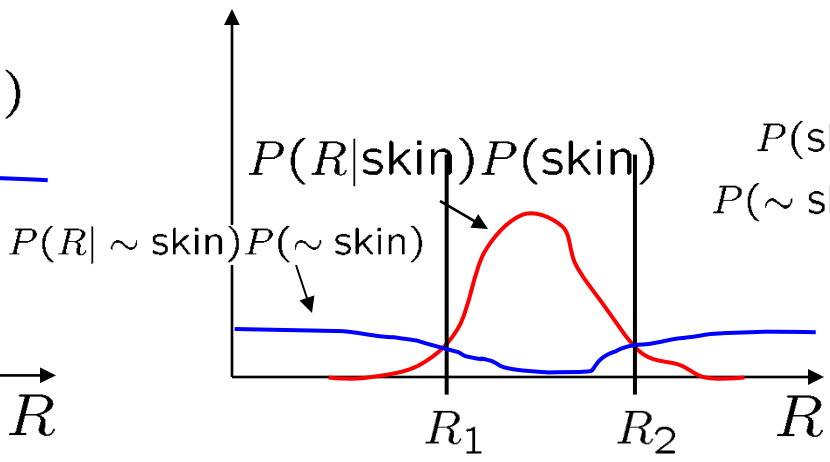
The prior: P(skin)

- Could use domain knowledge
  - P(skin) may be larger if we know the image contains a person
  - for a portrait, P(skin) may be higher for pixels in the center
- Could learn the prior from the training set. How?
  - P(skin) may be proportion of skin pixels in training set

Bayesian estimation



likelihood



posterior (unnormalized)

Bayesian estimation = minimize probability of misclassification

- Goal is to choose the label (skin or ~skin) that maximizes the posterior
  - this is called **Maximum A Posteriori (MAP) estimation**
- Suppose the prior is uniform: P(skin) = P(~skin) = 0.5
  - in this case $P(\text{skin}|R) = cP(R|\text{skin})$ $P(\sim\text{skin}|R) = cP(R|\sim\text{skin})$
  - maximizing the posterior is equivalent to maximizing the likelihood
    - $P(\text{skin}|R) > P(\sim\text{skin}|R)$ if and only if $P(R|\text{skin}) > P(R|\sim\text{skin})$
  - this is called **Maximum Likelihood (ML) estimation**

Skin detection results

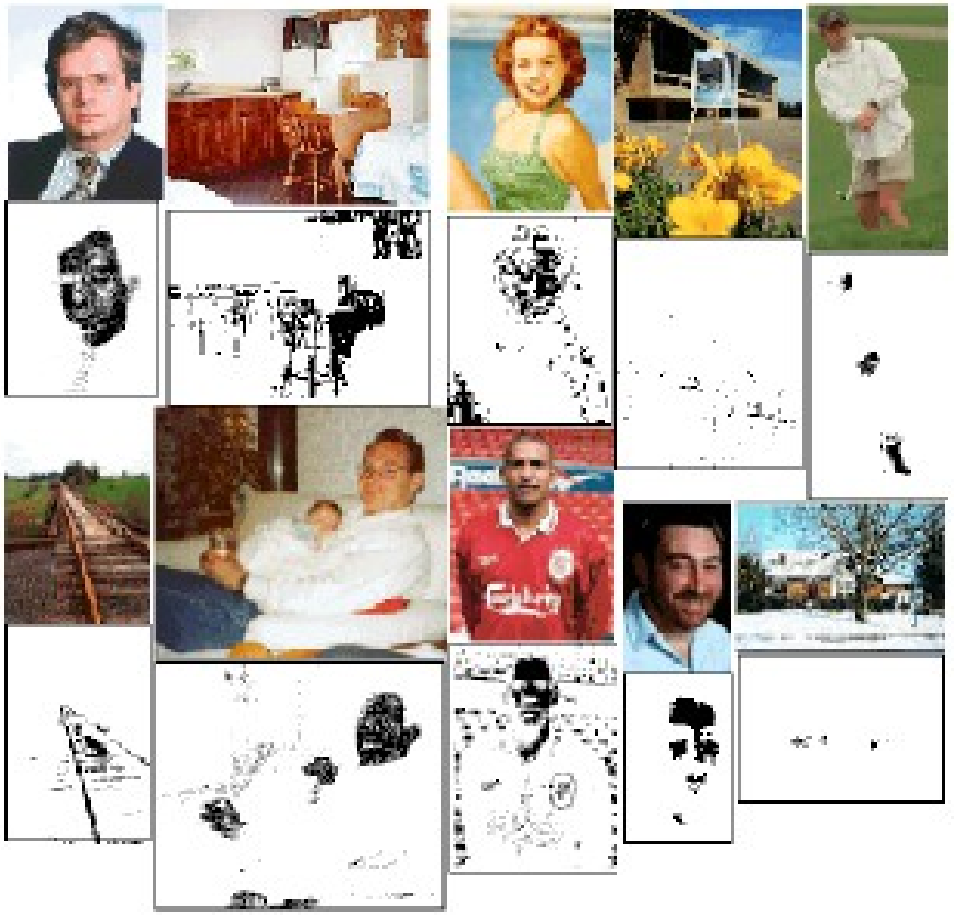
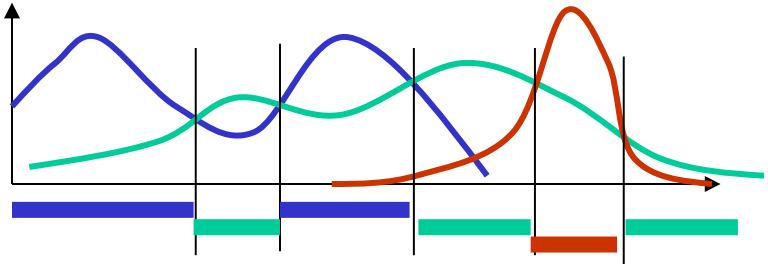


Figure 25.3. The figure shows a variety of images together with the output of the skin detector of Jones and Rehg applied to the image. Pixels marked black are skin pixels, and white are background. Notice that this process is relatively effective, and could certainly be used to focus attention on, say, faces and hands. *Figure from "Statistical color models with application to skin detection," M.J. Jones and J. Rehg, Proc. Computer Vision and Pattern Recognition, 1999 © 1999, IEEE*

General classification

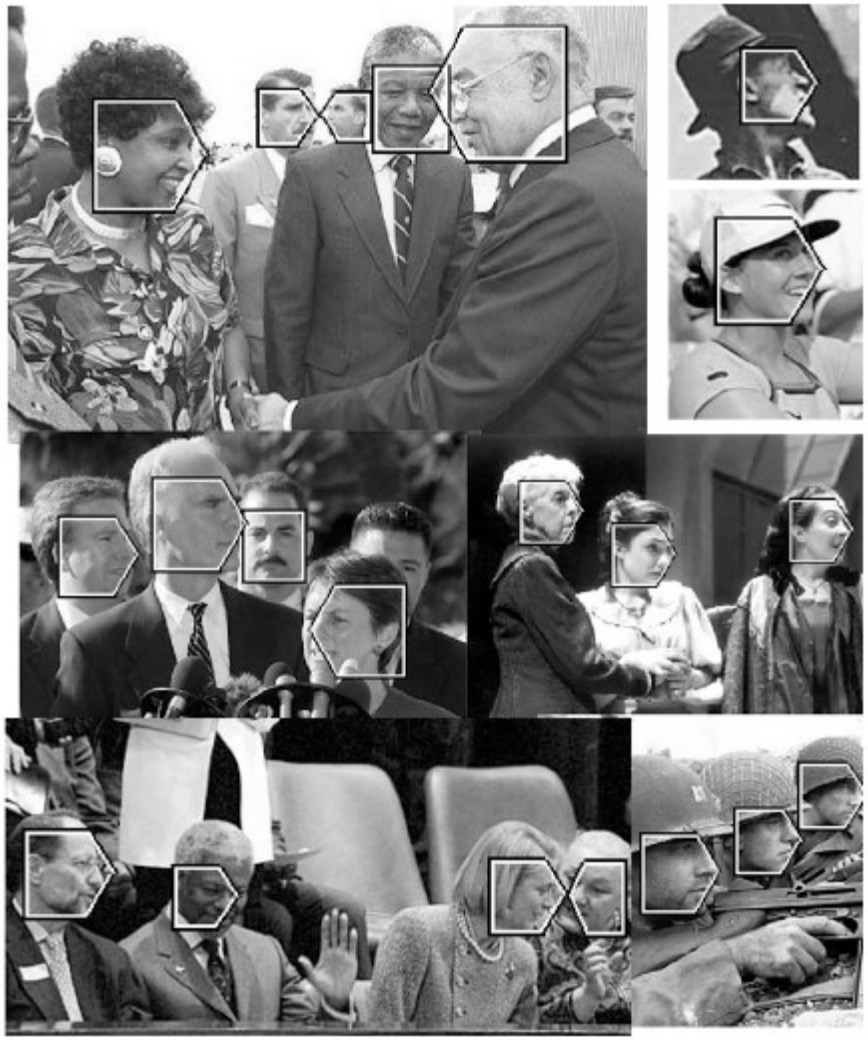
This same procedure applies in more general circumstances

- More than two classes
- More than one dimension



Example: face detection

- Here, X is an image region
  - dimension = # pixels
  - each face can be thought of as a point in a high dimensional space



H. Schneiderman and T.Kanade

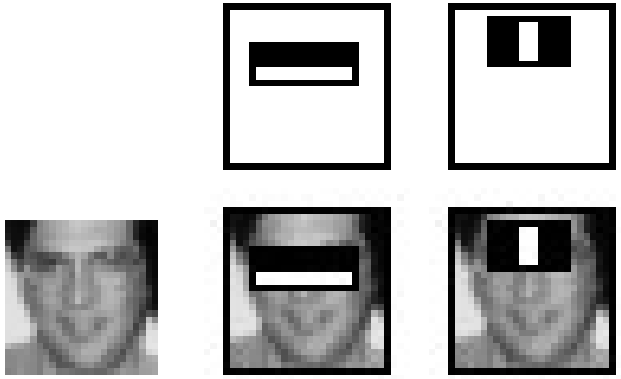
H. Schneiderman, T. Kanade. "A Statistical Method for 3D Object Detection Applied to Faces and Cars". IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000)
http://www-2.cs.cmu.edu/afs/cs.cmu.edu/user/hws/www/CVPR00.pdf

Issues: metrics

What's the best way to compare images?

- need to define appropriate features
- depends on goal of recognition task

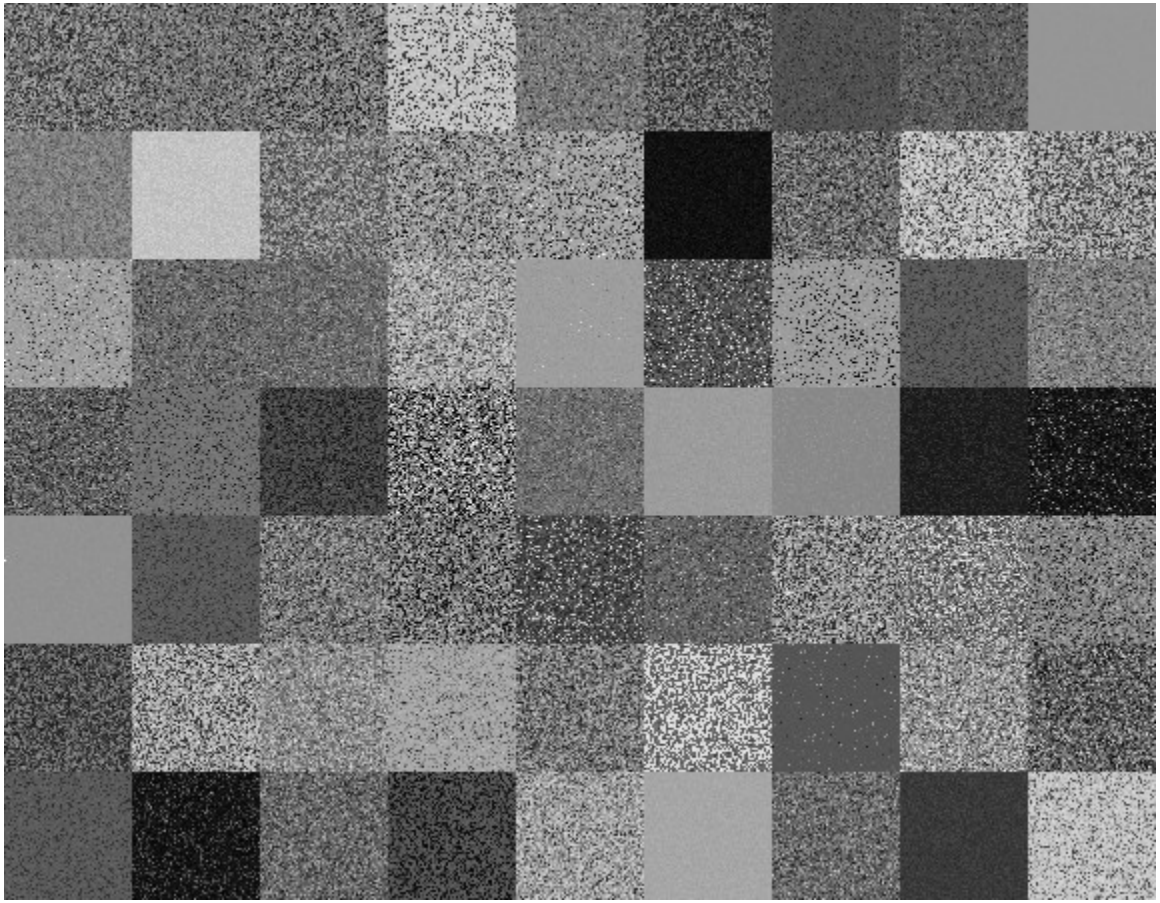
exact matching
complex features work well
(SIFT, MOPS, etc.)



classification/detection
simple features work well
(Viola/Jones, etc.)

Issues: metrics

What do you see?



Issues: metrics

What do you see?

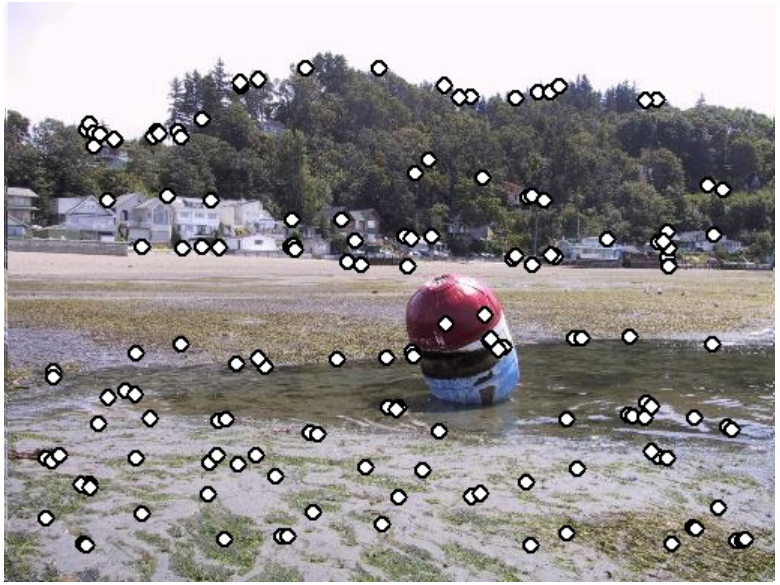
Metrics

Lots more feature types that we haven't mentioned

- moments, statistics
  - metrics: Earth mover's distance, ...
- edges, curves
  - metrics: Hausdorff, shape context, ...
- 3D: surfaces, spin images
  - metrics: chamfer (ICP)
- ...

We'll discuss more in Part 2

Issues: feature selection



If all you have is one image:
non-maximum suppression, etc.

If you have a training set of images:
AdaBoost, etc.

Issues: data modeling

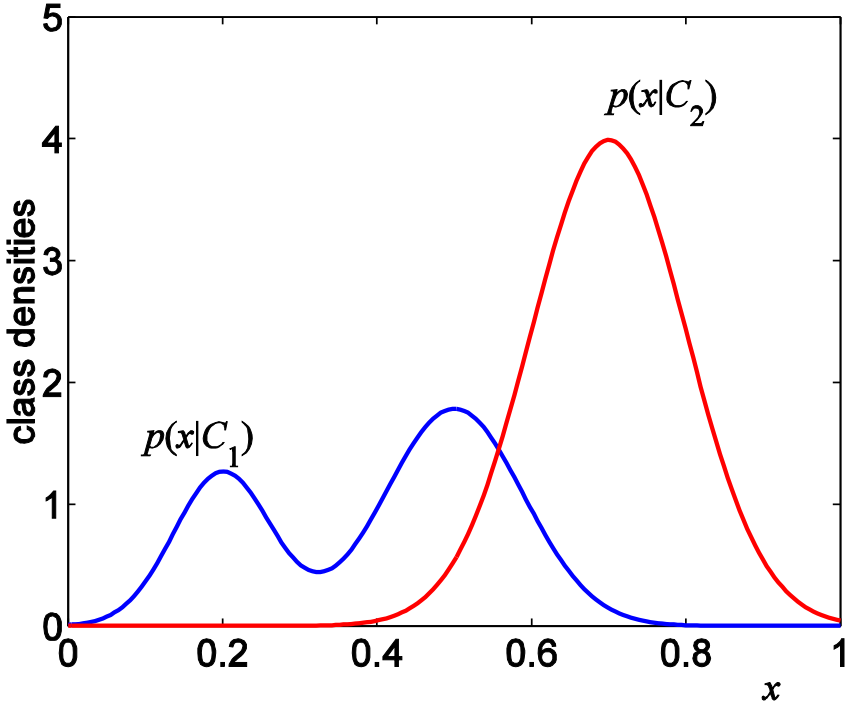
Generative methods

- model the “shape” of each class
  - histograms, PCA, mixtures of Gaussians
  - graphical models (HMM’s, belief networks, etc.)
  - ...

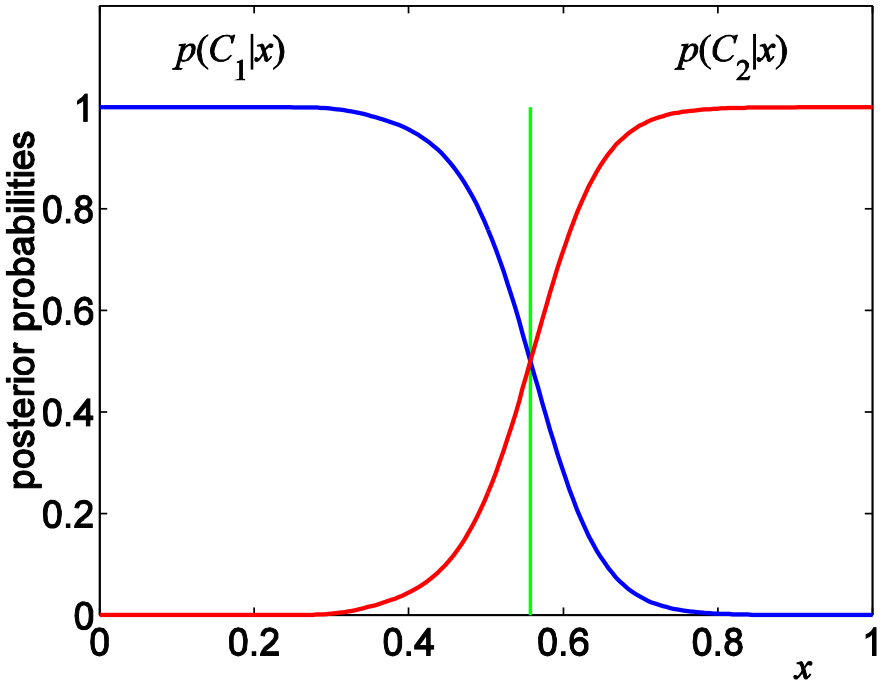
Discriminative methods

- model boundaries between classes
  - perceptrons, neural networks
  - support vector machines (SVM’s)

Generative vs. Discriminative



Generative Approach
model individual classes, priors



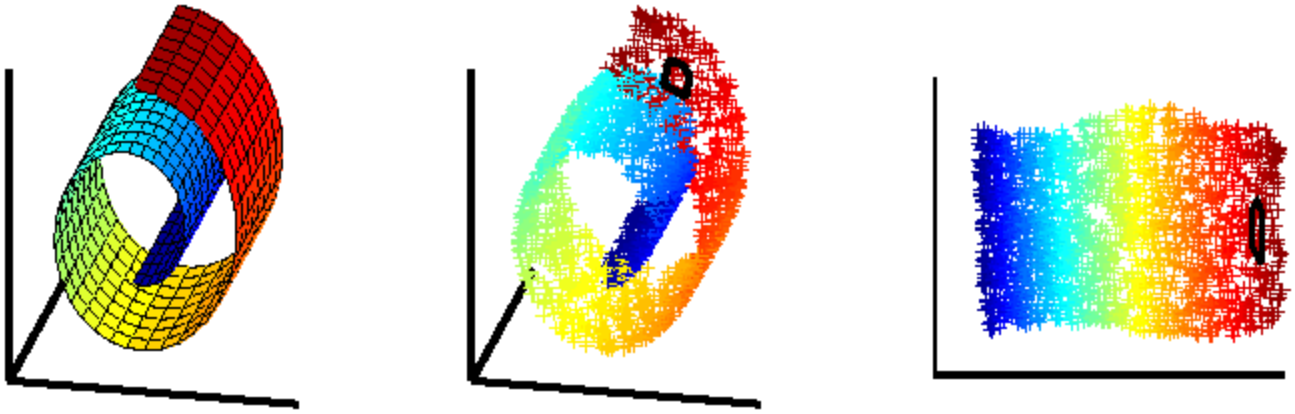
Discriminative Approach
model posterior directly

from Chris Bishop

Issues: dimensionality

What if your space isn't *flat*?

- PCA may not help



Nonlinear methods
LLE, MDS, etc.

Issues: speed

Case study: Viola Jones face detector

Next few slides adapted Grauman & Liebe's tutorial

- http://www.vision.ee.ethz.ch/~bleibe/teaching/tutorial-aaai08/

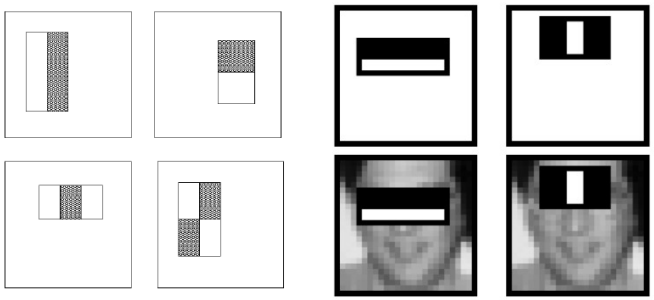
Also see Paul Viola's talk (video)

- http://www.cs.washington.edu/education/courses/577/04sp/contents.html#DM

Face detection

Where are the faces? Not who they are, that's recognition or identification.

Feature extraction



Sums of rectangular regions

How do we compute the sum of the pixels in the red box?

After some pre-computation, this can be done in constant time for any box.

This “trick” is commonly used for computing Haar wavelets (a fundemental building block of many object recognition approaches.)

243	239	240	225	206	185	188	218	211	206	216	225
242	239	218	110	67	31	34	152	213	206	208	221
243	242	123	58	94	82	132	77	108	208	208	215
235	217	115	212	243	236	247	139	91	209	208	211
233	208	131	222	219	226	196	114	74	208	213	214
232	217	131	116	77	150	69	56	52	201	228	223
232	232	182	186	184	179	159	123	93	232	235	235
232	236	201	154	216	133	129	81	175	252	241	240
235	238	230	128	172	138	65	63	234	249	241	245
237	236	247	143	59	78	10	94	255	248	247	251
234	237	245	193	55	33	115	144	213	255	253	251
248	245	161	128	149	109	138	65	47	156	239	255
190	107	39	102	94	73	114	58	17	7	51	137
23	32	33	148	168	203	179	43	27	17	12	8
17	26	12	160	255	255	109	22	26	19	35	24

Sums of rectangular regions

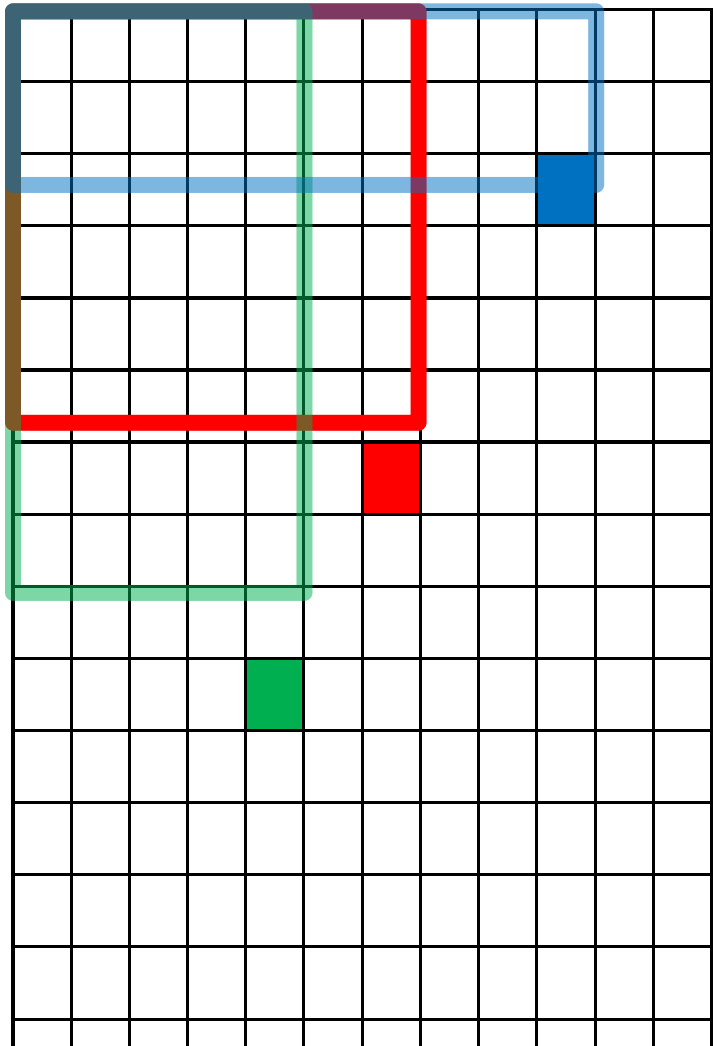
The trick is to compute an "integral image." Every pixel is the sum of its neighbors to the upper left.

Sequentially compute using:

$$I(x,y)\ I(x,\ -$$

$$I(x-1,y)+I(x,y-1)-$$

$$x-1,y-1)$$

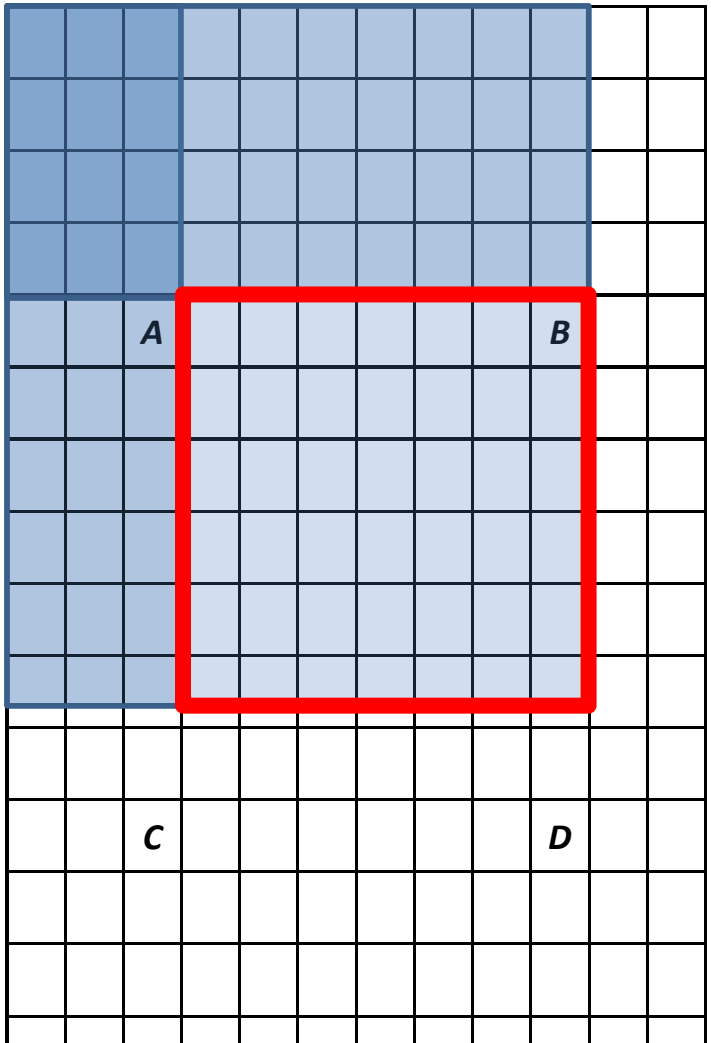


Sums of rectangular regions

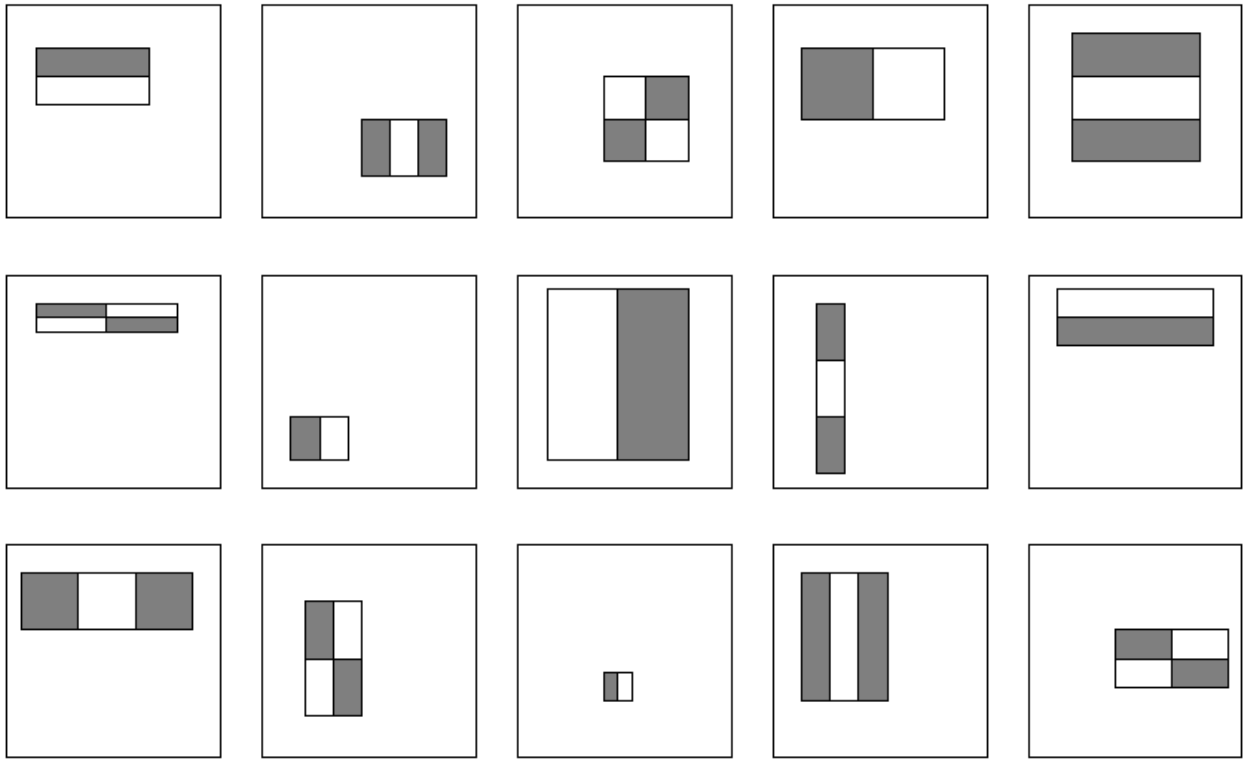
Solution is found using:

$$A + D - B - C$$

What if the position of the box lies between pixels?

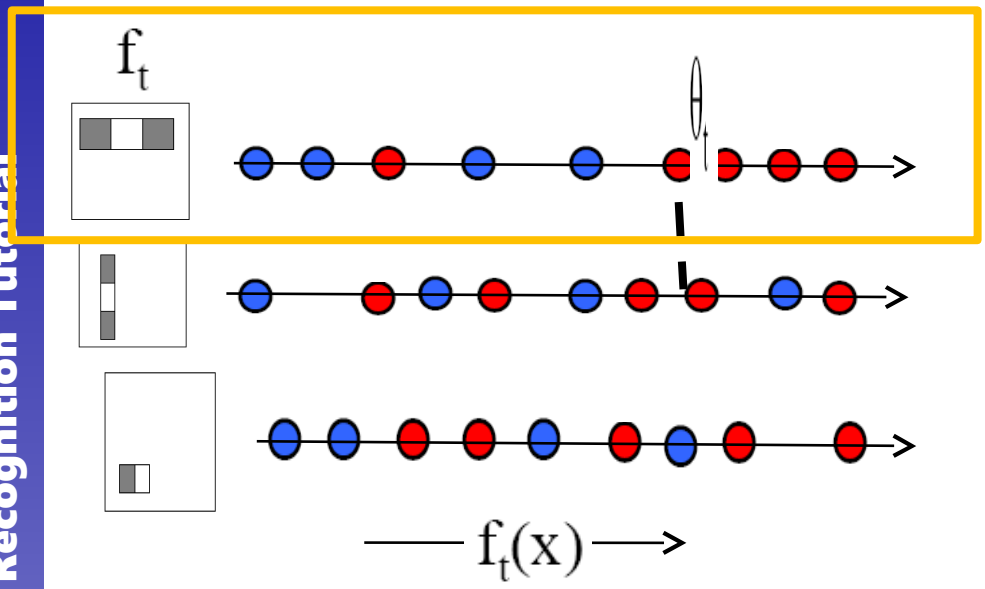


Large library of filters



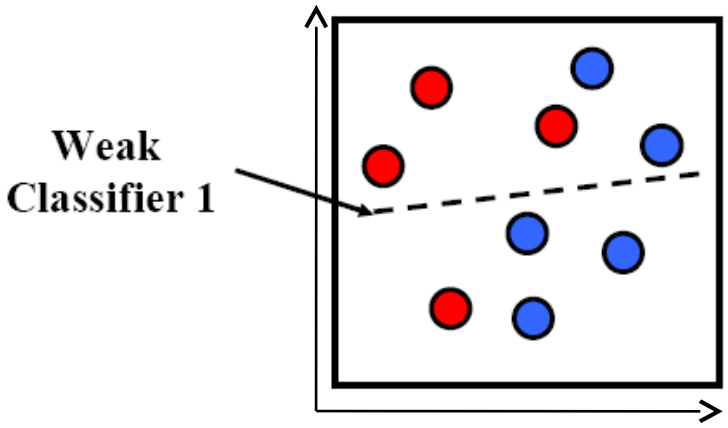
AdaBoost for feature+classifier selection

- Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of *weighted* error.

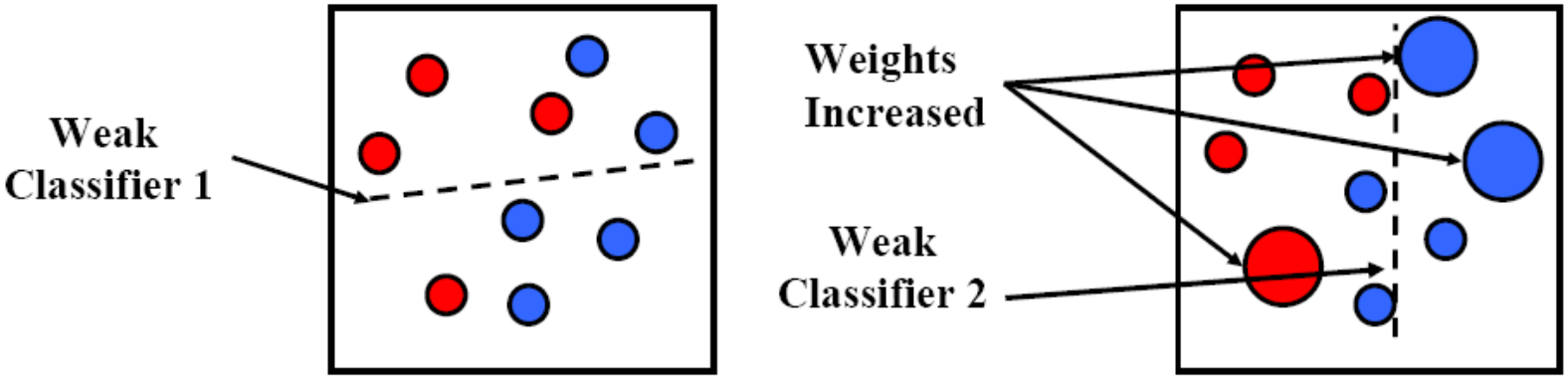


$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

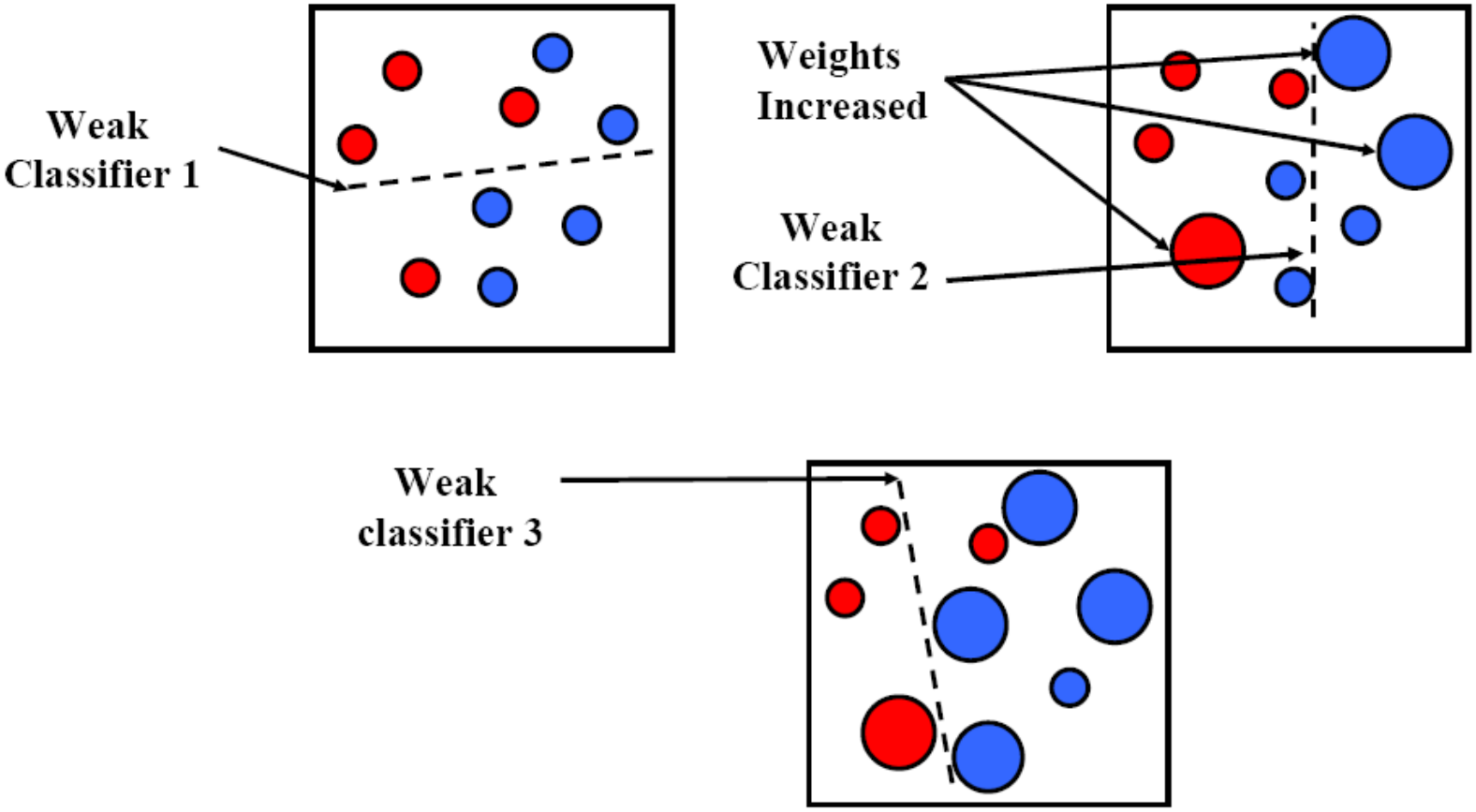
AdaBoost: Intuition



AdaBoost: Intuition



AdaBoost: Intuition

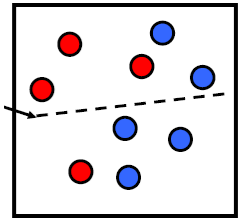


- The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

AdaBoost Algorithm



- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:

  1. Normalize the weights,

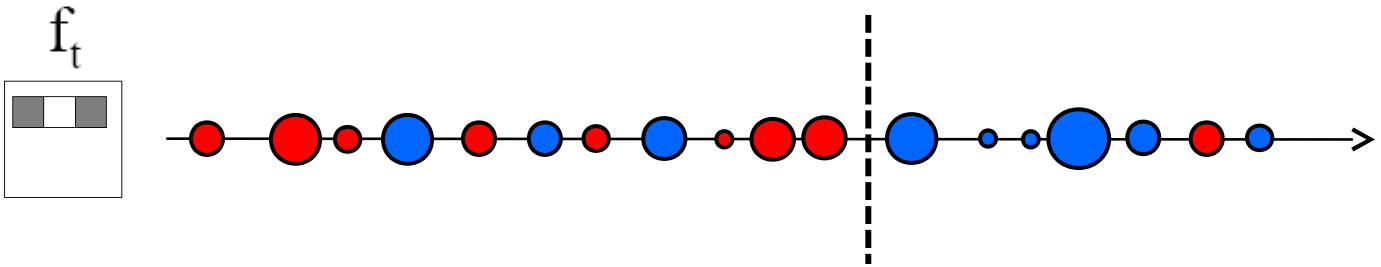
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

  so that w_t is a probability distribution.

  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:

  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.



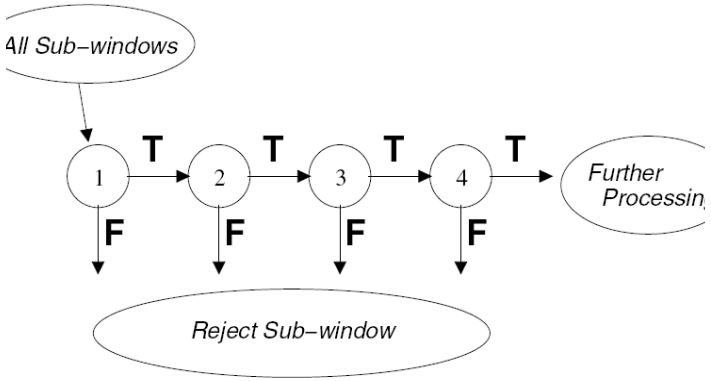
e

e

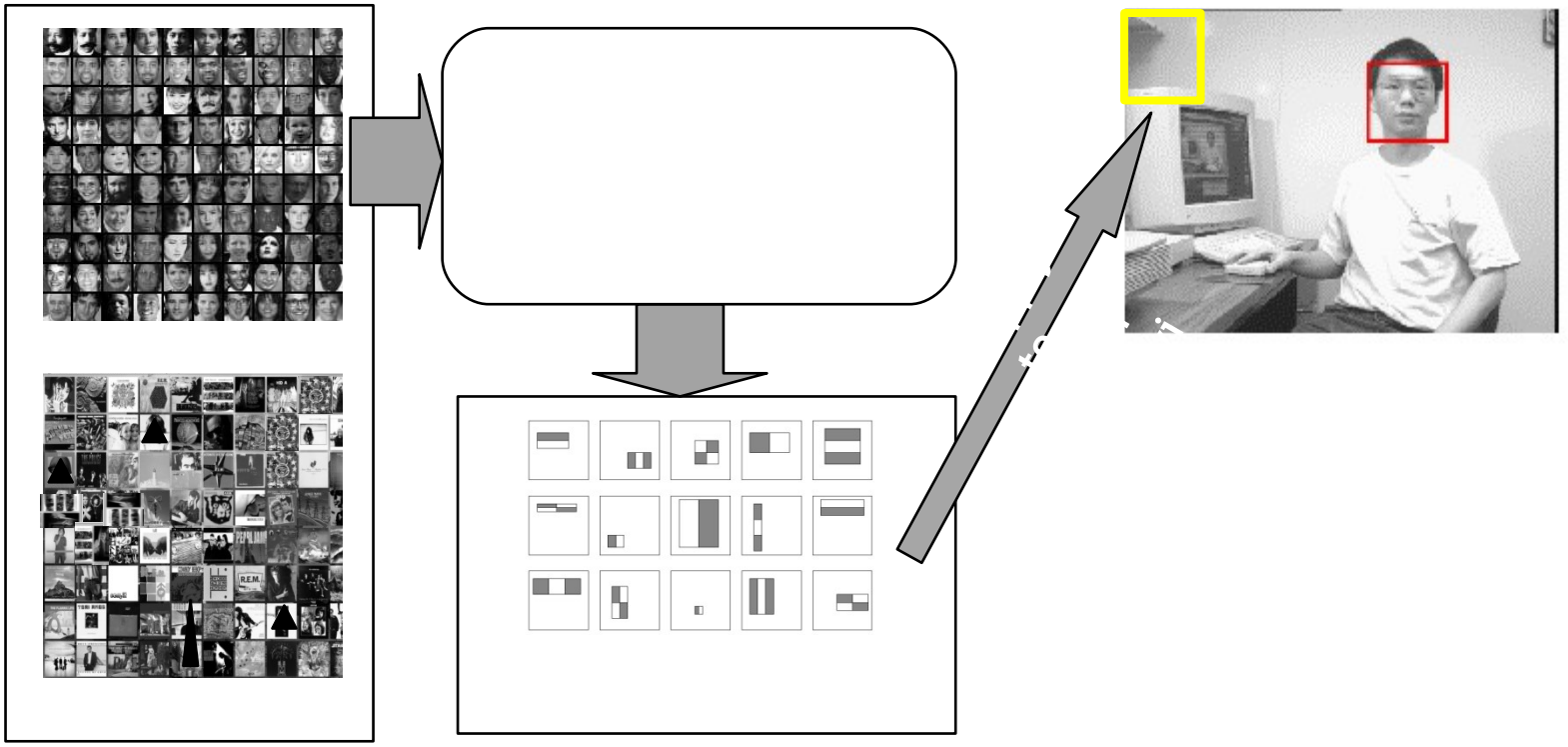
Cascading classifiers for detection

For efficiency, apply less accurate but faster classifiers first to immediately discard windows that clearly appear to be negative; e.g.,

- Filter for promising regions with an initial inexpensive classifier
- Build a chain of classifiers, choosing cheap ones with low false negative rates early in the chain



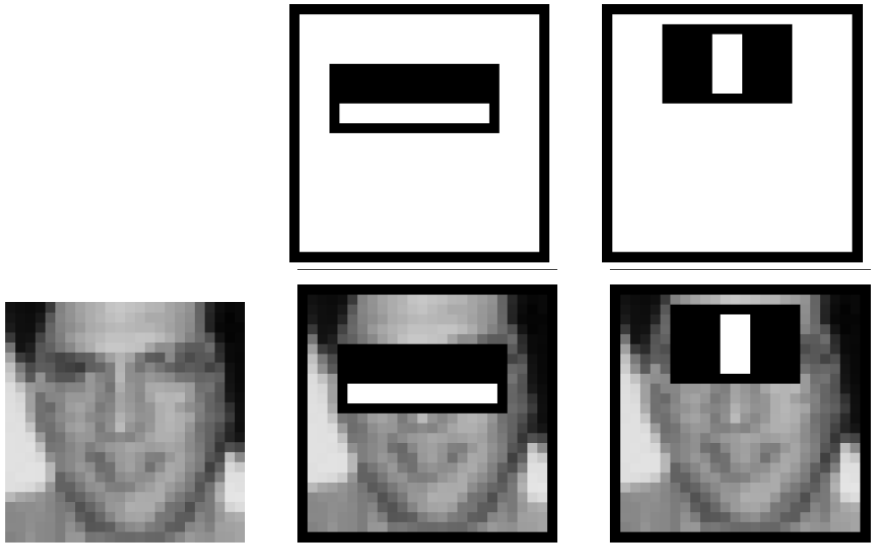
Viola-Jones Face Detector: Summary

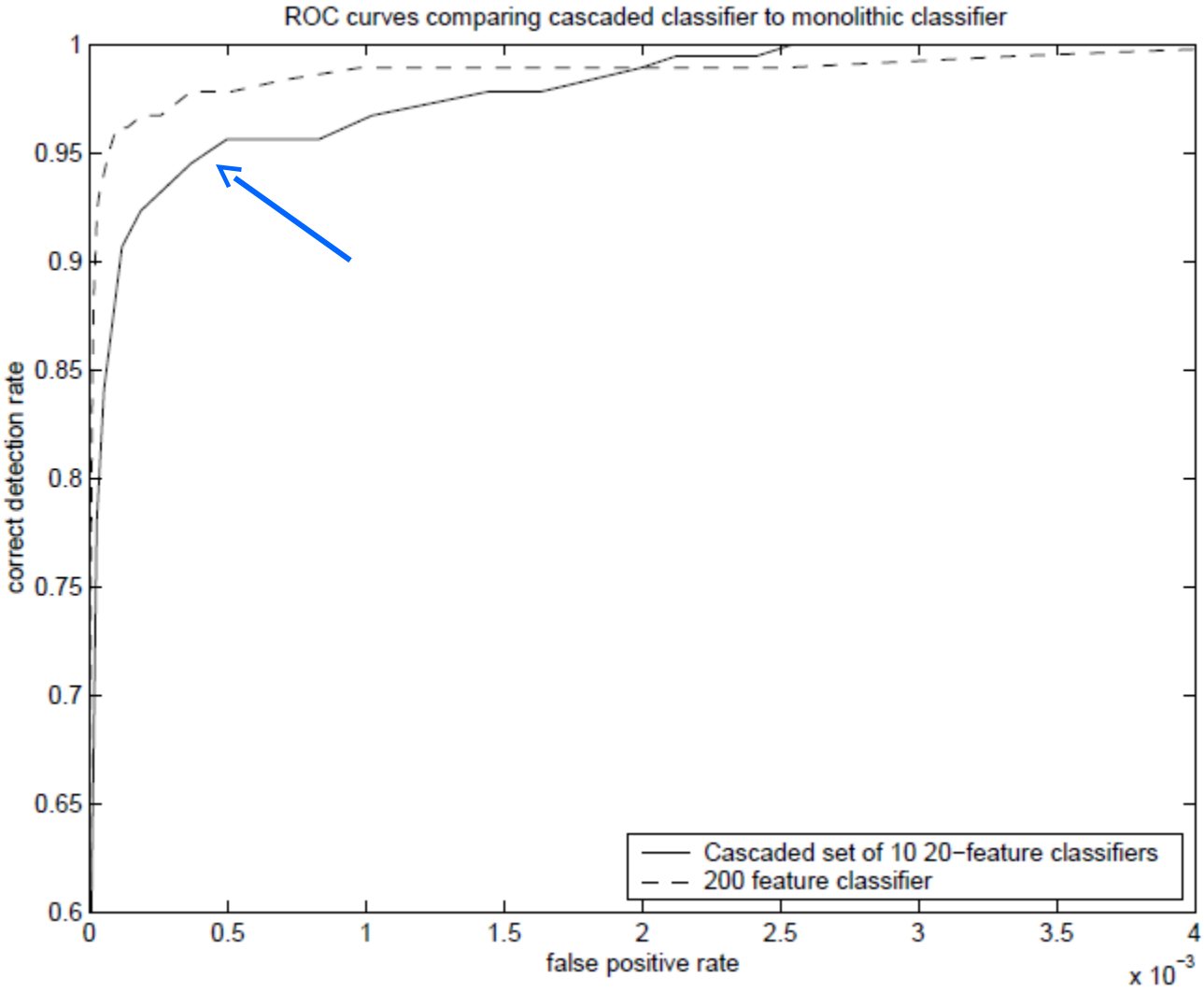


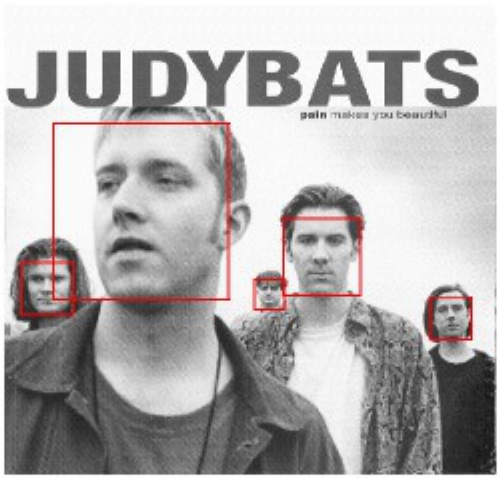
- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- [Implementation available in OpenCV: http://www.intel.com/technology/computing/opencv/]

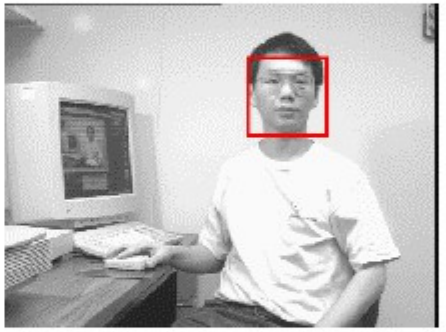

terra.es

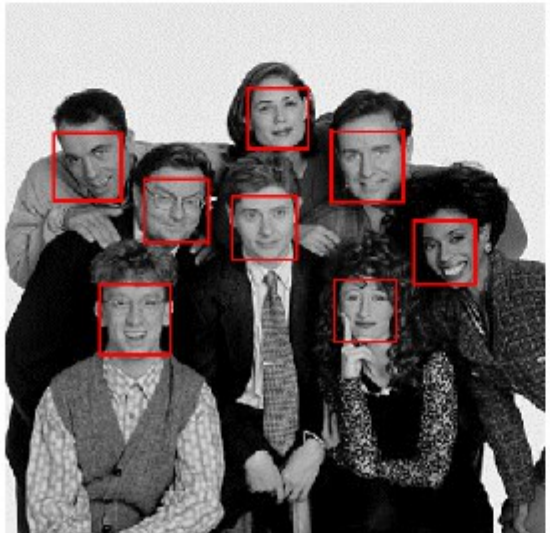
Viola-Jones Face Detector: Results

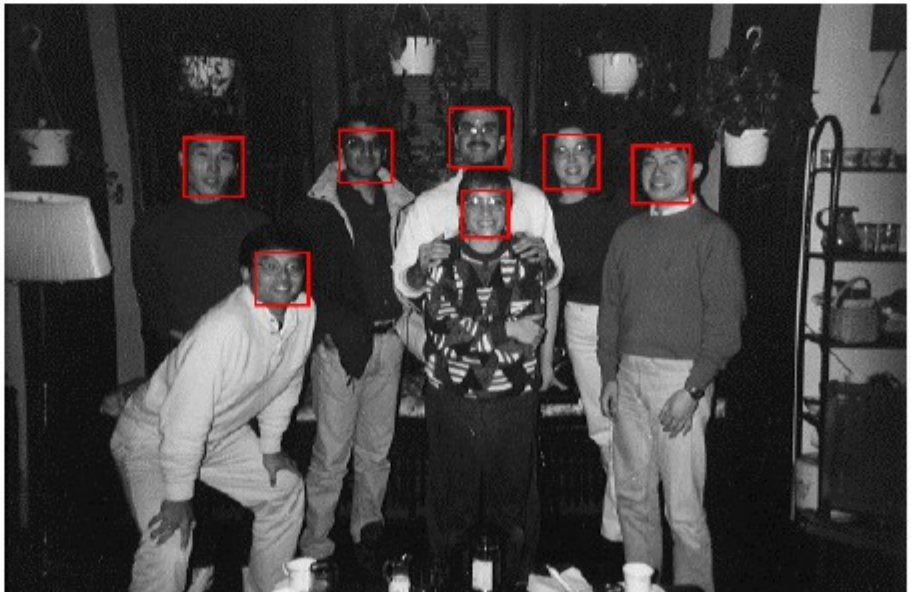


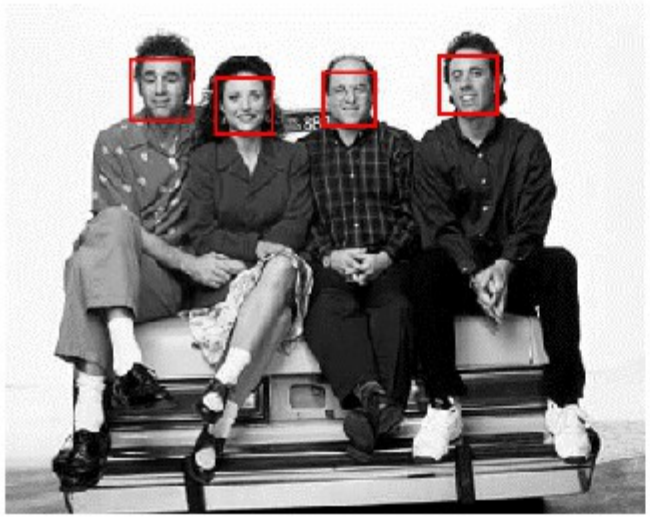

ROC curves comparing cascaded classifier to monolithic classifier
correct detection rate
false positive rate
x 10^-3
Cascaded set of 10 20−feature classifiers
200 feature classifier
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0
0.5
1
1.5
2
2.5
3
3.5
4

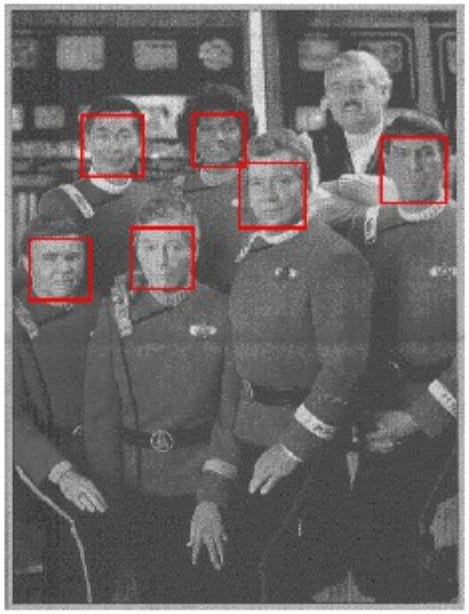

JUDYBATS
pain makes you beautiful

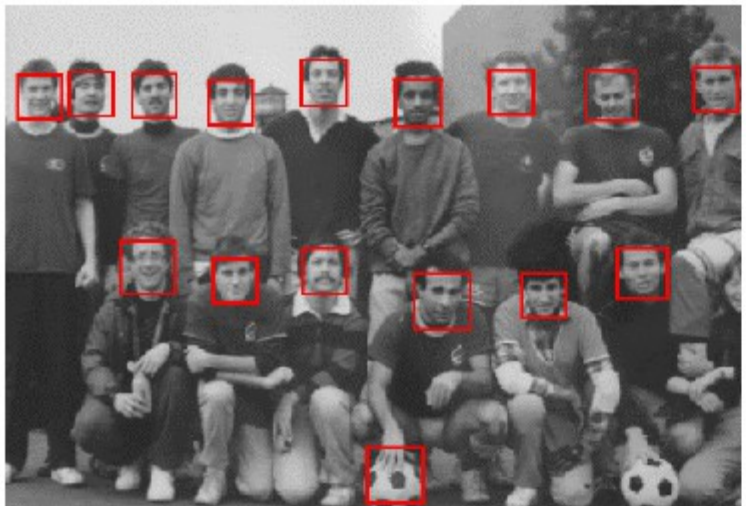


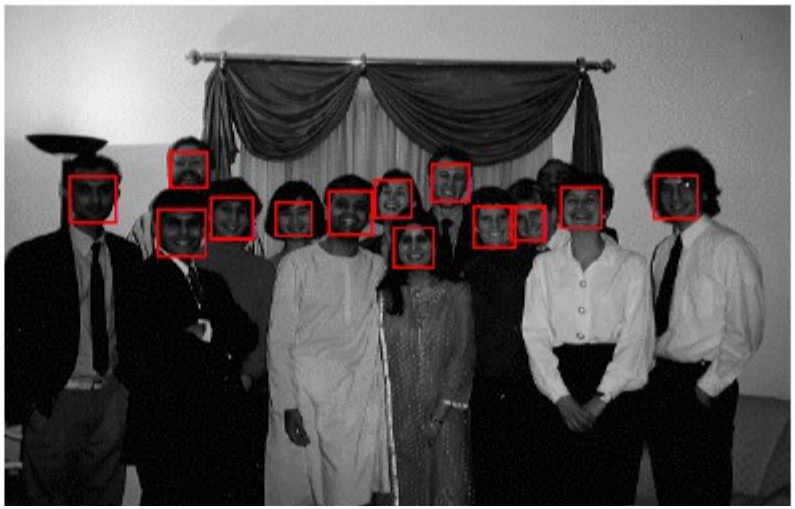




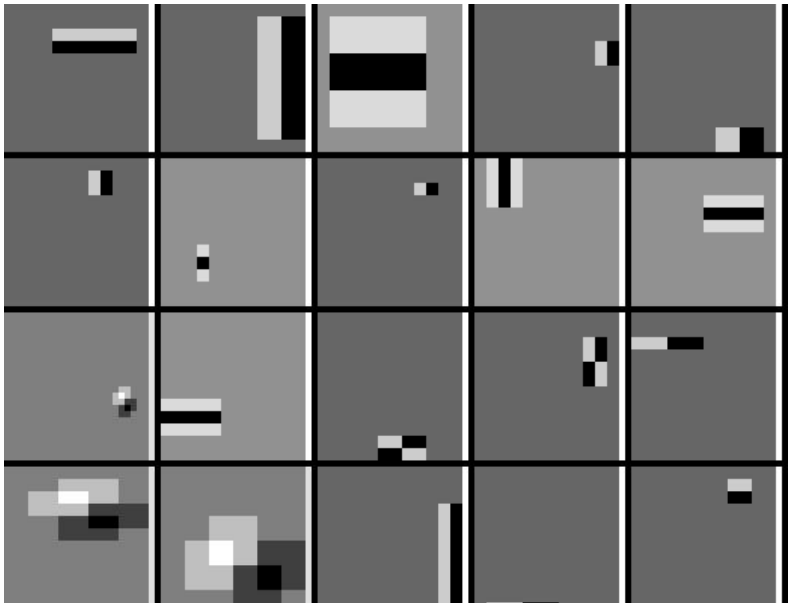


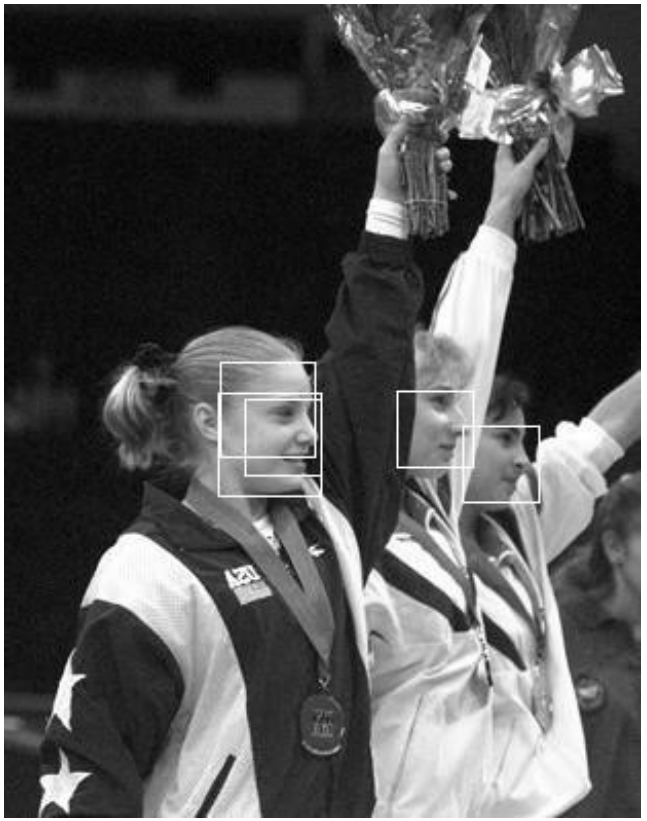






Detecting profile faces?





Computer Vision using Deep learning

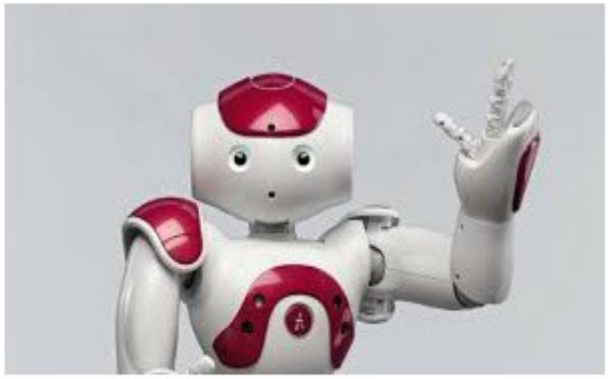
Dr Lijiya A
Assistant Professor
Department of Computer Science and Engineering
National Institute of technology Calicut

What is Computer Vision

- Enabling Computers/Machines to see and extract/interpret information from the visuals.

Self Driving Cars from Google

Unmanned Aerial Vehicles



Humanoid Robots

AI and Computer Vision

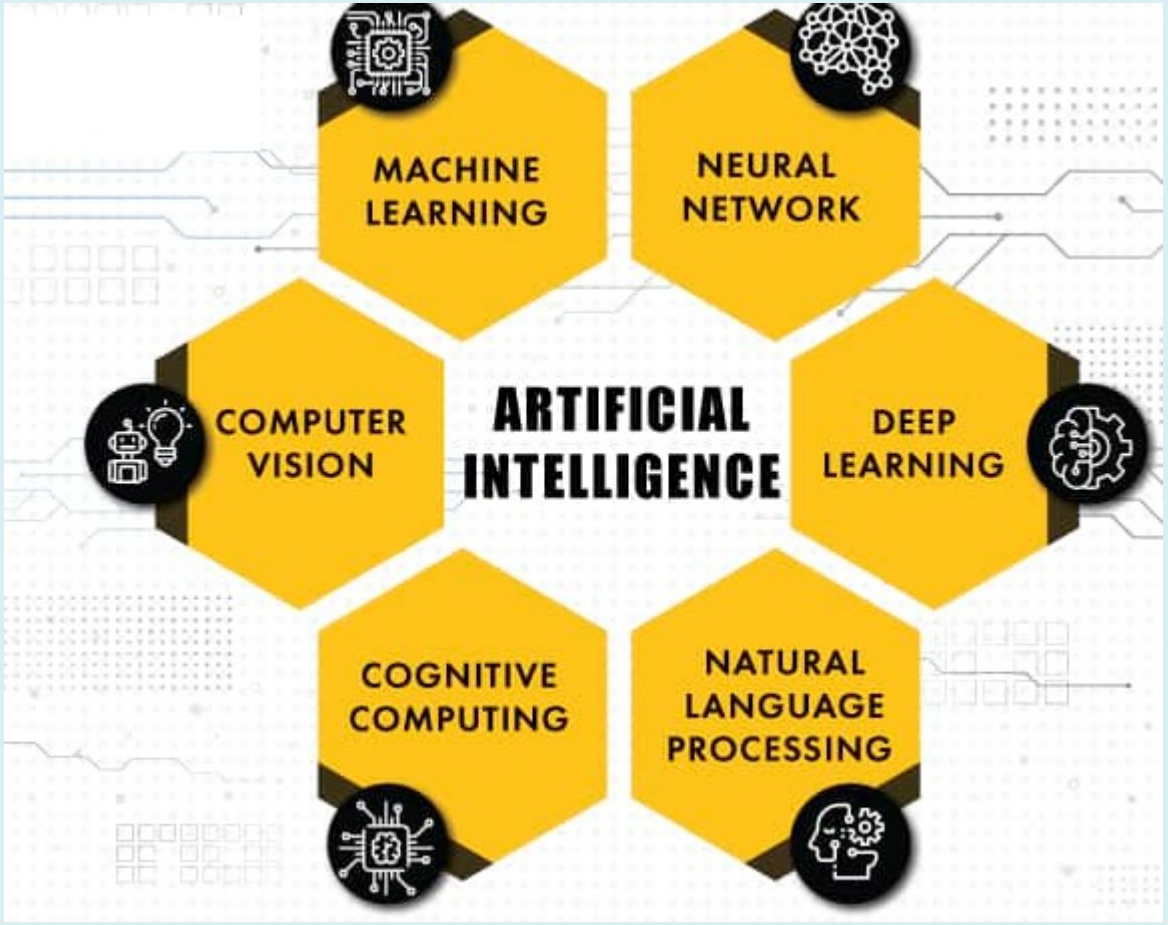


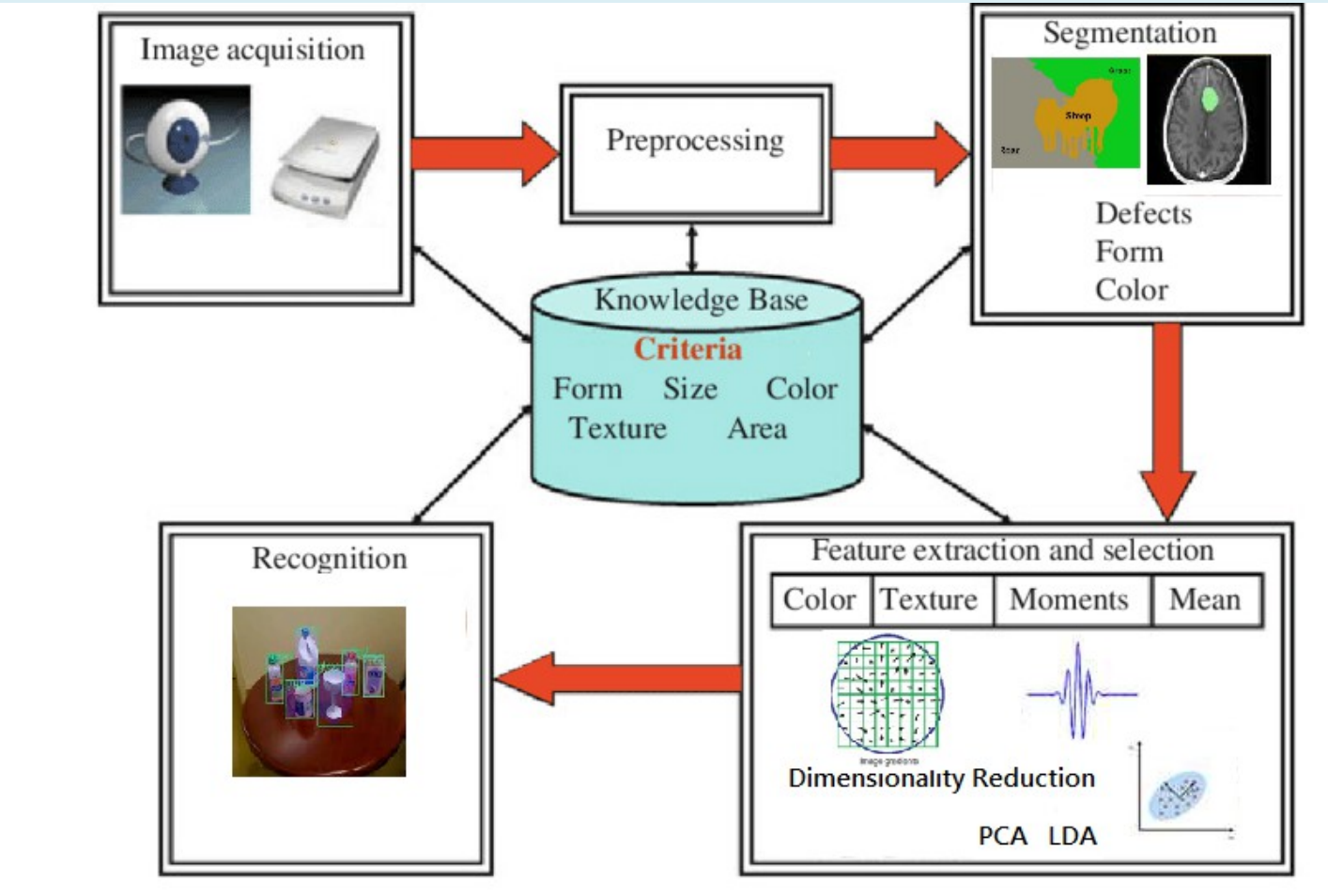
Image Processing Vs Computer Vision

Image Processing	Computer Vision
Input and output are Images	Input may be an image or video and output may be task specific knowledge inferred from the visual.
Focus is on processing the image	Focus is on understanding the scene.
Image Enhancement, segmentation, compression, Watermarking etc.	Verification, detection, labeling, prediction, measurement etc.

Applications of Computer Vision

- Optical character recognition (OCR)
- Machine inspection
- Retail (e.g. automated checkouts)
- 3D model building (photogrammetry)
- Medical imaging
- Automotive safety
- Motion capture
- Image Inpainting

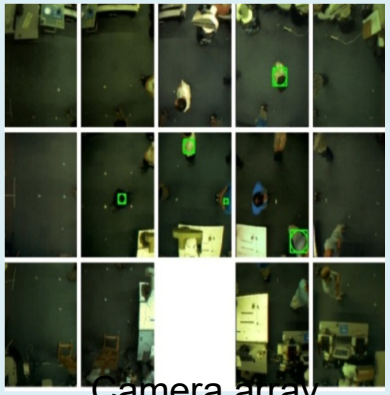
Different Stages in a CV system



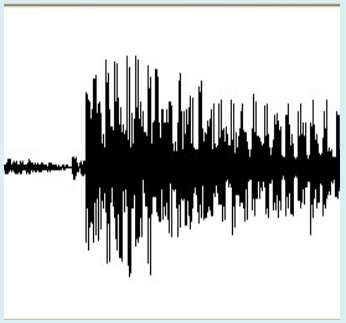
Data Acquisition :Computer vision is more than pictures

Images

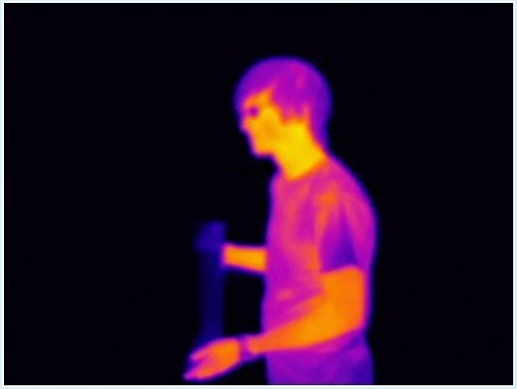
Video



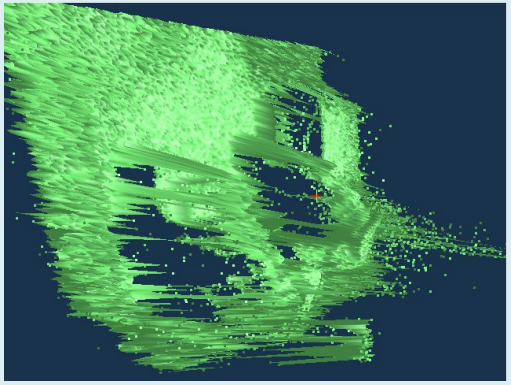
Camera array



Audio



Thermal Infrared



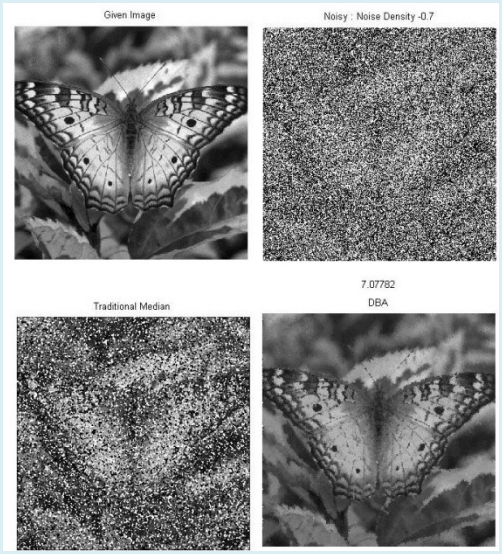
3d range scans
(flash lidar)

3d range scan

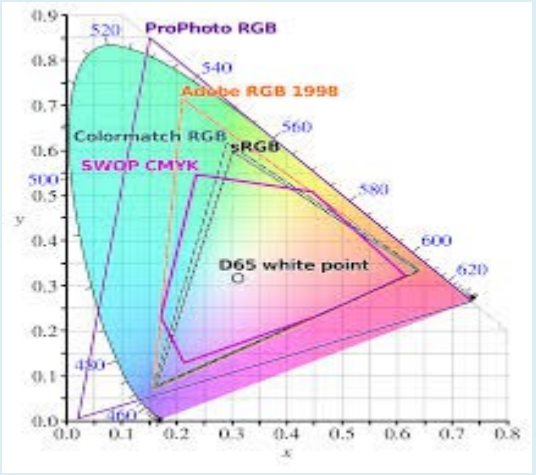
(laser scanner)

Preprocessing

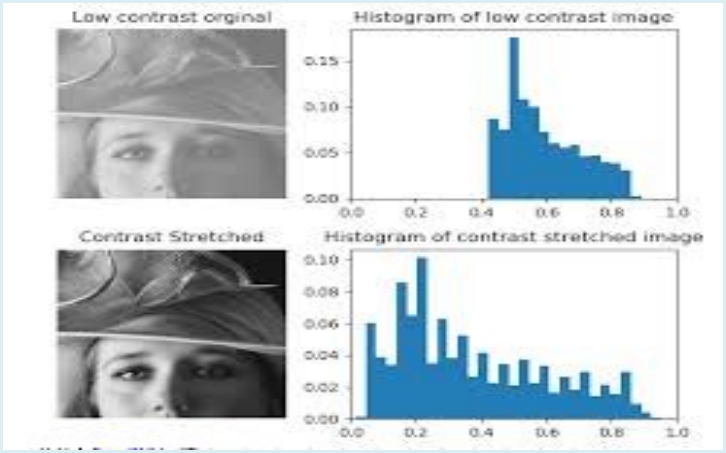
Noise Removal



Color Space Conversion



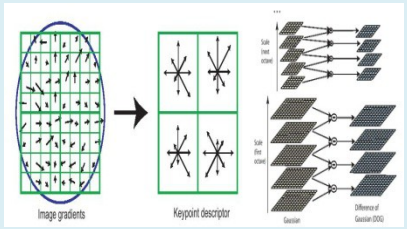
Contrast Strectching



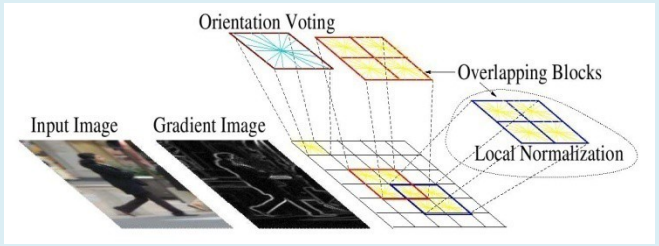
Feature Extraction

Feature Descriptor Should be:

- **Complete**
- **Invariant to affine transformations**
- **Compact**
- **Robust**
- **Low Computational Cost**



SIFT



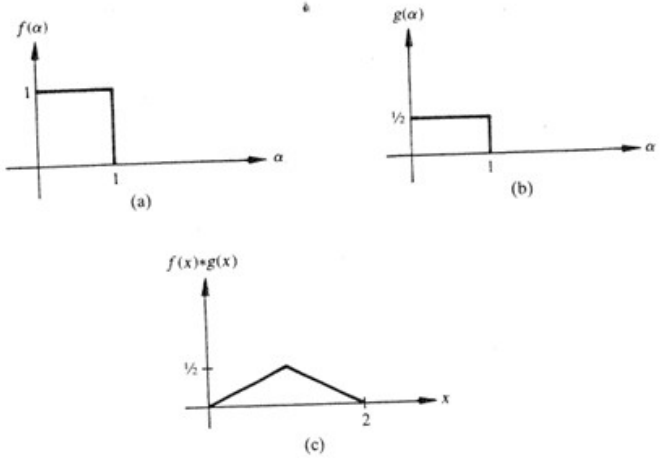
HoG

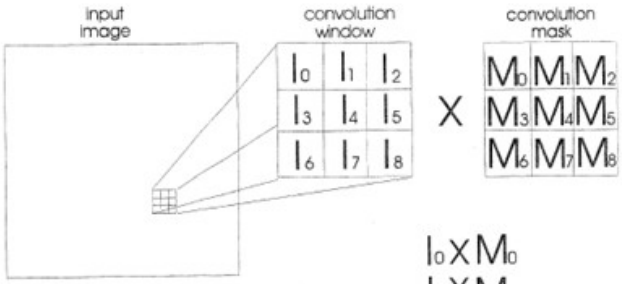
Feature	Description
Energy (ENR)	$ENR = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} G(i,j)^2$
Entropy (ENT)	$ENT = -\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} G(i,j)\, ln[G(i,j)]$
Contrast (CON)	$CON = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} G(i,j)(i-j)^2$
Difference entropy (DENT)	$DENT = -\sum_{i=0}^{n-1} G_{x-y}(i)\, ln[G_{x-y}(i)]$
Difference variance (DVAR)	$DVAR = -\sum_{i=0}^{n-1} G_{x-y}(i)(i-DENT)^2$
Maximum probability (MAXP)	$MAXP = MAX_{i,j} G(i,j)$
Sum entropy (SENT)	$SENT = -\sum_{i=2}^{2n} G_{x+y}(i)\, ln[G_{x+y}(i)]$
Sum average (SVAR)	$SVAR = \sum_{i=0}^{n-1} iG_{x+y}(i)$
Homogeneity (HOM)	$HOM = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} G(i,j)/(1+(i-j)^2$
Correlation (COR)	$COR = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} \frac{G(i,j)[(i-\mu_x)(j-\mu_y)}{\sigma_x\sigma_y}$

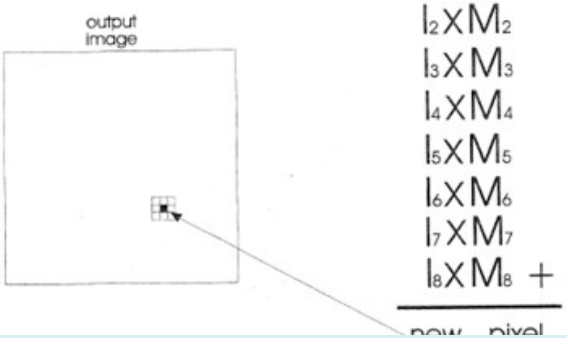
GLCM

Convolution

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(a)g(x-a)da$$



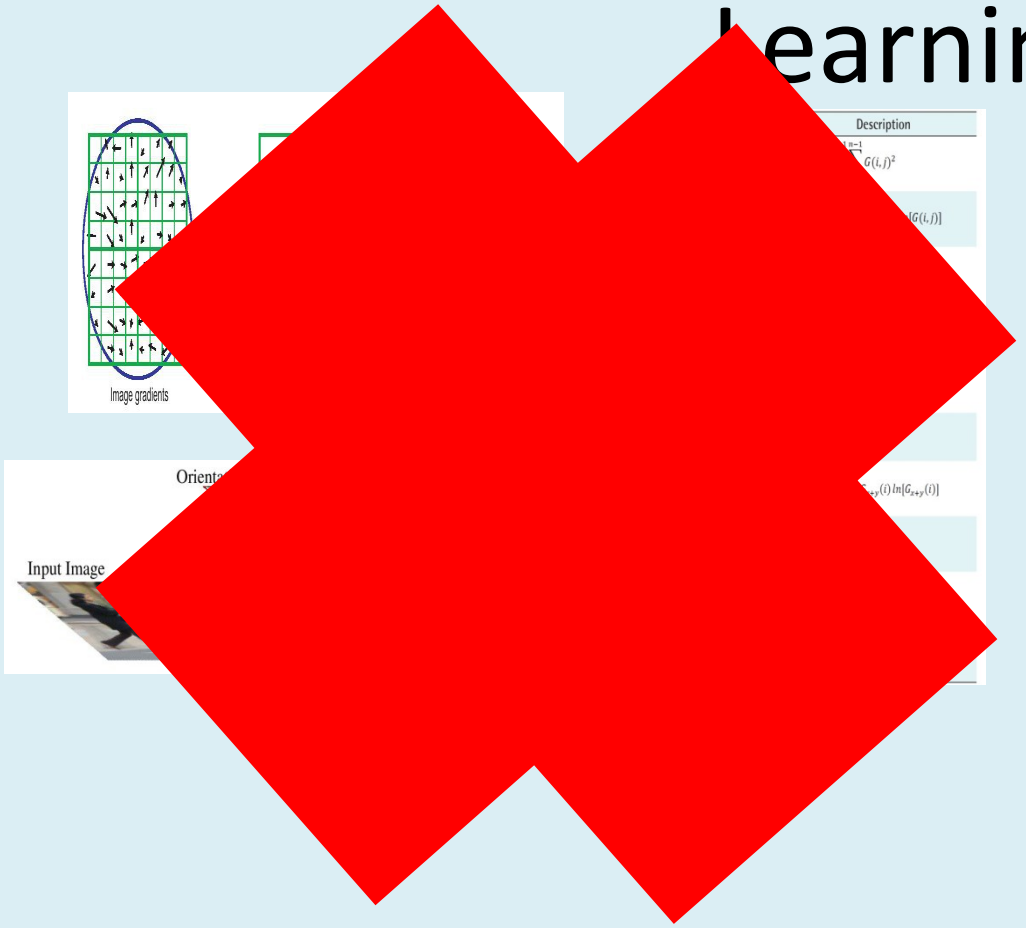


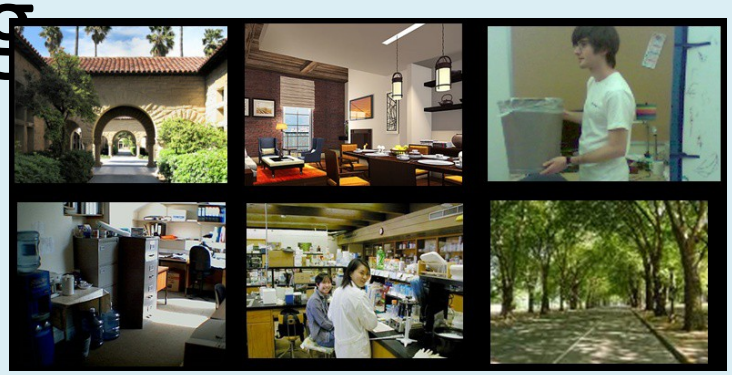


$$I_0 X M_0$$
$$I_1 X M_1$$
$$I_2 X M_2$$
$$I_3 X M_3$$
$$I_4 X M_4$$
$$I_5 X M_5$$
$$I_6 X M_6$$
$$I_7 X M_7$$
$$I_8 X M_8 +$$

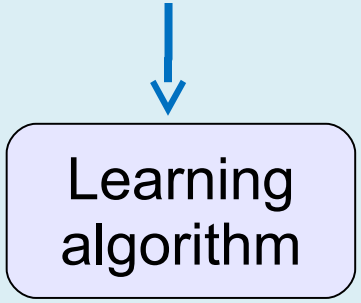
new pixel

The goal of Unsupervised Feature Learning





Unlabeled images

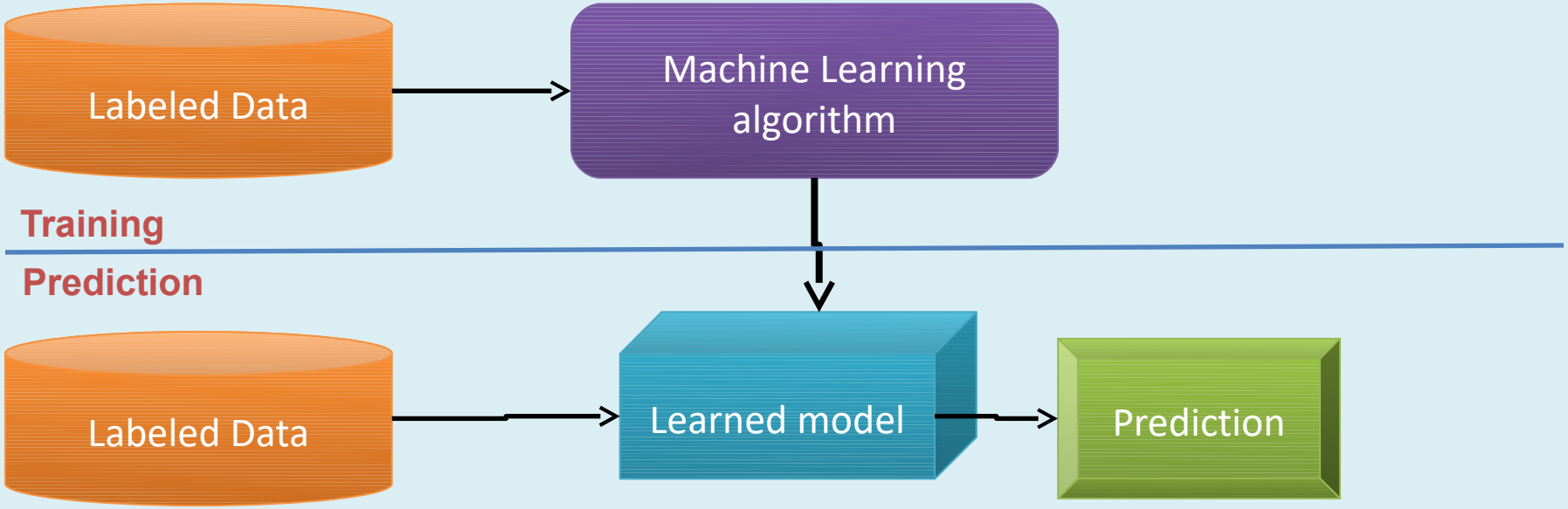


Feature representation

Can we automatically learn good feature representations?

Machine Learning Basics

Machine learning is a field of computer science that gives computers the ability to **learn new concepts mainly from the given set of examples without being explicitly programmed**



Methods that can learn from and make predictions on data

Types of Learning

Supervised: Learning with a **labeled training** set
Example: Object detection and classification with already labeled images as training set.
ANN, Bayes classifier, KNN, SVM, Decision trees.

Unsupervised: Discover **patterns** in **unlabeled** data
Clustering algorithms

Reinforcement learning: learn to **act** based on **feedback/reward**
Q learning

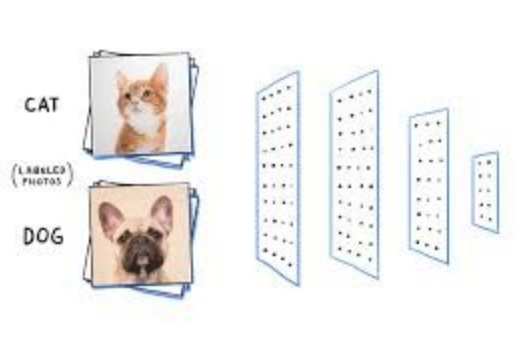


Image Classification

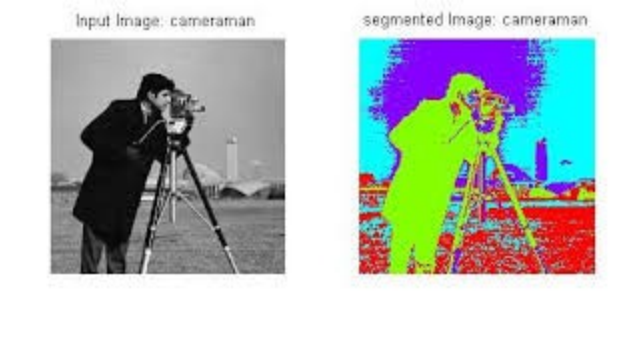
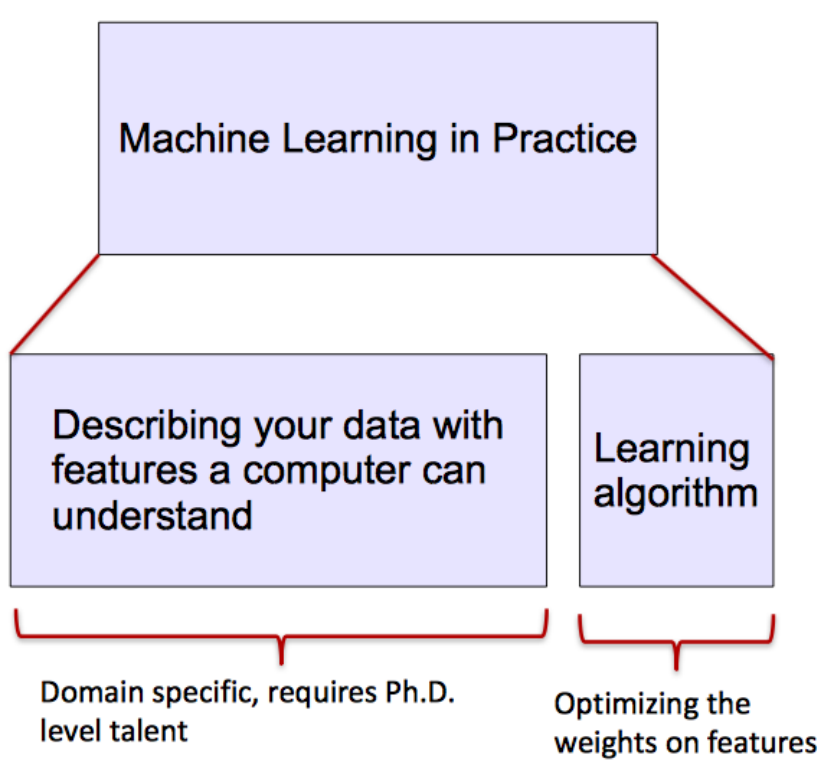


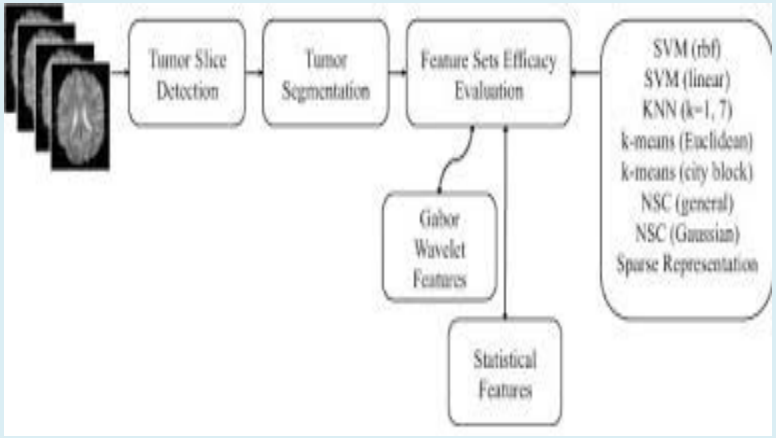
Image segmentation using clustering

Smart Traffic Signals in India using Deep Reinforcement Learning and Advanced Computer Vision

ML vs. Deep Learning

Most machine learning methods work well because of **human-designed representations** and **input features (hand crafted)**

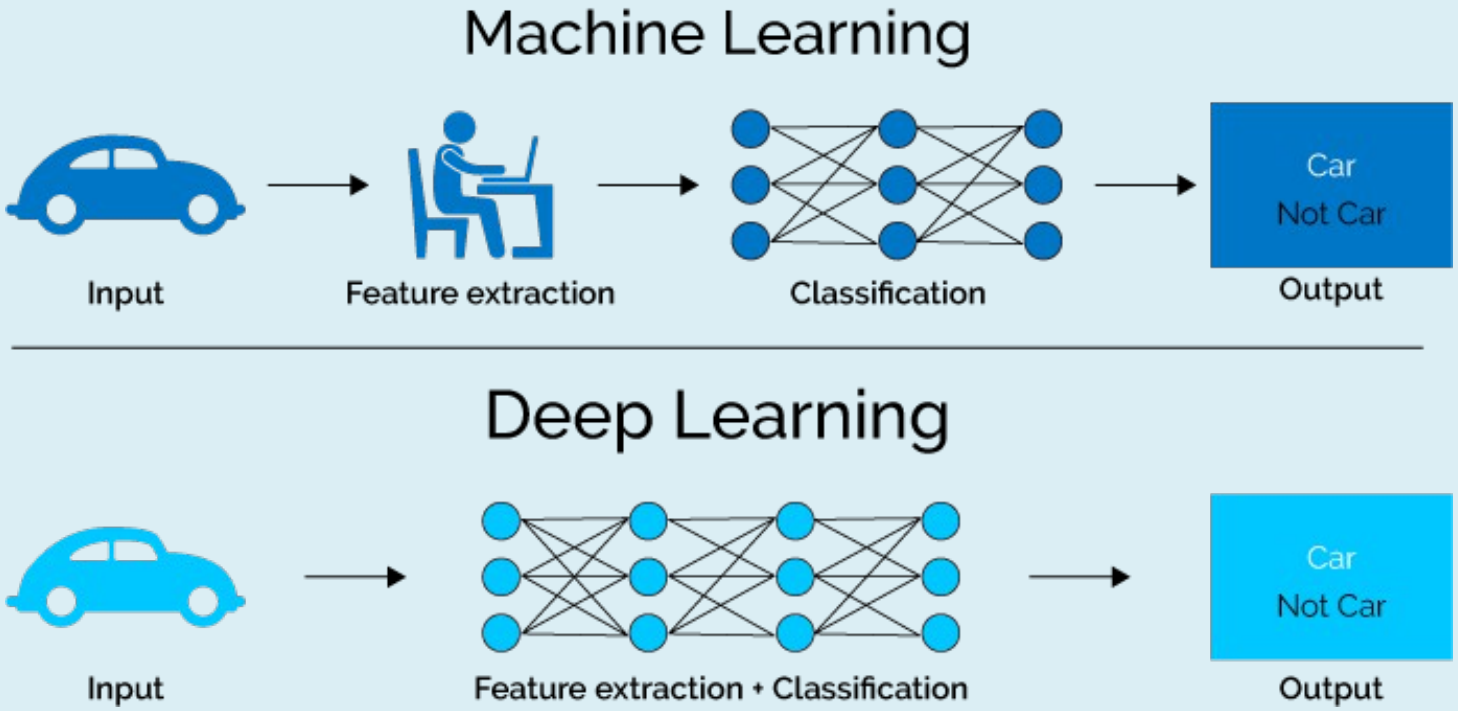




A tumor detection system using ML

What is Deep Learning (DL) ?

A subfield of machine learning capable of learning **representations** of data.
Exceptionally effective and independent at **learning patterns**.
Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers.**
If you provide the system **tons of information**, it begins to understand it and respond in useful ways.

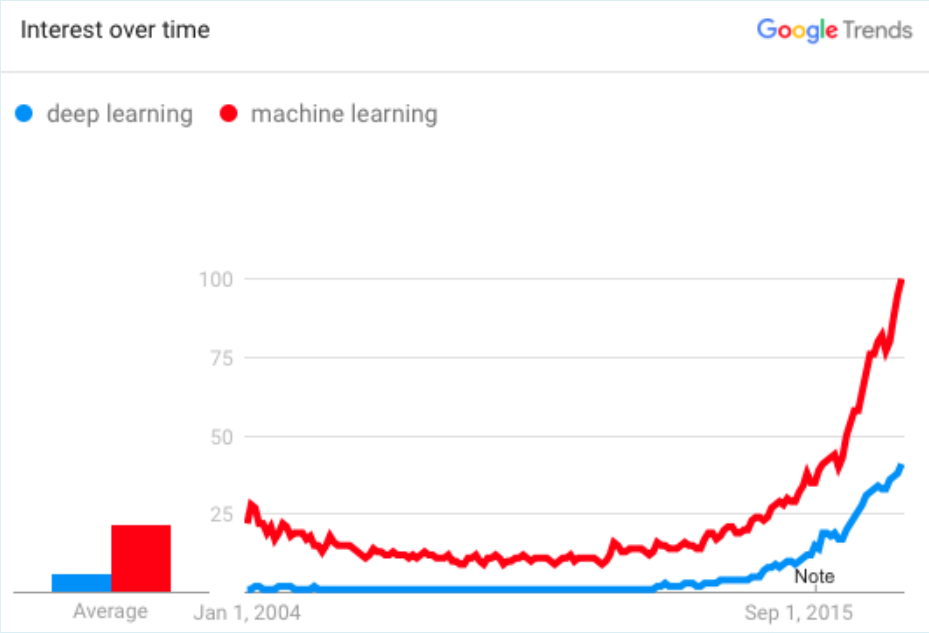


https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

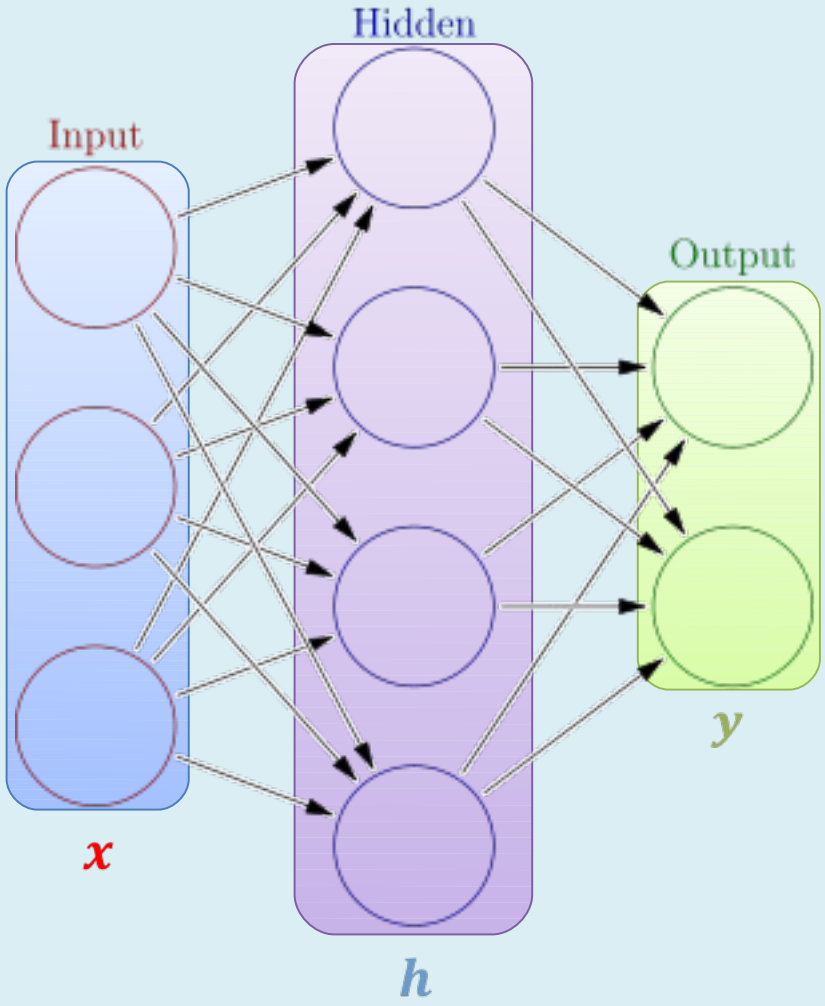
Why is DL useful?

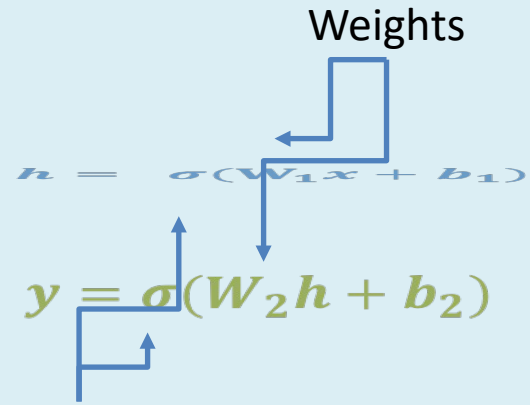
- Manually designed features are often **over-specified**, **incomplete** and take a **long time to design** and validate.
- Designer should have clear understanding of the problem and a strong foundation in IP and knowledge about the different Feature extraction techniques.
- Learned Features are **easy to adapt**, **fast** to learn
- Deep learning provides a very **flexible**, (almost?) **universal**, learnable framework for representing world, visual and linguistic information.
- DL requires large amounts of training data

In ~2010 DL started outperforming other ML techniques
first in speech and vision, then NLP



Neural Network Intro





Activation functions

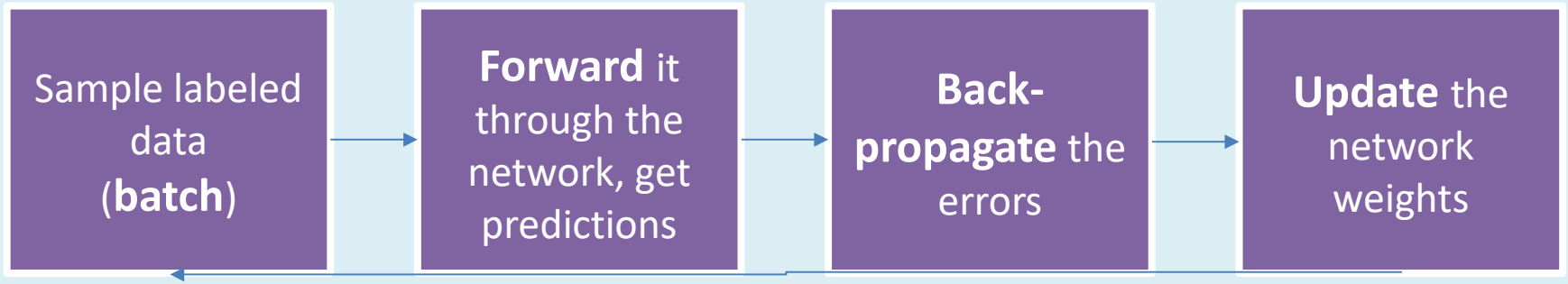
How do we train?

4 + 2 = 6 neurons (not counting inputs)
[3 x 4] + [4 x 2] = 20 weights
4 + 2 = 6 biases

26 learnable **parameters**

Demo

Training

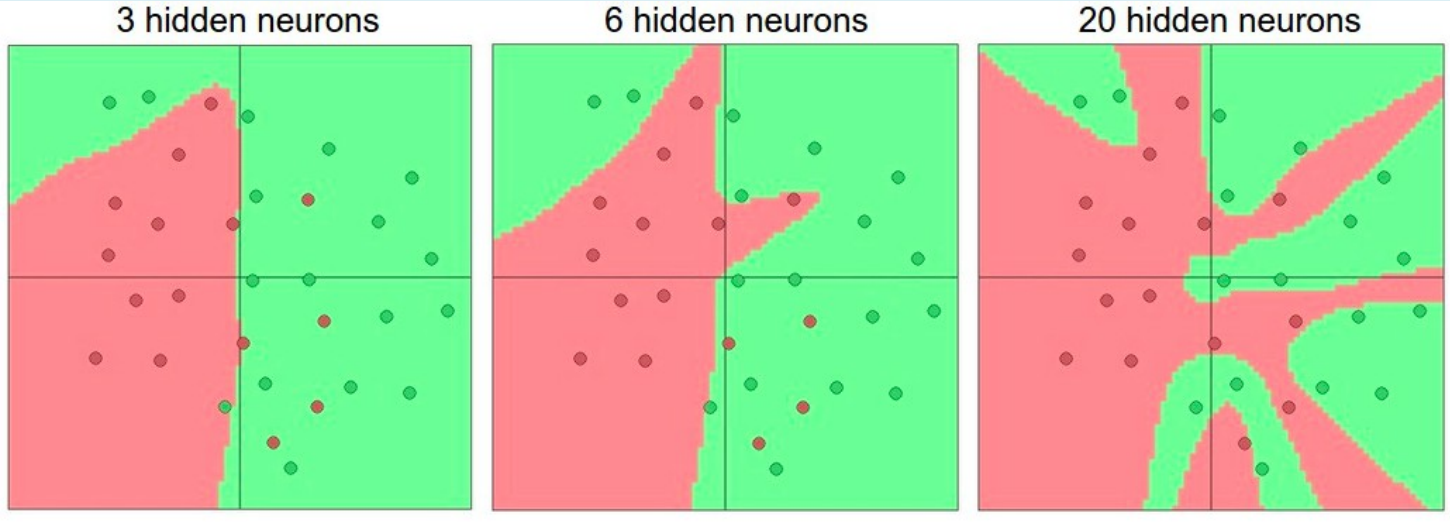


Optimize (min. or max.) **objective/cost function** $J(\theta)$
Generate **error signal** that measures difference between predictions and target values

Use error signal to change the **weights** and get more accurate predictions
Subtracting a fraction of the **gradient** moves you towards the **(local) minimum of the cost function**

Activation functions

Non-linearities needed to learn complex (non-linear) representations of data, otherwise the NN would be just a linear function $W_1 W_2 x = Wx$

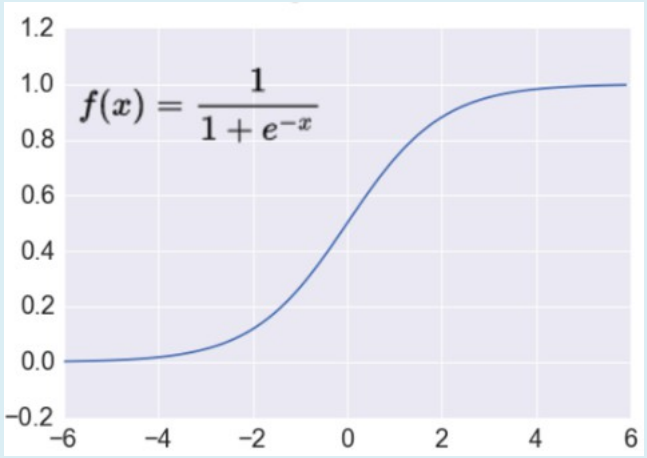


http://cs231n.github.io/assets/nn1/layer_sizes.jpeg

More layers and neurons can approximate more complex functions

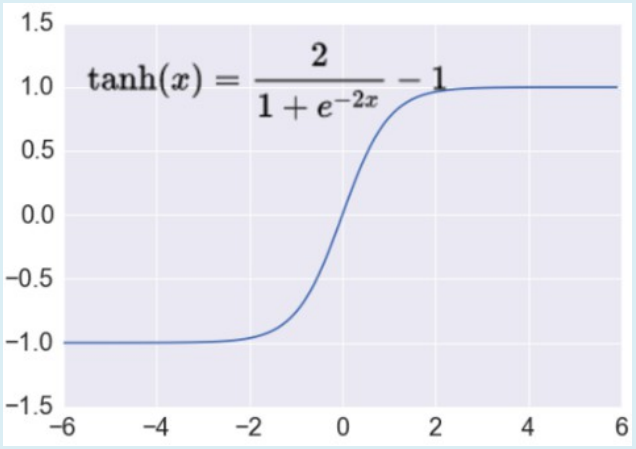
Full list: https://en.wikipedia.org/wiki/Activation_function

Activation Functions

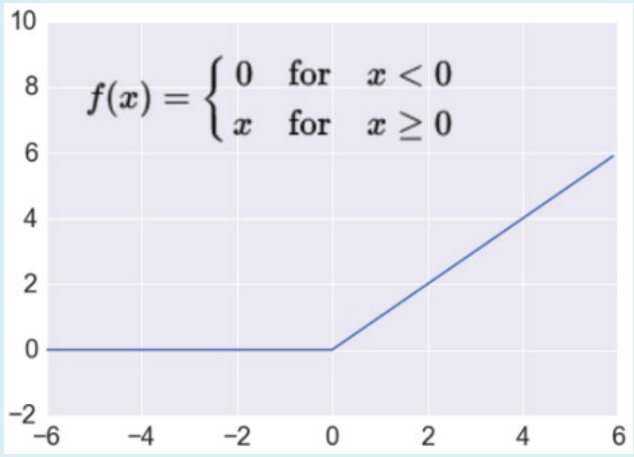


Takes a real-valued number and "squashes" it into range between 0 and 1

Sigmoid



Tanh Takes a real-valued number and "squashes" it into range between -1 and 1.

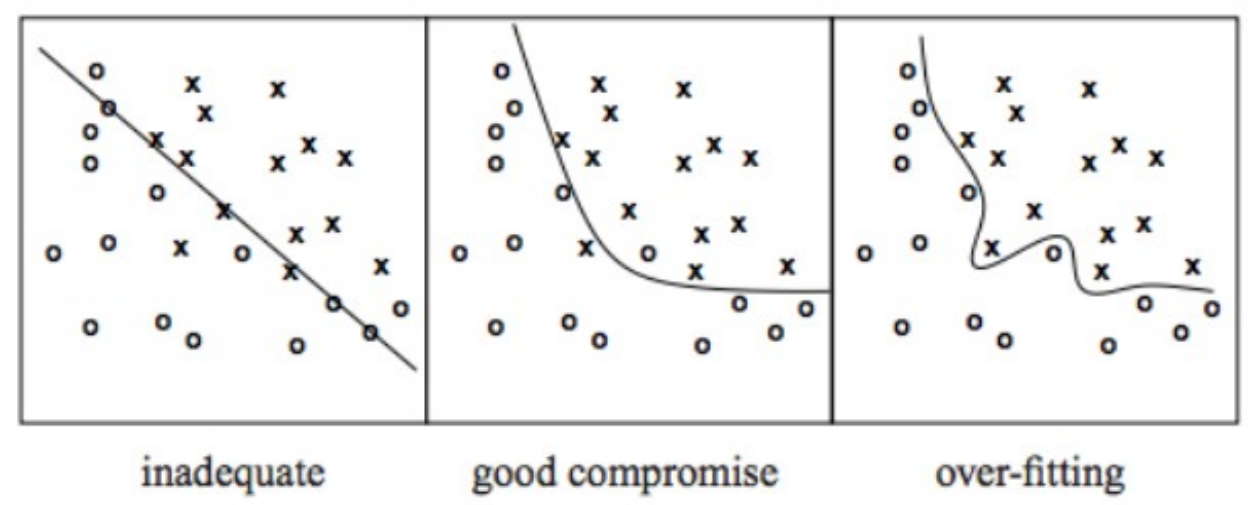


ReLU

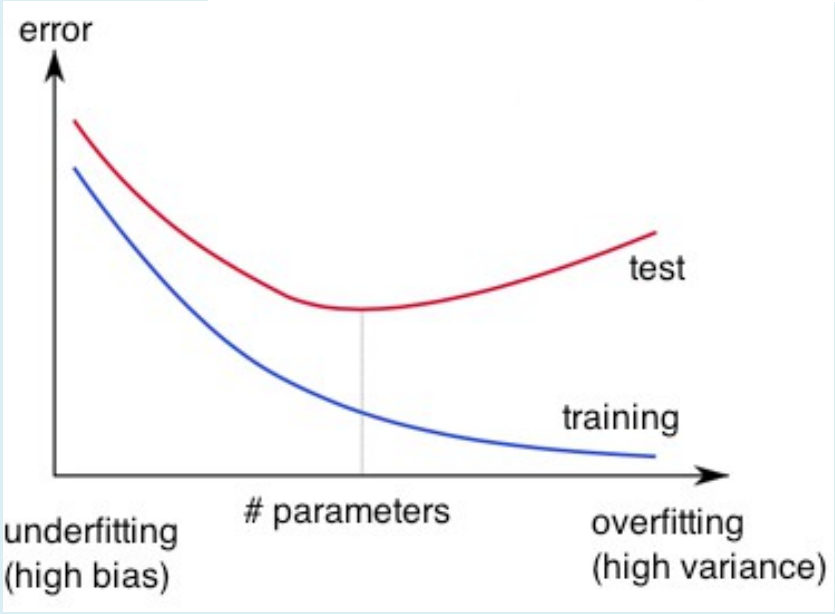
Takes a real-valued number and thresholds it at zero

http://adilmoujahid.com/images/activation.png

Overfitting



http://wiki.bethanycrane.com/overfitting-of-data



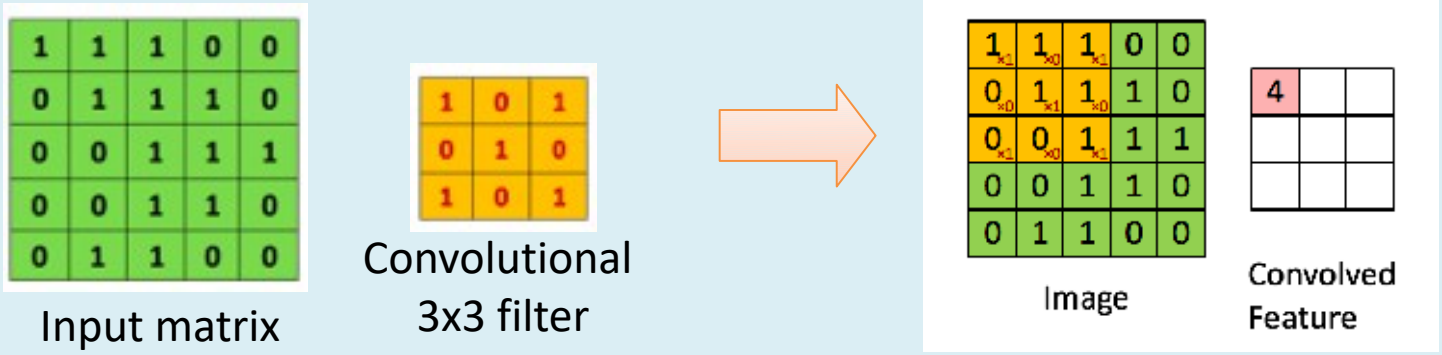
https://www.neuraldesigner.com/images/learning/selection_error.svg

Learned hypothesis may **fit** the training data very well, even outliers (**noise**) but fail to **generalize** to new examples (test data)

Convolutional Neural Networks (CNNs)

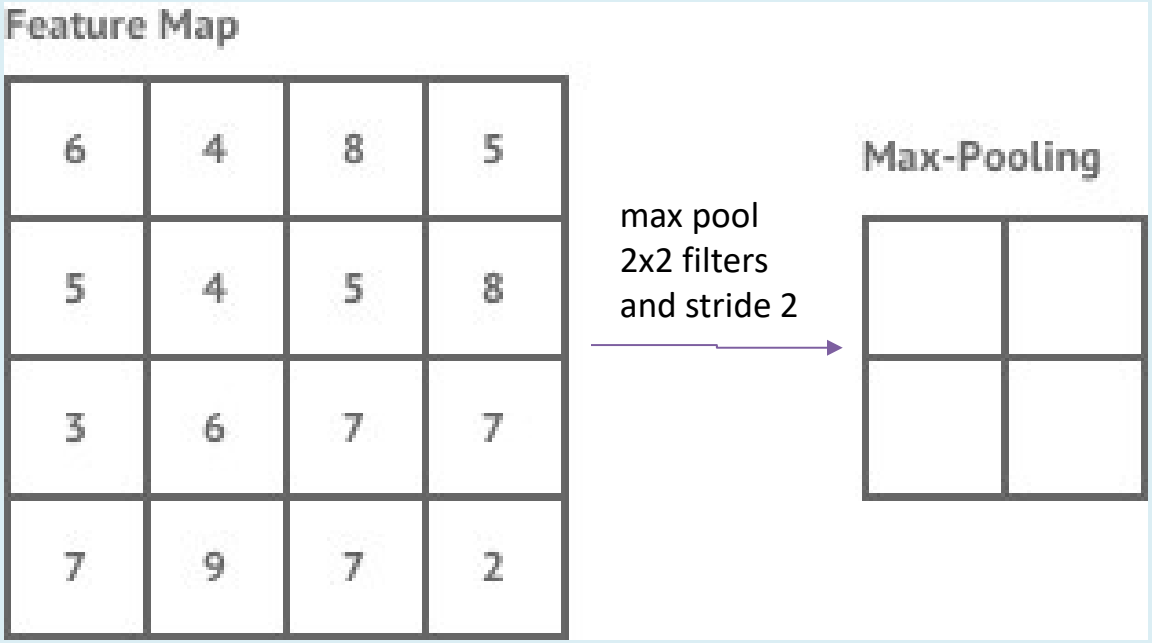
Main CNN idea for text:
Compute vectors for n-grams and group them afterwards

Example: “this takes too long” compute vectors for:
This takes, takes too, too long, this takes too, takes too long, this takes too long



http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

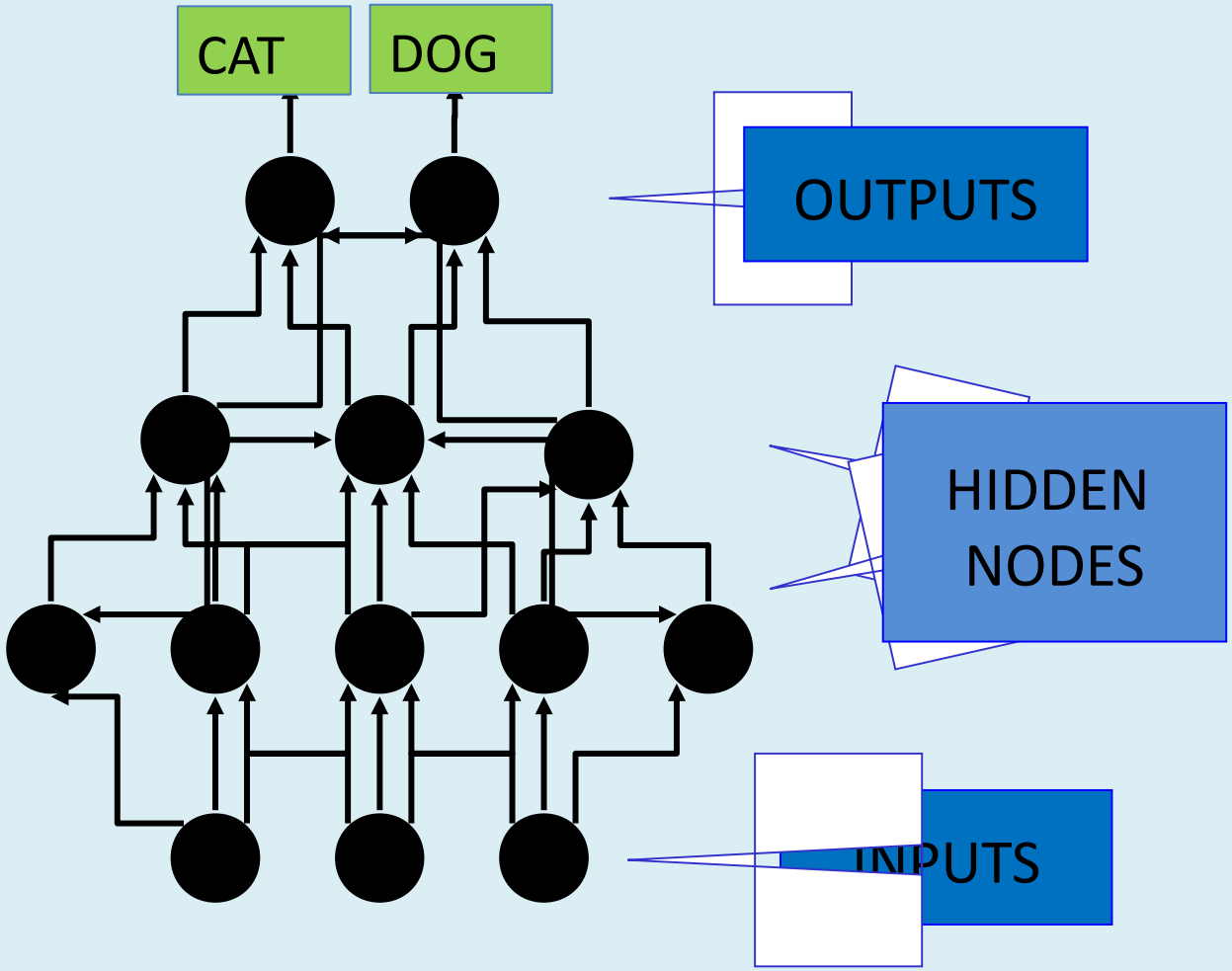
Convolutional Neural Networks (CNNs)



https://shafeentejani.github.io/assets/images/pooling.gif

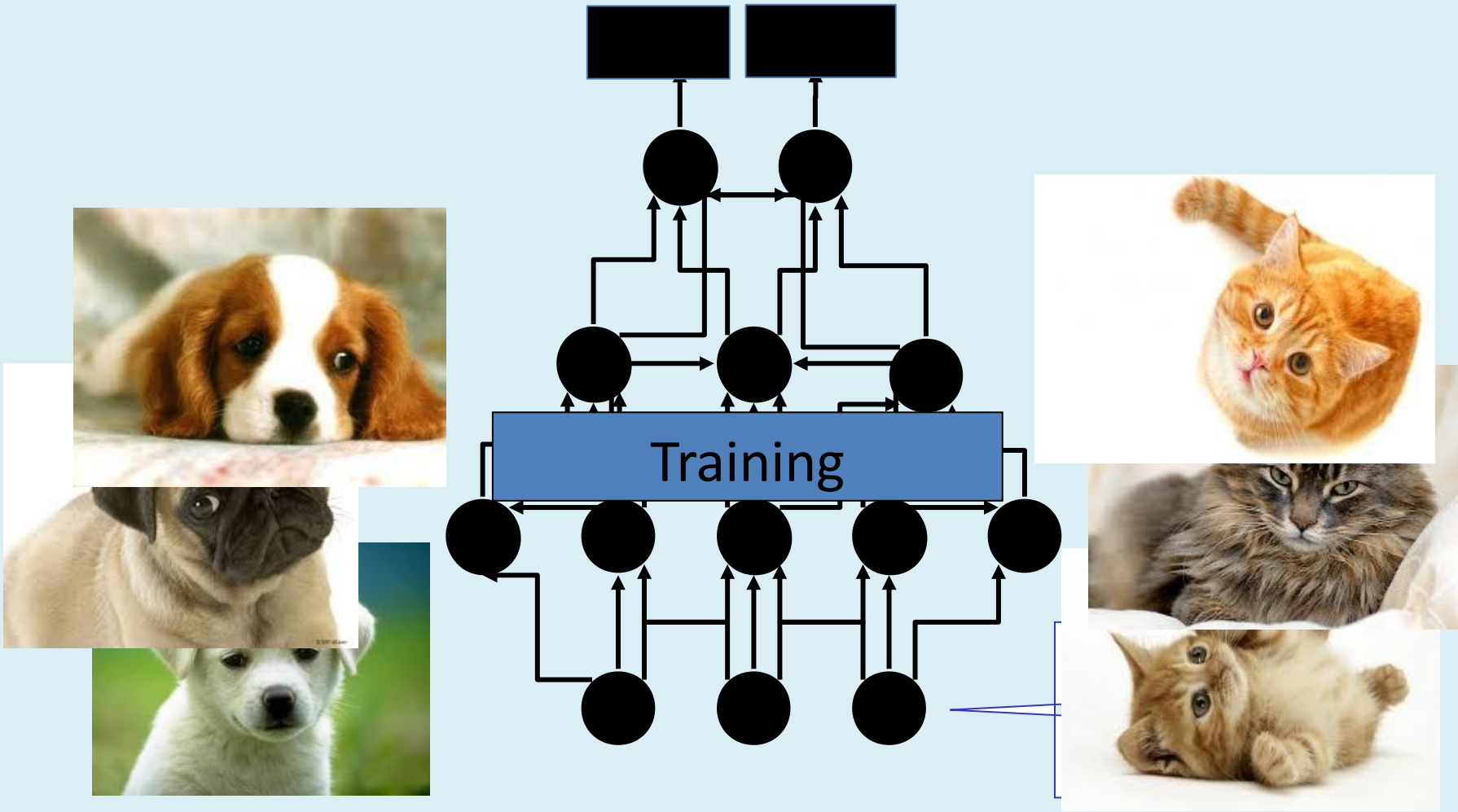
Deep Learning Basics

Deep Learning – is a set of machine learning algorithms based on multi-layer networks

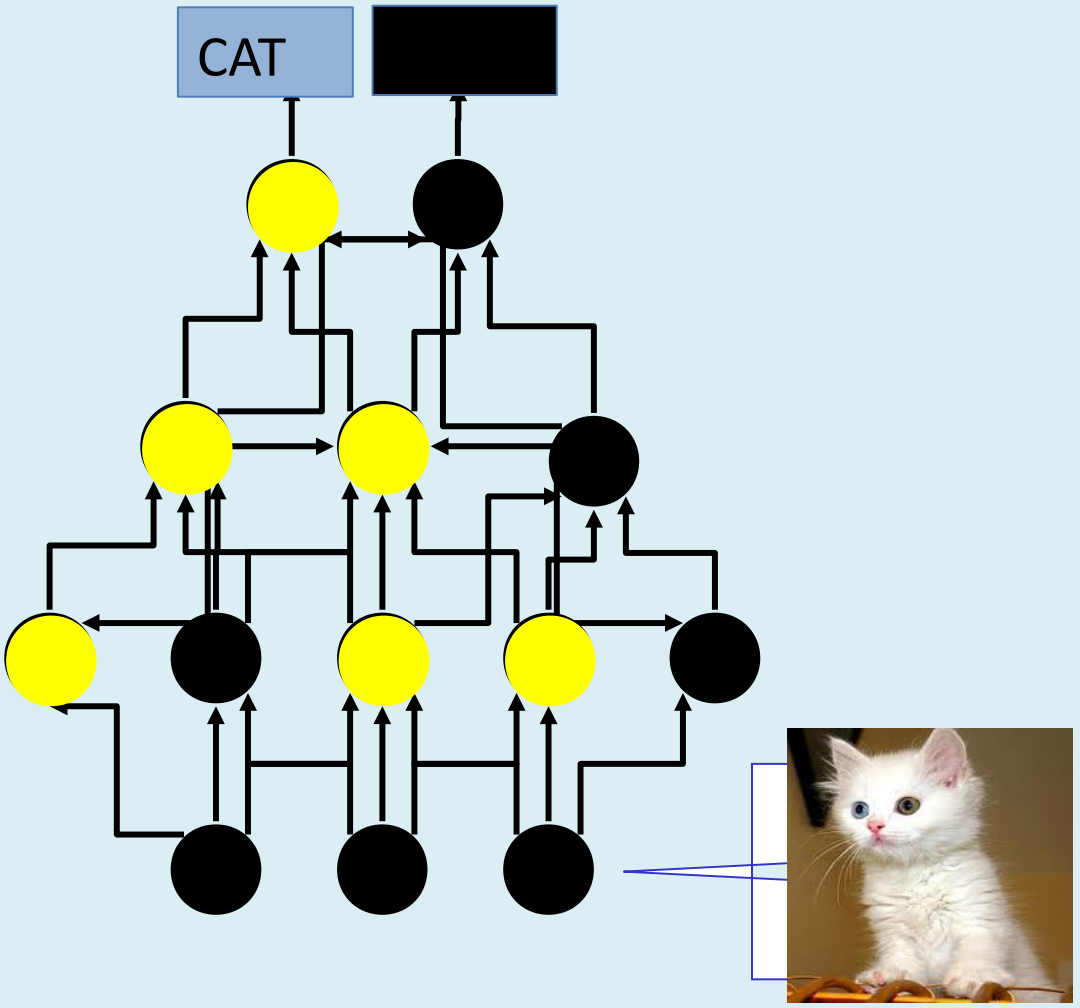


Deep Learning Basics

Deep Learning – is a set of machine learning algorithms based on multi-layer networks

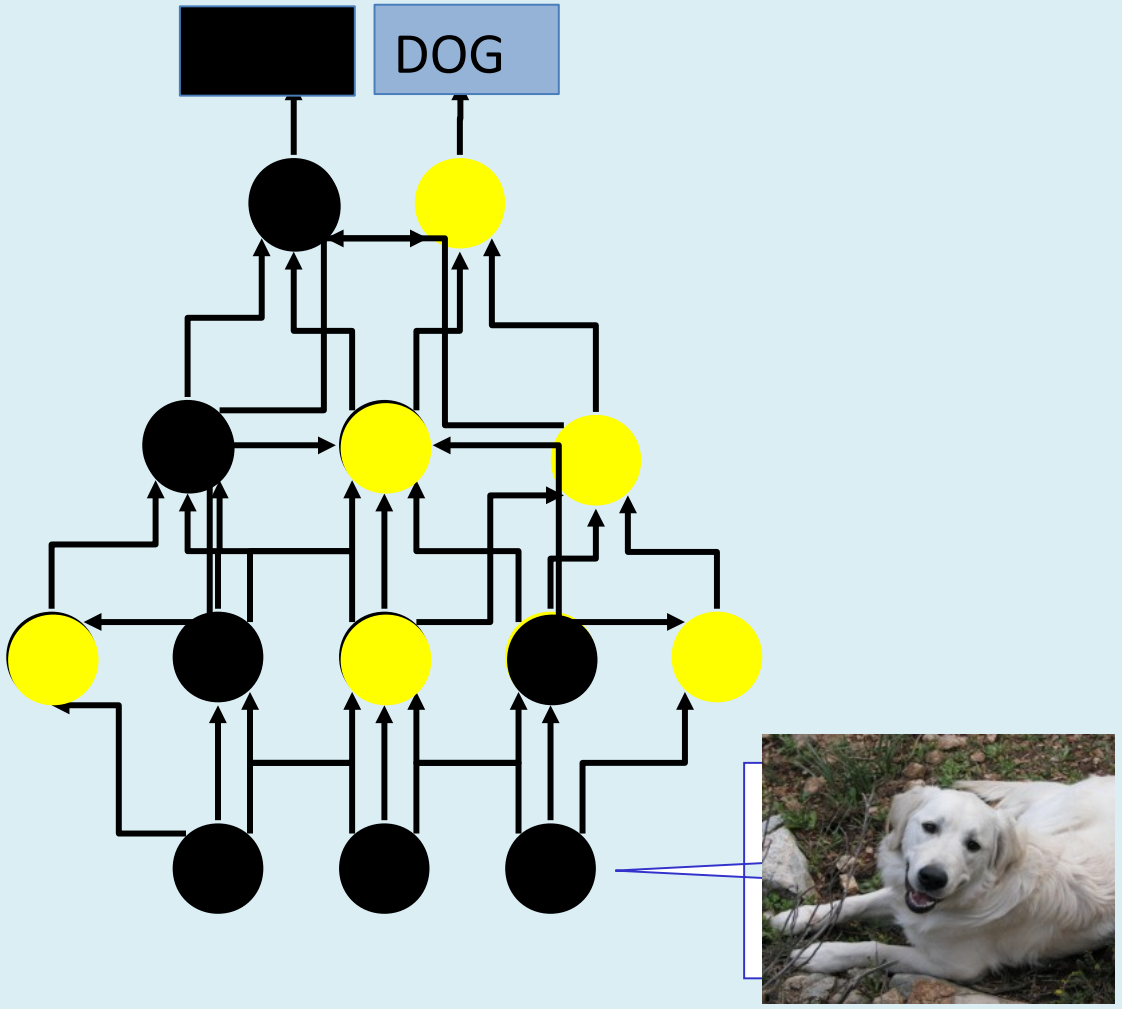


Deep Learning Basics



Deep Learning Basics

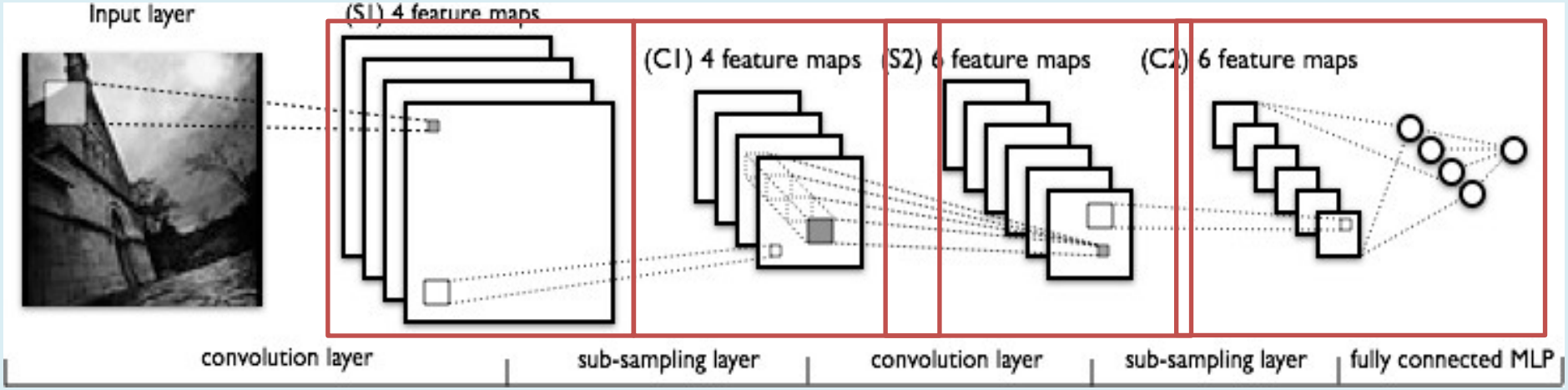
Deep Learning – is a set of machine learning algorithms based on multi-layer networks



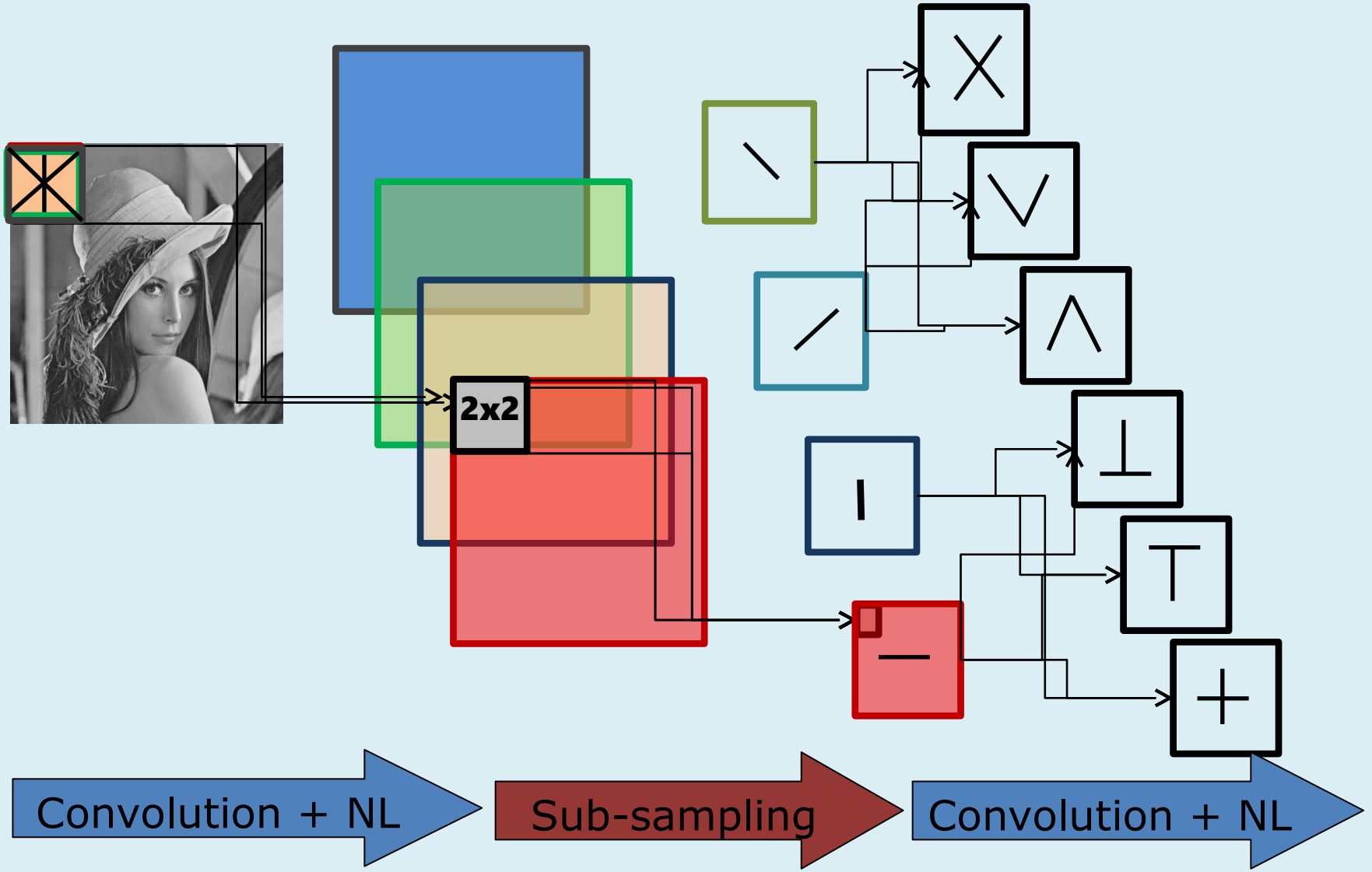
What is Convolutional NN ?

CNN - multi-layer NN architecture

- Convolutional + Non-Linear Layer
- Sub-sampling Layer
- Convolutional +Non-L inear Layer
- Fully connected layers

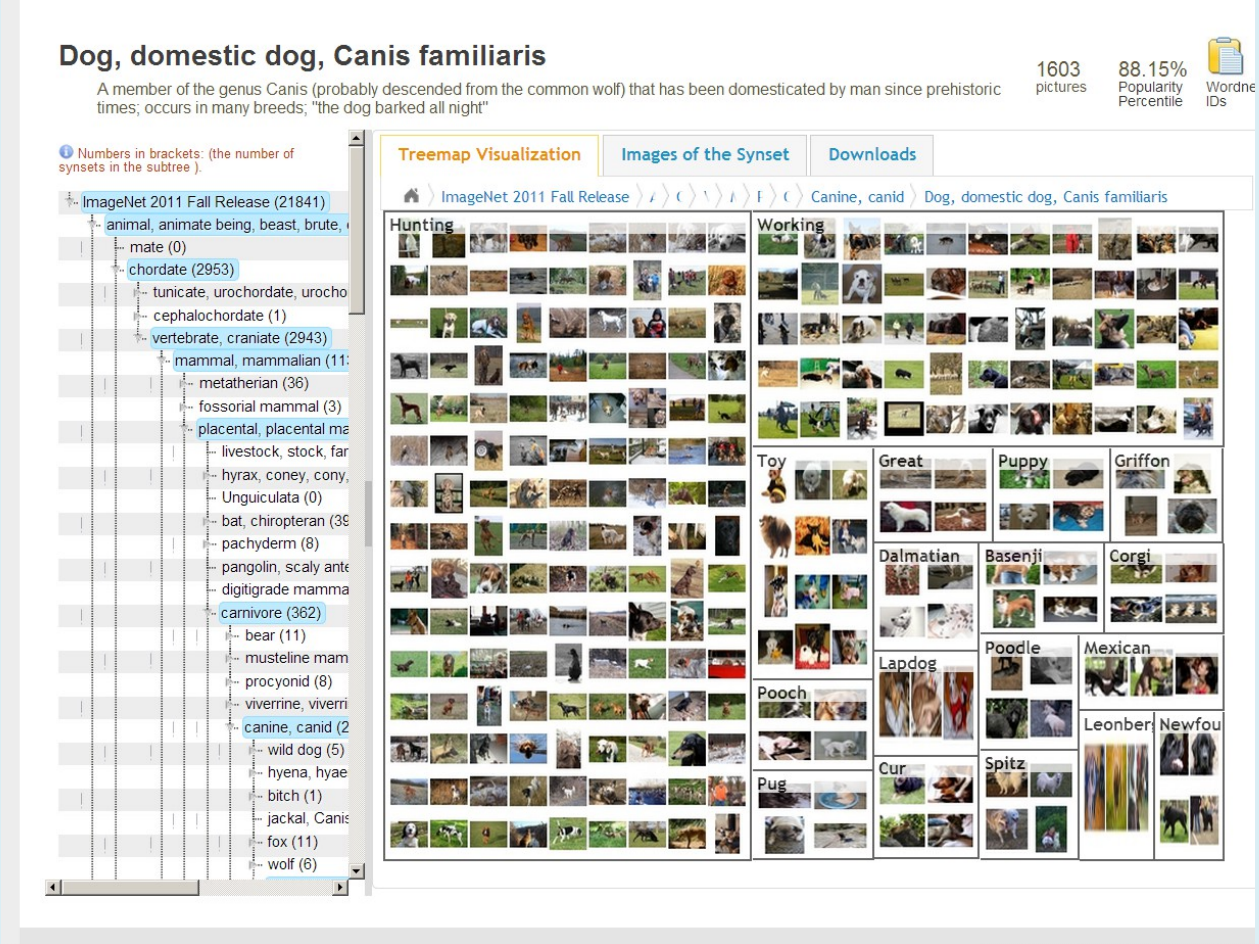


What is Convolutional NN ?



ImageNet Database

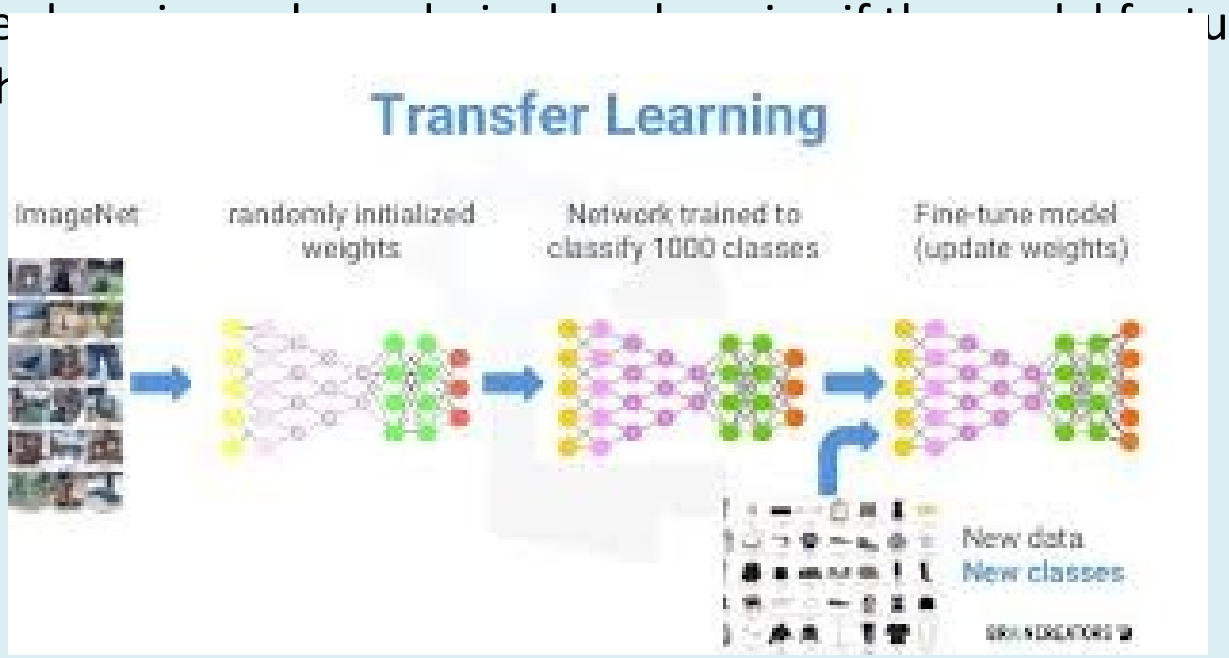
Imagenet data base: 14 mln labeled images, 20K categories



http://www.image-net.org/

Transfer learning

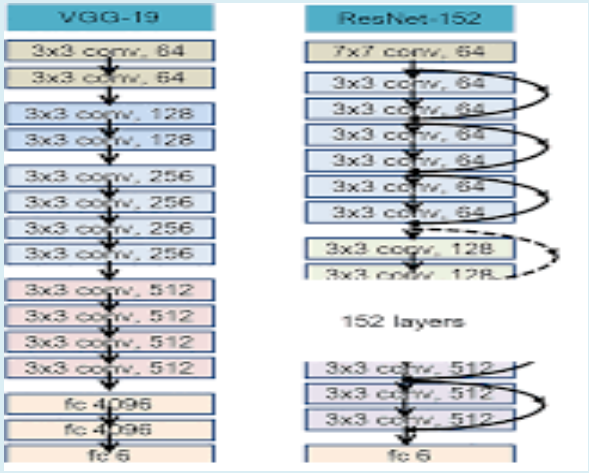
- Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task.
- Transfer learning is an optimization that allows rapid progress or improved performance when modeling the second task.
- Transfe[illegible]ures learned from th[illegible]



https://machinelearningmastery.com/transfer-learning-for-deep-learning/

Transfer learning Approaches in DL

- **Develop Model Approach**
  - Select Source Task.
  - Develop Source Model.
  - Reuse Model.
  - Tune Model.
- **Pre-trained Model Approach**
  - Select Source Model.
  - Reuse Model.
  - Tune Model..



Three examples of most popular Pre-trained models include:
Oxford VGG Model
Google Inception Model
Microsoft ResNet Model

Stereo Vision

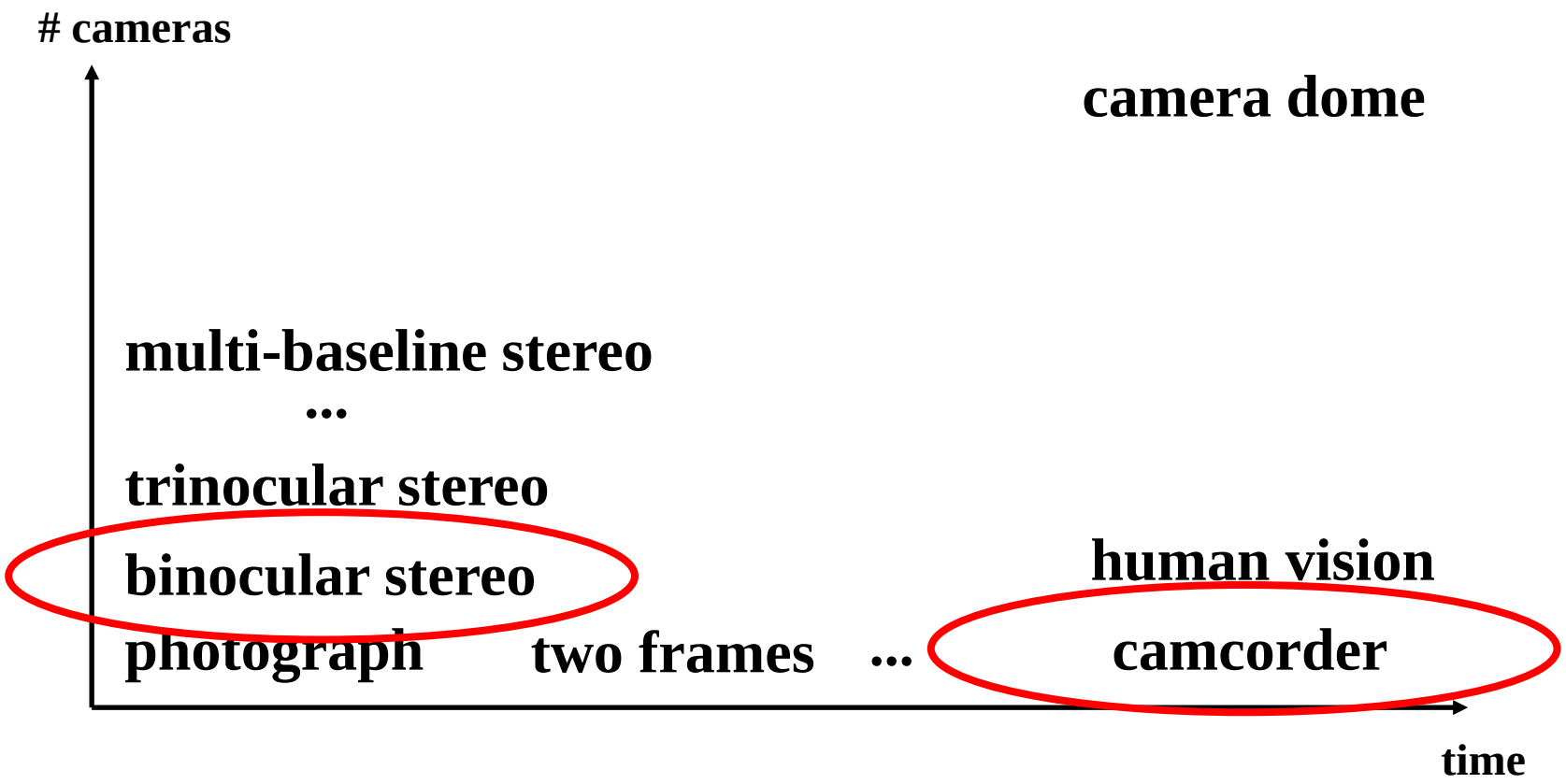
ECE 847:
Digital Image Processing

Stan Birchfield
Clemson University

Outline

- Stereo basics
- Binocular stereo matching
- Advanced stereo techniques

Modeling from multiple views



στερεος – Greek for solid

Stereoscope

Invented by Wheatstone in 1838

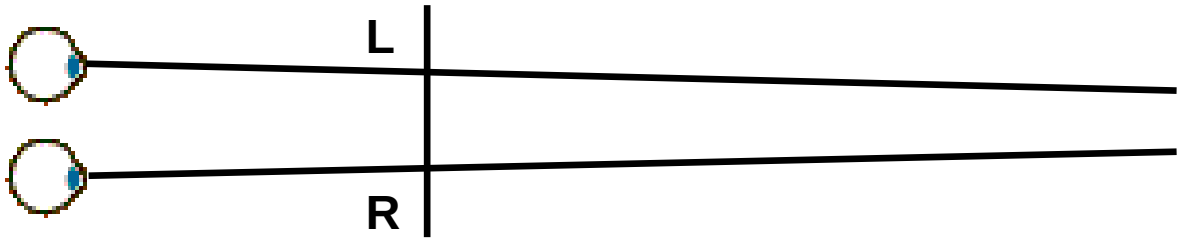
Modern version

Can you fuse these?

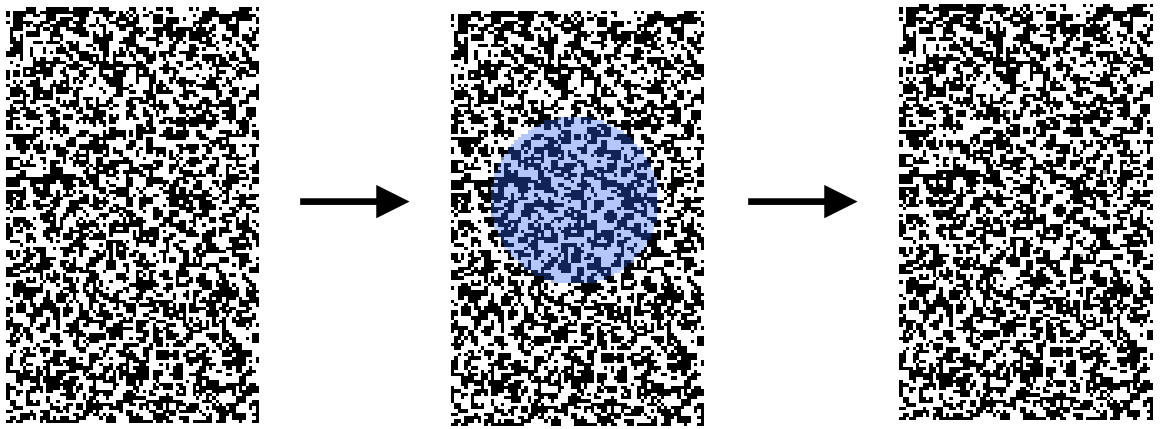
left **right**

No special instrument needed

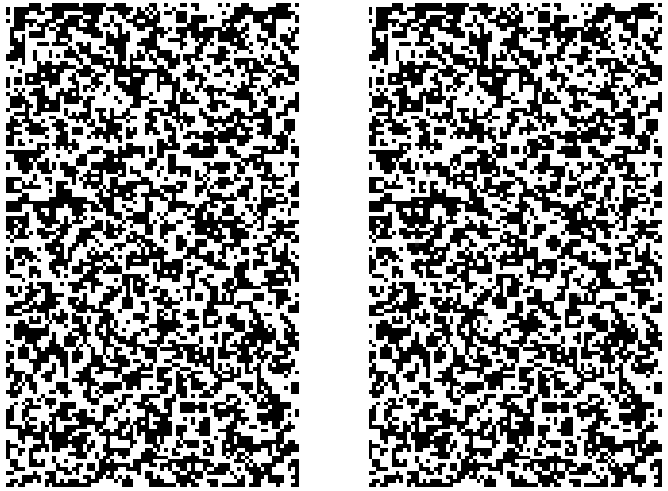
Just relax your eyes



Random dot stereogram

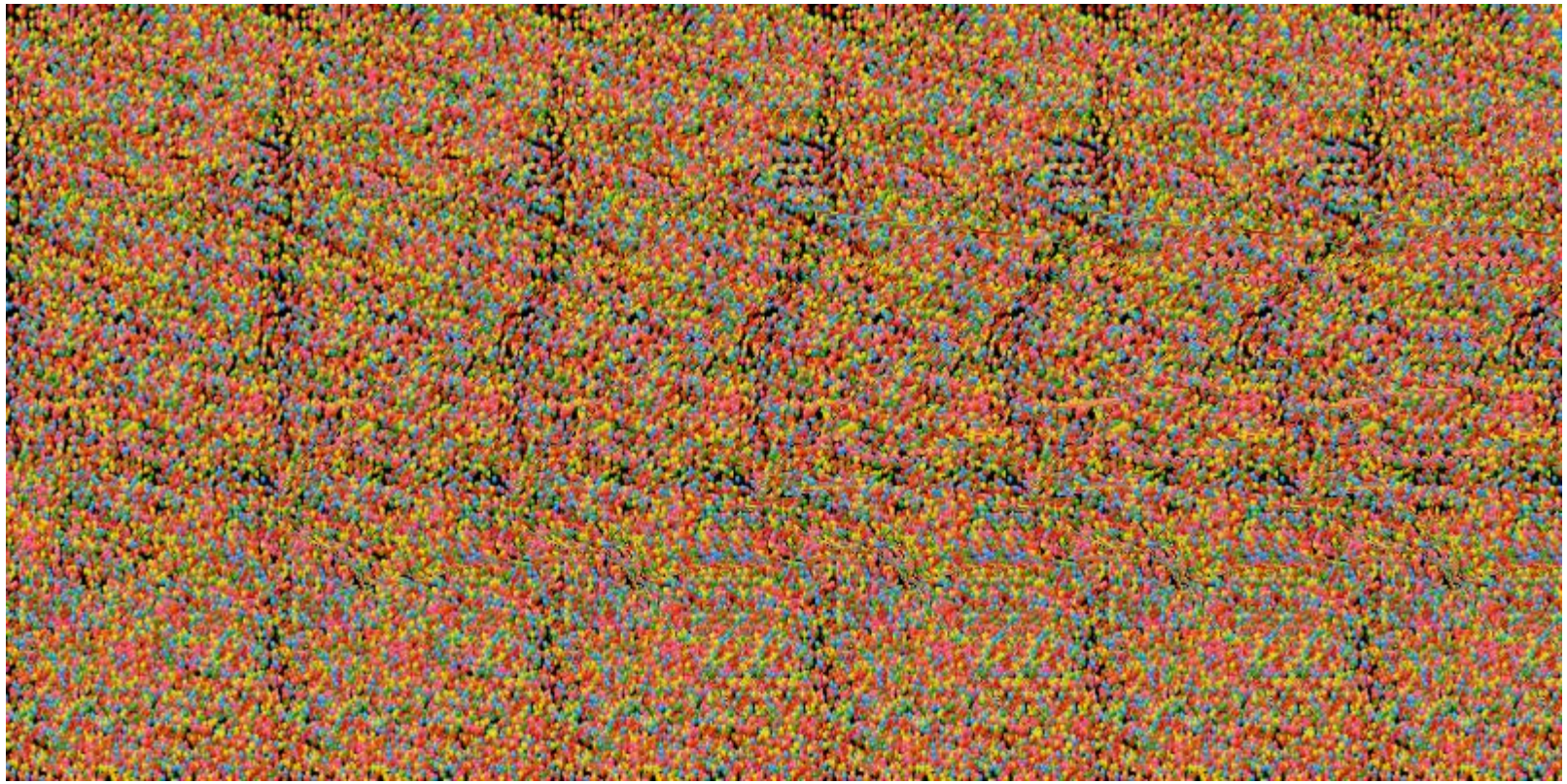


invented by Bela Julesz in 1959



http://www.magiceye.com/faq.htm

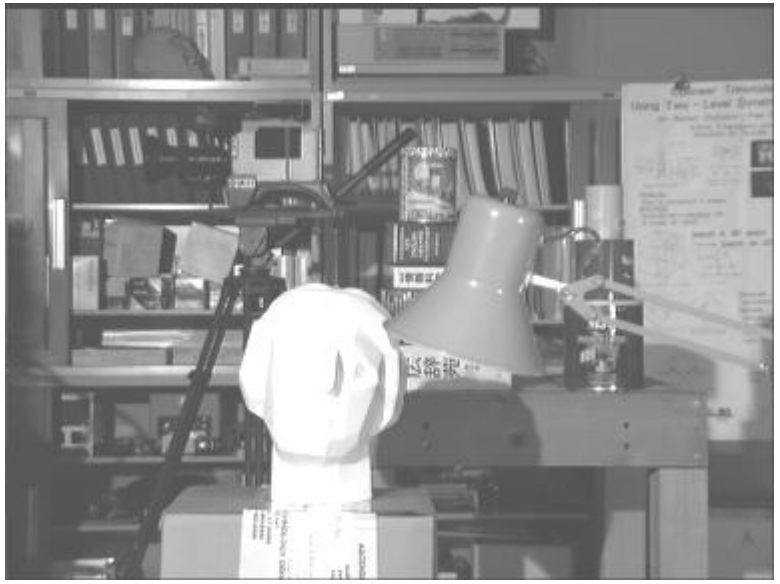
Autostereogram



Do you see the shark?

http://en.wikipedia.org/wiki/Autostereogram

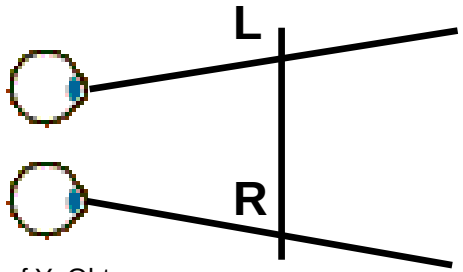
Can you cross-fuse these?



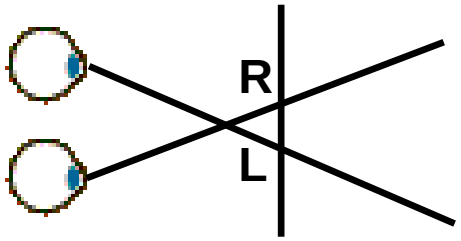
right **left**

Note: **Cross-fusion is necessary if distance between images is greater than inter-ocular distance**

impossible:

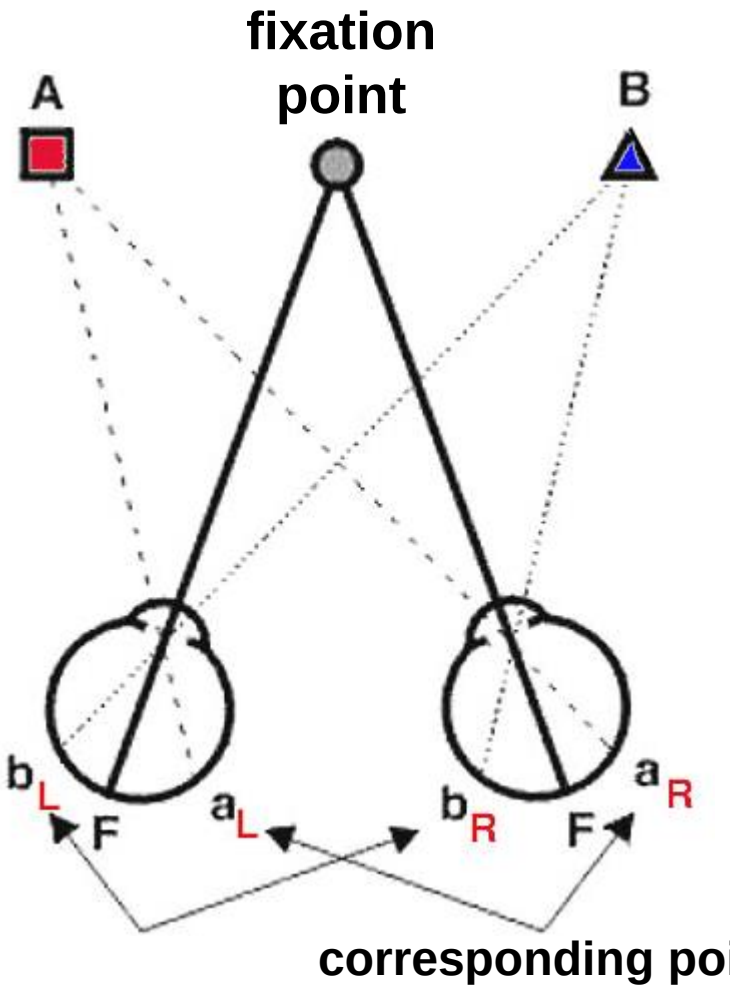


instead, trick the brain:



Tsukuba stereo images courtesy of Y. Ohta and Y. Nakamura at the University of Tsukuba

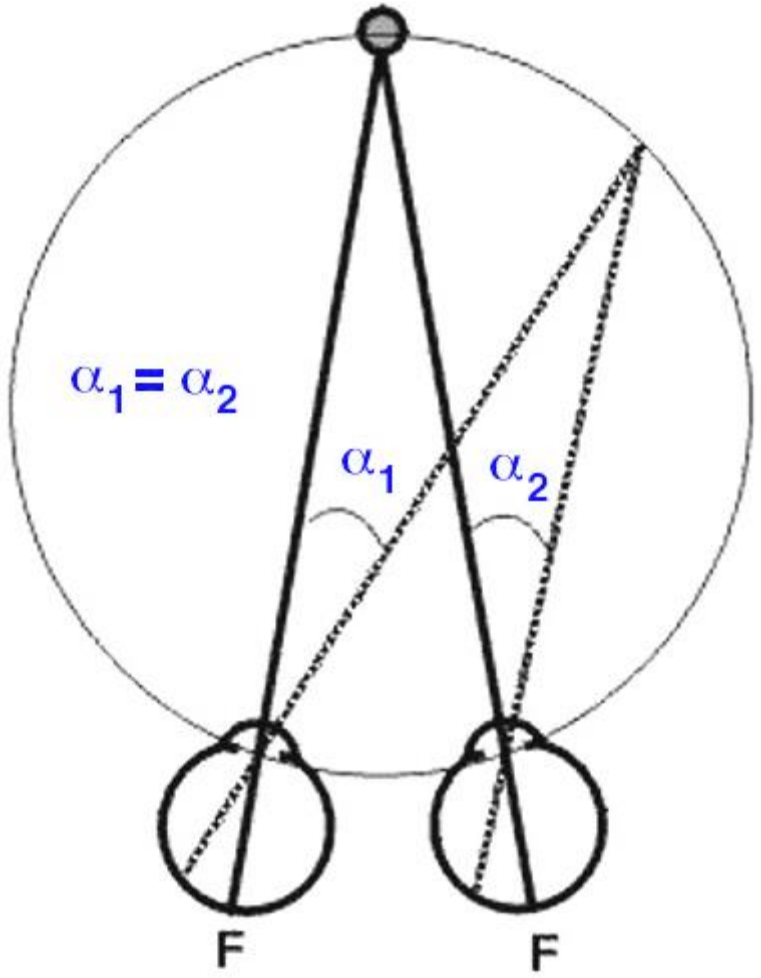
Human stereo geometry



http://webvision.med.utah.edu/space_perception.html

Horopter

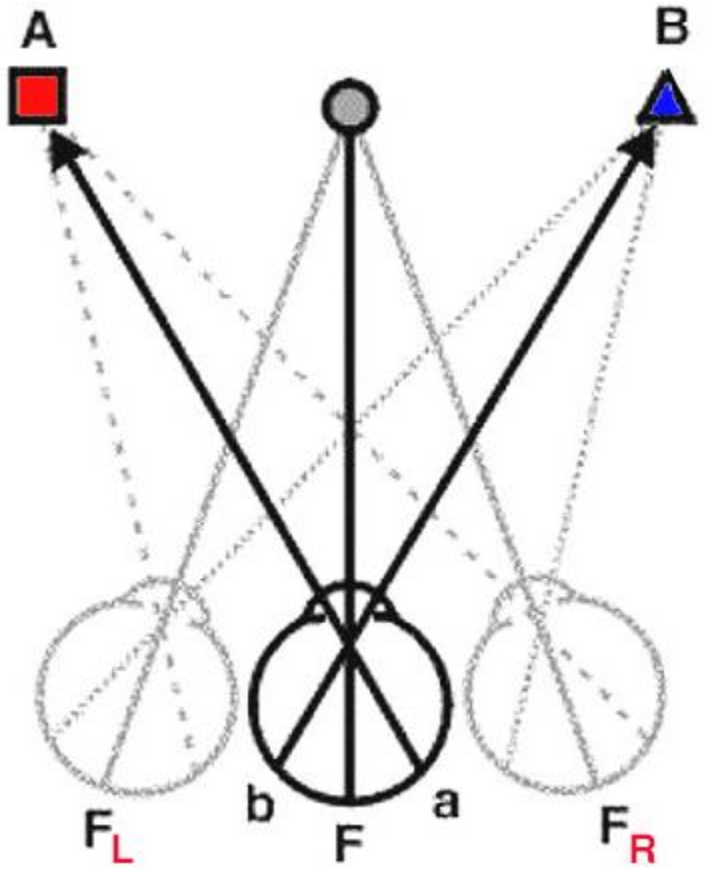
- Horopter: surface where disparity is zero
- For round retina, the theoretical horopter is a circle (Vieth-Muller circle)



http://webvision.med.utah.edu/space_perception.html

Cyclopean image

http://bearah718.tripod.com/sitebuildercontent/sitebuilderpictures/cyclops.jpg



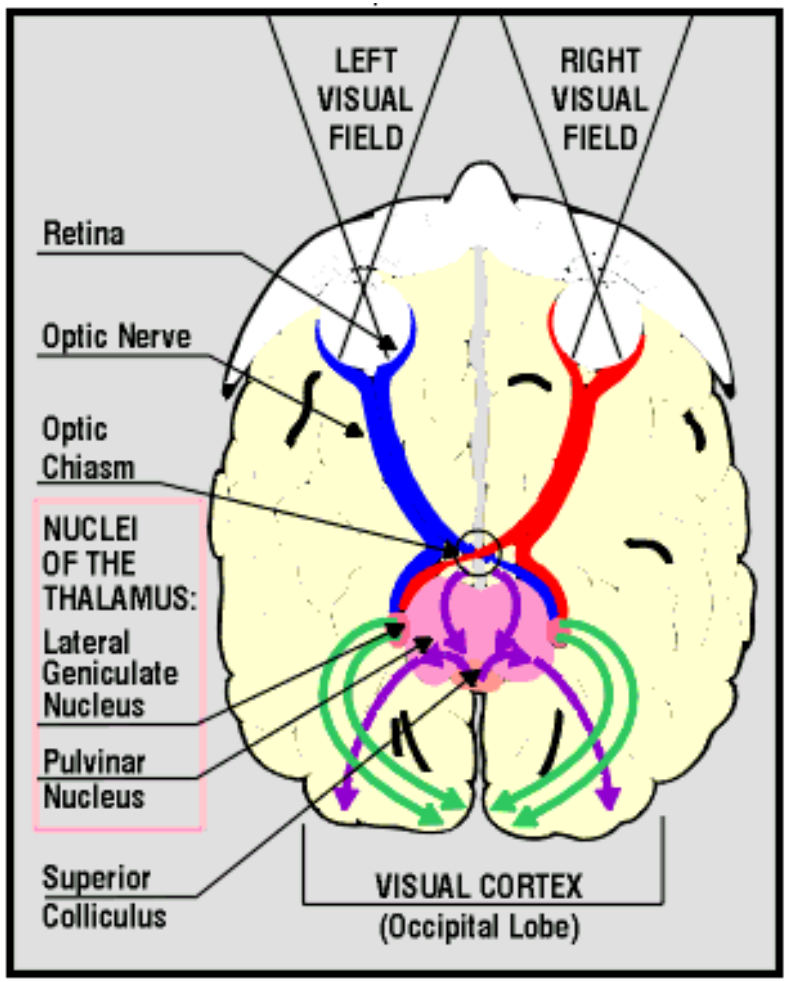
http://webvision.med.utah.edu/space_perception.html

Panum's fusional area (volume)

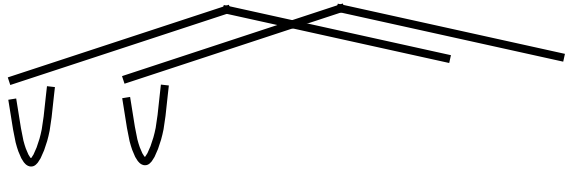
- Human visual system is only capable of fusing the two images with a narrow range of disparities around fixation point
- This area (volume) is Panum's fusional area
- Outside this area we get double-vision (diplopia)

http://www.allaboutvision.com/conditions/double-vision.htm

Human visual pathway

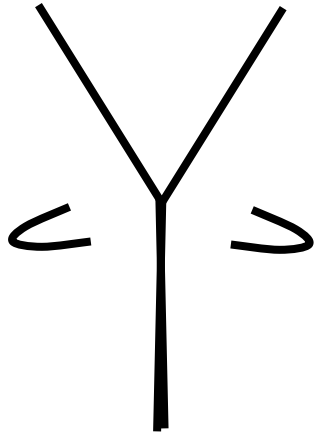


Prey and predator



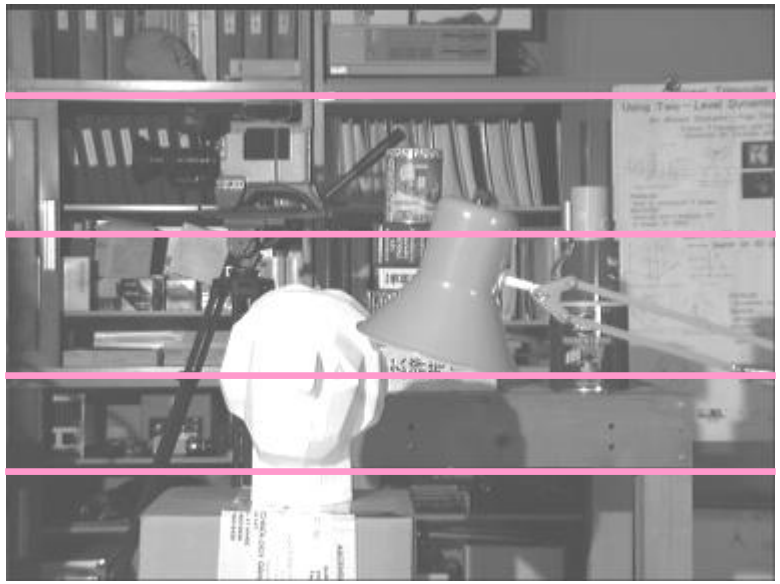
Cheetah: More accurate depth estimation

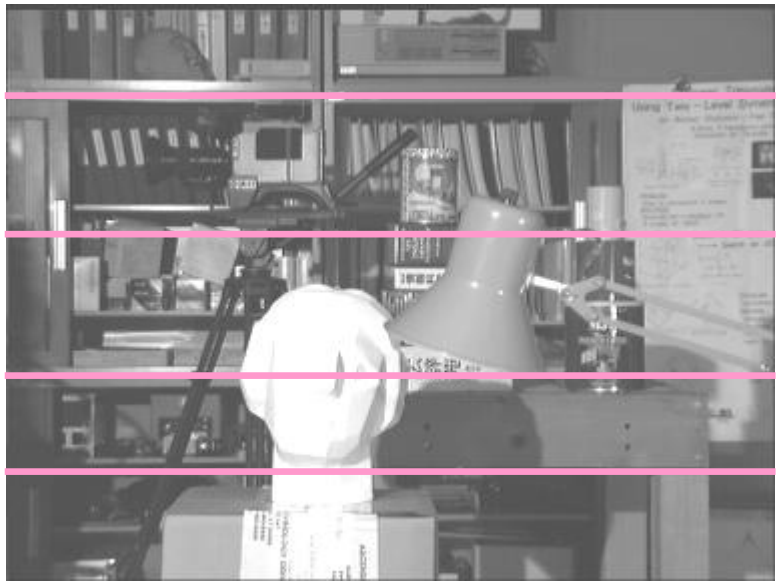
Antelope: larger field of view



photos courtesy California Academy of Science

Example: Motion parallel to image scanlines





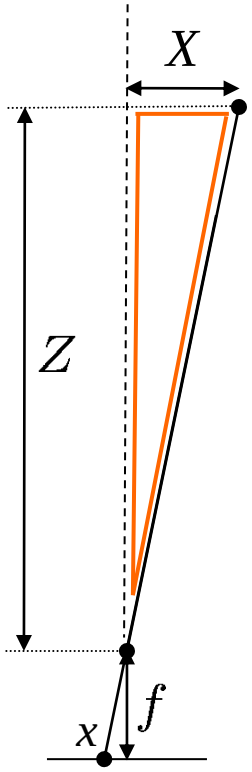
Epipoles are at infinity

Scanlines are the epipolar lines

In this case, the images are said to be “rectified”

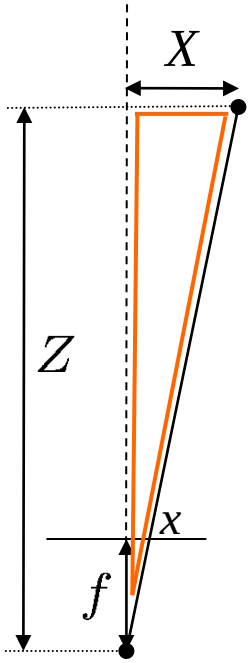
Tsukuba stereo images courtesy of Y. Ohta and Y. Nakamura at the University of Tsukuba

Perspective projection



$$x = -f\frac{X}{Z}$$

$$y = -f\frac{Y}{Z}$$

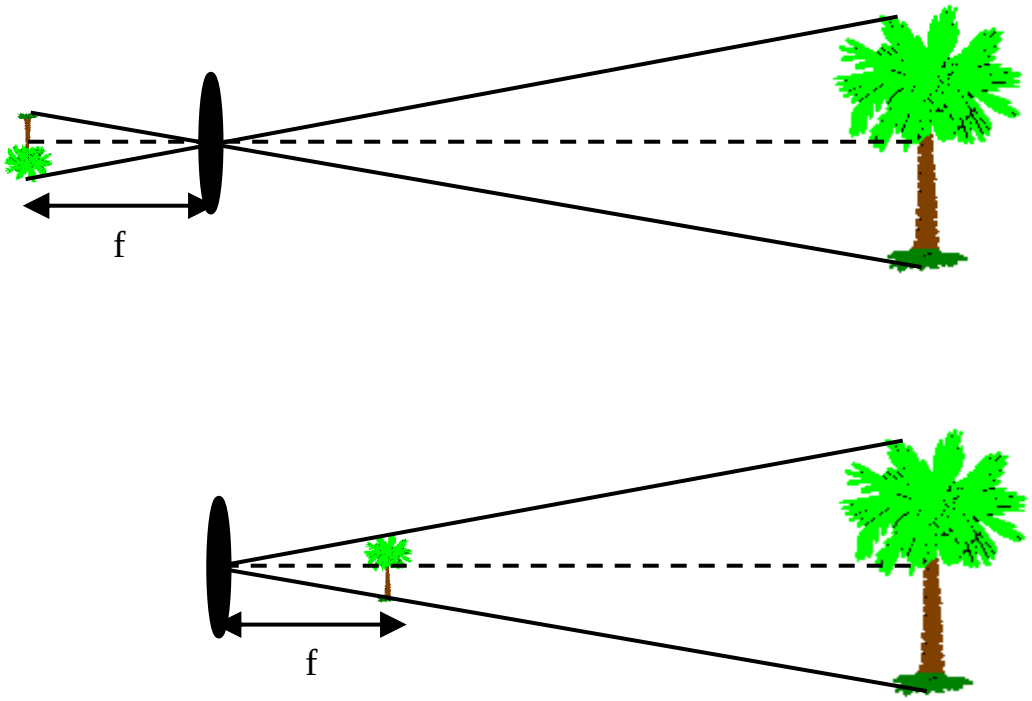


$$x = f\frac{X}{Z}$$

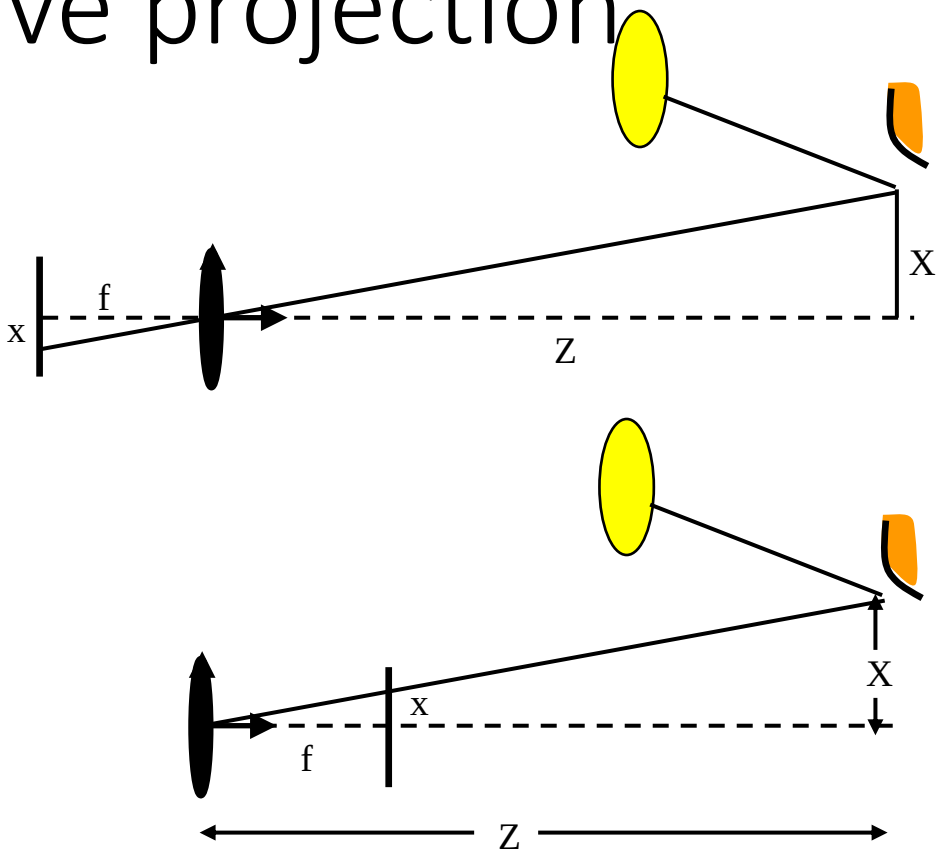
$$y = f\frac{Y}{Z}$$

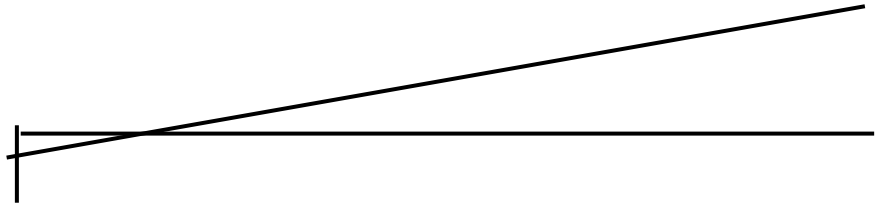
M. Pollefeys, http://www.cs.unc.edu/Research/vision/comp256fall03/

Perspective projection

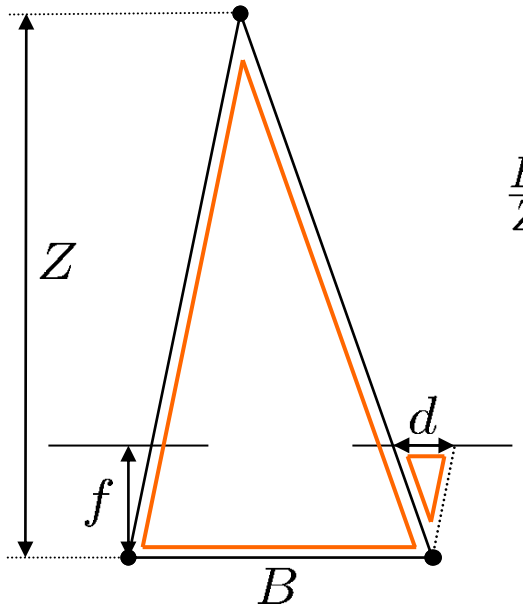


Perspective projection

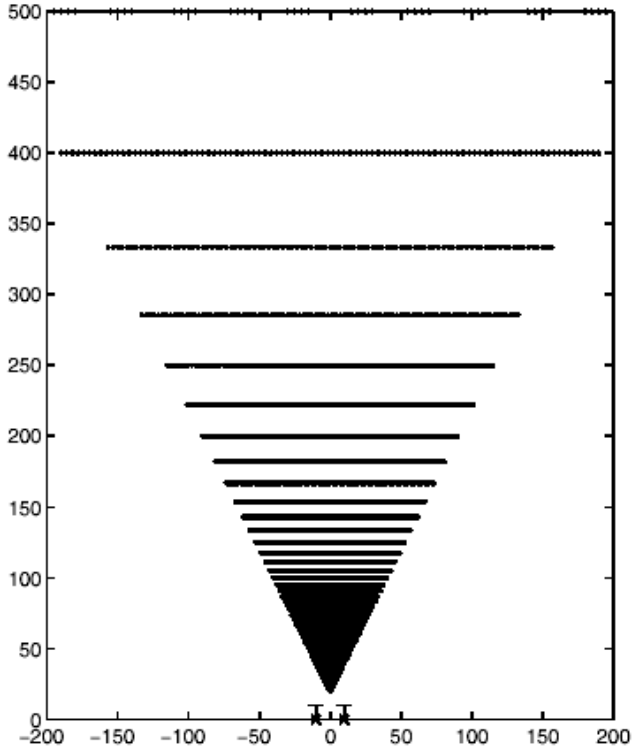




Standard stereo geometry



$$\frac{B}{Z} = \frac{d}{f}$$

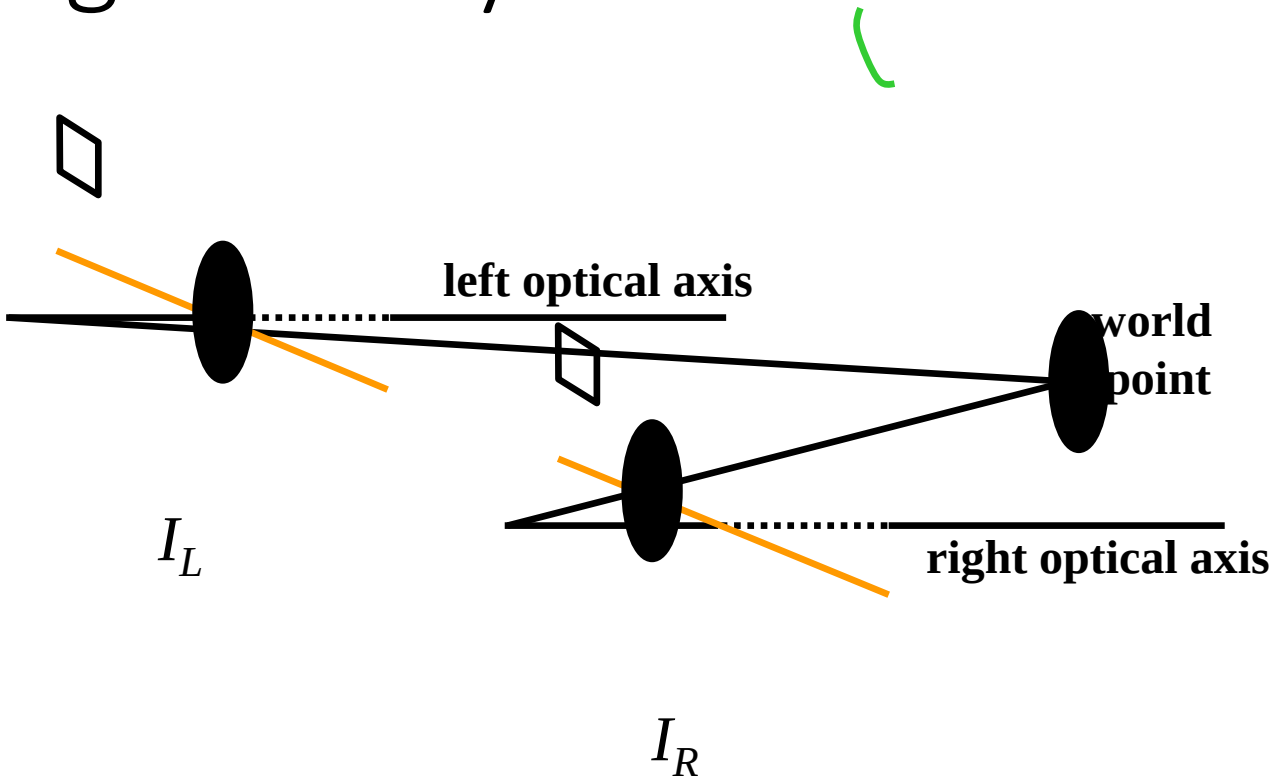


$$d = x_L - x_R = f\frac{X_L}{Z} - f\frac{X_R}{Z} = f\frac{X_L - X_R}{Z} = f\frac{B}{Z}$$

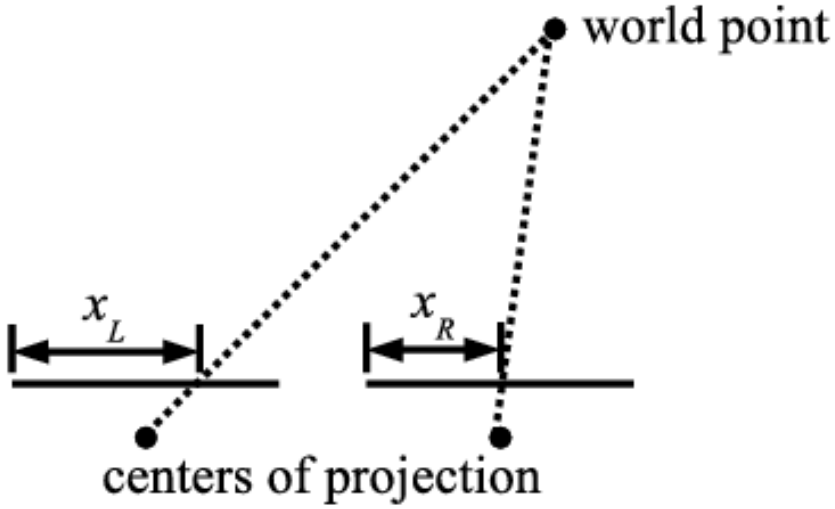
- **disparity is inversely proportional to depth**
- **stereo vision is less useful for distant objects**

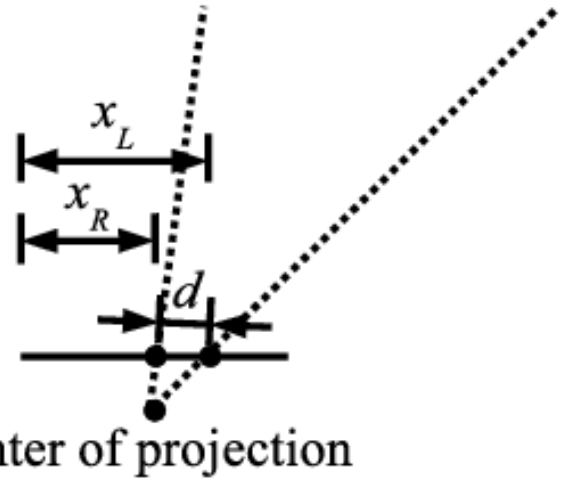
M. Pollefeys, http://www.cs.unc.edu/Research/vision/comp256fall03/

Rectified geometry



Rectified geometry



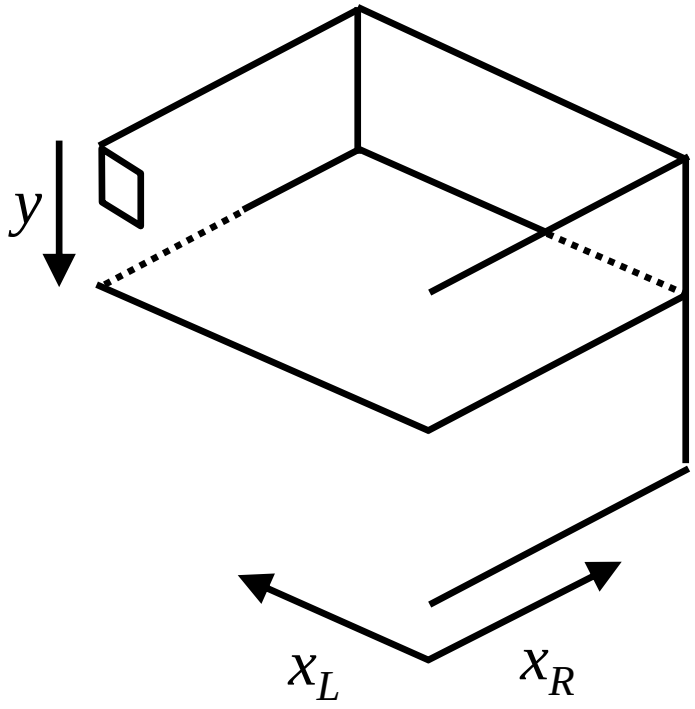


two cameras **overlapped (for display)**

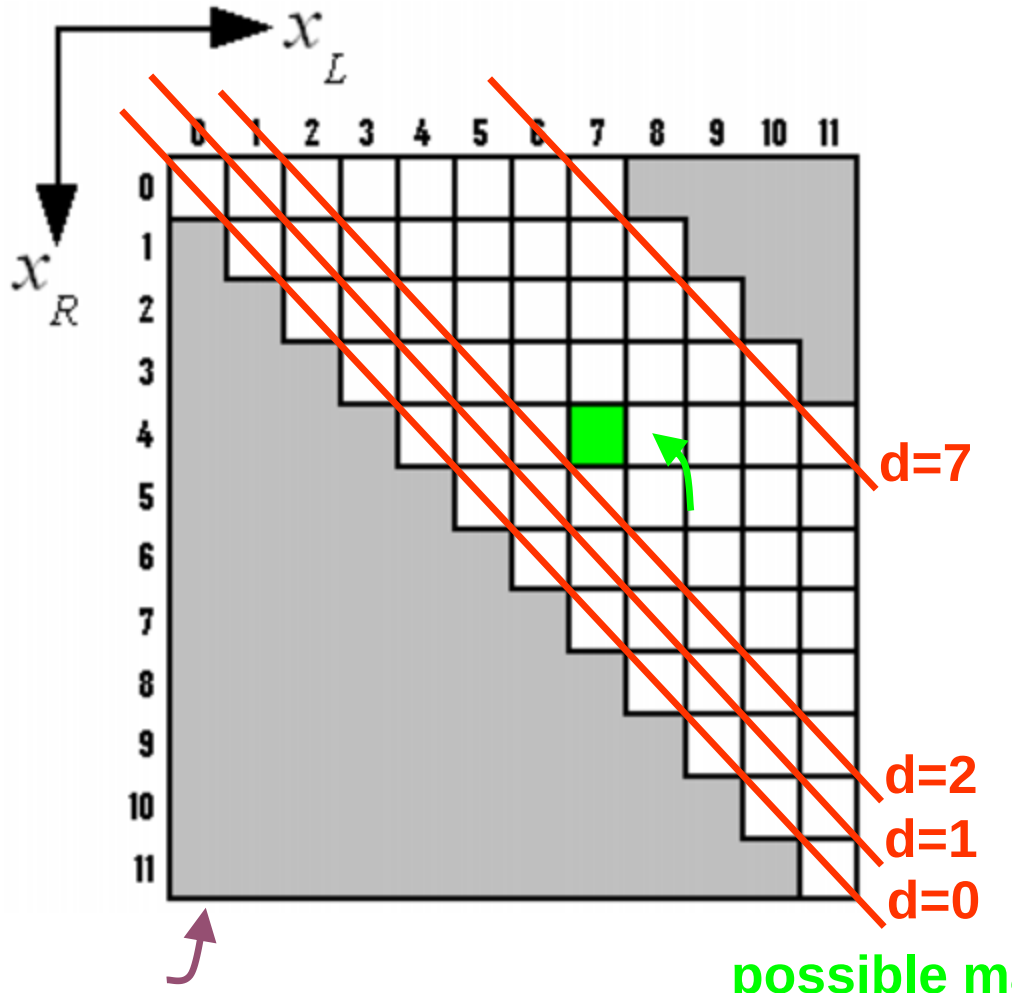
$$d = x_1 - x_2 = f\,(X_1 - X_2) / Z = f\,b / Z$$

disparity **baseline** **depth**

Matching space



Matching space



$$d = x_L - x_R$$

impossible matches

possible match between pixel 7 in left scanline and pixel 4 in right scanline

Stereo Matching

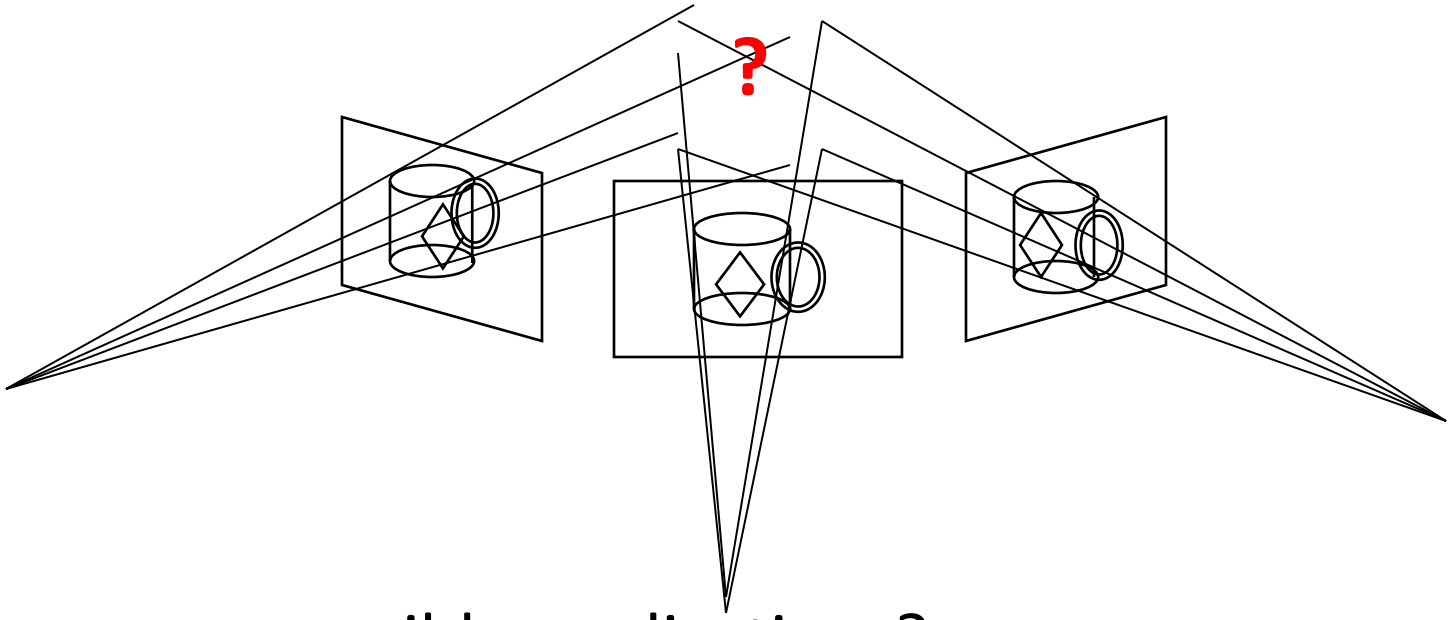
Computer Vision

CSE576, Spring 2005

Richard Szeliski

Stereo Matching

- Given two or more images of the same scene or object, compute a representation of its shape



- What are some possible applications?

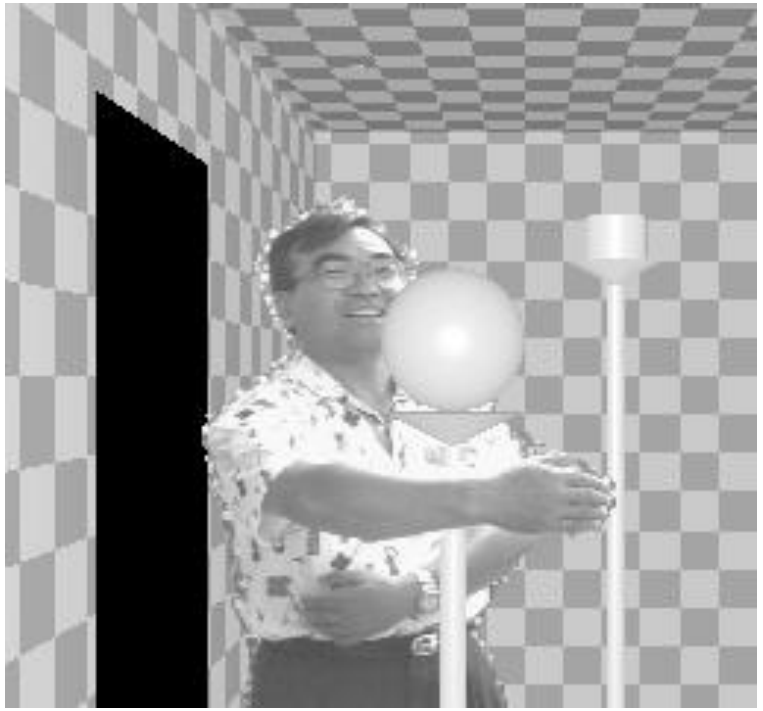
Face modeling

- From one stereo pair to a 3D head model

[Frederic Deverney, INRIA]

Z-keying: mix live and synthetic

- Takeo Kanade, CMU (Stereo Machine)



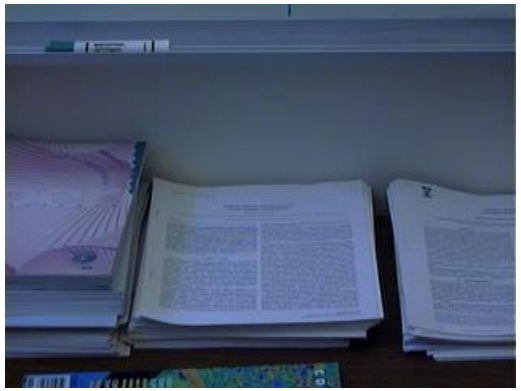
View Interpolation

- Given two images with correspondences, *morph* (warp and cross-dissolve) between them [Chen & Williams, SIGGRAPH'93]

[Matthies,Szeliski,Kanade'88]

More view interpolation

- Spline-based depth map

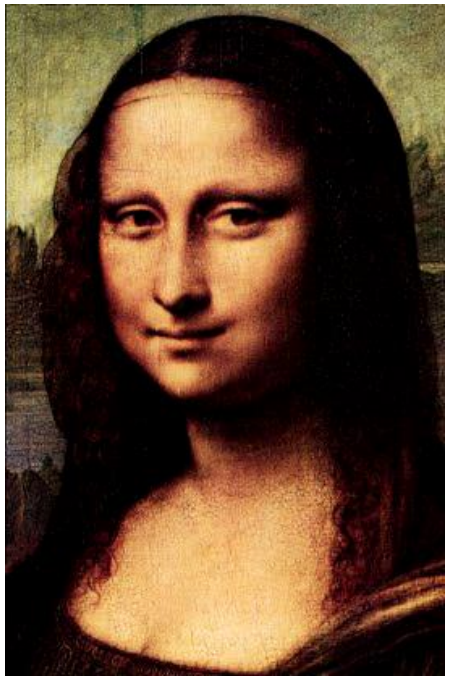


input depth image novel view

- [Szeliski & Kang ‘95]

View Morphing

- Morph between pair of images using epipolar geometry [Seitz & Dyer, SIGGRAPH'96]

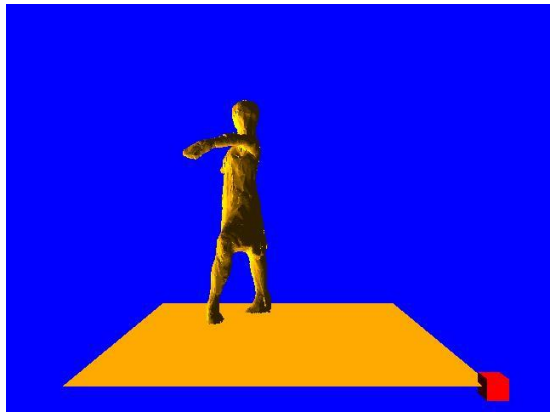


Video view interpolation


Massive Arabesque

Virtualized Reality™

- [Takeo Kanade *et al.*, CMU]
  - collect video from 50+ stream
  - reconstruct 3D model sequences



  - steerable version used for SuperBowl XXV "eye vision"

- http://www.cs.cmu.edu/afs/cs/project/VirtualizedR/www/VirtualizedR.html

Real-time stereo

Nomad robot searches for meteorites in Antartica
http://www.frc.ri.cmu.edu/projects/meteorobot/index.html

- Used for robot navigation (and other tasks)
  - Software-based real-time stereo techniques

Additional applications

- Real-time people tracking (systems from Pt. Gray Research and SRI)
- “Gaze” correction for video conferencing [Ott,Lewis,Cox InterChi’93]
- Other ideas?

Stereo Matching

- Given two or more images of the same scene or object, compute a representation of its shape

- What are some possible representations?
  - depth maps
  - volumetric models
  - 3D surface models
  - planar (or offset) layers

Stereo Matching

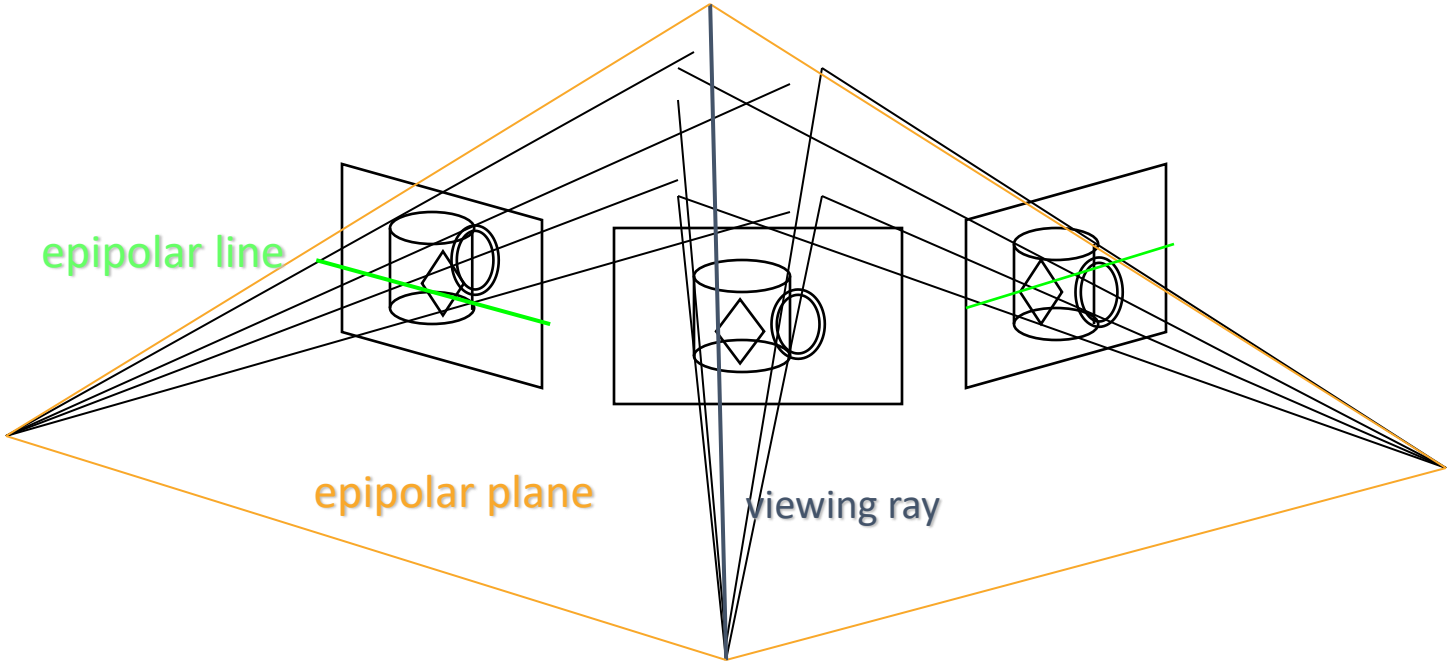
- What are some possible algorithms?
  - match “features” and interpolate
  - match edges and interpolate
  - match all pixels with windows (coarse-fine)
  - use optimization:
    - iterative updating
    - dynamic programming
    - energy minimization (regularization, stochastic)
    - graph algorithms

Outline (remainder of lecture)

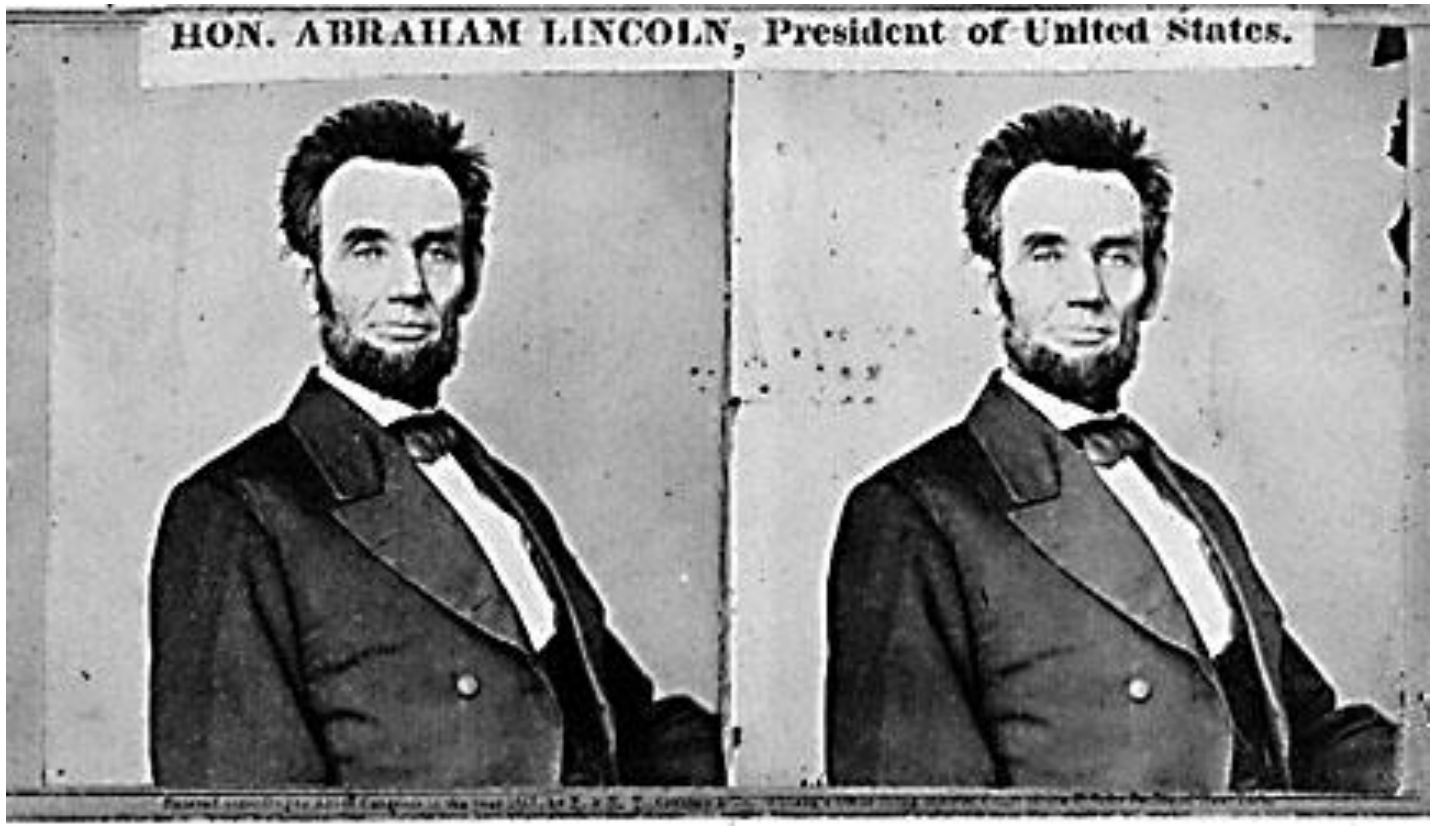
- Image rectification
- Matching criteria
- Local algorithms (aggregation)
  - iterative updating
- Optimization algorithms:
  - energy (cost) formulation & Markov Random Fields
  - mean-field, stochastic, and graph algorithms
- Multi-View stereo & occlusions

Stereo: epipolar geometry

- Match features along epipolar lines



Stereo image pair



Anaglyphs

http://www.rainbowsymphony.com/freestuff.html

(Wikipedia for images)

Public Library, Stereoscopic Looking Room, Chicago, by Phillips, 1923

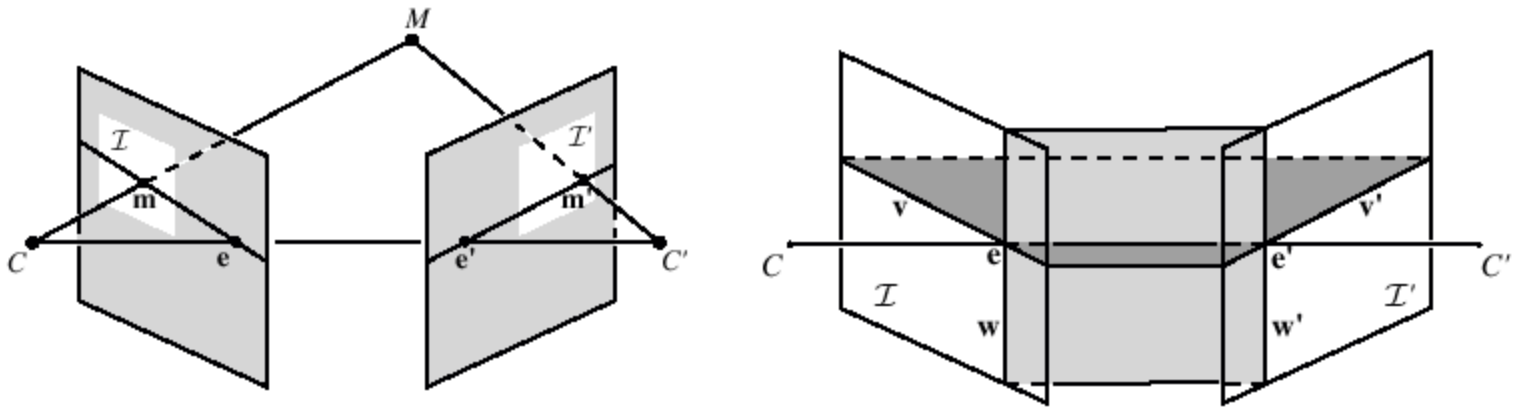
Stereo: epipolar geometry

- for *two* images (or images with collinear camera centers), can find epipolar lines
- epipolar lines are the projection of the *pencil* of planes passing through the centers

- **Rectification:** warping the input images (perspective transformation) so that epipolar lines are horizontal

Rectification

- Project each image onto same plane, which is parallel to the epipole
- Resample lines (and shear/stretch) to place lines in correspondence, and minimize distortion



- [Loop and

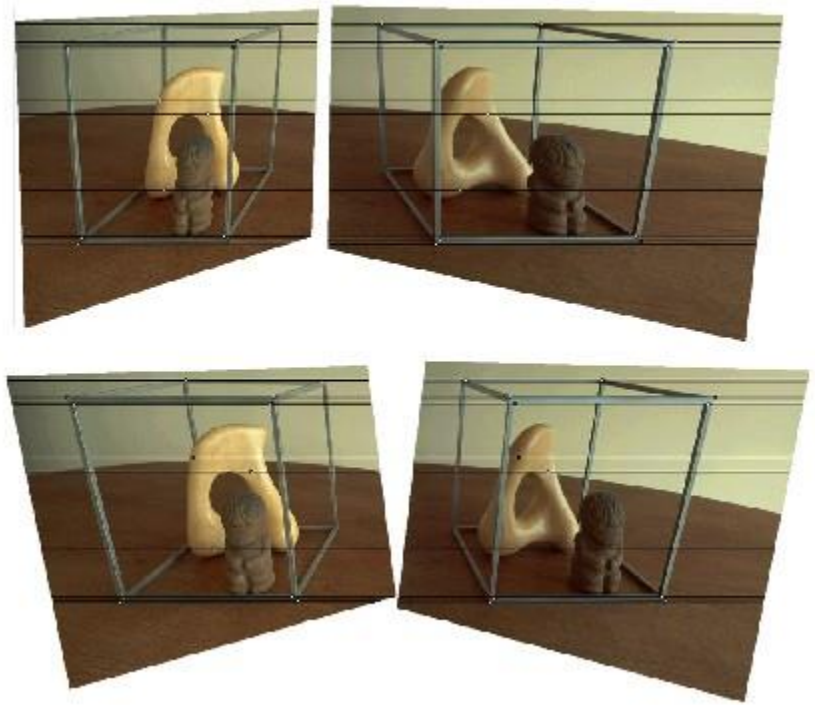
Rectification

(a) Original image pair overlayed with several epipolar lines.

(b) Image pair transformed by the specialized projective mapping $\mathbf{H}_p$ and $\mathbf{H}'_p$. Note that the epipolar lines are now parallel to each other in each image.

BAD!

Rectification



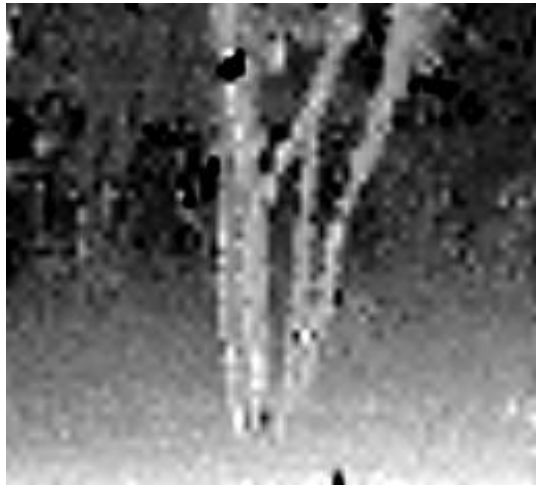
(c) Image pair transformed by the similarity $\mathbf{H}_r$ and $\mathbf{H}'_r$. Note that the image pair is now rectified (the epipolar lines are horizontally aligned).

(d) Final image rectification after shearing transform $\mathbf{H}_a$ and $\mathbf{H}'_a$. Note that the image pair remains rectified, but the horizontal distortion is reduced.

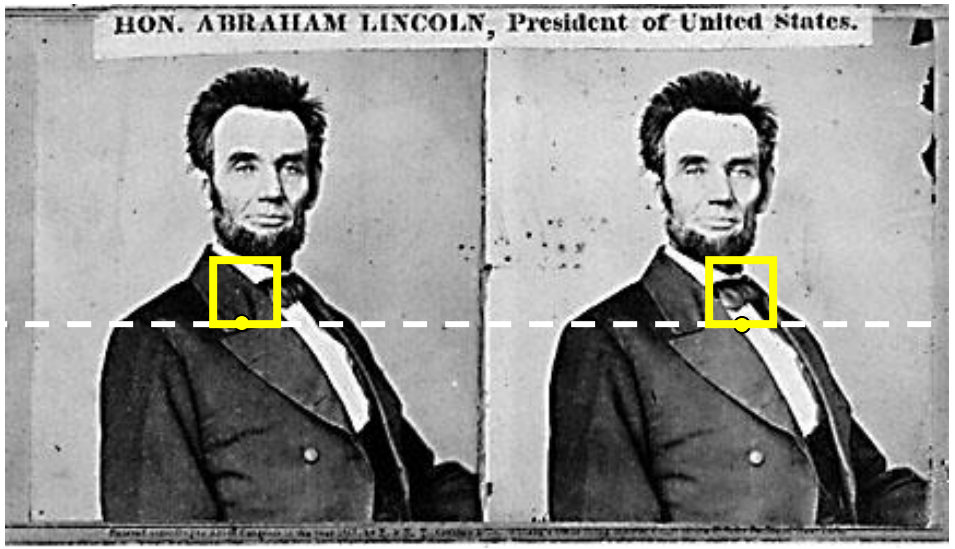
GOOD!

Finding correspondences

- apply feature matching criterion (e.g., correlation or Lucas-Kanade) at *all* pixels simultaneously
- search only over epipolar lines (many fewer candidate positions)



Your basic stereo algorithm



For each epipolar line

For each pixel in the left image

- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match ***windows***

- This should look familar...

Image registration (revisited)

- How do we determine correspondences?
  - *block matching* or *SSD* (sum squared differences)

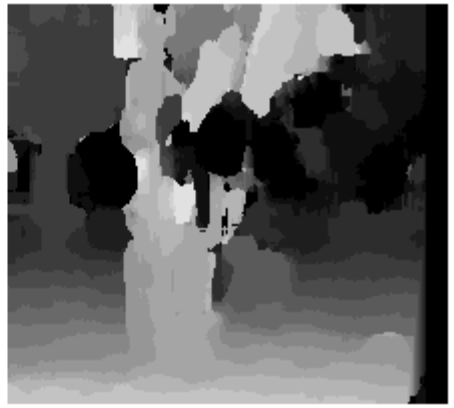
$$E(x,y;d) = \sum_{(x',y')\in N(x,y)} [I_L(x'+d,y') - I_R(x',y')]^2$$

d is the *disparity* (horizontal motion)

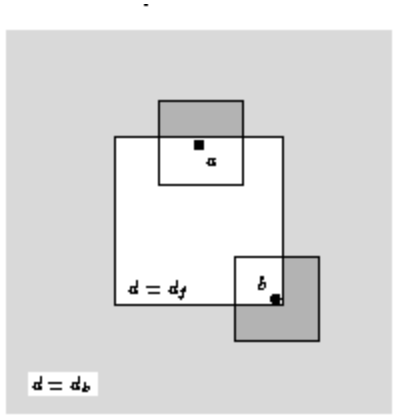
- How big should the neighborhood be?

Neighborhood size

- Smaller neighborhood: more details
- Larger neighborhood: fewer isolated mistakes
- 
- w = 3



- w = 20

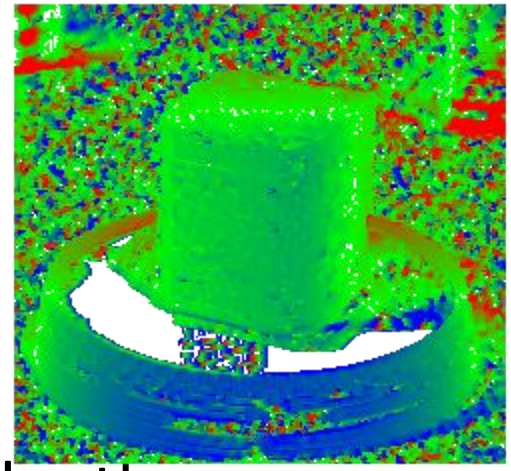
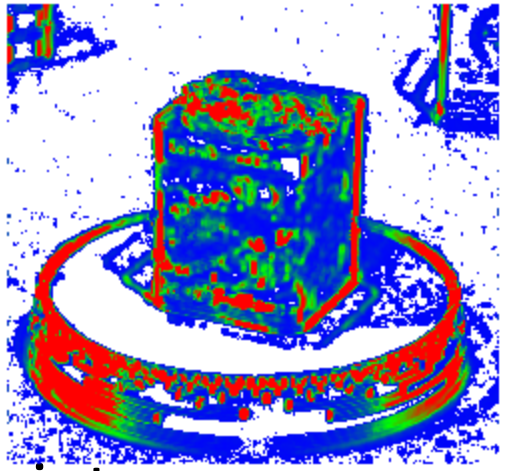


Matching criteria

- Raw pixel values (correlation)
- Band-pass filtered images [Jones & Malik 92]
- “Corner” like features [Zhang, …]
- Edges [many people…]
- Gradients [Seitz 89; Scharstein 94]
- Rank statistics [Zabih & Woodfill 94]

Stereo: certainty modeling

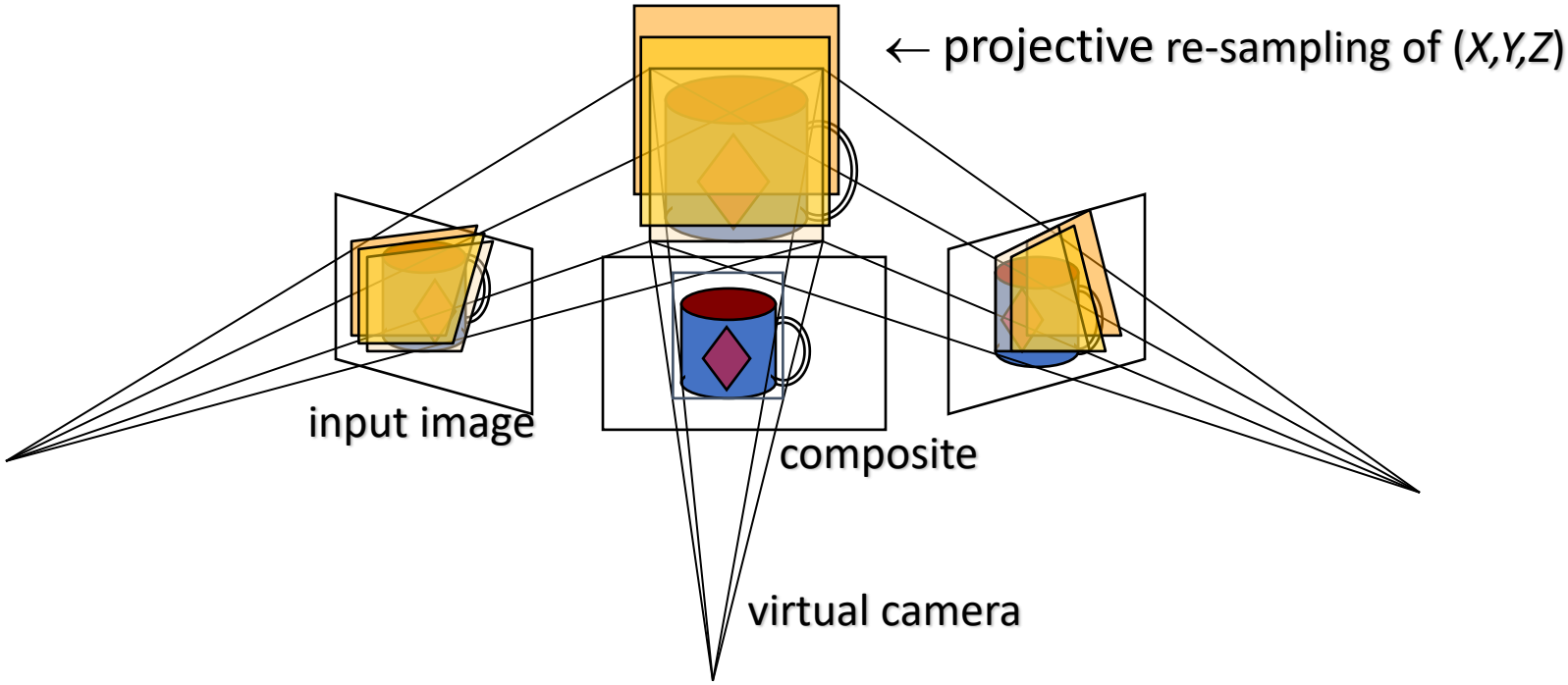
- Compute certainty map from correlations

- input depth map certainty map

Plane Sweep Stereo

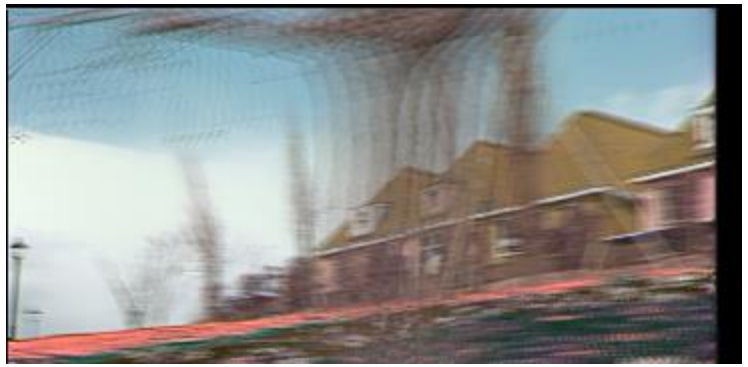
- Sweep family of planes through volume



- each plane defines an image ⇒ composite homography

Plane Sweep Stereo

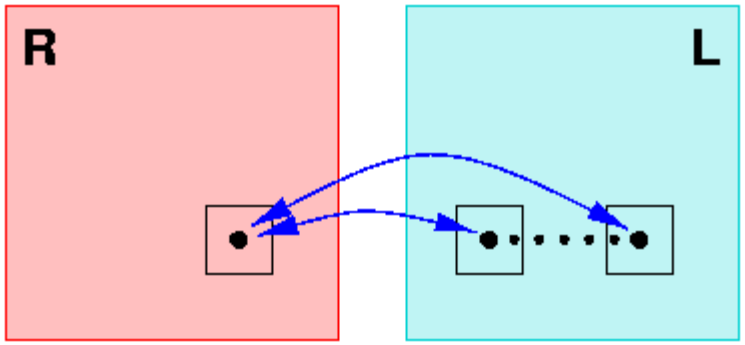
- For each depth plane
  - compute composite (mosaic) image — *mean*

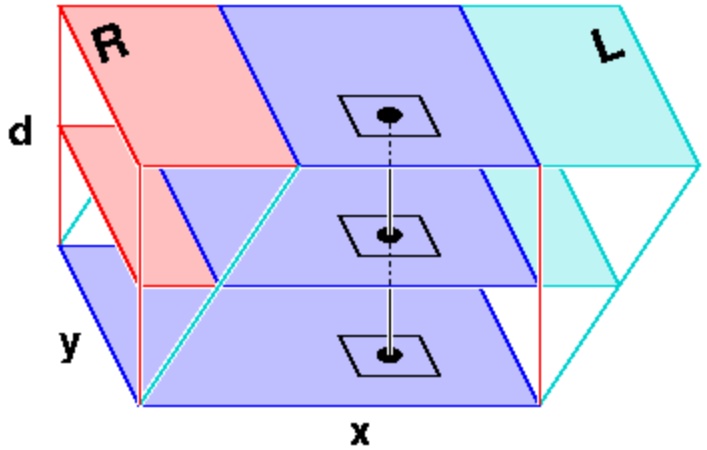


  - compute error image — *variance*
  - convert to confidence and aggregate spatially
- Select winning depth at each pixel

Plane sweep stereo

- Re-order (pixel / disparity) evaluation loops





for every pixel,
for every disparity
compute cost

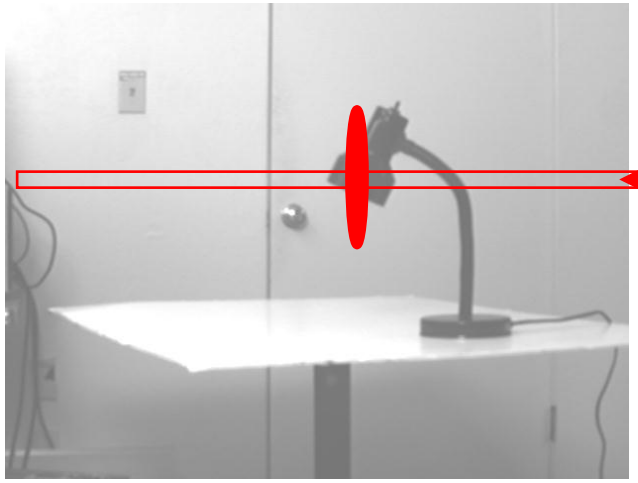
for every disparity
for every pixel
compute cost

Outline

- Stereo basics
- Binocular stereo matching
- Advanced stereo techniques

Binocular rectified stereo

left

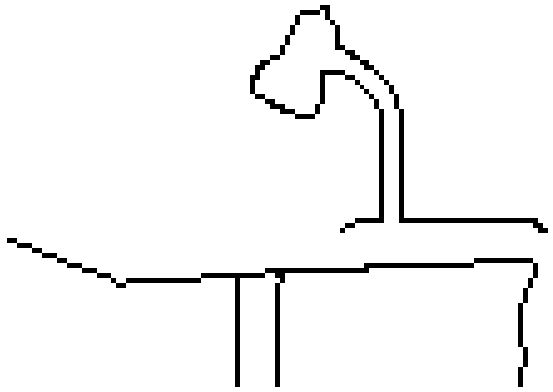


right

epipolar constraint

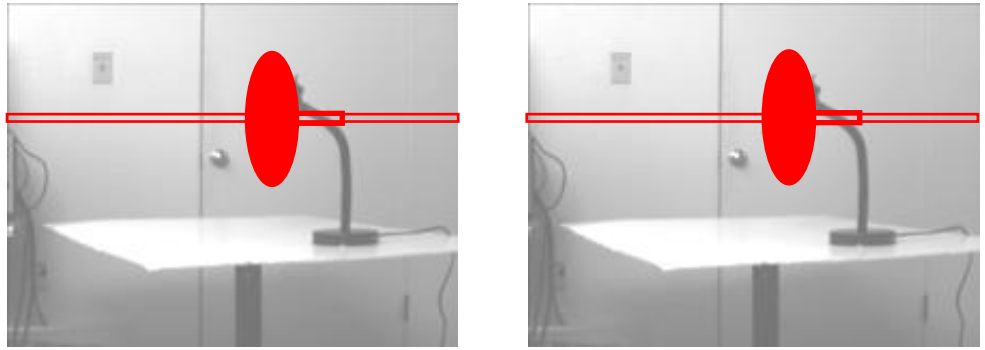
1D search: look for similar pixel in other image

disparity map

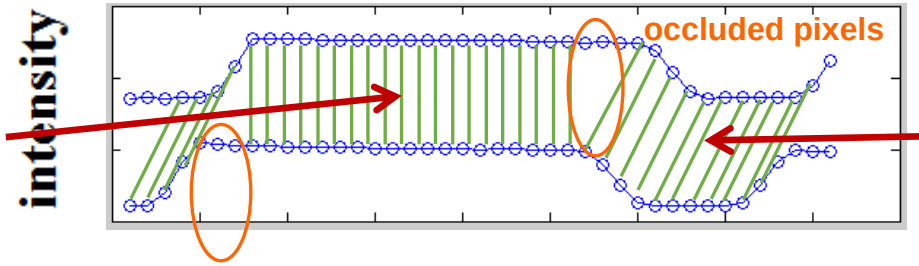


depth discontinuities

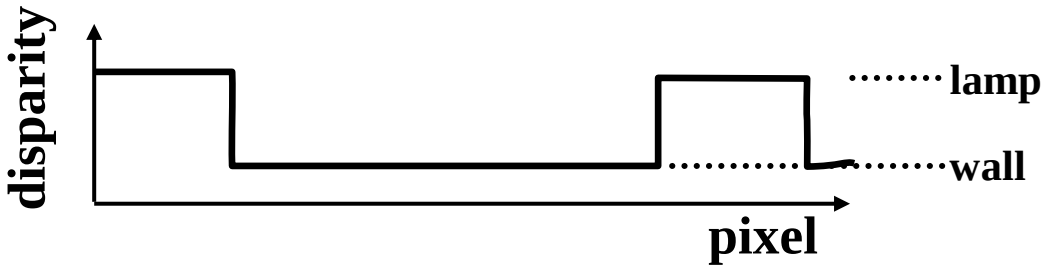
Disparity function



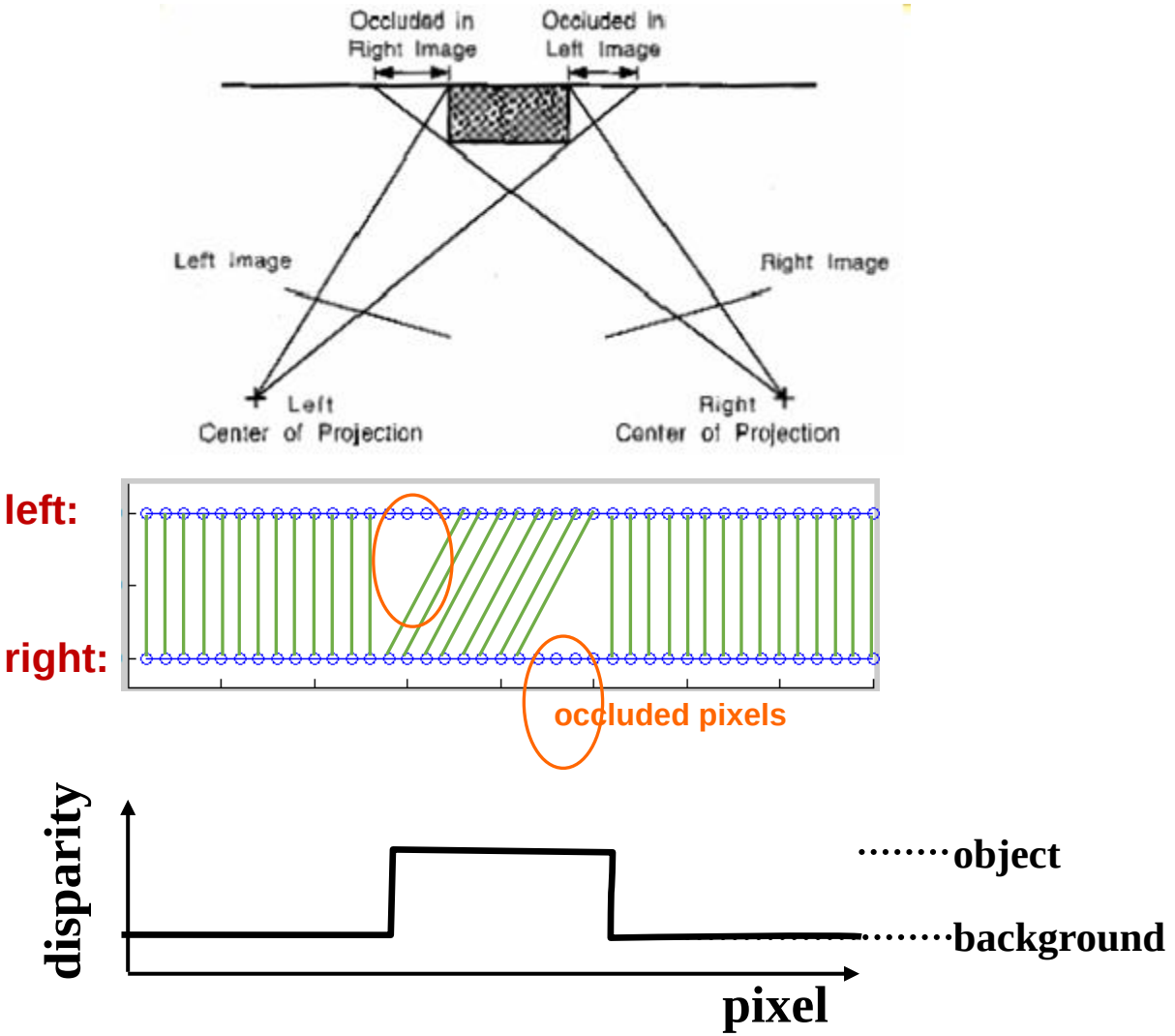
smaller slope =
smaller disparity =
farther from camera



higher slope =
larger disparity =
closer to camera

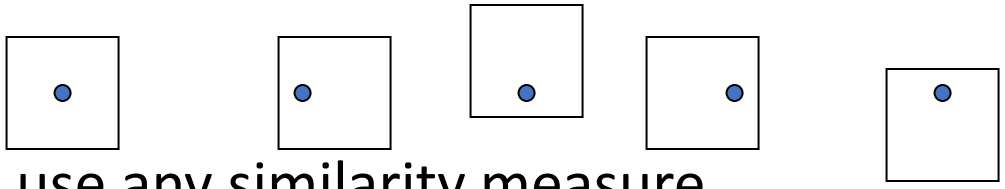


Occlusions


Occluded in
Right Image
Occluded in
Left Image
Left Image
Right Image
Left
Center of Projection
Right
Center of Projection
left:
right:
occluded pixels
disparity
object
background
pixel

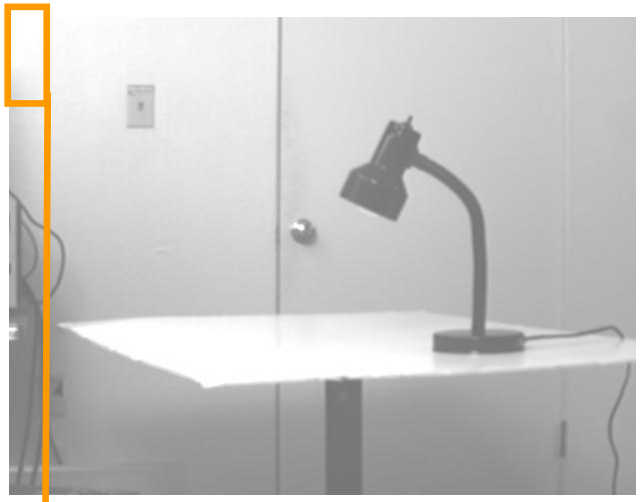
Matching a pixel

- Pixel's value is not unique
  - Only 256 values but ~100,000 pixels!
  - Also, noise affects value
- Solution: use more than one pixel
- Assume neighbors have similar disparity
  - Correlation window around pixel



  - Can use any similarity measure

Block matching

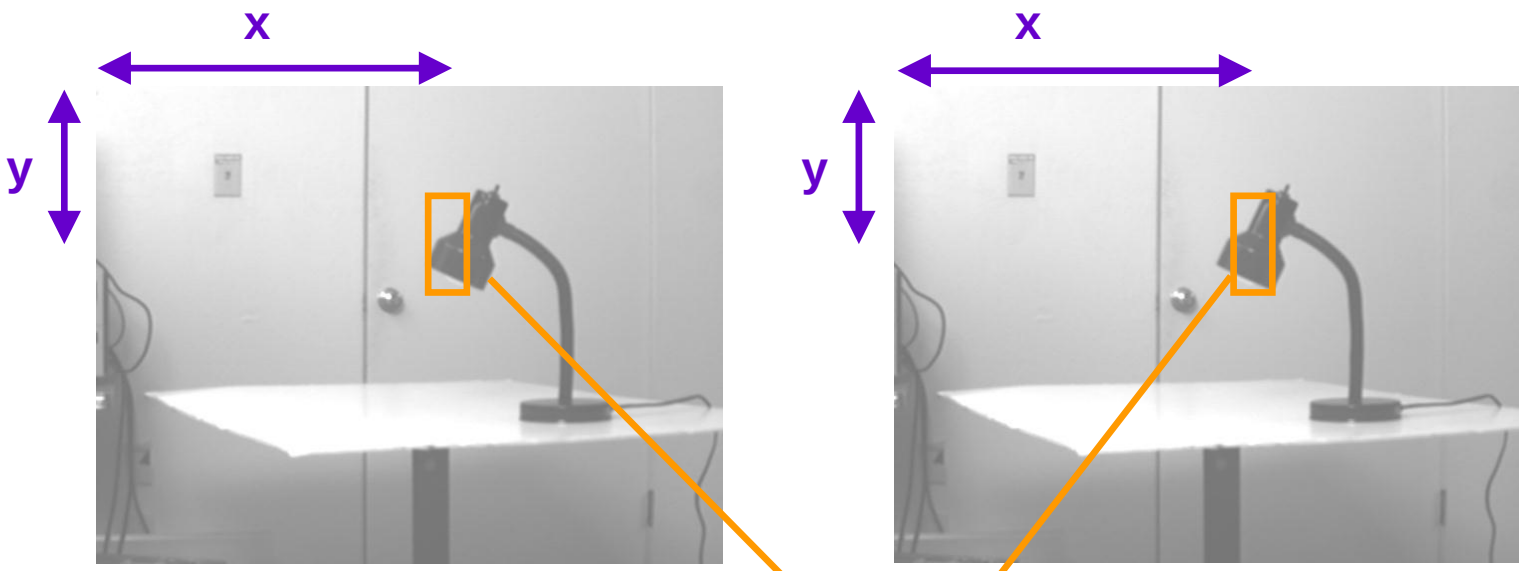


left

disparity map

- **compute best disparity for each pixel**
- **store result in disparity map**

Block matching (cont.)



Note: Window only moves left. Why?

Block matching

$$d_L(x,y) = \arg \min_{0 \le d \le d_{\max}} dissim\left(I_L(x,y), I_R(x-d,y)\right)$$

disparity

dissimilarity

```
Function disparity_map = BlockMatch1(img_left, img_right; min_disp, max_disp)
  for y = 0 to height-1
    for x = 0 to width-1
      ghat = infinity
      for d = min_disp to max_disp
        g = 0
        for j = -w to w
          for i = -w to w
            g = g + dissimilarity(img_left(x+i, y+j), img_right(x+i-d, y+j))
        if g < ghat,
          ghat = g
          dhat = d
      disparity_map(x, y) = dhat
```

5 nested for loops!!!!!

Block matching

$$d_L(x,y) = \arg \min_{0 \le d \le d_{\max}} dissim\left(I_L(x,y), I_R(x-d,y)\right)$$

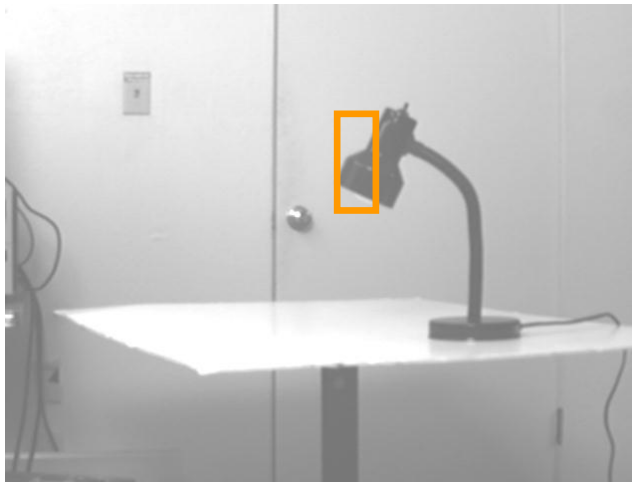
disparity

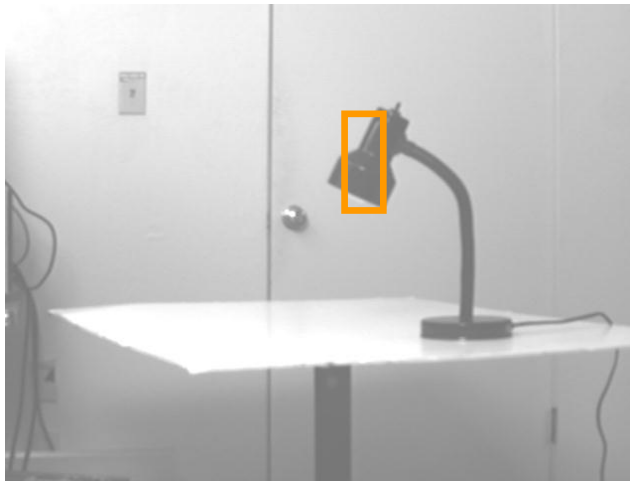
dissimilarity

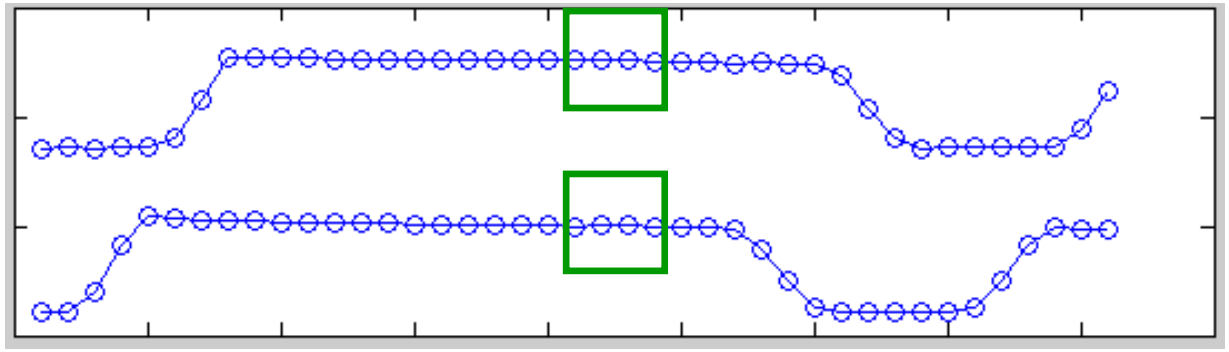
```
BlockMatch1(I_L, I_R, d_min, d_max)
 1  for (x, y) ∈ I_L do
 2        ĝ ← ∞
 3        for d ← d_min to d_max do
 4              g ← 0
 5              for (x̃, ỹ) ∈ W do
 6                    g ← g + dissim(I_L(x + x̃, y + ỹ), I_R(x + x̃ − d, y + ỹ))
 7              if g < ĝ then
 8                    ĝ ← g
 9                    d̂ ← d
10        d_L(x, y) ← d̂
11  return d_L
```

5 nested for loops!!!!!

Eliminating redundant computations



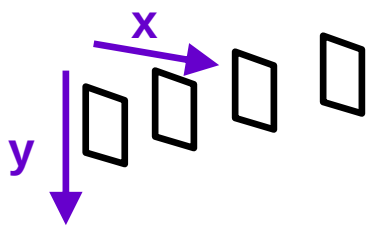




for same disparity, overlapping windows recompute the same dissimilarities for many pixels

Block matching: another view

- Alternatively,
  - precompute
    $\Delta(x,y,d) = dissim(I_L(x,y), I_R(x-d,y))$
    for all *x, y, d*
  - then for each *(x,y)* select the best *d*



More efficient block matching

```
Function dbar = ComputeDbar(img_left, img_right; min_disp, max_disp)
  for d=min_disp:max_disp,
    // compare pixels
    for y=0:height-1,
     for x=0:width-1,
       dbar(x, y, d) = dissimilarity(img_left(x, y), img_right(x-d, y)
    // convolve with 2D box filter to sum over window
    tmp = convolve dbar(:, :, d) with 1D kernel [1 ... 1]
    dbar(:, :, d) = convolve tmp with 1D kernel [1 ... 1]^T
```

} separable

```
Function disparity_map = BlockMatch2(img_left, img_right; min_disp, max_disp)
dbar = ComputeDbar(img_left, img_right; min_disp, max_disp)
  for y=0:height-1,
   for x=0:width-1,
     disparity_map(x, y) = arg min of dbar(x, y, :)
```

Key idea: **Summation over window is convolution with box filter, which is separable**

(only 3 nested for loops!!!)

Running sum improves efficiency even more

More efficient block matching

```
BlockMatch2(I_L, I_R, d_min, d_max)
1  Δ ← ComputeSummedDissimilarities(I_L, I_R, d_min, d_max)
2  for (x, y) ∈ I_L do
3        d_L(x, y) ← arg min_d Δ(x, y, d)
4  return d_L

ComputeSummedDissimilarities(I_L, I_R, d_min, d_max)
1  for d ← d_min to d_max do
2        for (x, y) ∈ I_L do
3              Δ(x, y, d) ← dissim(I_L(x, y), I_R(x − d, y))
4        Δ(:, :, d) ← Convolve(Δ(:, :, d), 1_{w×w})
5  return Δ
```

separable

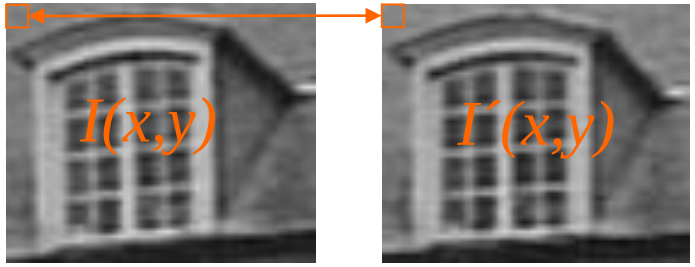
Key idea: **Summation over window is convolution with box filter, which is separable**

(only 3 nested for loops!!!)

Running sum improves efficiency even more

Comparing image regions

Compare intensities pixel-by-pixel



Dissimilarity measures

Sum of Square Differences

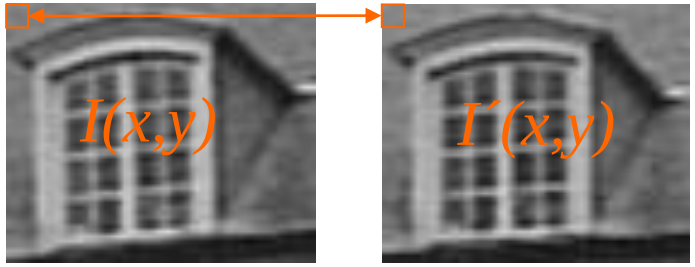
$$SSD = \iint_W \left[I'(x,y) - I(x,y) \right]^2 dxdy$$

Note: **SAD is fast approximation (replace square with absolute value)**

M. Pollefeys, http://www.cs.unc.edu/Research/vision/comp256fall03/

Comparing image regions

Compare intensities pixel-by-pixel



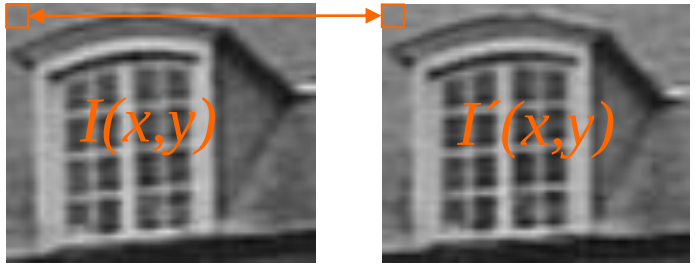
Dissimilarity measures

If energy does not change much, then minimizing SSD equals maximizing cross-correlation

M. Pollefeys, http://www.cs.unc.edu/Research/vision/comp256fall03/

Comparing image regions

Compare intensities pixel-by-pixel



Similarity measures

Zero-mean Normalized Cross Correlation

$$NCC = \frac{N(I', I)}{\sqrt{N(I', I')N(I, I)}}$$

$$N(A, B) = \iint_W \left(A(x, y) - \overline{A}\right)\left(B(x, y) - \overline{B}\right)dxdy$$

M. Pollefeys, http://www.cs.unc.edu/Research/vision/comp256fall03/

Dissimilarity measures

Most common:

$$D(\mathbf{x}_L, \mathbf{x}_R) = [I_L(x_L, y_L) - I_R(x_R, y_R)]^2 \qquad \text{SSD}$$

$$D(\mathbf{x}_L, \mathbf{x}_R) = |I_L(x_L, y_L) - I_R(x_R, y_R)| \qquad \text{SAD}$$

$$D(\mathbf{x}_L, \mathbf{x}_R) = -I_L(x_L, y_L) I_R(x_R, y_R) \qquad \text{cross correlation}$$

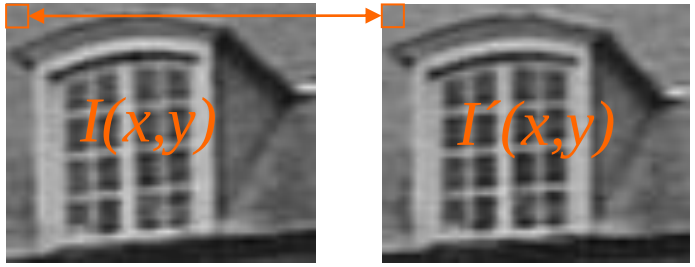
Connection between SSD and cross correlation:

$$\begin{aligned} D(\mathbf{x}_L, \mathbf{x}_R) &= [I_L(x_L, y_L) - I_R(x_R, y_R)]^2 \\ &= [I_L(x_L, y_L)]^2 + [I_R(x_R, y_R)]^2 - 2 I_L(x_L, y_L) I_R(x_R, y_R) \\ &\propto -I_L(x_L, y_L) I_R(x_R, y_R) \end{aligned}$$

Also normalized correlation, rank, census, sampling-insensitive ...

Comparing image regions

Compare intensities pixel-by-pixel



Similarity measures

Census

$$C_I(i,j) = (I(x+i, y+j) > I(x,y))$$

125	126	125
127	128	130
129	132	135

→

0	0	0
0		1
1	1	1

→ [00001111]

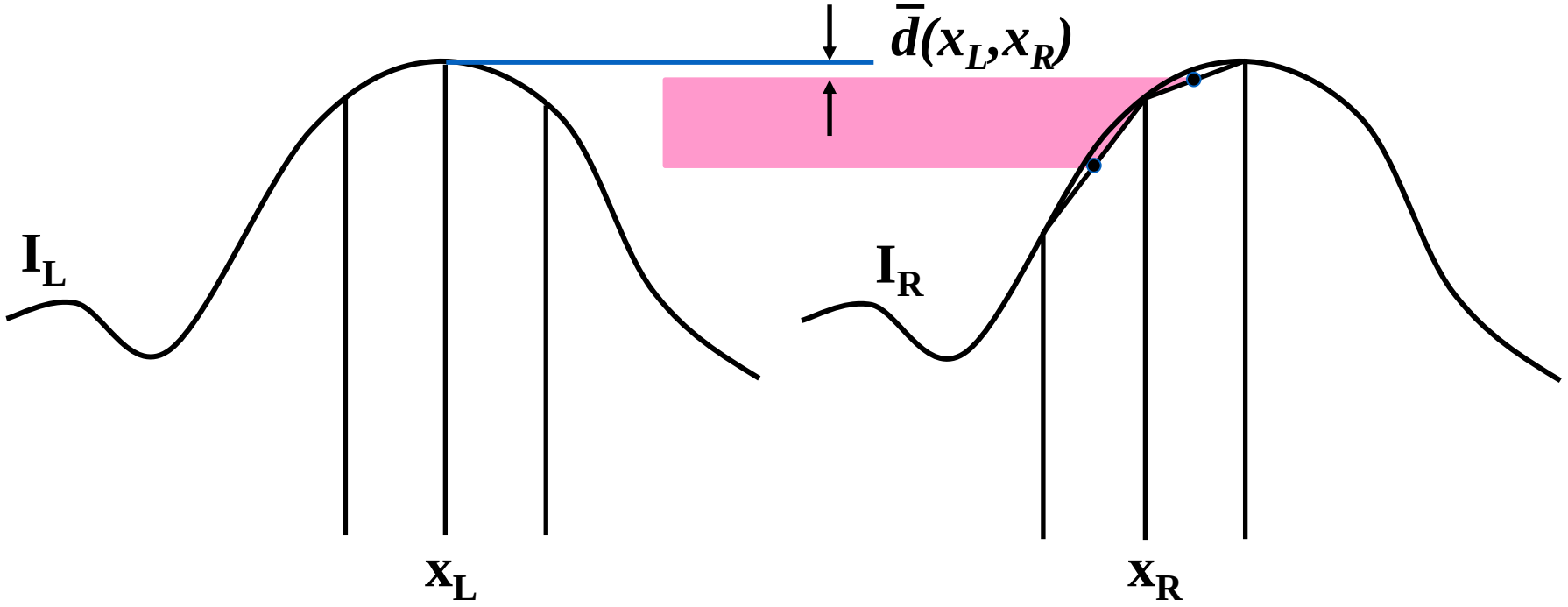
only compare bit signature

using XOR, SAD, or Hamming distance (all equivalent)

(Real-time chip from TZYX based on Census)

M. Pollefeys, http://www.cs.unc.edu/Research/vision/comp256fall03/

Sampling-Insensitive Pixel Dissimilarity



Our dissimilarity measure: $d(x_L,x_R) = \min\{\bar{d}(x_L,x_R)\ ,\bar{d}(x_R,x_L)\}$

[Birchfield & Tomasi 1998]

Dissimilarity Measure Theorems

Given: **An interval A such that**
$[x_L - ½ , x_L + ½] \subseteq A$, and
$[x_R - ½ , x_R + ½] \subseteq A$

Theorem 1:

If $| x_L - x_R | \leq ½$, then $d(x_L,x_R) = 0$

(when A is convex or concave)

Theorem 2: $| x_L - x_R | \leq ½$ iff $d(x_L,x_R) = 0$

(when A is linear)

[Birchfield & Tomasi 1998]

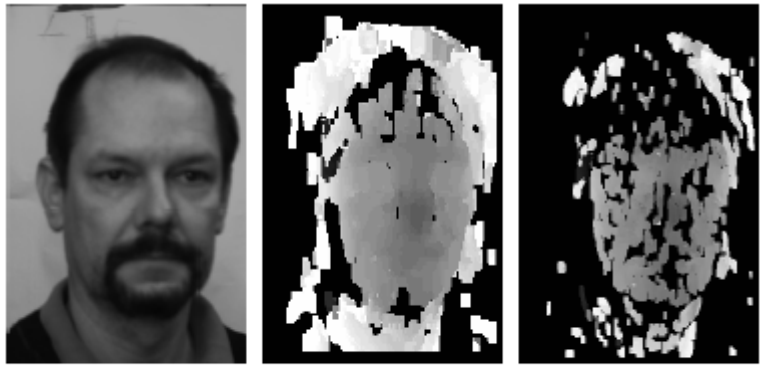
Aggregation window sizes

Small windows

- disparities similar
- more ambiguities
- accurate when correct

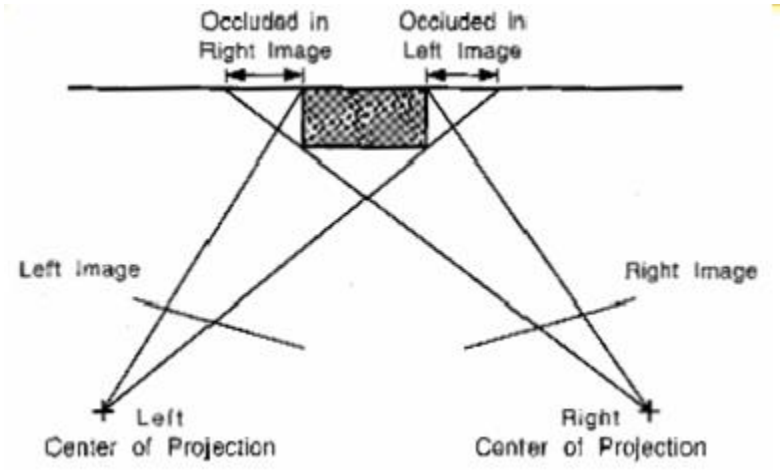
Large windows

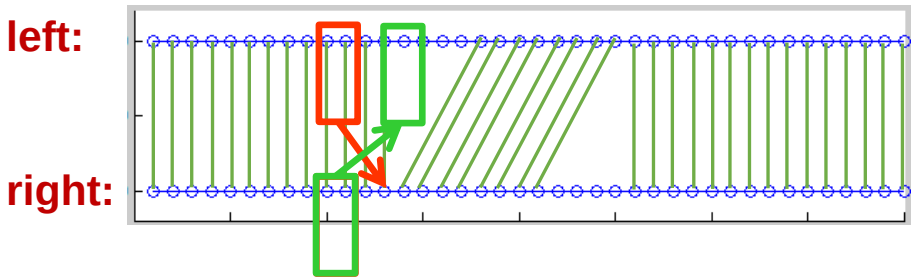
- larger disp. variation
- more discriminant
- often more robust
- use shiftable windows to deal with discontinuities



(Illustration from Pascal Fua)

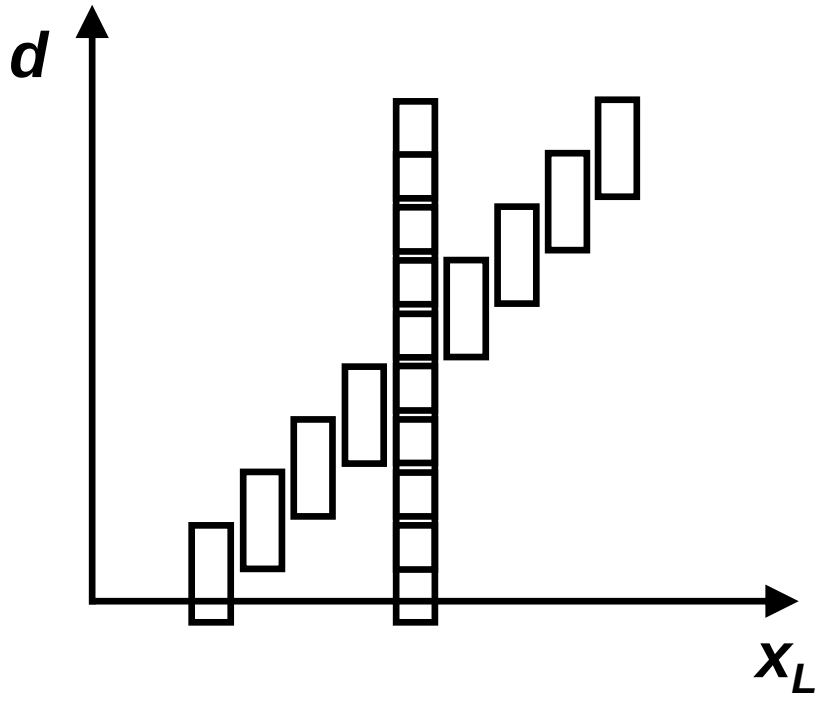
Occlusions





If pixel matches do not agree in both directions, then unreliable

Left-right consistency check



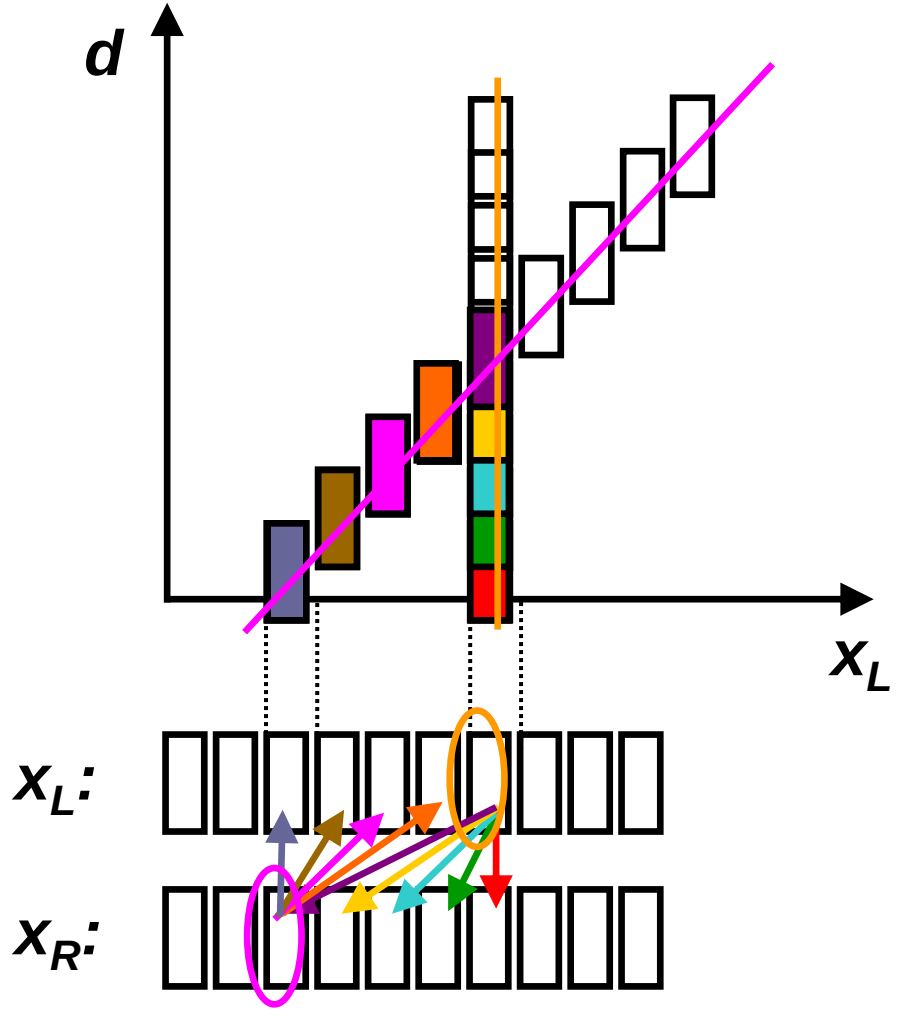
- **Search left-to-right, then right-to-left**
- **Retain disparity only if they agree**

Do minima coincide?

Conceptually,

```
dm_L = BlockMatch(img_left, img_right; 0, max_disp)
dm_R = BlockMatch(img_right, img_left; -max_disp, 0)
for y=0:height-1,
for x=0:width-1,
  if dm_L(x, y) != - dm_R(x - dm_L(x, y), y)
    dm_L(x, y) = NOT_MATCHED
```

Left-right consistency check



for pixel (x,y) in left image, choices are

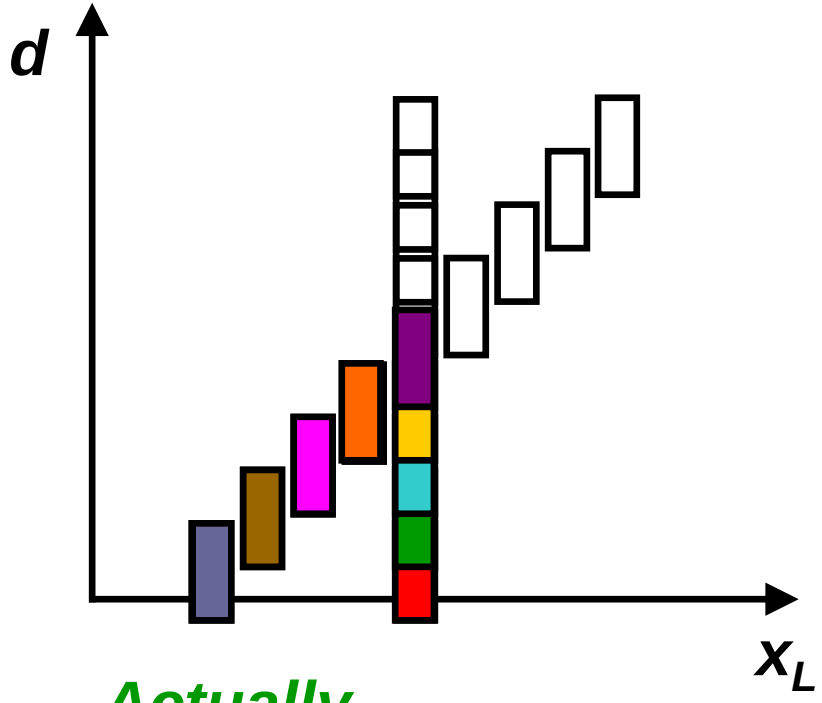
Δ(x,y,0),
Δ(x,y,1),
Δ(x,y,2),
…,
Δ(x,y,max_disp)

for pixel (x,y) in right image, choices are

Δ(x,y,0),
Δ(x+1,y,1),
Δ(x+2,y,2),
…,
Δ(x+max_disp,y,max_disp)

because $x_L = x_R$ + disparity

Left-right consistency check



Actually,

```
Function disparity_map = BlockMatchWithRightLeftCheck(img_left, img_right; max_disp)
△= ComputeDbar(img_left, img_right; 0, max_disp)
for y=0:height-1,
for x=0:width-1,
  // find left answer
  d_left = arg min( △(x,y,0), △(x,y,1), ..., △(x,y,max_disp) )
  d_right = arg min( △(x-d_left,y,0), △(x-d_left+1,y,1), ..., △(x-d_left+max_disp,y,max_disp)
  disp_map(x,y) = (d_left == d_right) ? d_left : NOT_MATCHED
```

With left-right check

inefficient:

BlockMatchWithLeftRightCheck1$(I_L, I_R, d_{\max})$

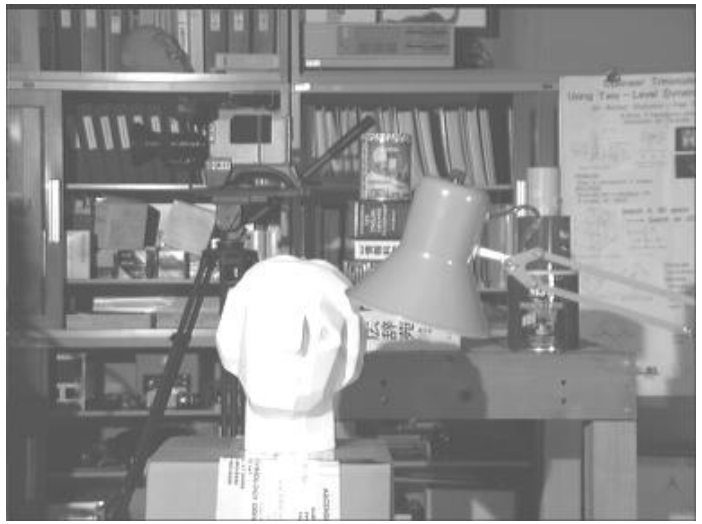
```
1  d_L ← BlockMatch2(I_L, I_R, 0, d_max)
2  d_R ← BlockMatch2(I_R, I_L, −d_max, 0)
3  for (x, y) ∈ I_L do
4        if d_L(x, y) ≠ −d_R(x − d_L(x, y), y) then
5              d_L(x, y) ← NOT-MATCHED
6  return d_L
```

more efficient:

BlockMatchWithLeftRightCheck2$(I_L, I_R, d_{\max})$

```
1  Δ ← ComputeSummedDissimilarities(I_L, I_R, 0, d_max)
2  for (x, y) ∈ I_L do
3        δ_L ← arg min{Δ(x, y, 0), Δ(x, y, 1), ..., Δ(x, y, d_max)}
4        δ_R ← arg min{Δ(x − δ_L, y, 0), Δ(x − δ_L + 1, y, 1), ..., Δ(x − δ_L + d_max, y, d_max)}
5        if δ_L == δ_R then
6              d_L(x, y) ← δ_L
7        else
8              d_L(x, y) ← NOT-MATCHED
9  return d_L
```

Results: correlation



left

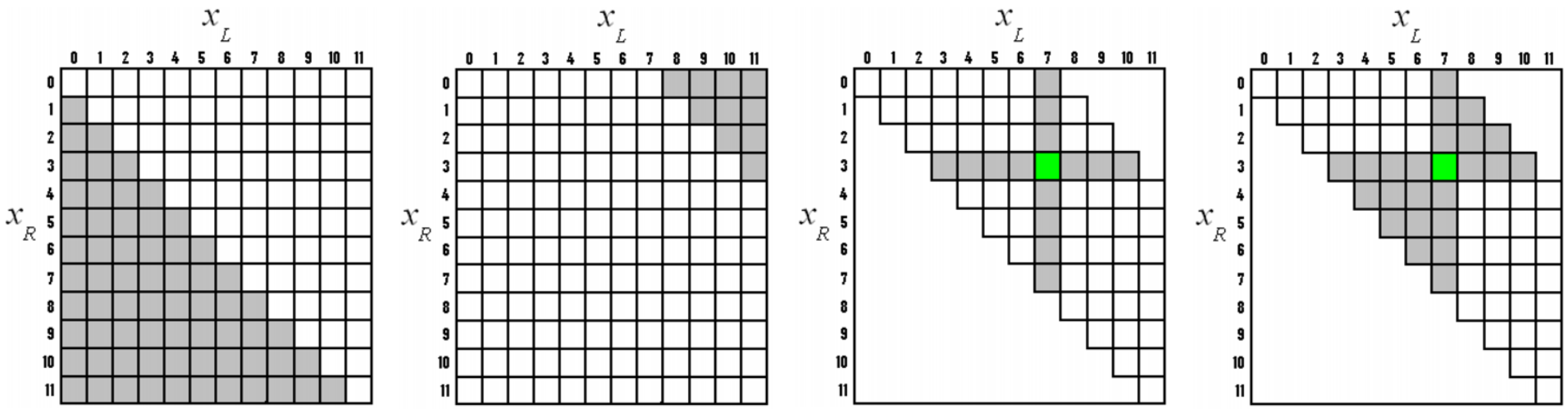
disparity map

with left-right consistency check

Constraints

- Epipolar – match must lie on epipolar line
- Piecewise constancy – neighboring pixels should usually have same disparity
- Piecewise continuity – neighboring pixels should usually have similar disparity
- Disparity – impose allowable range of disparities
  (Panum's fusional area)
- Disparity gradient – restricts slope of disparity
- Figural continuity – disparity of edges across scanlines
- Uniqueness – each pixel has no more than one match (violated by windows and mirrors)
- Ordering – disparity function is monotonic
  (precludes thin poles)

Stereo constraints



cheirality

$$d = x_L - x_R \geq 0$$

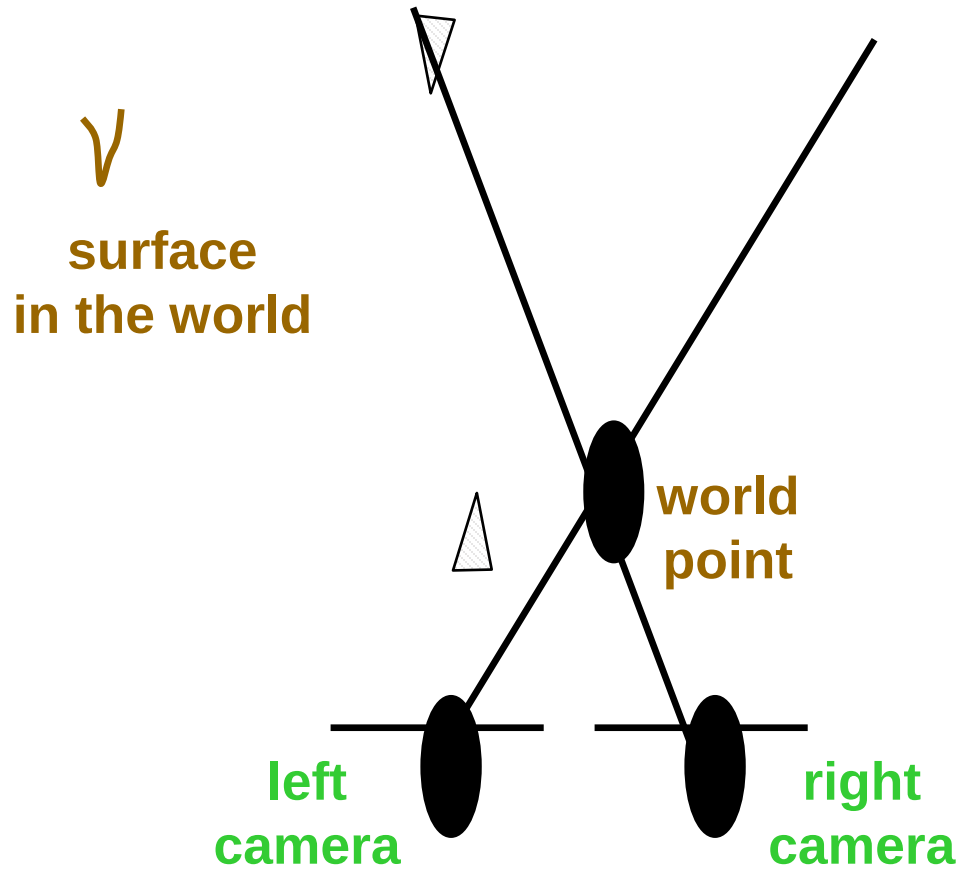
maximum disparity

uniqueness

ordering (monotonicity)

When are these violated?

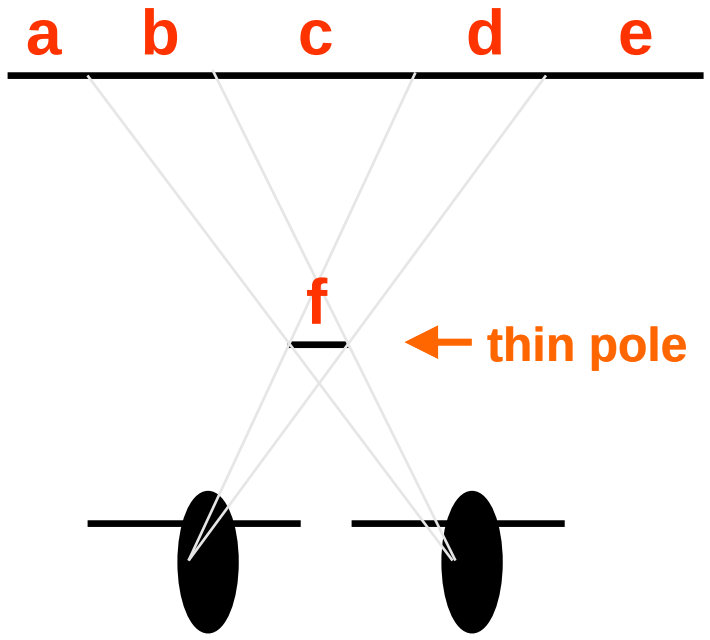
Forbidden zone

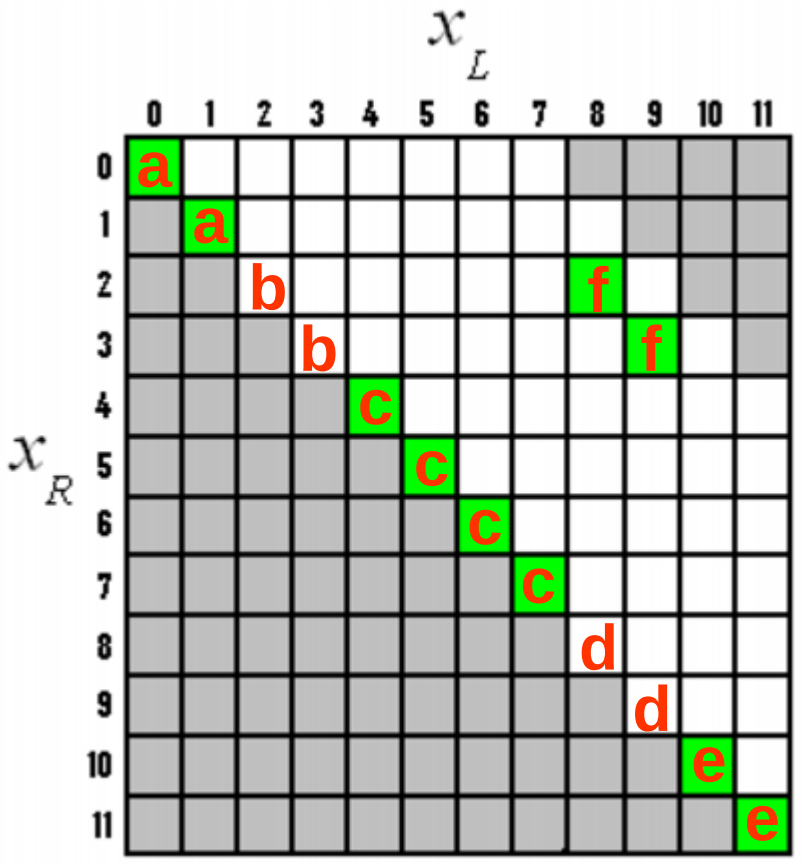


**no matches
are possible in the
hourglass-shaped region
(if surface is continuous)**

(Related to ordering constraint)

Violation of ordering constraint





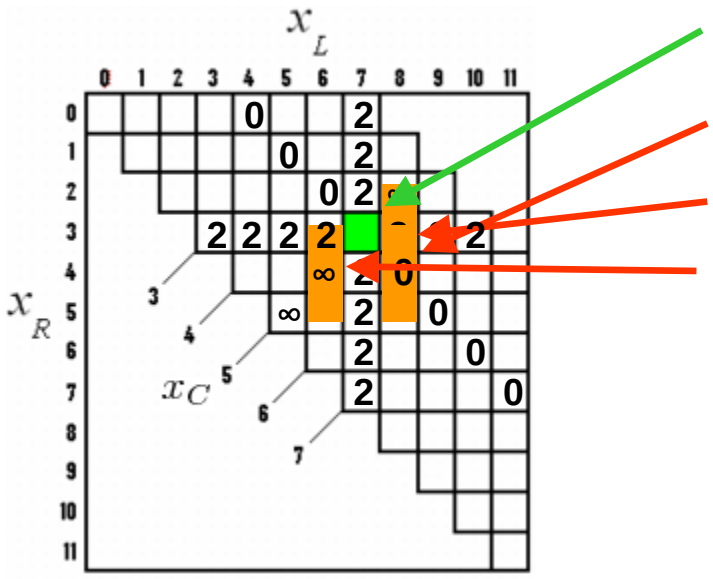
Disparity gradient

$$x_C = \frac{1}{2}(x_L + x_R)$$ ← Cyclopean coordinate

x_1 in I_L matches x'_1 in I_R: $d_1 = x_1 - x'_1$

x_2 in I_L matches x'_2 in I_R: $d_2 = x_2 - x'_2$

disparity gradient: $$\left|\frac{\partial d}{\partial x_c}\right| = \frac{d_2 - d_1}{\frac{1}{2}(x_2 + x'_2) - \frac{1}{2}(x_1 + x'_1)} = \frac{2(d_2 - d_1)}{x_2 + x'_2 - x_1 - x'_1}$$



$d_1 = 7-3 = 4$ $x_c = (7+3)/2 = 5$

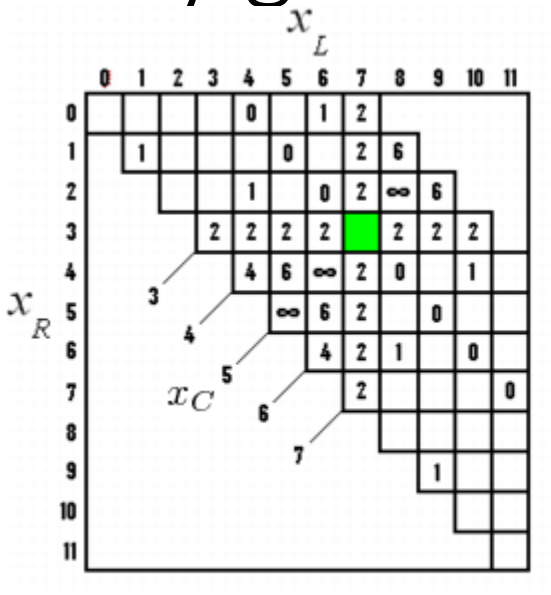
$d_2 = 8-4 = 4$ $x_c = (8+4)/2 = 6$ d.g. = 0

$d_2 = 8-3 = 5$ $x_c = (8+3)/2 = 5.5$ d.g. = 2

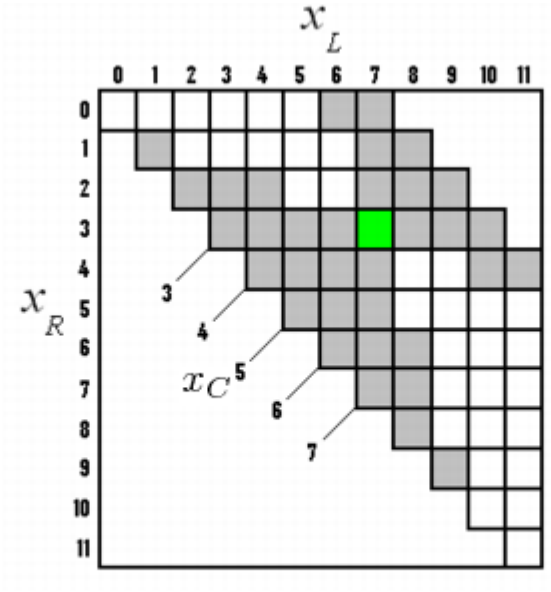
$d_2 = 6-4 = 2$ $x_c = (6+4)/2 = 5$ d.g. = ∞

…

Disparity gradient constraint

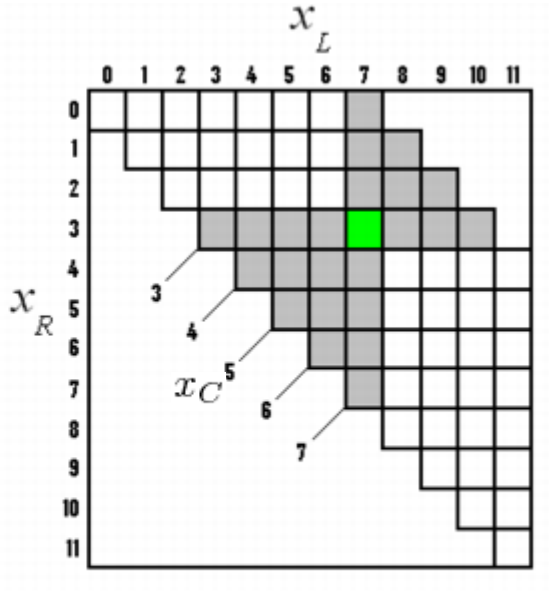


$$\left|\frac{\partial d}{\partial x_c}\right|$$



$$\left|\frac{\partial d}{\partial x_c}\right| \leq 1$$

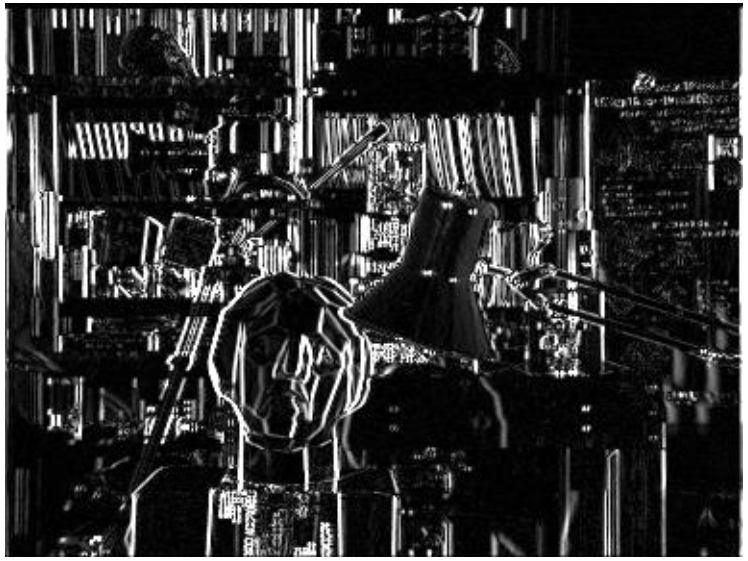
(human visual system imposes this)



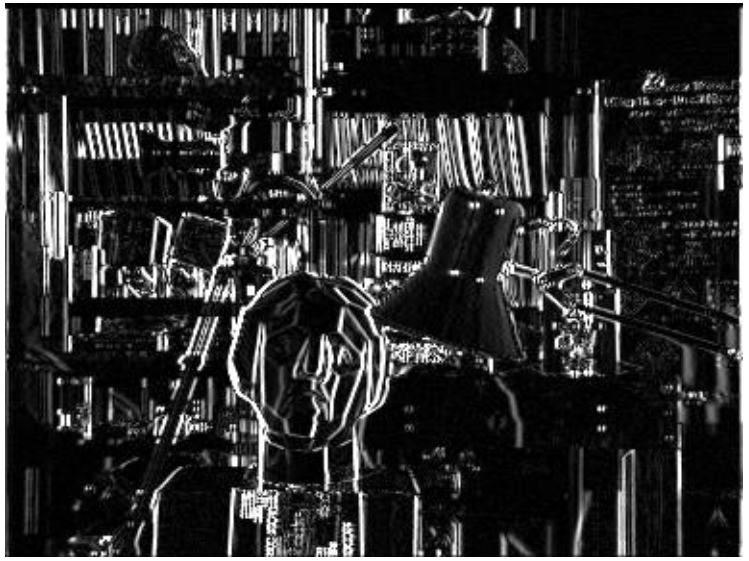
$$\left|\frac{\partial d}{\partial x_c}\right| \leq 2$$

(same as ordering constraint)

Figural continuity constraint



right



left

[University of Tsukuba]

Outline

- Stereo basics
- Binocular stereo matching
- Advanced stereo techniques

Dynamic Programming: 1D Search

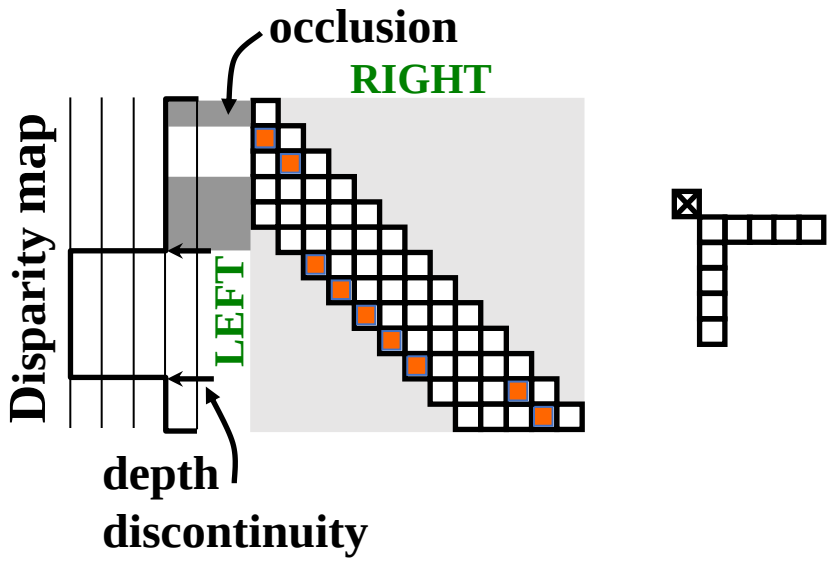
string editing:

		c	a	r	t
	0	1	2	3	4
c	1	0	1	2	3
a	2	1	0	1	2
t	3	2	1	1	1

c a t
| | \
c a r t

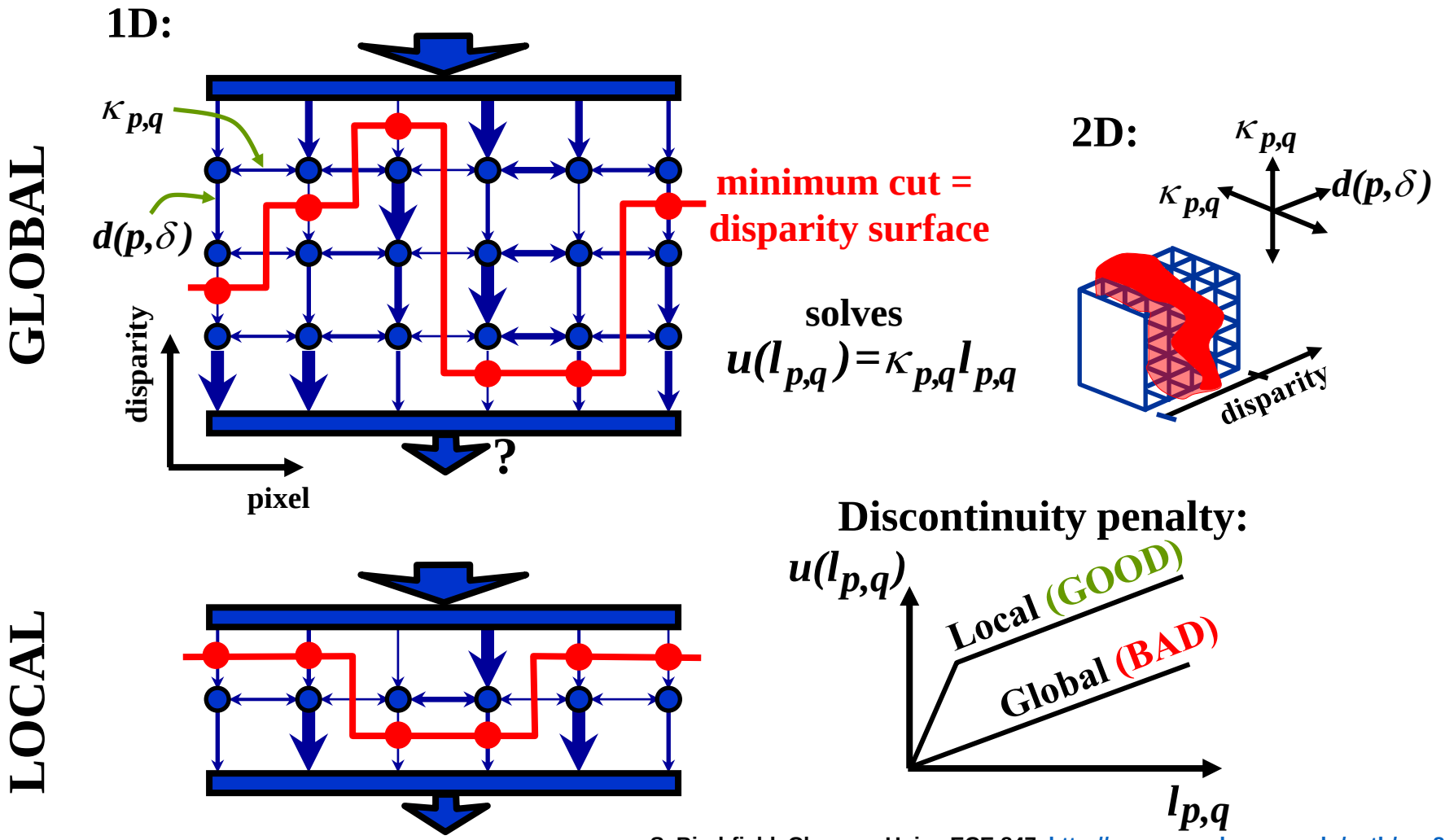
penalties:
- mismatch = 1
- insertion = 1
- deletion = 1

stereo matching:

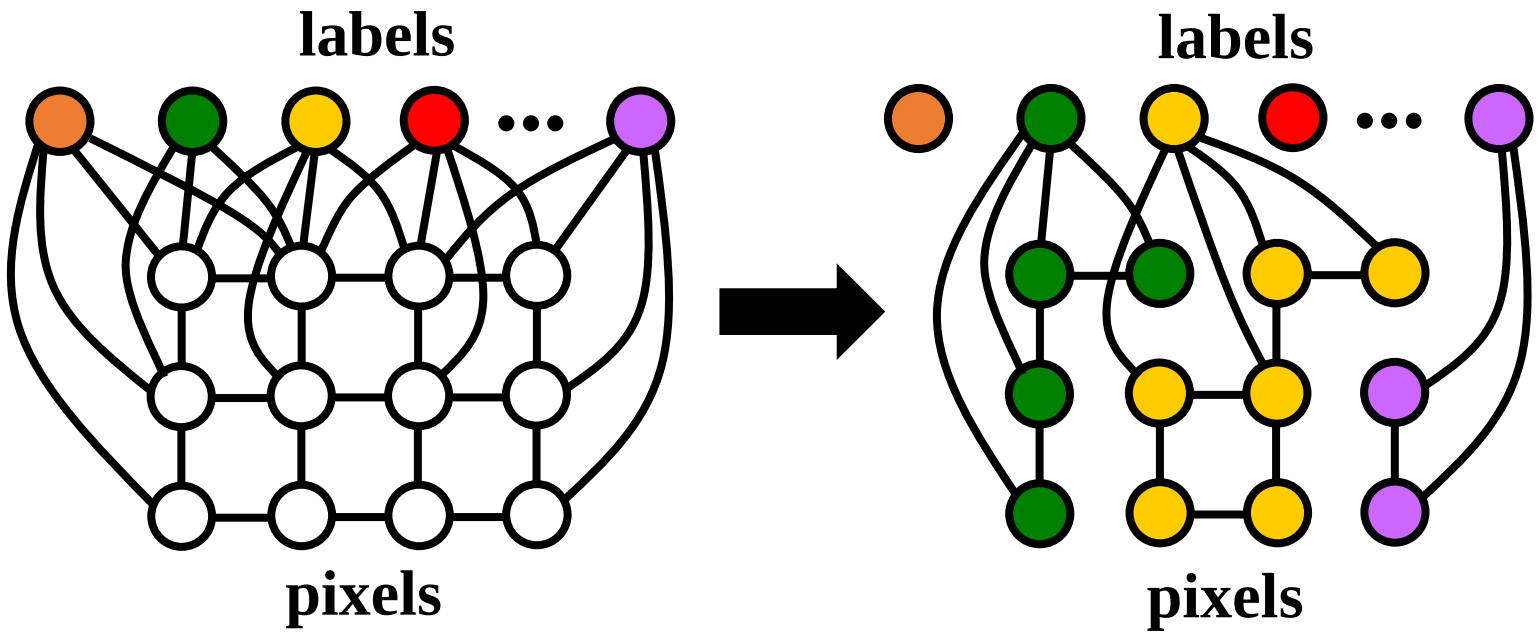


Minimizing a 2D Cost Functional

$$\text{Minimize: } \min_{\delta} E_{data} + E_{smoothness} = \sum_{p} d(p,\delta) + \sum_{\{p,q\} \in N} \kappa_{p,q} u(l_{p,q})$$

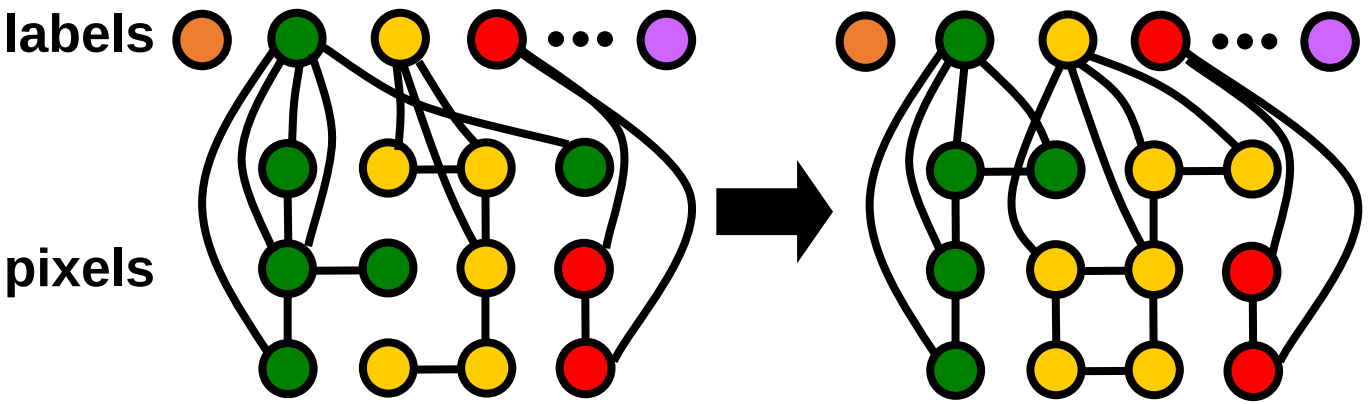


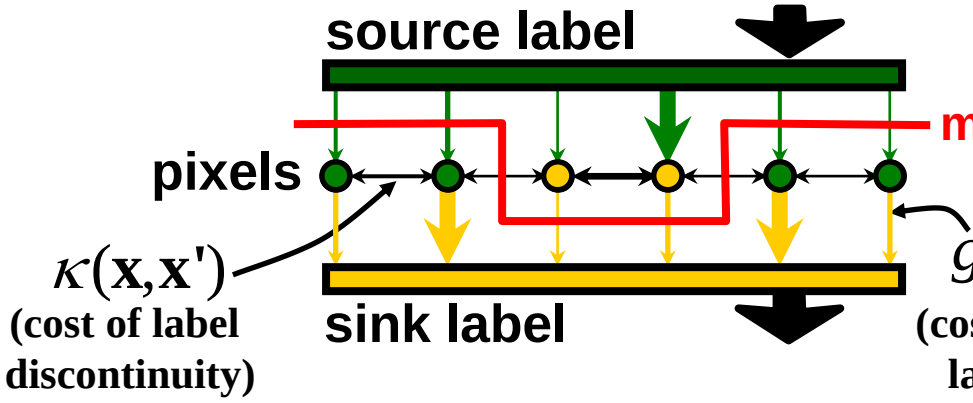
Multiway-Cut: 2D Search



[Boykov, Veksler, Zabih 1998]

Multiway-Cut Algorithm

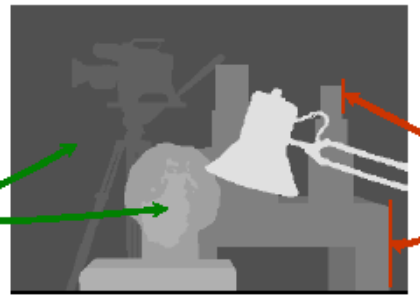




Minimizes $\sum_{\mathbf{x}} g(\mathbf{x}, f(\mathbf{x})) + \sum_{(\mathbf{x},\mathbf{x}')} \kappa(\mathbf{x},\mathbf{x}')[f(\mathbf{x}) \neq f(\mathbf{x}')]$

Energy minimization

Disparity continuous in most places,



except at depth discontinuities

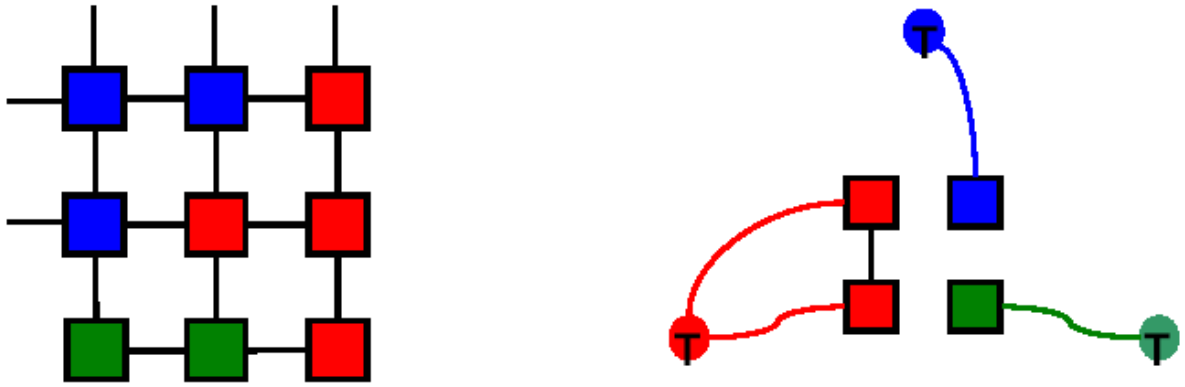
1. Matching pixels should have similar intensities.
2. Most nearby pixels should have similar disparities

→ Minimize

$$\sum [I_1(x + D(x,y), y) - I_2(x,y)]^2$$
$$+ \lambda \sum [D(x+1,y) - D(x,y)]^2$$
$$+ \mu \sum [D(x,y+1) - D(x,y)]^2$$

(Slide from Pascal Fua)

Graph Cut



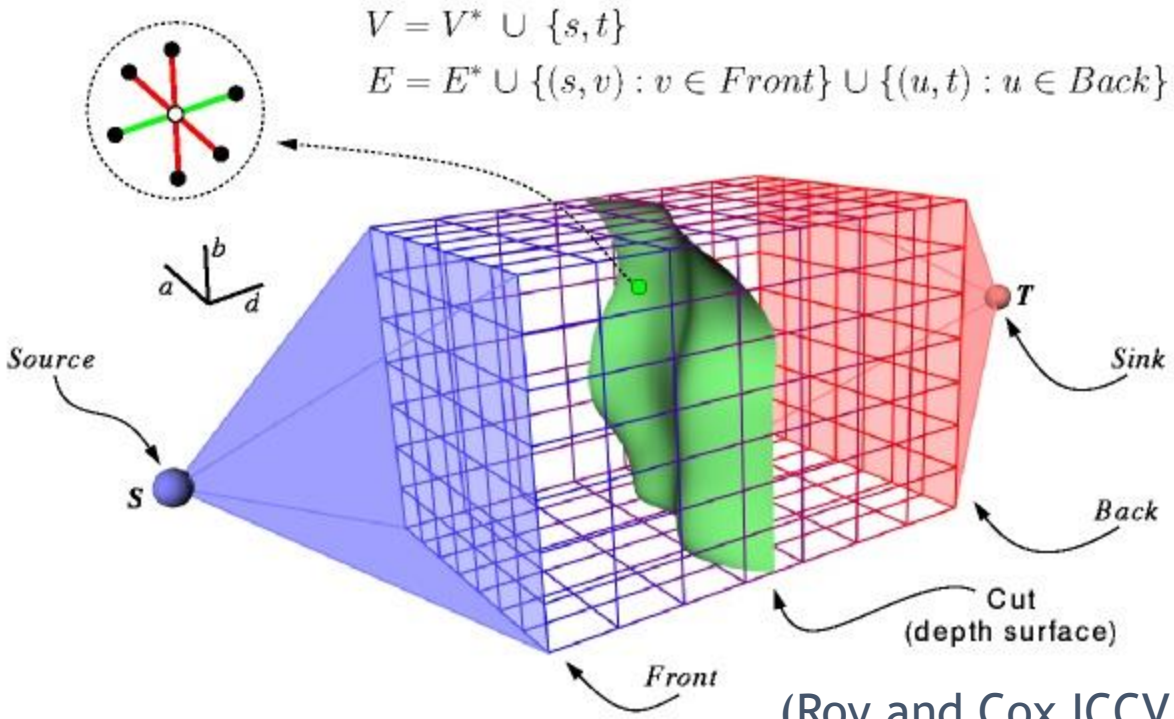
1. Stereo is a labeling problem
2. Graph cut corresponds to a labeling.

→ **Assign edge weights cleverly so that the min-weight cut gives the minimum energy!**

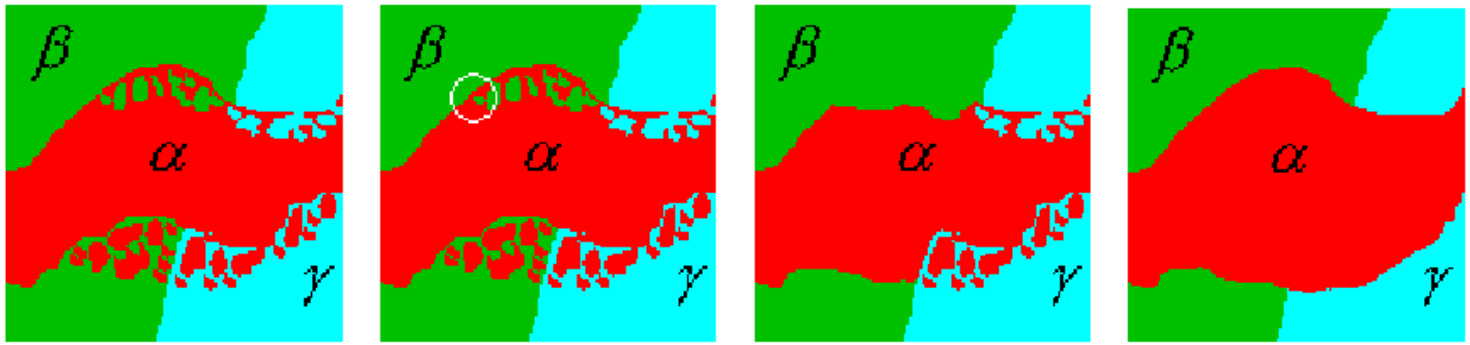
(general formulation requires multi-way cut!)

(Slide from Pascal Fua)

Simplified graph cut



(Roy and Cox ICCV'98)

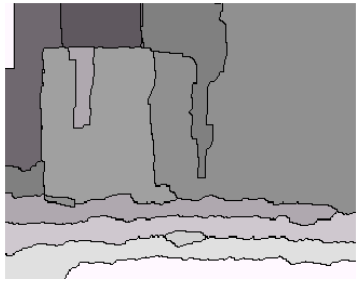


(a) initial labeling (b) standard move (c) α-β-swap (d) α-expansion

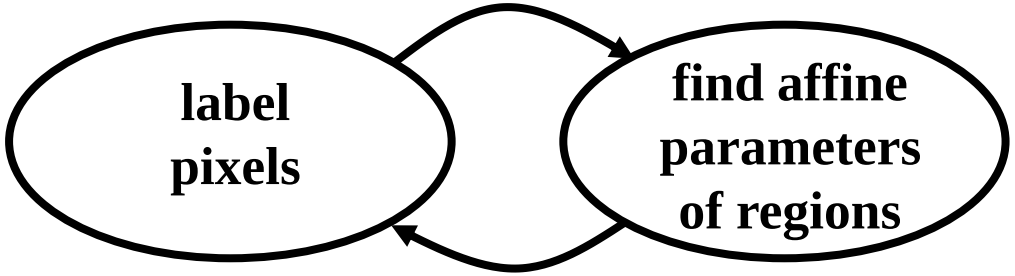
(Boykov et al ICCV'99)

Correspondence as Segmentation

- **Problem:**
  disparities (fronto-parallel) O(Δ)
  surfaces (slanted) $O(\Delta \sigma^2 n)$
  => computationally intractable!

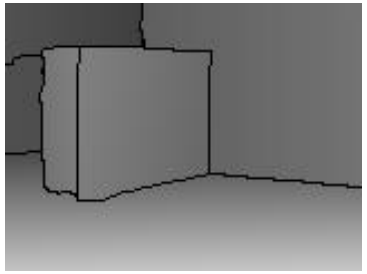


- **Solution: iteratively determine which labels to use**

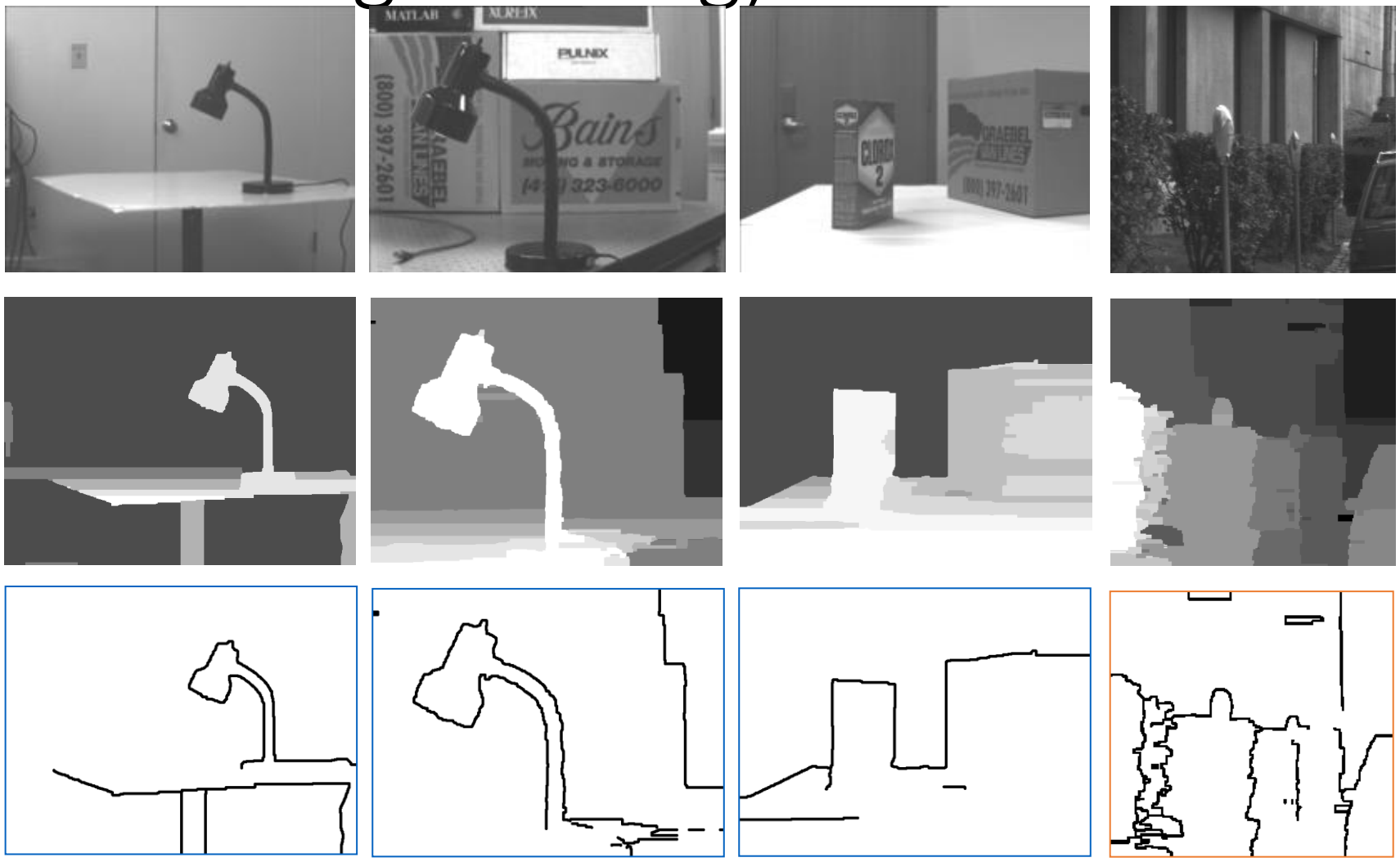


multiway-cut (Expectation)

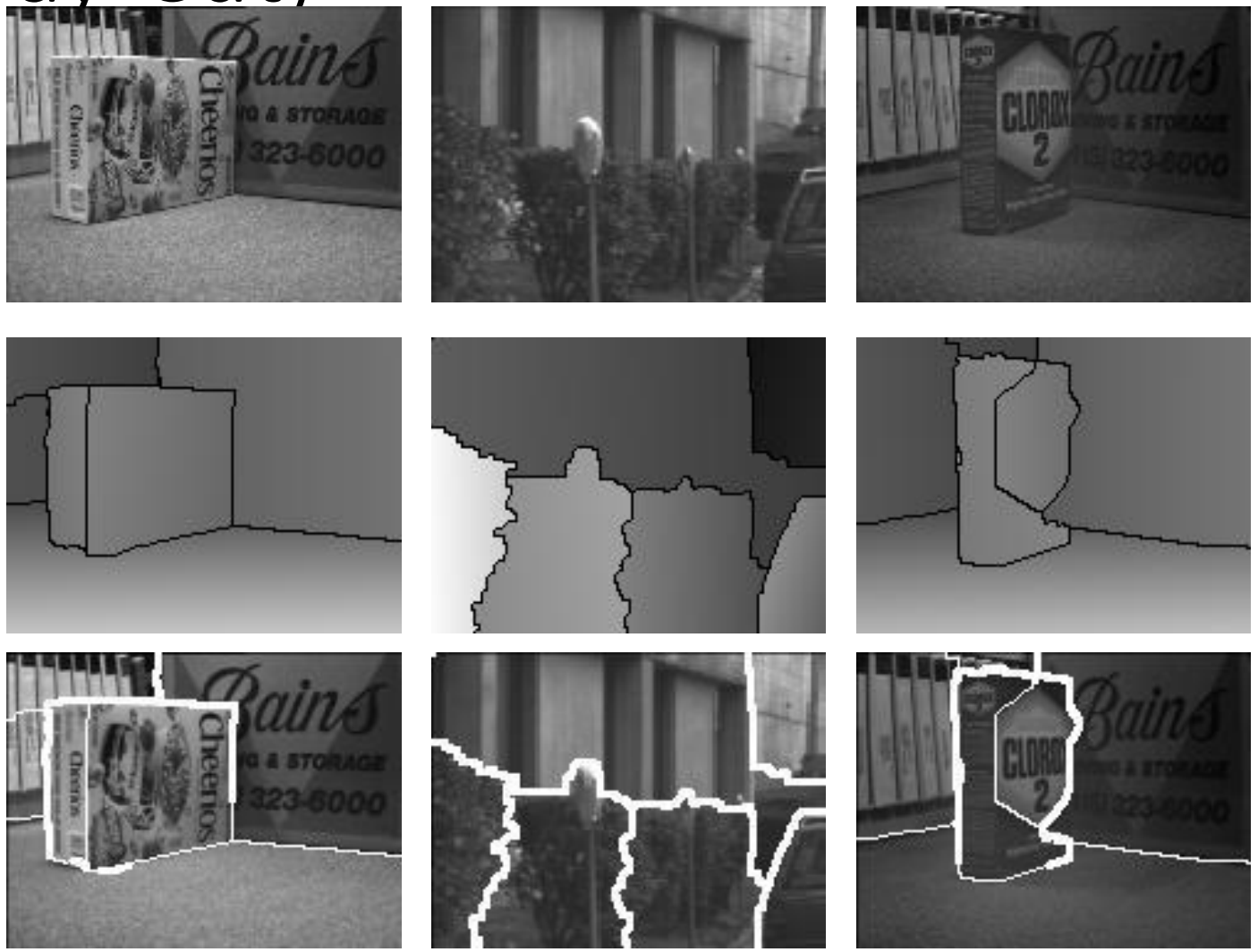
Newton-Raphson (Maximization)



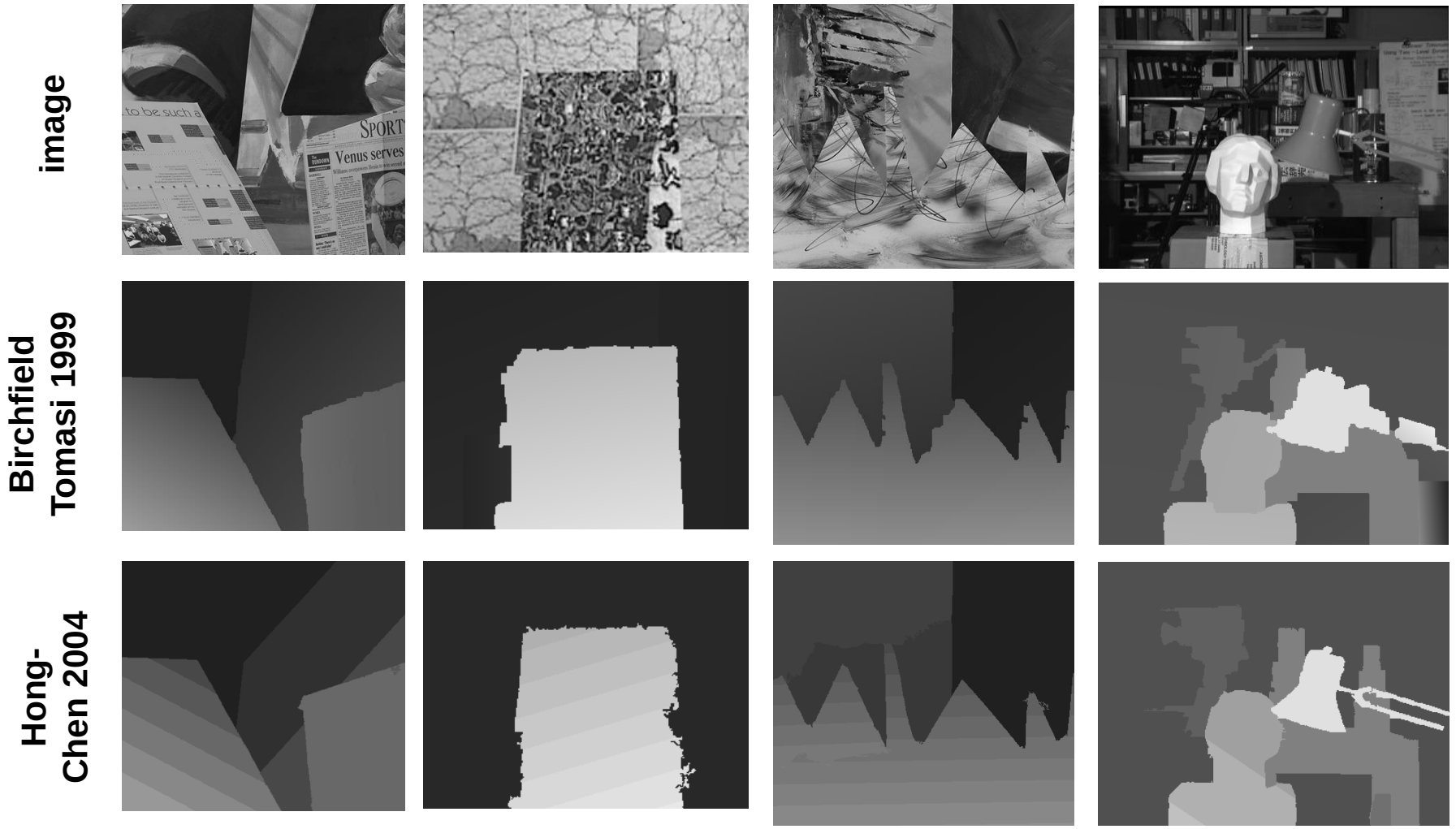
Stereo Results (Dynamic Programming)



Stereo Results (Multiway-Cut)



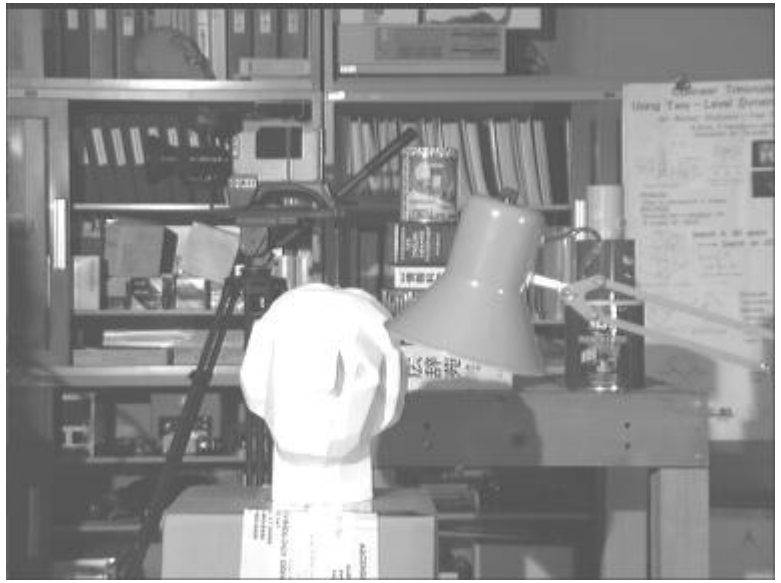
Stereo Results on Middlebury Database



Untextured regions remain a challenge

Dynamic programming **Multiway-cut**

Results: dynamic programming

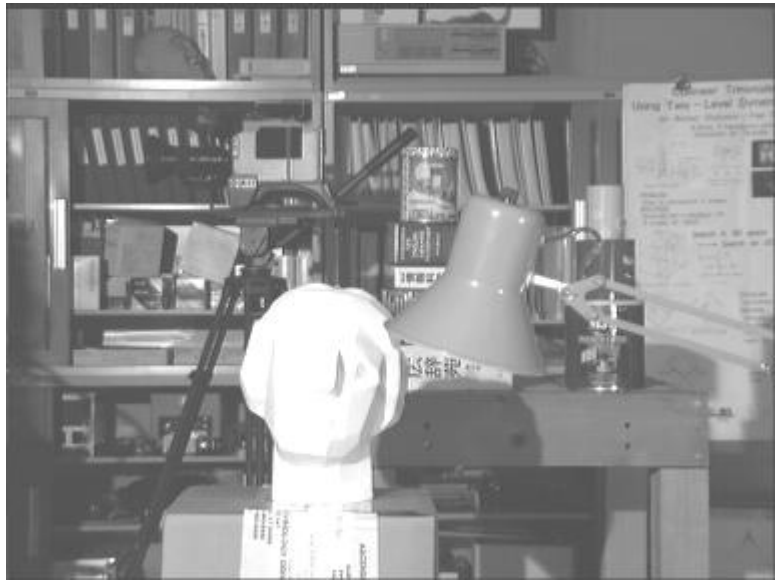


left

disparity map

[Bobick & Intille]

Results: multiway cut



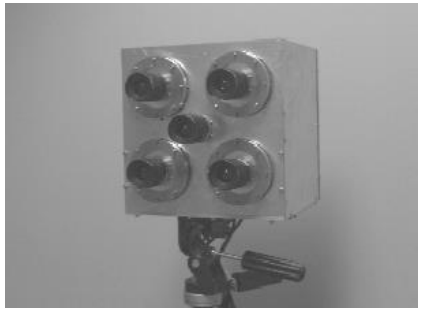
left

disparity map

[Kolmogorov & Zabih]

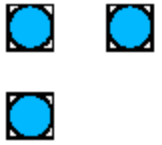
Results: multiway cut (untextured)

Multi-camera configurations



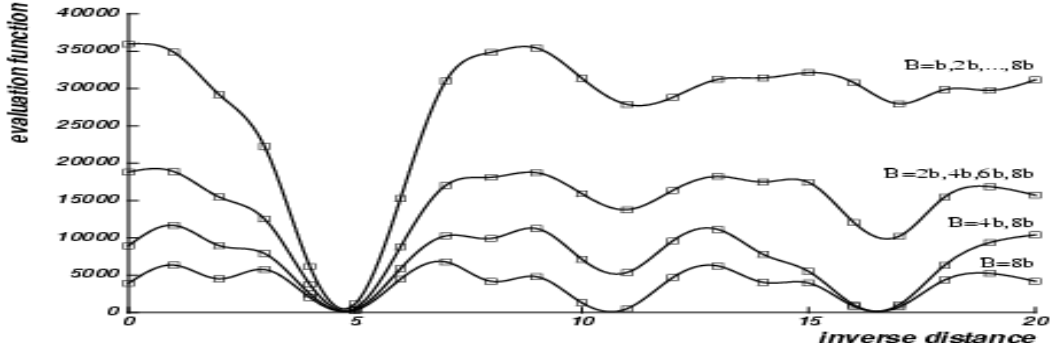
3 cameras give both robustness and precision

4 cameras give additional redundancy



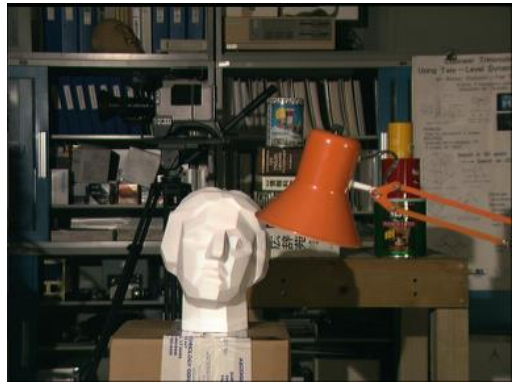
3 cameras in a T arrangement allow the system to see vertical lines.

(illustration from Pascal Fua)

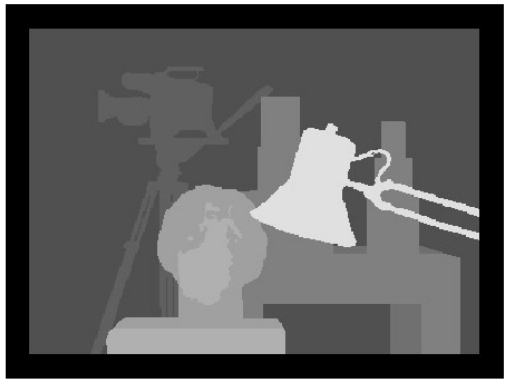


Okutami and Kanade

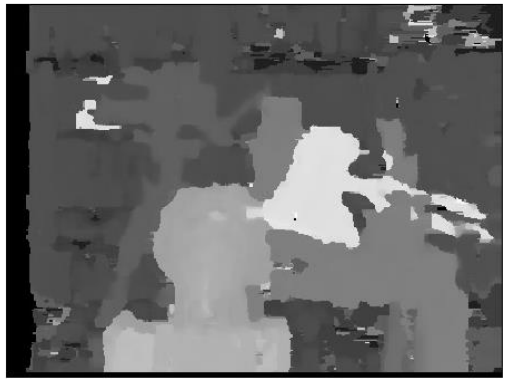
M. Pollefeys, http://www.cs.unc.edu/Research/vision/comp256fall03/



Tsukuba dataset



True disparities



16 – Fast Correlation

*2 – Dynamic progr.

*1 – SSD+MF

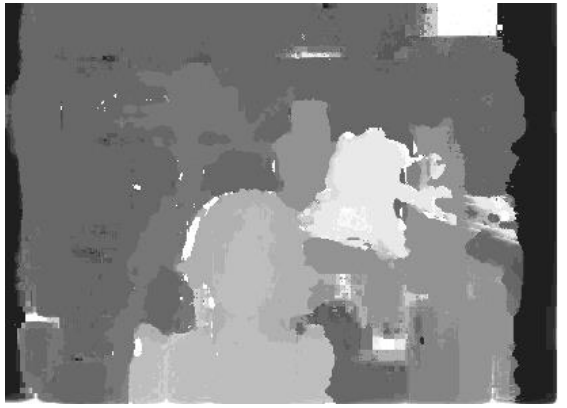
11 – GC + occlusions

19 – Belief propagation

Real-time stereo on GPU

(Yang and Pollefeys, CVPR2003)

- Computes Sum-of-Square-Differences (use pixelshader)
- Hardware mip-map generation for aggregation over window
- Trade-off between small and large support window



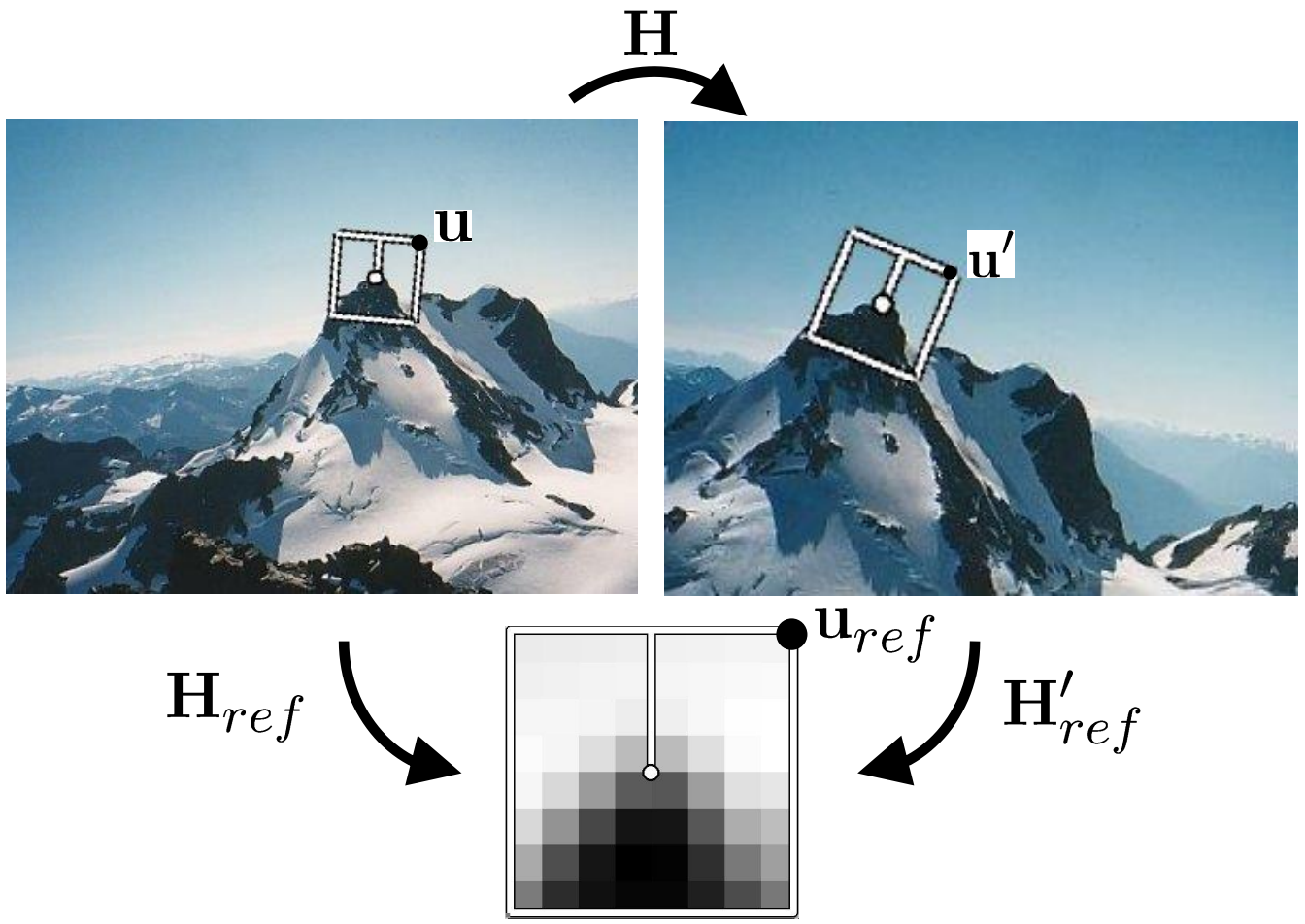
290M disparity hypothesis/sec (Radeon9800pro)
e.g. 512x512x36disparities at 30Hz

GPU is great for vision too!

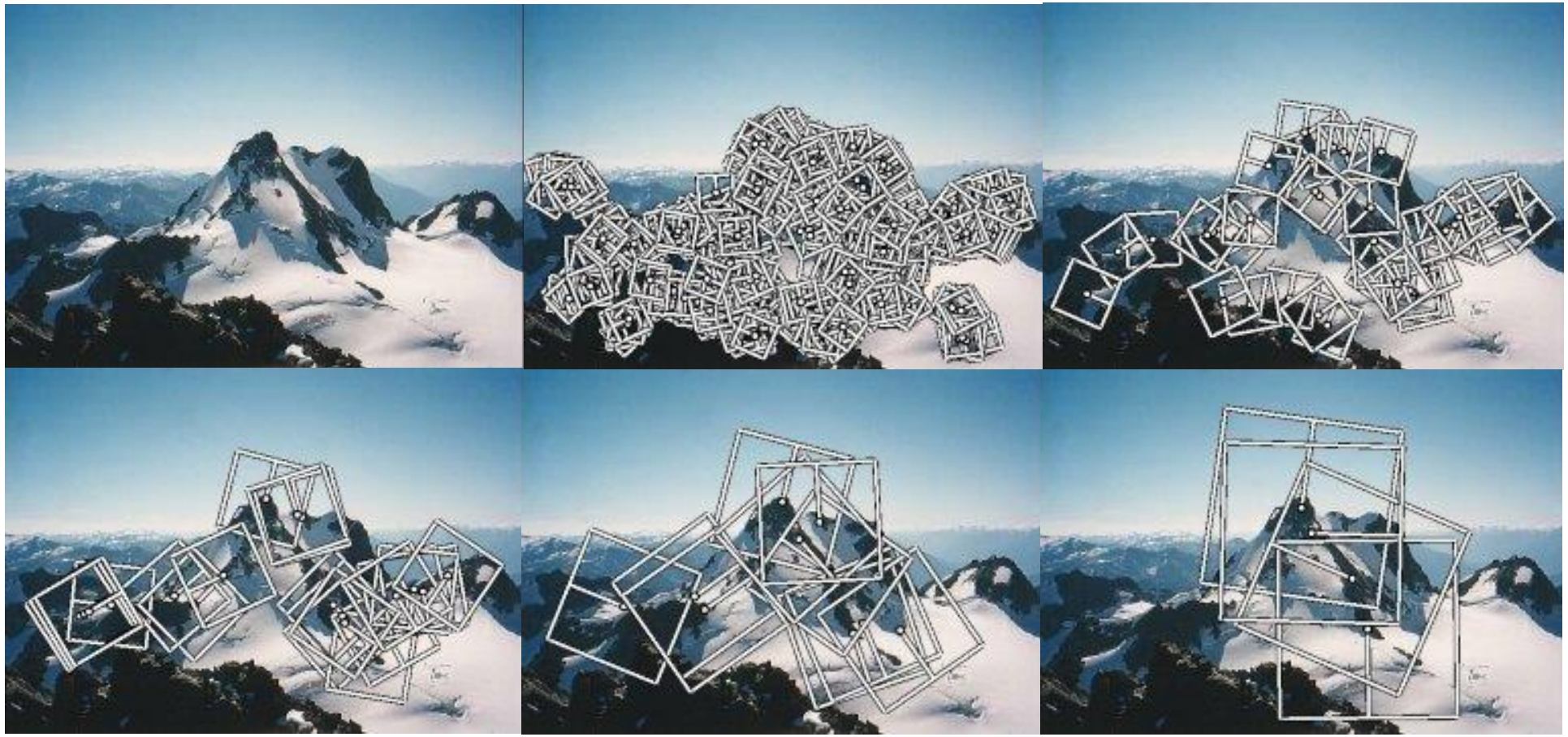
The SIFT (Scale Invariant Feature Transform) Detector and Descriptor

developed by David Lowe
University of British Columbia
Initial paper ICCV 1999
Newer journal paper IJCV 2004

Review: Matt Brown's Canonical Frames



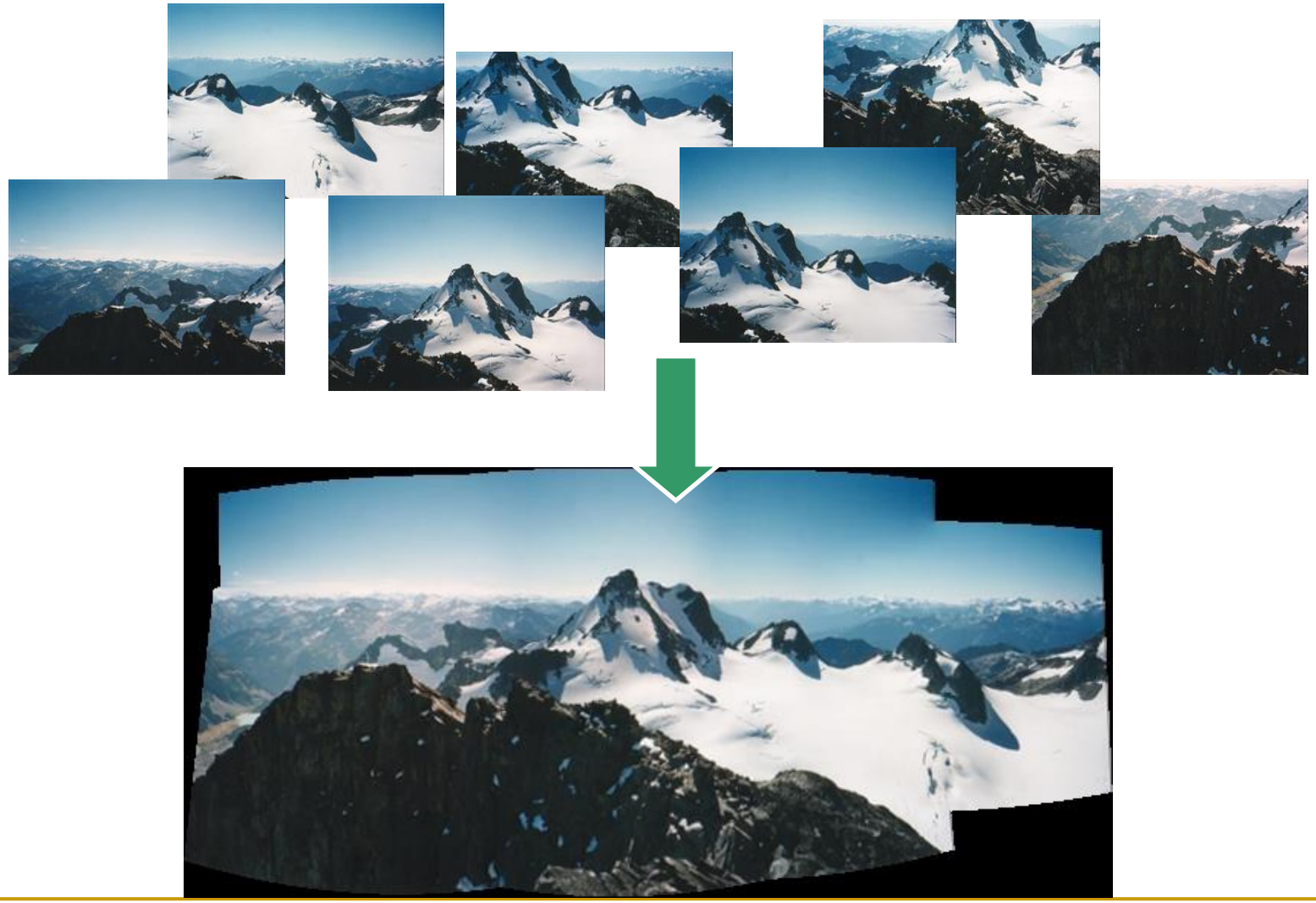
Multi-Scale Oriented Patches



- Extract oriented patches at multiple scales

[Brown, Szeliski, Winder CVPR 2005]

Application: Image Stitching



[Microsoft Digital Image Pro version 10]

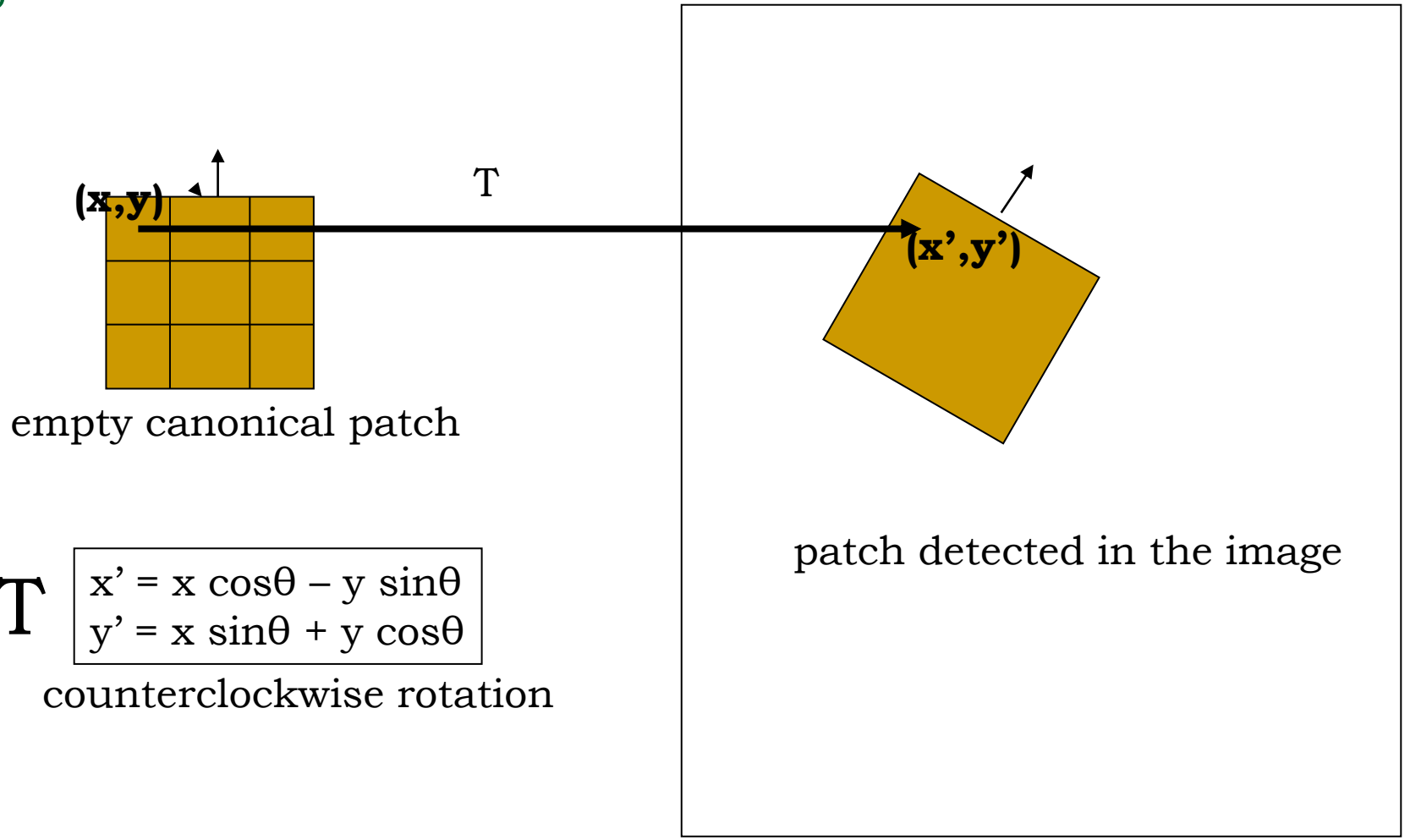
Ideas from Matt's Multi-Scale Oriented Patches

- 1. Detect an interesting patch with an interest operator. Patches are translation invariant.
- 2. Determine its dominant orientation.
- 3. Rotate the patch so that the dominant orientation points upward. This makes the patches rotation invariant.
- 4. Do this at multiple scales, converting them all to one scale through sampling.
- 5. Convert to illumination "invariant" form

Implementation Concern: How do you rotate a patch?

- Start with an “empty” patch whose dominant direction is “up”.
- For each pixel in your patch, compute the position in the detected image patch. It will be in floating point and will fall between the image pixels.
- Interpolate the values of the 4 closest pixels in the image, to get a value for the pixel in your patch.

Rotating a Patch

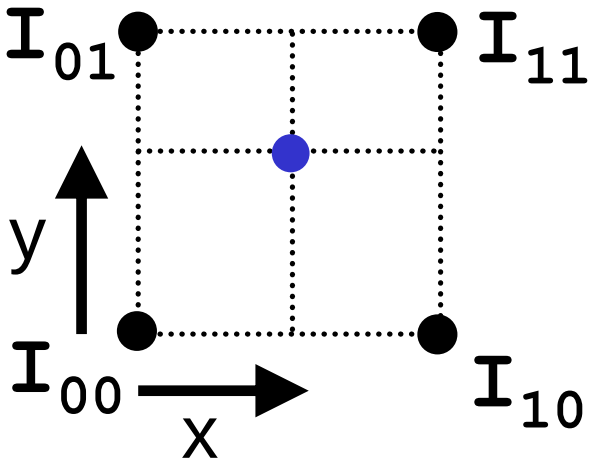


$$T \quad \begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned}$$

counterclockwise rotation

Using Bilinear Interpolation

- Use all 4 adjacent samples



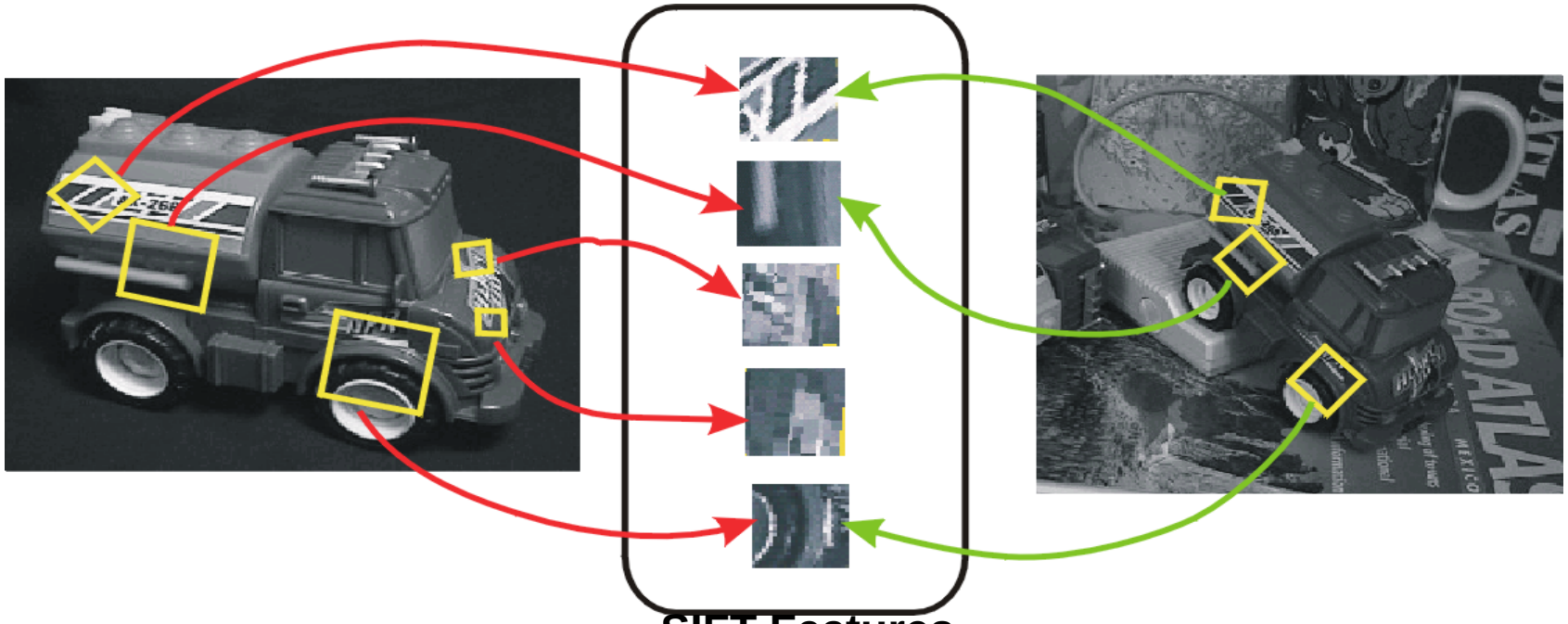
SIFT: Motivation

- The Harris operator is not invariant to scale and correlation is not invariant to rotation[1].
- For better image matching, Lowe's goal was to develop an interest operator that is invariant to scale and rotation.
- Also, Lowe aimed to create a descriptor that was robust to the variations corresponding to typical viewing conditions. The descriptor is the most-used part of SIFT.

[1]But Schmid and Mohr developed a rotation invariant descriptor for it in 1997.

Idea of SIFT

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



Claimed Advantages of SIFT

- **Locality:** features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance
- **Extensibility:** can easily be extended to wide range of differing feature types, with each adding robustness

Overall Procedure at a High Level

1. Scale-space extrema detection

   Search over multiple scales and image locations.

2. Keypoint localization

   Fit a model to detrmine location and scale.
   Select keypoints based on a measure of stability.

3. Orientation assignment

   Compute best orientation(s) for each keypoint region.

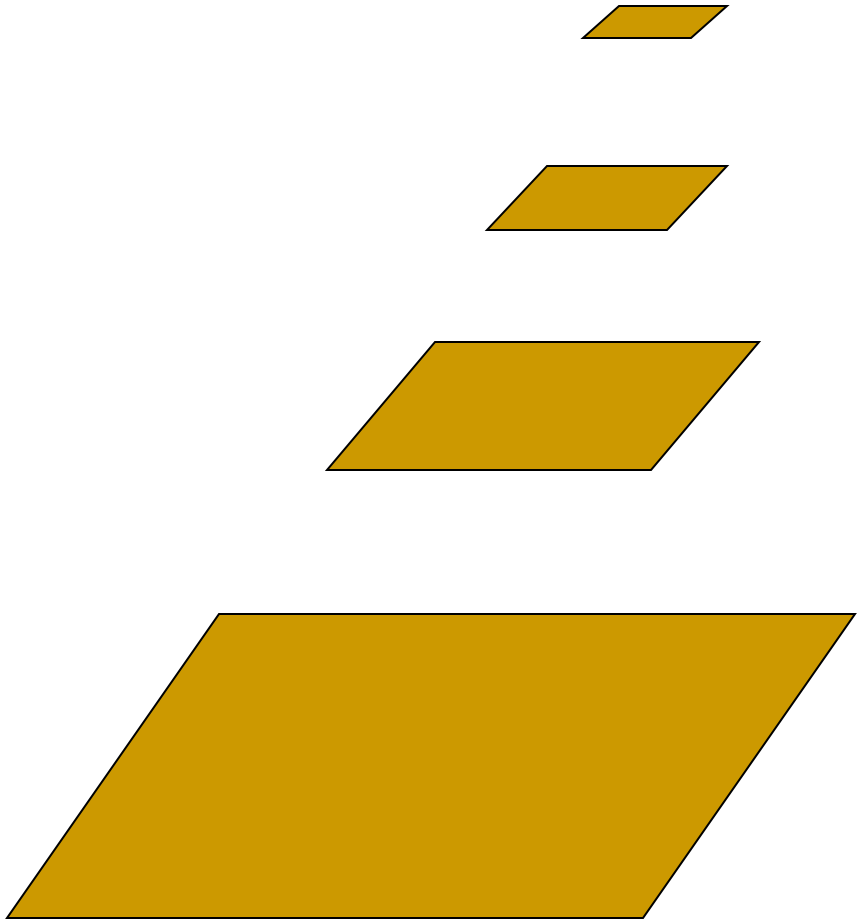
4. Keypoint description

   Use local image gradients at selected scale and rotation to describe each keypoint region.

1. Scale-space extrema detection

- Goal: Identify locations and scales that can be repeatably assigned under different views of the same scene or object.
- Method: search for stable features across multiple scales using a continuous function of scale.
- Prior work has shown that under a variety of assumptions, the best function is a Gaussian function.
- The scale space of an image is a function $L(x,y,\sigma)$ that is produced from the convolution of a Gaussian kernel (at different scales) with the input image.

Aside: Image Pyramids



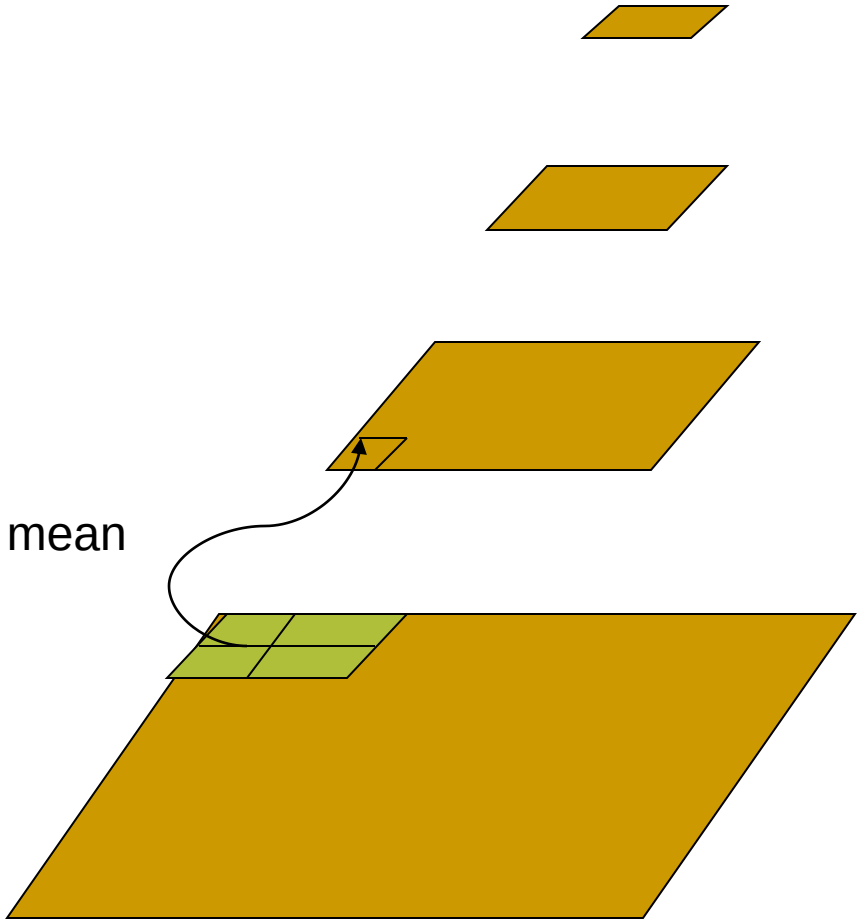
And so on.

3rd level is derived from the 2nd level according to the same funtion

2nd level is derived from the original image according to some function

Bottom level is the original image.

Aside: Mean Pyramid



And so on.

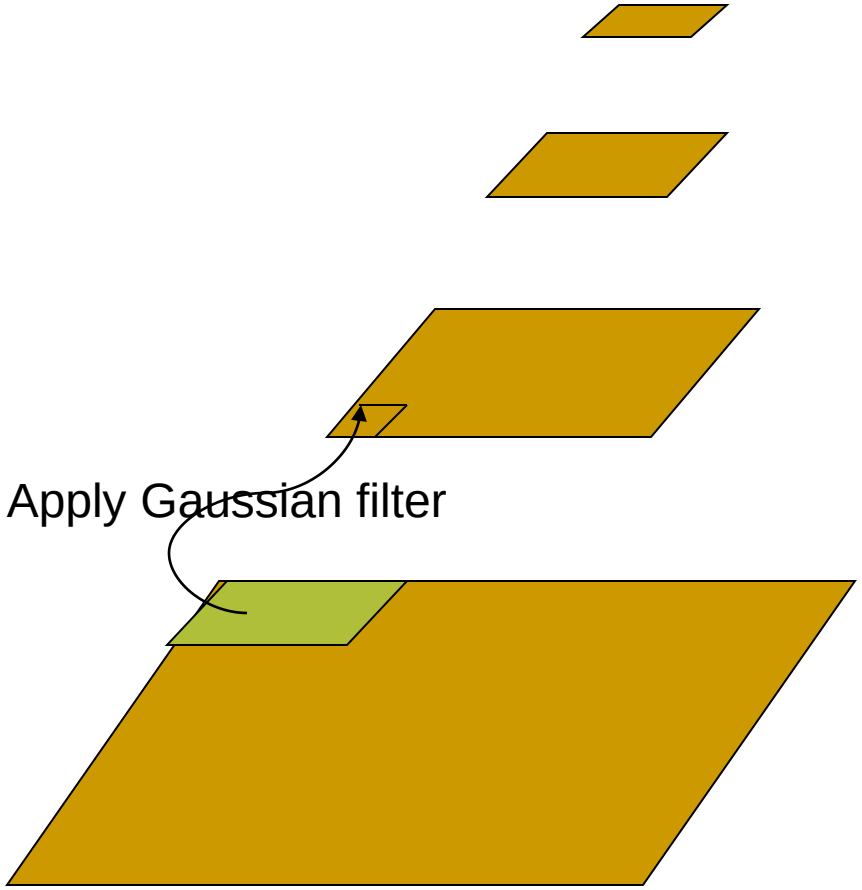
At 3rd level, each pixel is the mean of 4 pixels in the 2nd level.

At 2nd level, each pixel is the mean of 4 pixels in the original image.

Bottom level is the original image.

Aside: Gaussian Pyramid

At each level, image is smoothed and reduced in size.

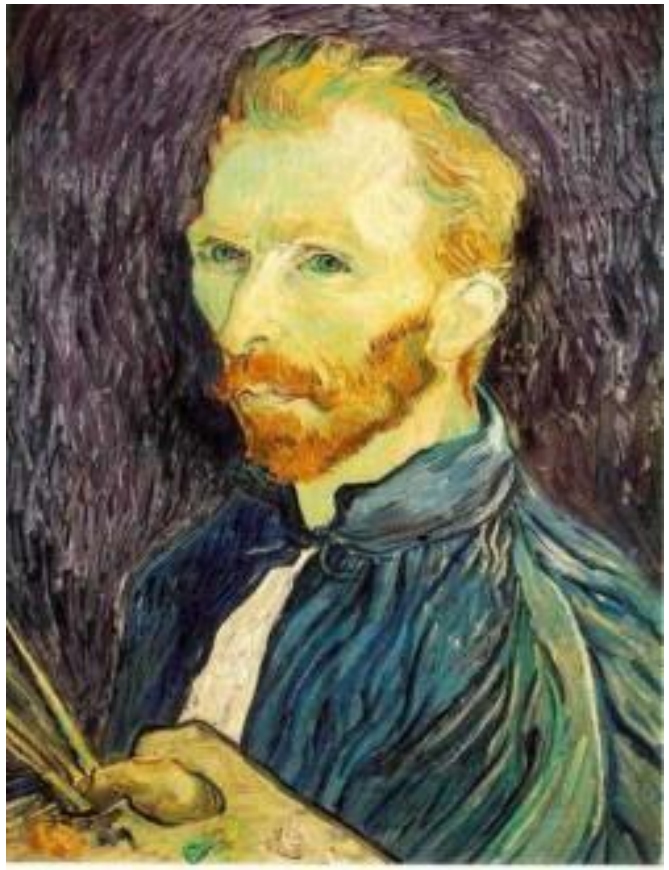


And so on.

At 2nd level, each pixel is the result of applying a Gaussian mask to the first level and then subsampling to reduce the size.

Bottom level is the original image.

Example: Subsampling with Gaussian pre-filtering



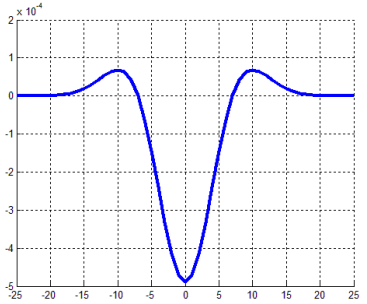
Gaussian 1/2

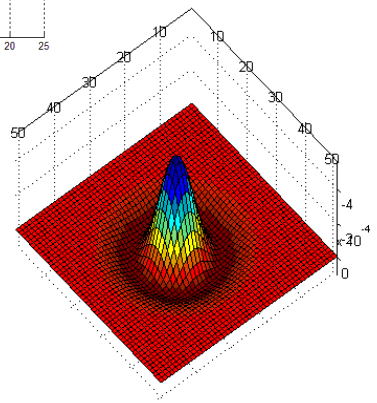
G 1/4

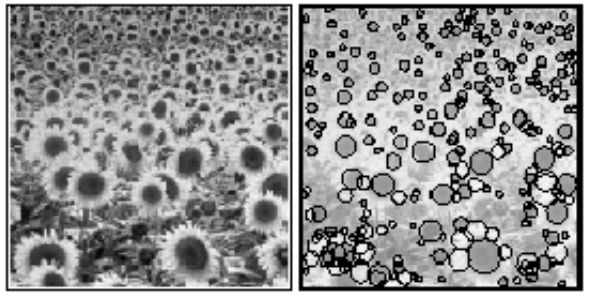
G 1/8

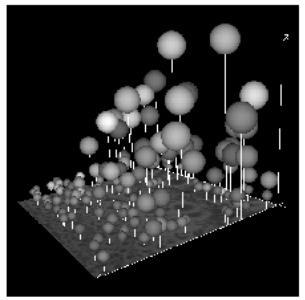
Lowe's Scale-space Interest Points

- **Laplacian of Gaussian** kernel
  - Scale normalised (x by scale^2)
  - Proposed by Lindeberg
- Scale-space detection
  - Find local maxima across scale/space
  - A good "blob" detector







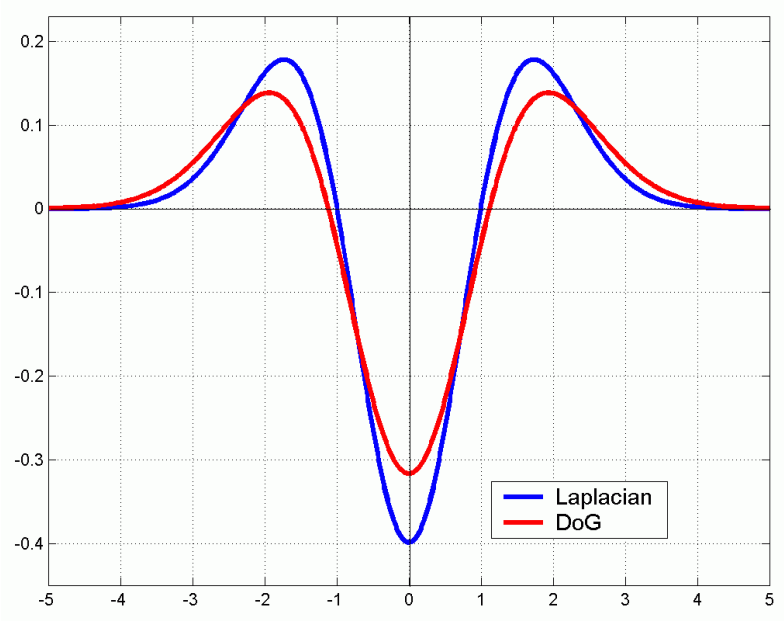


$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}}$$

$$\nabla^2 G(x, y, \sigma) = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$

[T. Lindeberg IJCV 1998]

Lowe's Scale-space Interest Points: Difference of Gaussians



- Gaussian is an ad hoc solution of heat diffusion equation
- He

$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G.$$

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

- k is not necessarily very small in practice

Lowe's Pyramid Scheme

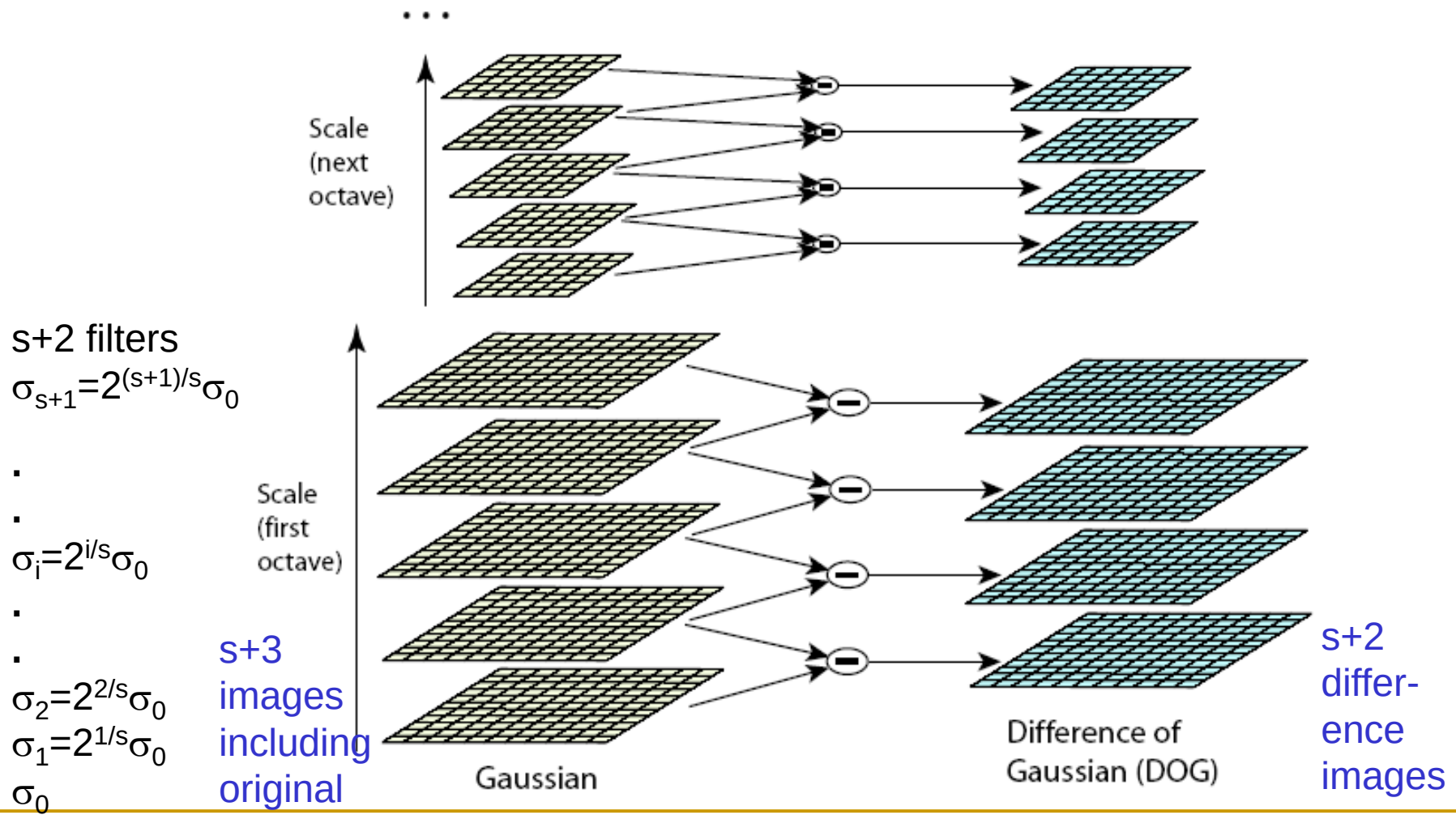
- Scale space is separated into octaves:
  - Octave 1 uses scale σ
  - Octave 2 uses scale 2σ
  - etc.

- In each octave, the initial image is repeatedly convolved with Gaussians to produce a set of scale space images.

- Adjacent Gaussians are subtracted to produce the DOG

- After each octave, the Gaussian image is down-sampled by a factor of 2 to produce an image ¼ the size to start the next level.

Lowe's Pyramid Scheme



s+2 filters

$\sigma_{s+1}=2^{(s+1)/s}\sigma_0$

.

.

$\sigma_i=2^{i/s}\sigma_0$

.

.

$\sigma_2=2^{2/s}\sigma_0$

$\sigma_1=2^{1/s}\sigma_0$

σ_0

s+3 images including original

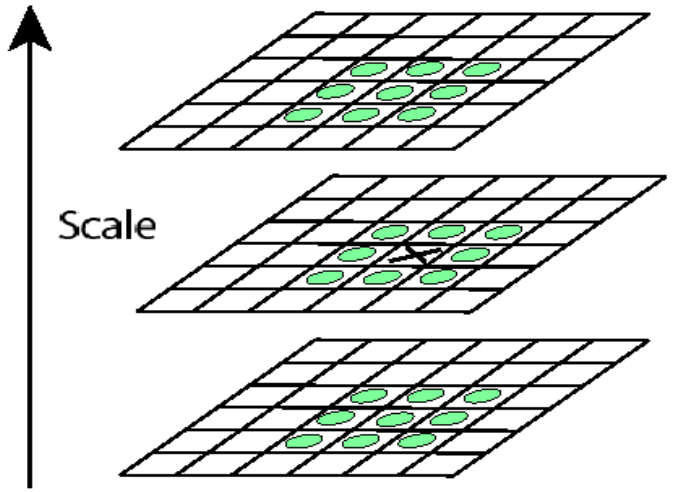
s+2 differ-ence images

The parameter **s** determines the number of images per octave.

Key point localization

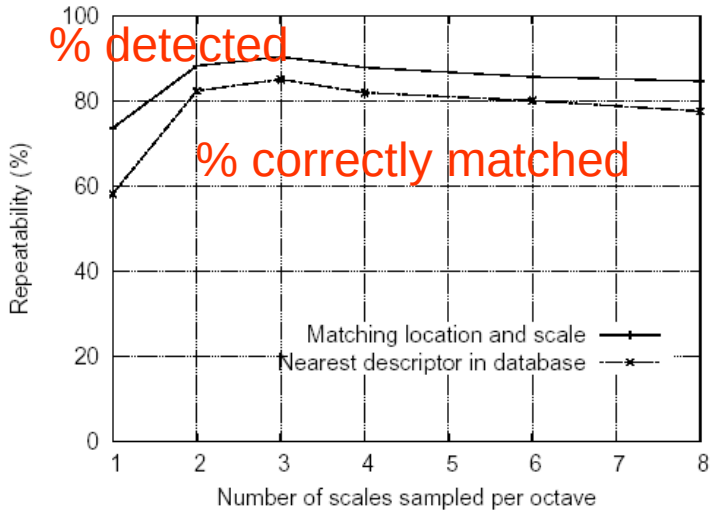
s+2 difference images.
top and bottom ignored.
s planes searched.

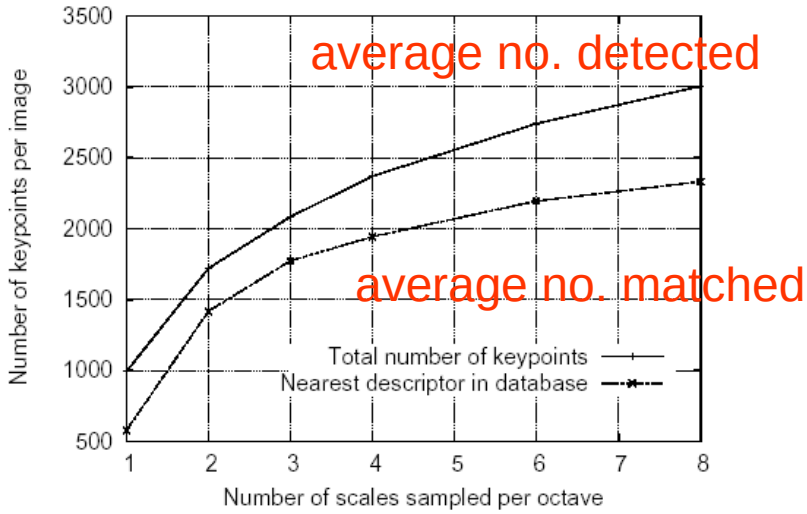
- Detect maxima and minima of difference-of-Gaussian in scale space
- Each point is compared to its 8 neighbors in the current image and 9 neighbors each in the scales above and below



For each max or min found, output is the **location** and the **scale**.

Scale-space extrema detection: experimental results over 32 images that were synthetically transformed and noise added.





- Sampling in scale for efficiency
  - How many scales should be used per octave? S=?
    - More scales evaluated, more keypoints found
    - S < 3, stable keypoints increased too
    - S > 3, stable keypoints decreased
    - S = 3, maximum stable keypoints found

Keypoint localization

- Once a keypoint candidate is found, perform a detailed fit to nearby data to determine
  - location, scale, and ratio of principal curvatures
- In initial work keypoints were found at location and scale of a central sample point.
- In newer work, they fit a 3D quadratic function to improve interpolation accuracy.
- The Hessian matrix was used to eliminate edge responses.

Eliminating the Edge Response

- Reject flats:
  - $|D(\hat{\mathbf{x}})|$ < 0.03
- Reject edges:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Let α be the eigenvalue with larger magnitude and β the smaller.

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

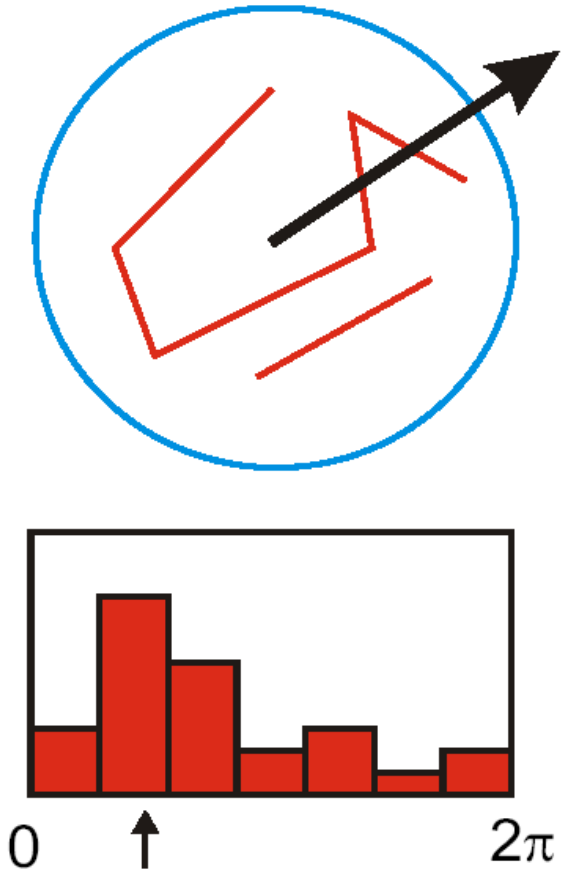
Let $r = \alpha/\beta$. So $\alpha = r\beta$

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$

$(r+1)^2/r$ is at a min when the 2 eigenvalues are equal.

  - r < 10
- What does this look like?

3. Orientation assignment



- Create histogram of local gradient directions at selected scale
- Assign canonical orientation at peak of smoothed histogram
- Each key specifies stable 2D coordinates (x, y, scale,orientation)

If 2 major orientations, use both.

Keypoint localization with orientation

233x189

832

initial keypoints

729

keypoints after
gradient threshold

536

keypoints after
ratio threshold

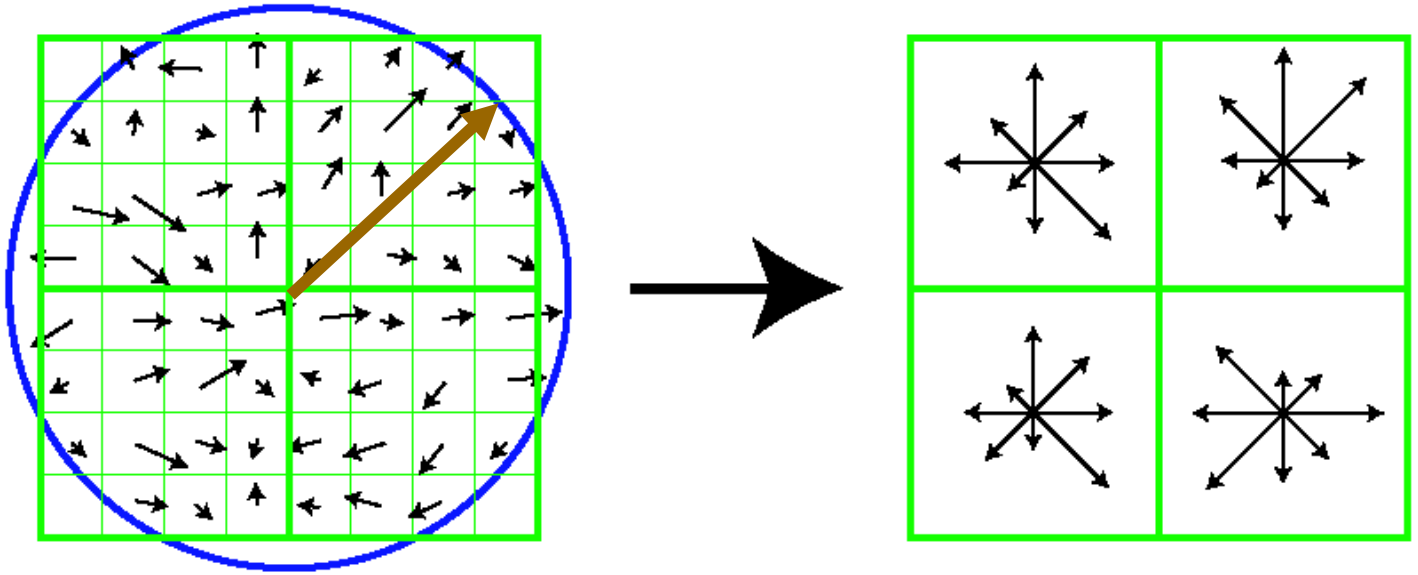
4. Keypoint Descriptors

- At this point, each keypoint has
  - location
  - scale
  - orientation
- Next is to compute a descriptor for the local image region about each keypoint that is
  - highly distinctive
  - invariant as possible to variations such as changes in viewpoint and illumination

Normalization

- Rotate the window to standard orientation
- Scale the window size based on the scale at which the point was found.

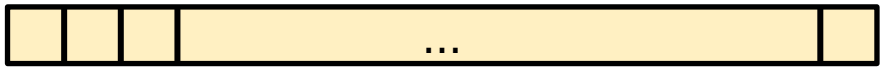
Lowe's Keypoint Descriptor (shown with 2 X 2 descriptors over 8 X 8)



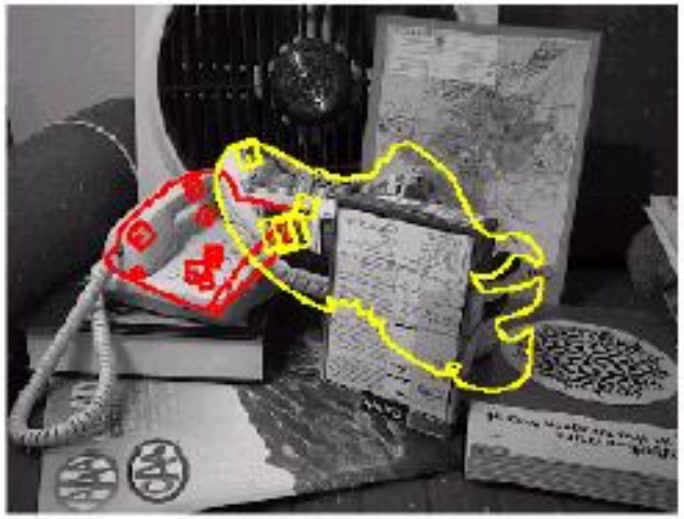
In experiments, 4x4 arrays of 8 bin histogram is used, a total of 128 features for one keypoint

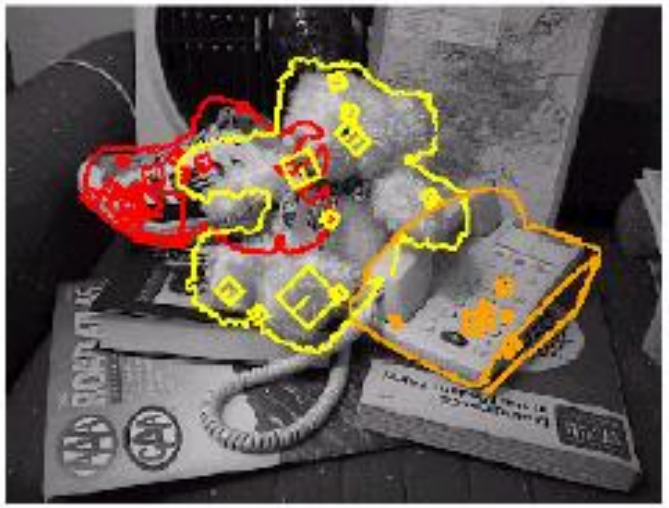
Lowe's Keypoint Descriptor

- use the normalized region about the keypoint
- compute gradient magnitude and orientation at each point in the region
- weight them by a Gaussian window overlaid on the circle
- create an orientation histogram over the 4 X 4 subregions of the window
- 4 X 4 descriptors over 16 X 16 sample array were used in practice. 4 X 4 times 8 directions gives a vector of 128 values.



Using SIFT for Matching “Objects”





Uses for SIFT

- Feature points are used also for:
  - Image alignment (homography, fundamental matrix)
  - 3D reconstruction (e.g. Photo Tourism)
  - Motion tracking
  - Object recognition
  - Indexing and database retrieval
  - Robot navigation
  - … many others

[Photo Tourism: Snavely et al. SIGGRAPH 2006]