



CS 558:

Computer Vision

5th Set of Notes

Instructor: Enrique Dunn

Webpage: www.cs.stevens.edu/~edunn

E-mail: edunn@stevens.edu

Office: North Bldg 219

Overview

- Hough Transform
- Template Matching
- Image Alignment
 - Based on slides by S. Lazebnik, K. Grauman and D. Hoiem

Voting schemes

- Let each feature vote for all the models that are compatible with it
- Hopefully the noise features will not vote consistently for any single model

Hough transform

- An early type of voting scheme
- General outline:
 - Discretize *parameter space* into bins
 - For each feature point in the image, put a vote in every bin in the parameter space that could have generated this point
 - Find bins that have the most votes

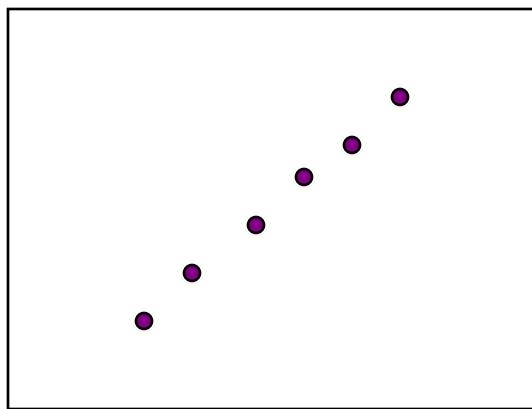
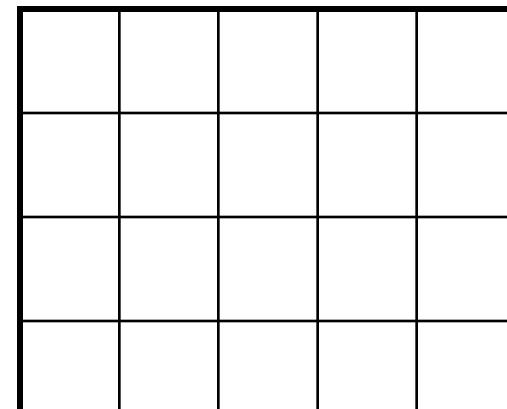
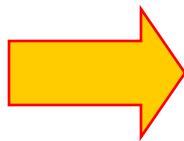


Image space

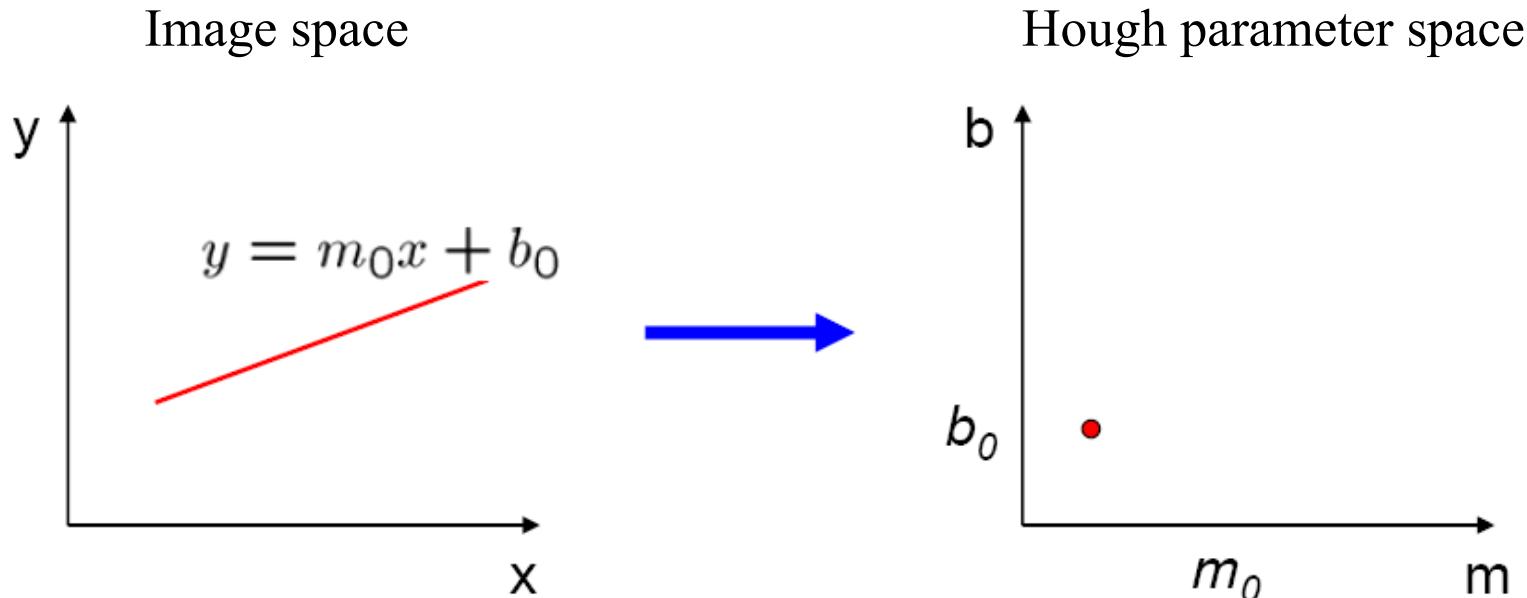


Hough parameter space

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures*, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

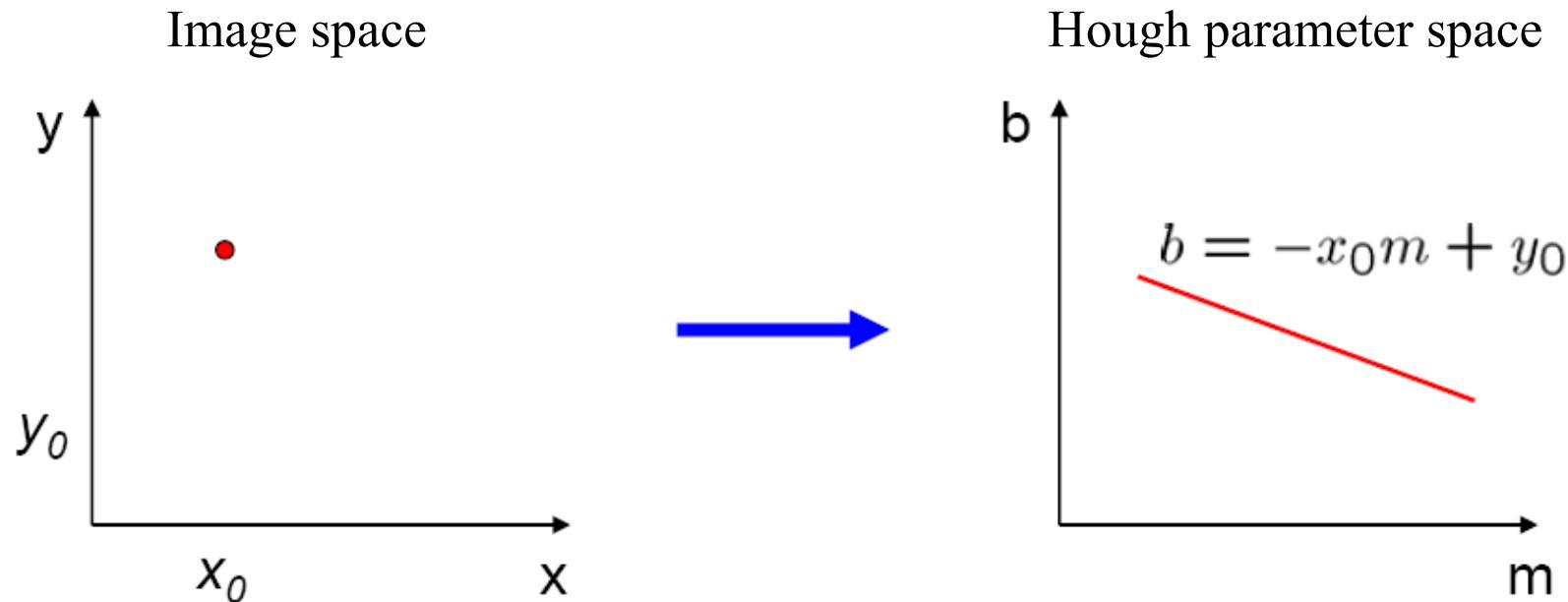
Parameter space representation

- A line in the image corresponds to a point in Hough space



Parameter space representation

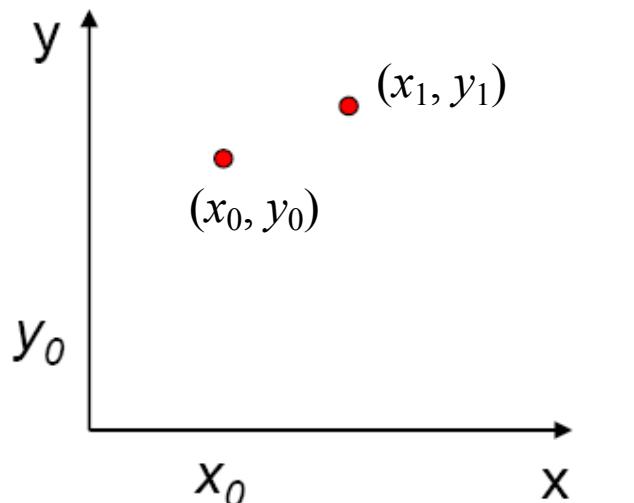
- What does a point (x_0, y_0) in the image space map to in the Hough space?
 - Answer: the solutions of $b = -x_0m + y_0$
 - This is a line in Hough space



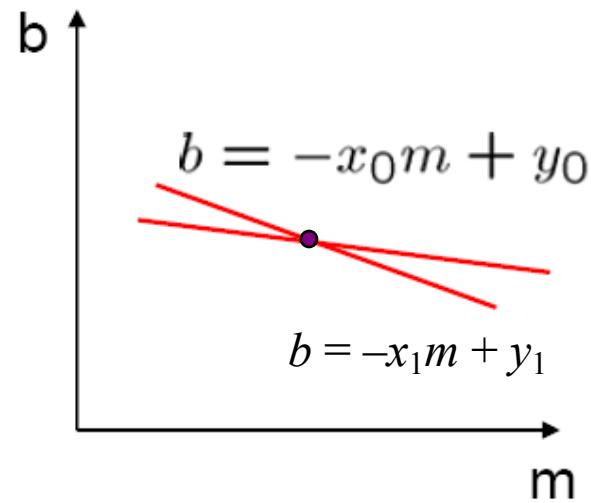
Parameter space representation

- Where is the line that contains both (x_0, y_0) and (x_1, y_1) ?
 - It is the intersection of the lines $b = -x_0m + y_0$ and $b = -x_1m + y_1$

Image space

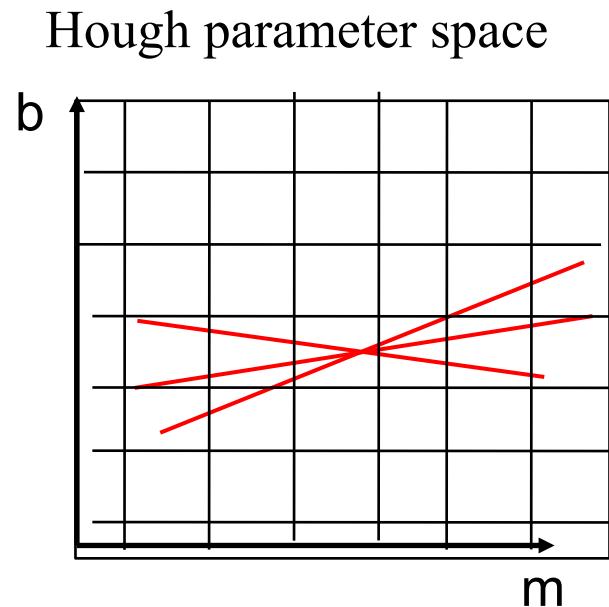
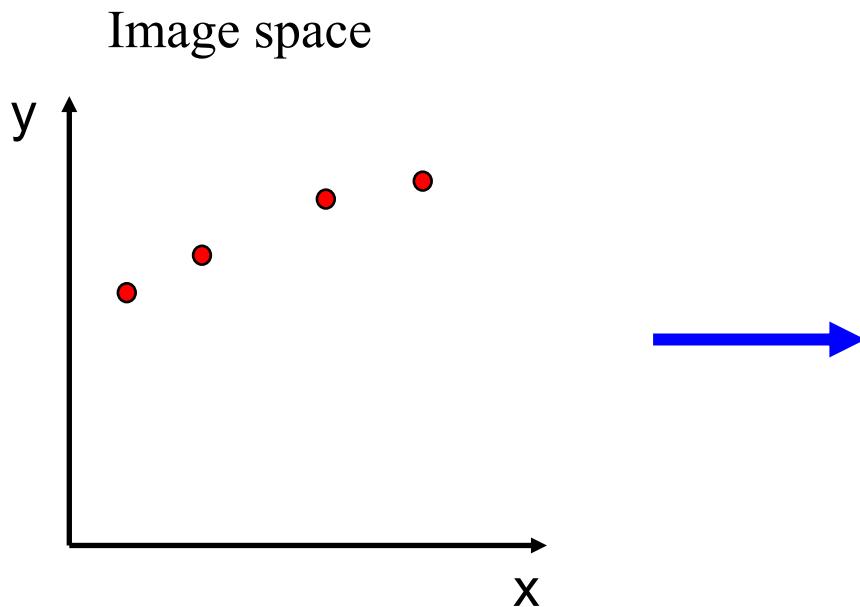


Hough parameter space



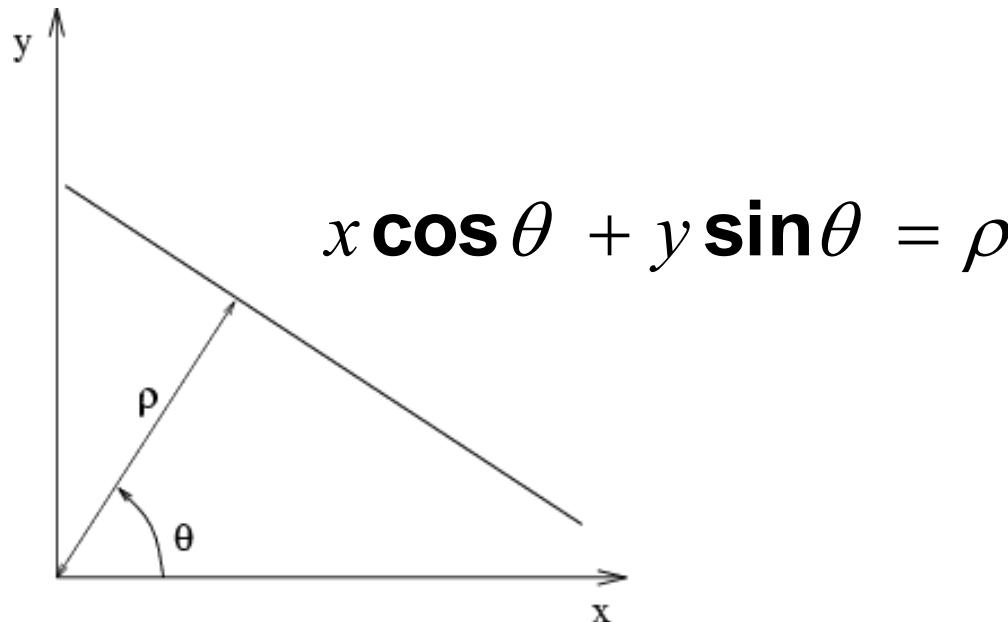
Parameter space representation

- Where is the line that contains both (x_0, y_0) and (x_1, y_1) ?
 - It is the intersection of the lines $b = -x_0m + y_0$ and $b = -x_1m + y_1$



Parameter space representation

- Problems with the (m,b) space:
 - Unbounded parameter domains
 - Vertical lines require infinite m
- Alternative: *polar representation*

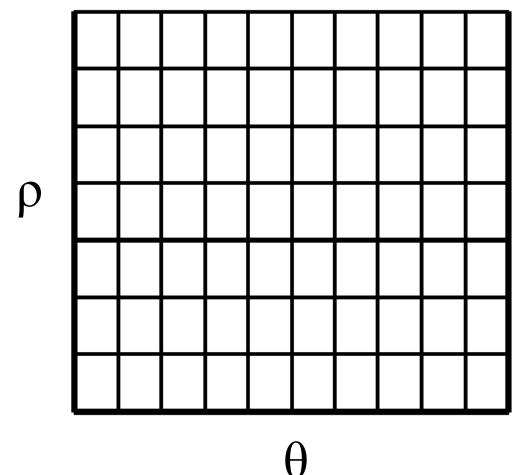


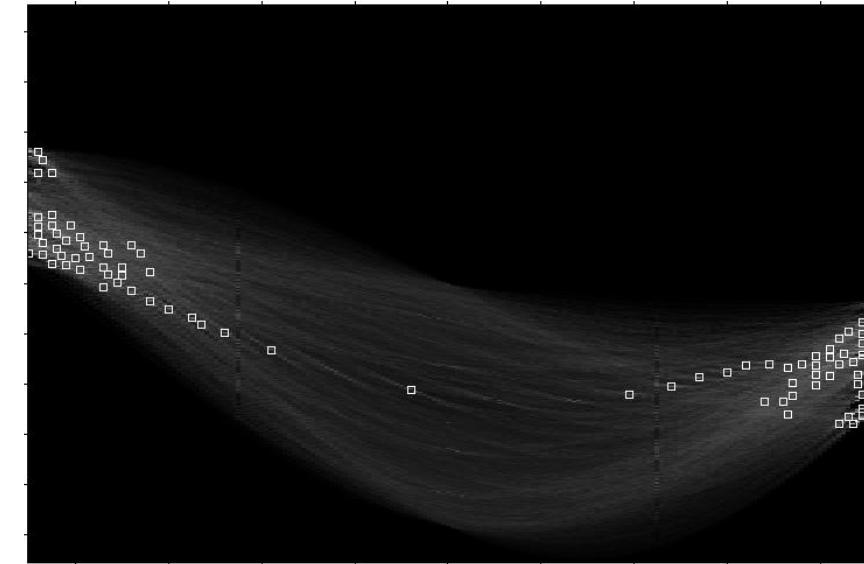
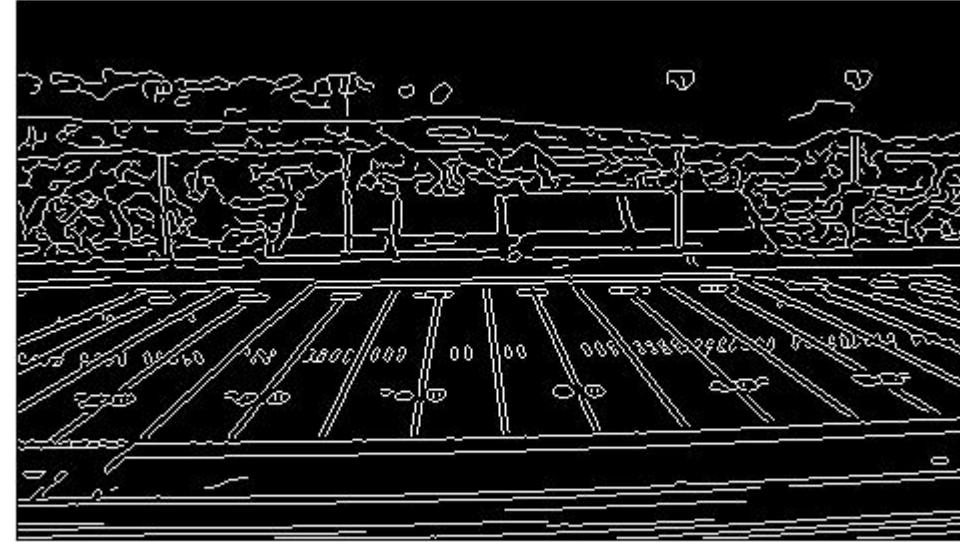
Each point (x,y) will add a sinusoid in the (θ,ρ) parameter space

Algorithm outline

- Initialize accumulator H to all zeros
- For each feature point (x, y) in the image
 - For $\theta = 0$ to 180
 - $\rho = x \cos \theta + y \sin \theta$
 - $H(\theta, \rho) = H(\theta, \rho) + 1$
 - end
 - end
- Find the value(s) of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
 - The detected line in the image is given by
$$\rho = x \cos \theta + y \sin \theta$$

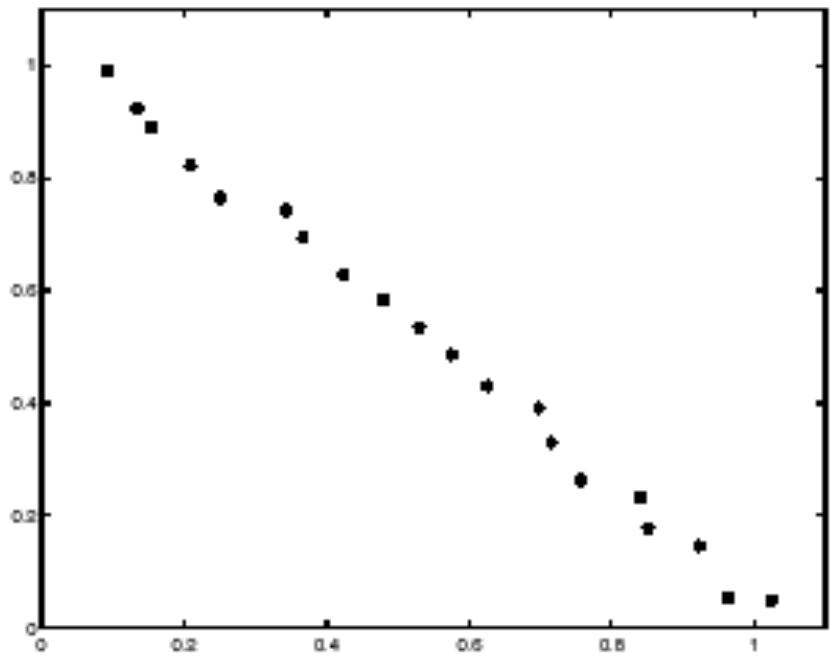
H : accumulator array (votes)



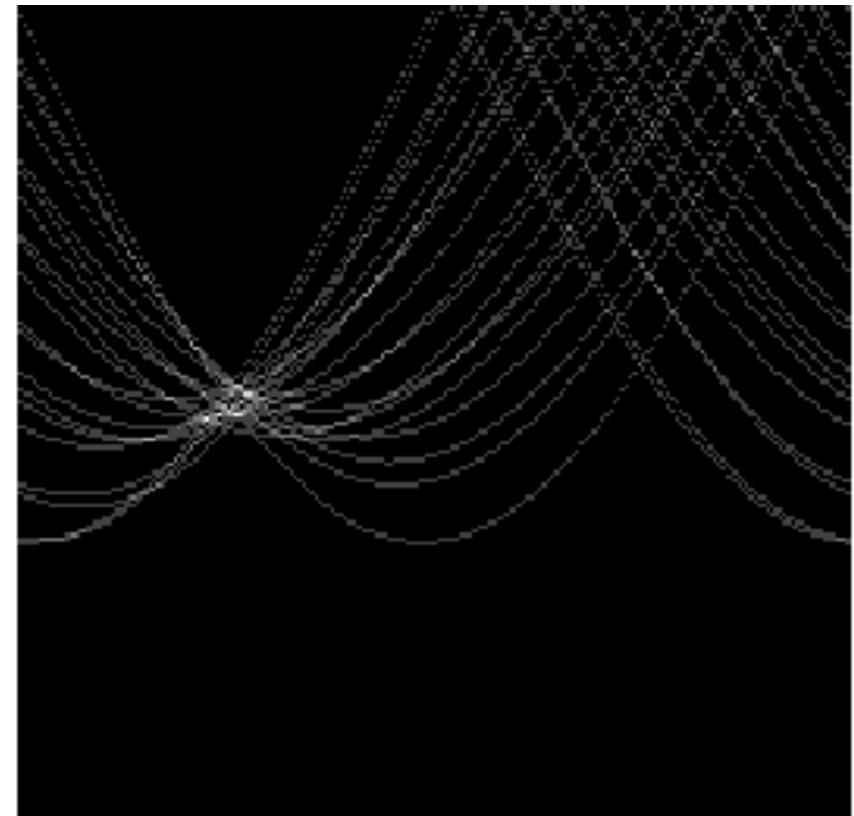


Showing longest segments found

Effect of noise



features

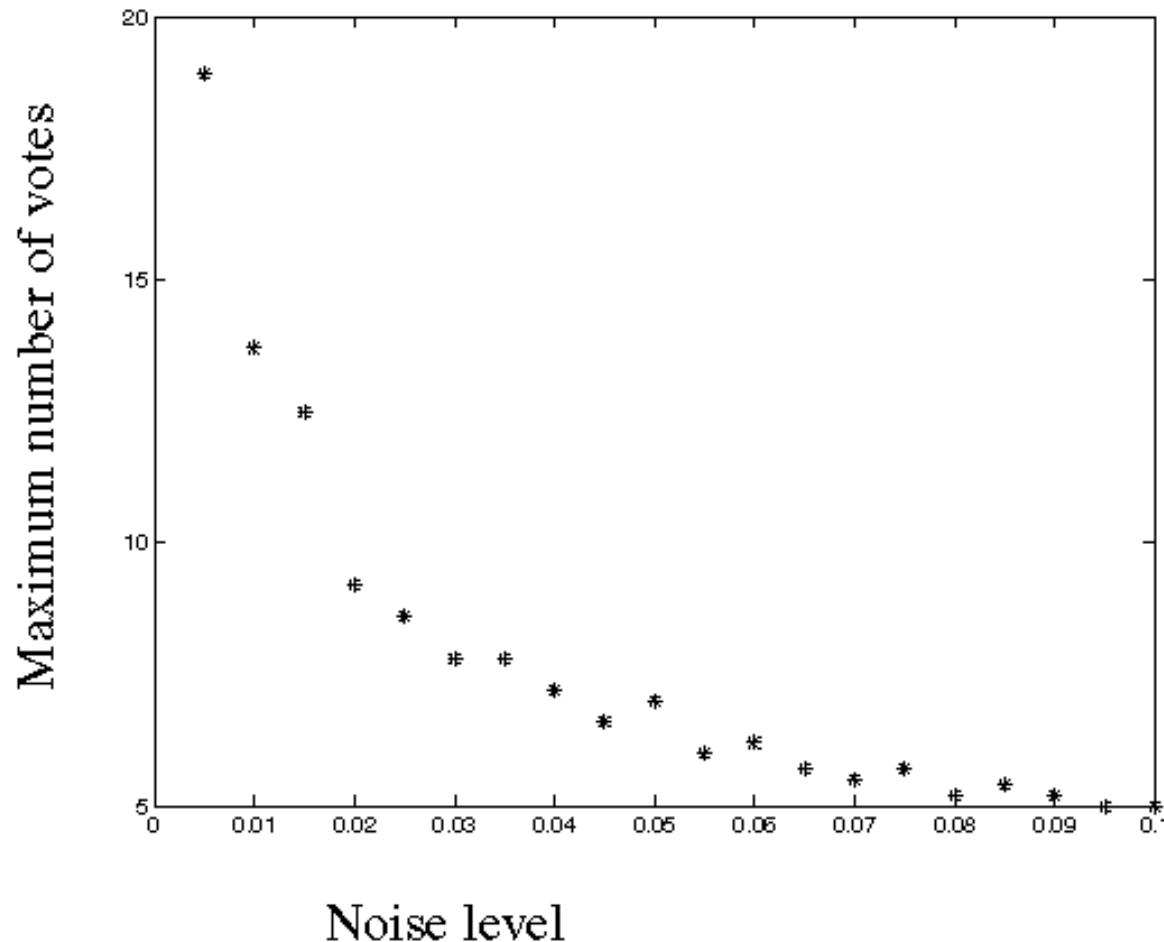


votes

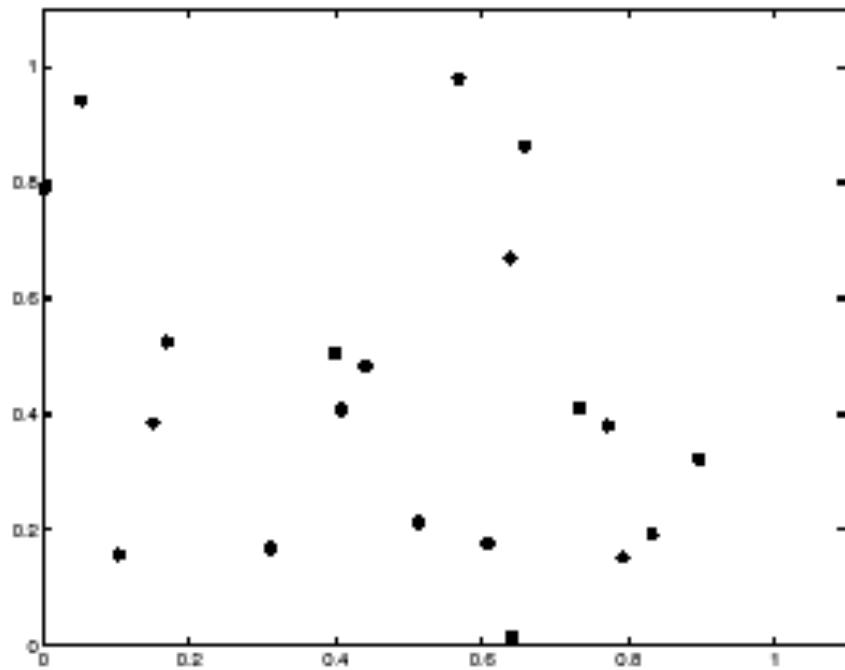
- Peak gets fuzzy and hard to locate

Effect of noise

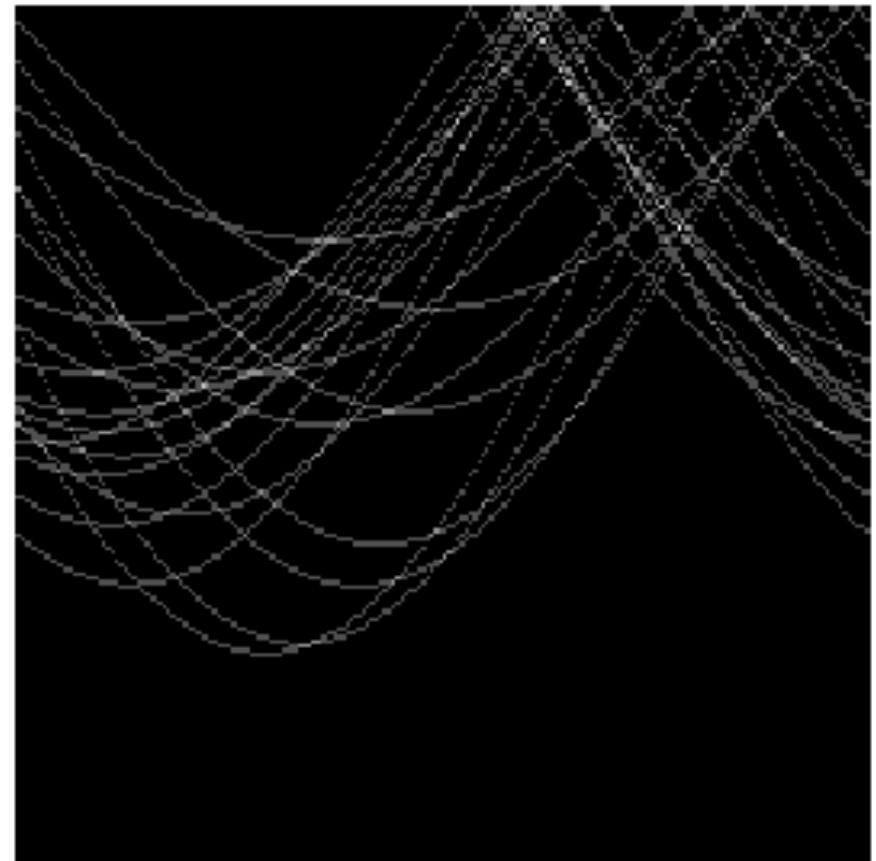
- Number of votes for a line of 20 points with increasing noise:



Random points



features

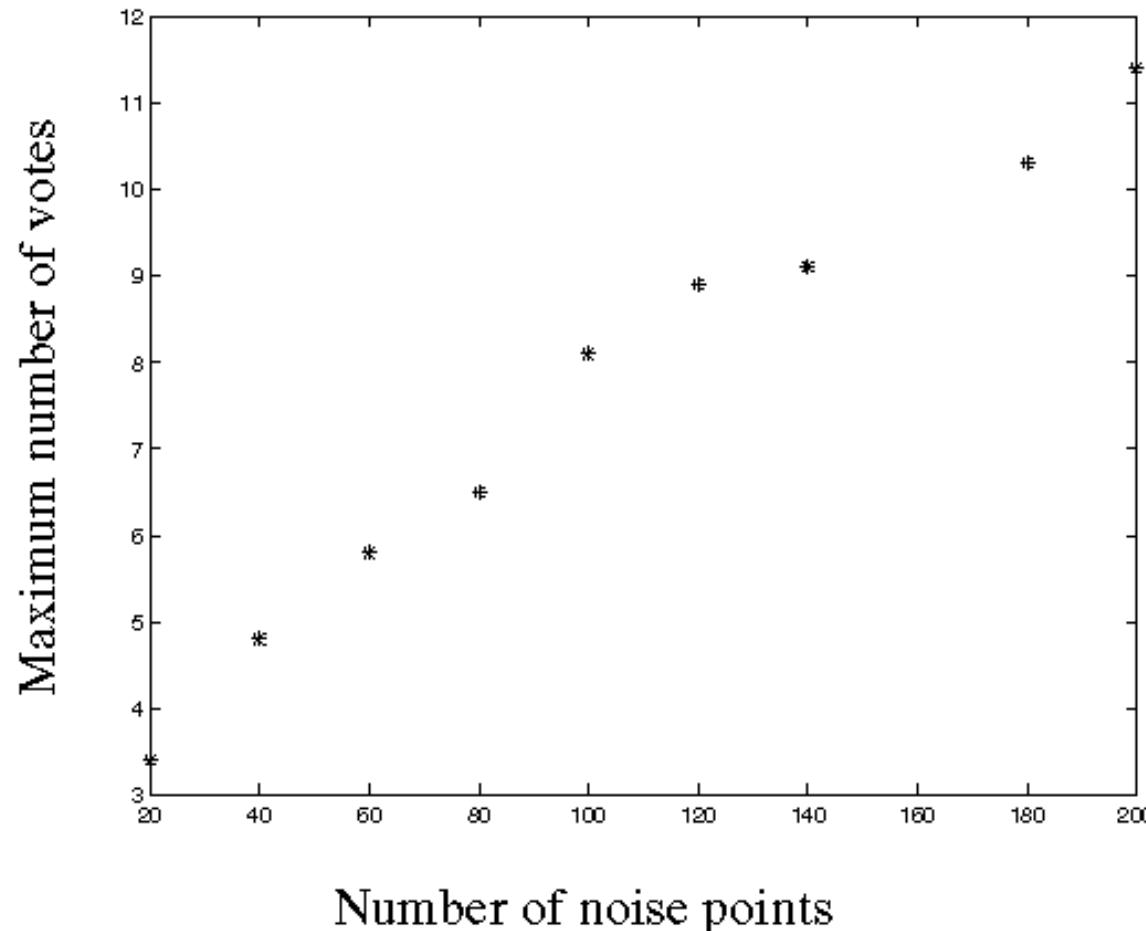


votes

- Uniform noise can lead to spurious peaks in the array

Random points

- As the level of uniform noise increases, the maximum number of votes increases too:

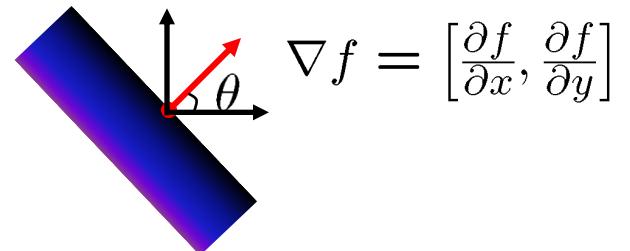


Dealing with noise

- Choose a good grid / discretization
 - Too coarse: large vote counts obtained when too many different lines correspond to a single bucket
 - Too fine: miss lines because some points that are not exactly collinear cast votes for different buckets
- Increment neighboring bins (smoothing in accumulator array)
- Try to get rid of irrelevant features
 - E.g., take only edge points with significant gradient magnitude

Incorporating image gradients

- Recall: when we detect an edge point, we also know its gradient direction
- But this means that the line is uniquely determined!



- Modified Hough transform:

- For each edge point (x, y)
 $\theta = \text{gradient orientation at } (x, y)$
 $\rho = x \cos \theta + y \sin \theta$
 $H(\theta, \rho) = H(\theta, \rho) + 1$

end

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

Hough transform for circles

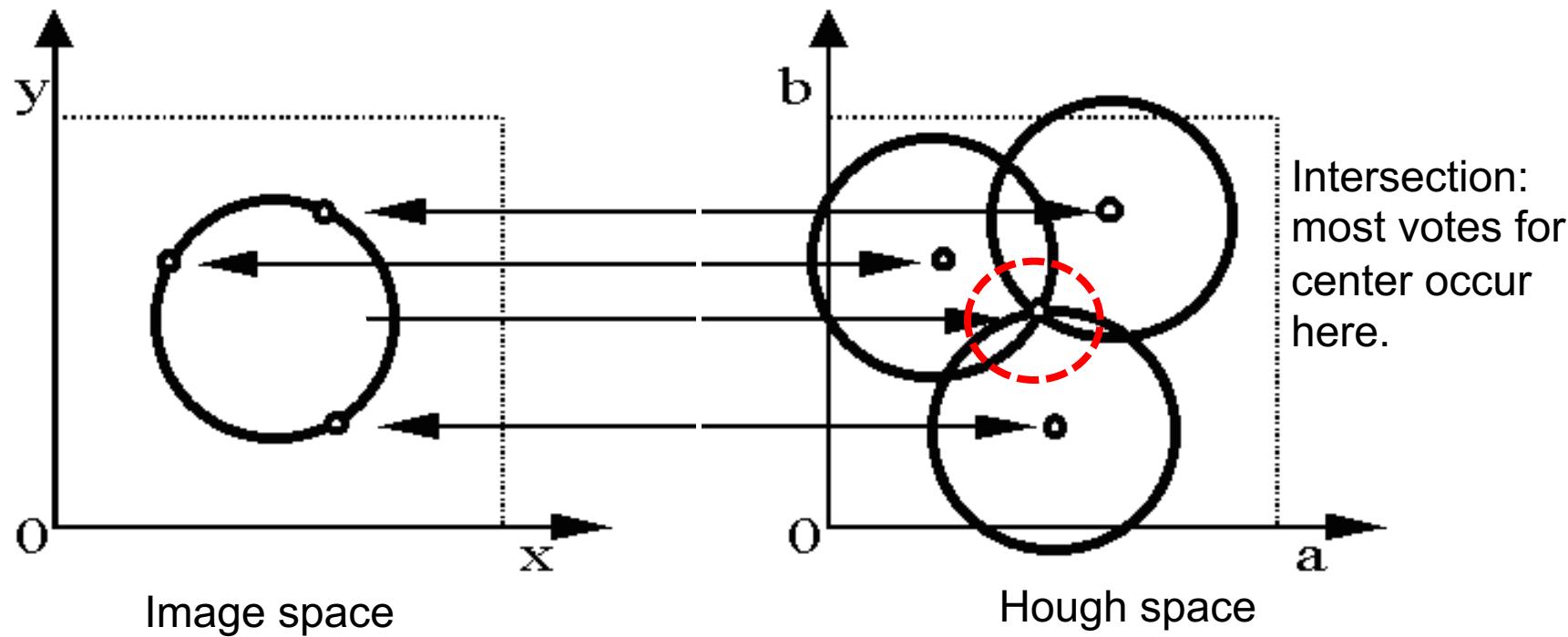
- How many dimensions will the parameter space have?
- Given an unoriented edge point, what are all possible bins that it can vote for?
- What about an *oriented* edge point?

Hough transform for circles

- Circle: center (a, b) and radius r

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

- For a fixed radius r , unknown gradient direction

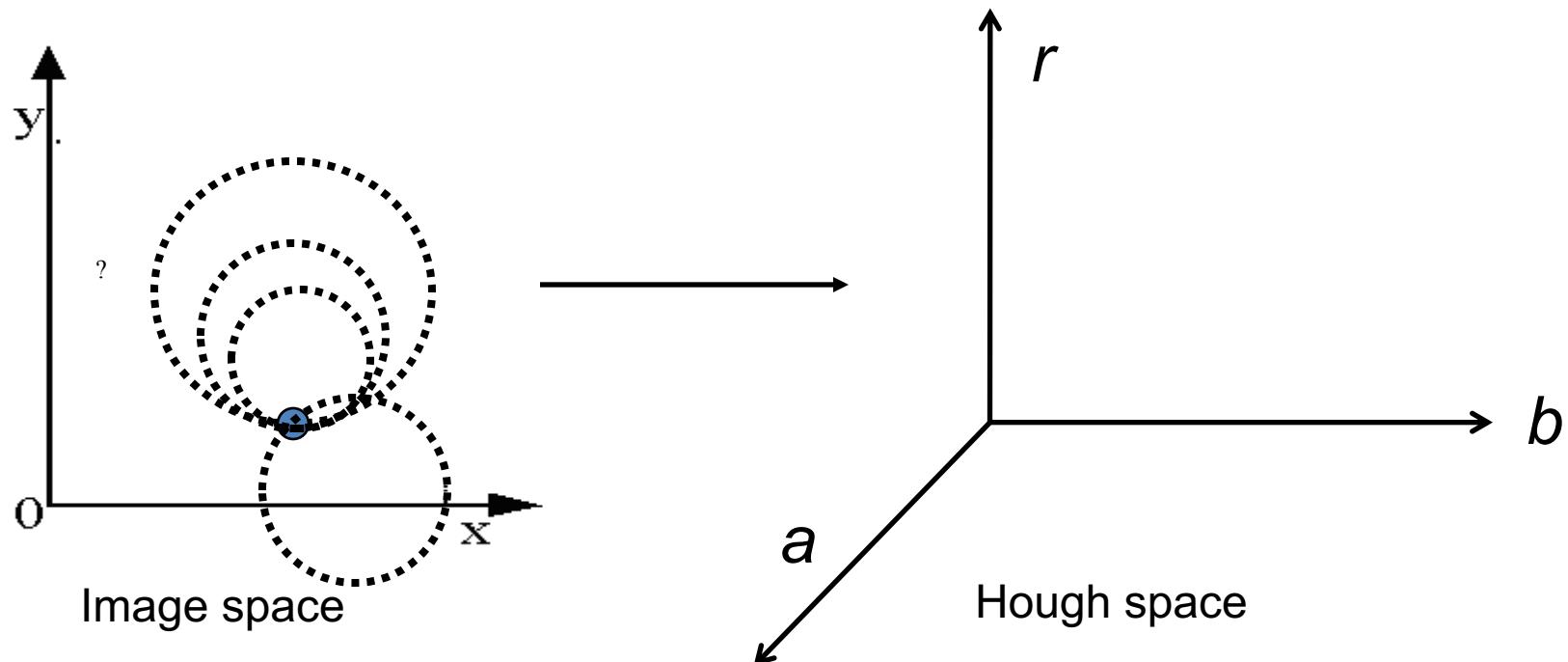


Hough transform for circles

- Circle: center (a, b) and radius r

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

- For an unknown radius r , unknown gradient direction

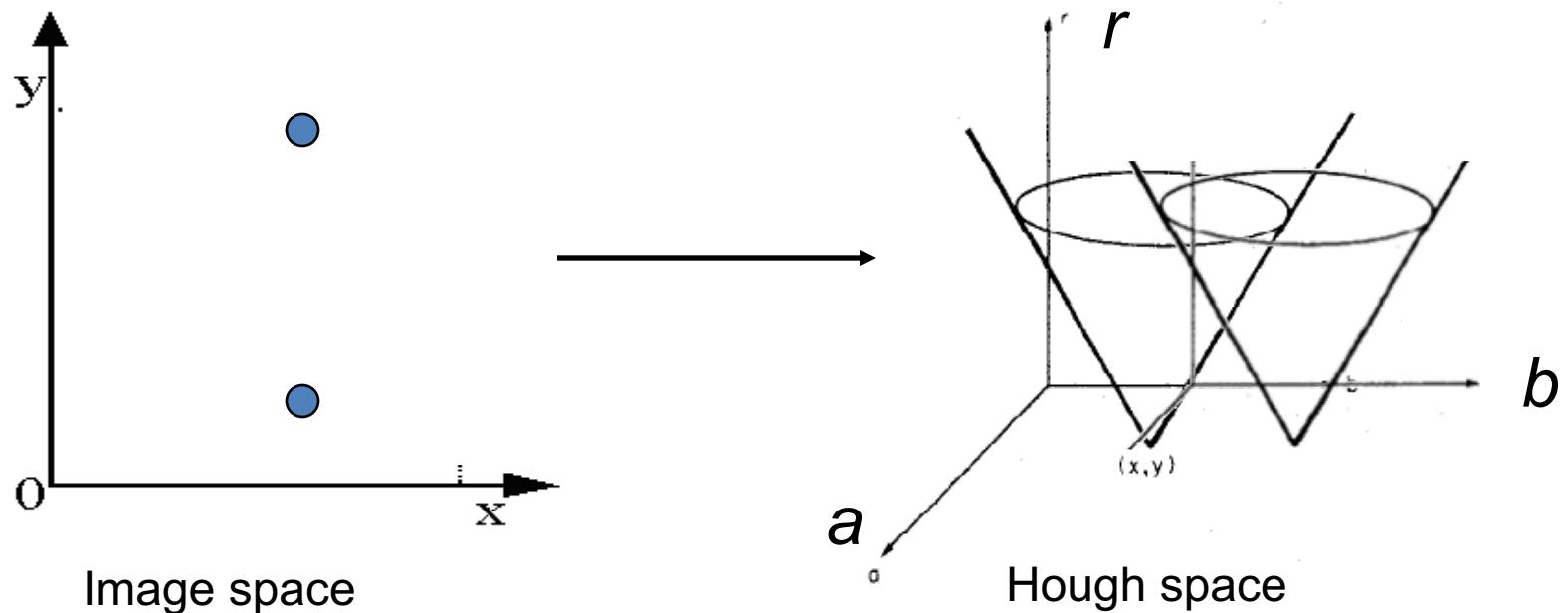


Hough transform for circles

- Circle: center (a, b) and radius r

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

- For an unknown radius r , unknown gradient direction



Hough transform for circles

- For an unknown radius r , known gradient direction

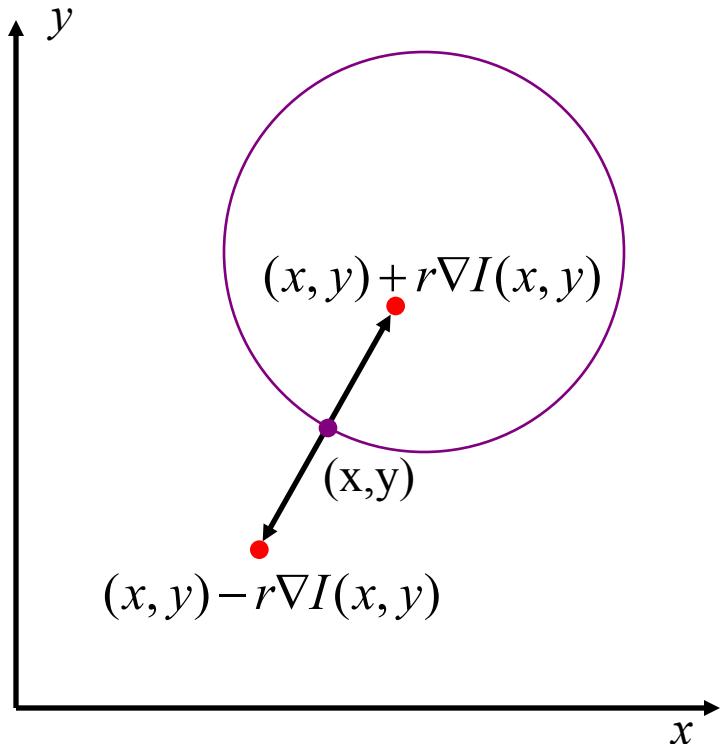
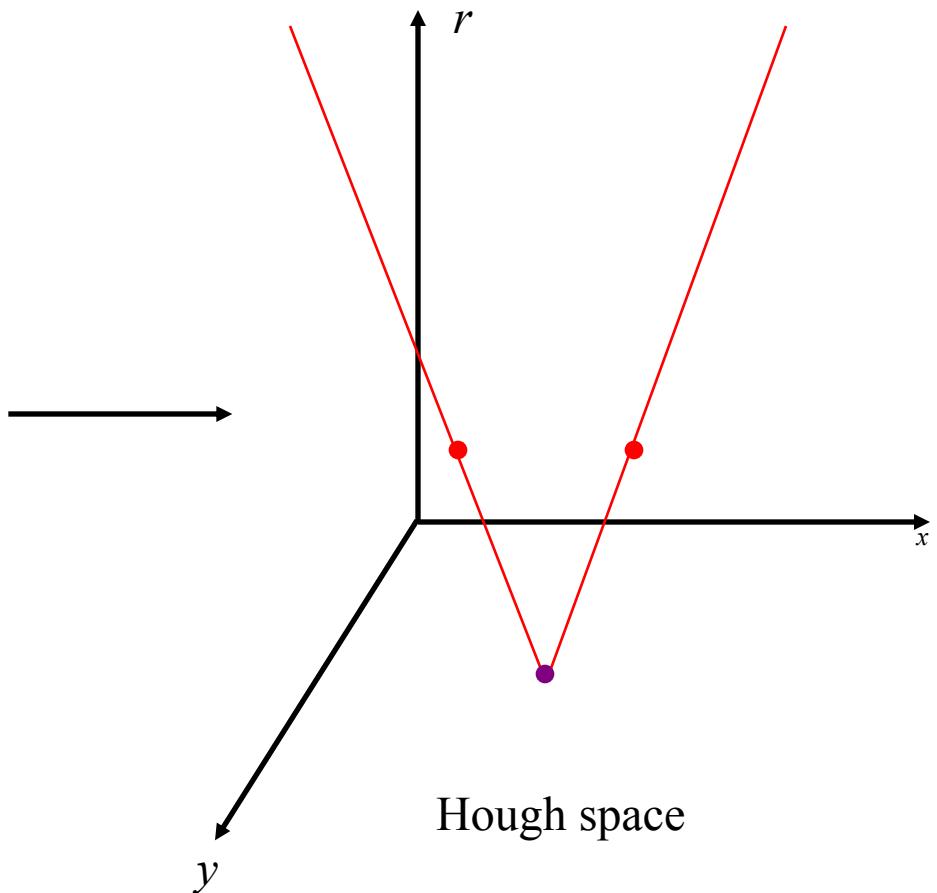


Image space



Hough space

Hough transform for circles

For every edge pixel (x,y) :

 For each possible radius value r :

 For each possible gradient direction θ :

// or use estimated gradient at (x,y)

$$a = x - r \cos(\theta) \text{ // column}$$

$$b = y + r \sin(\theta) \text{ // row}$$

$$H[a,b,r] += 1$$

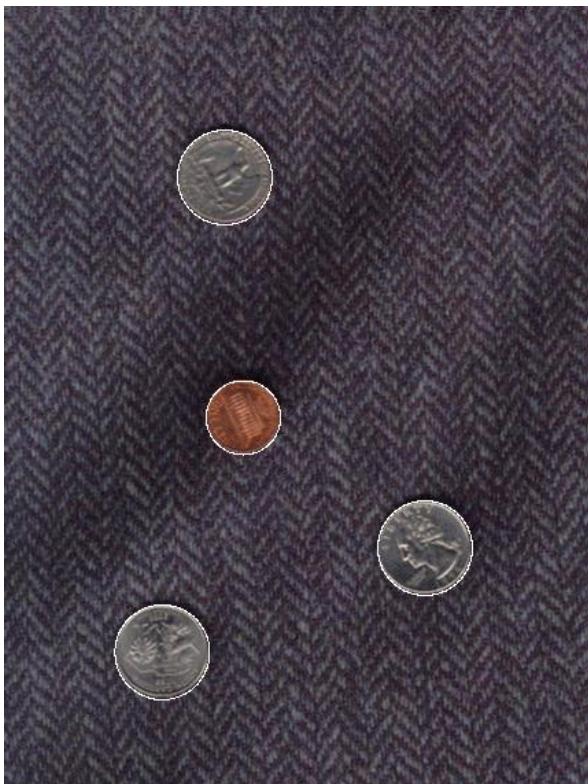
end

end

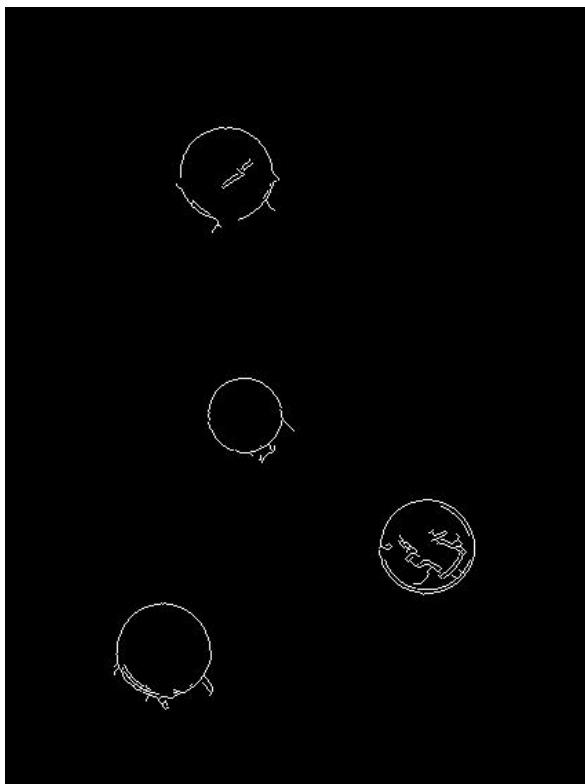
Time complexity per edge?

Example: detecting circles with Hough

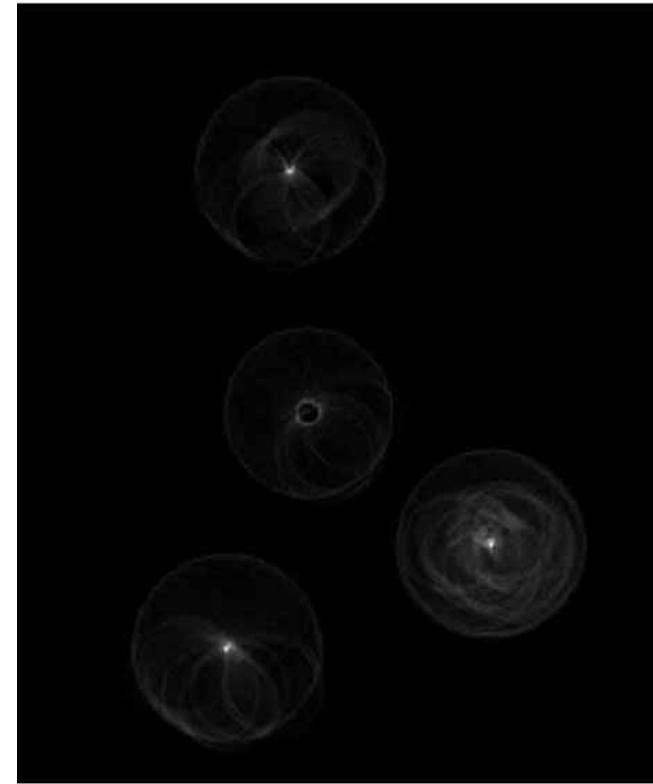
Combined detections



Edges



Votes: Quarter



Example: iris detection



Gradient+threshold

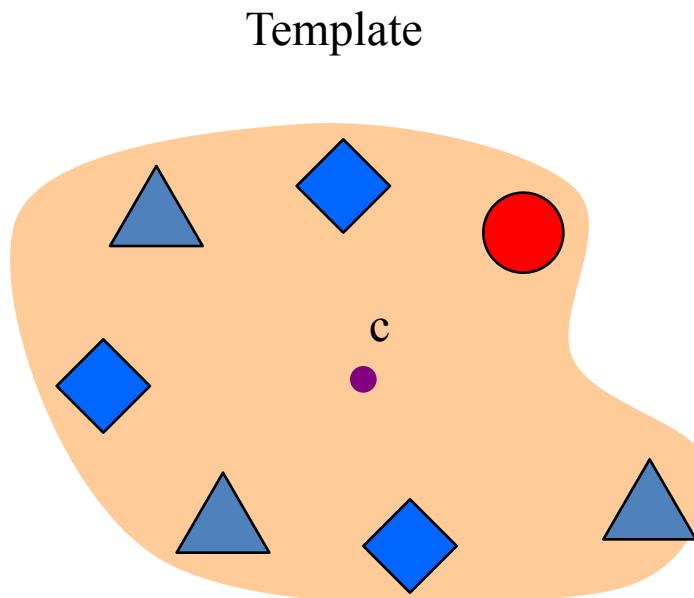
Hough space (fixed radius)

Max detections

- Hemerson Pistori and Eduardo Rocha Costa
<http://rsbweb.nih.gov/ij/plugins/hough-circles.html>

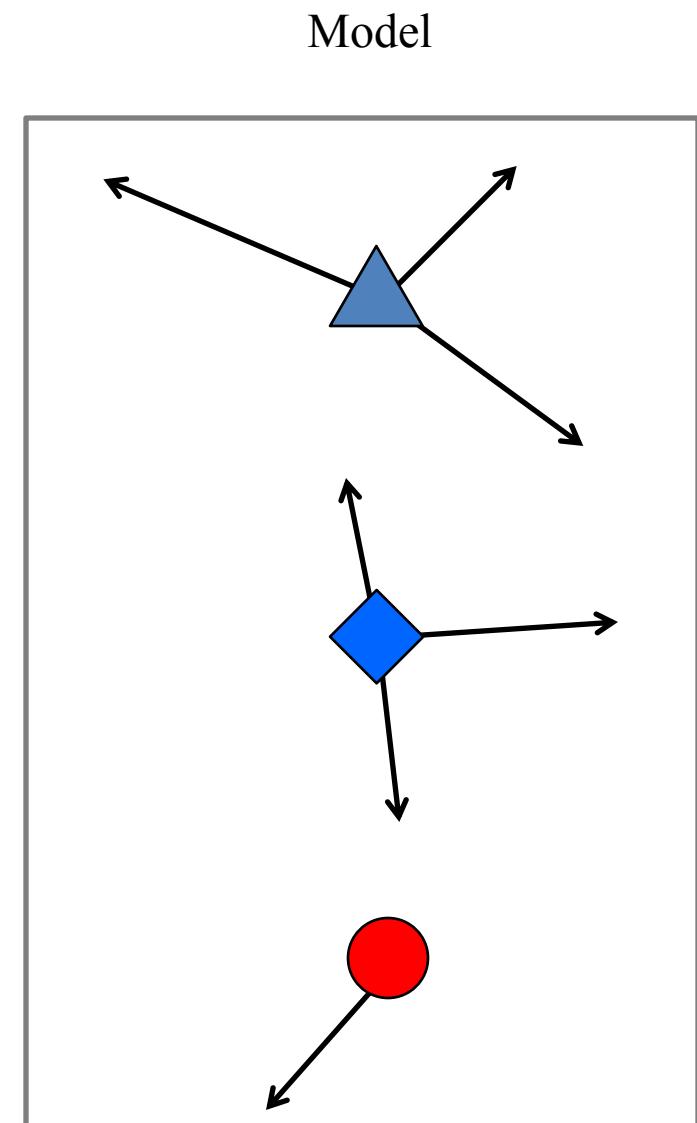
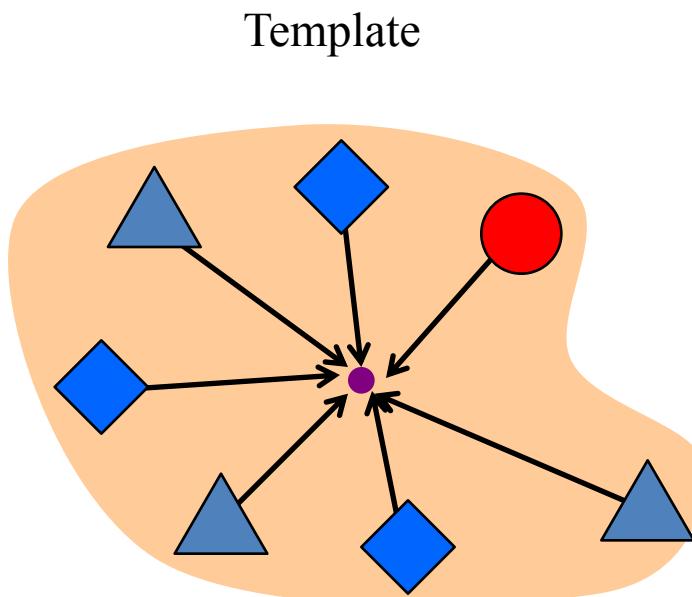
Generalized Hough transform

- We want to find a template defined by its reference point (center) and several distinct types of landmark points in stable spatial configuration



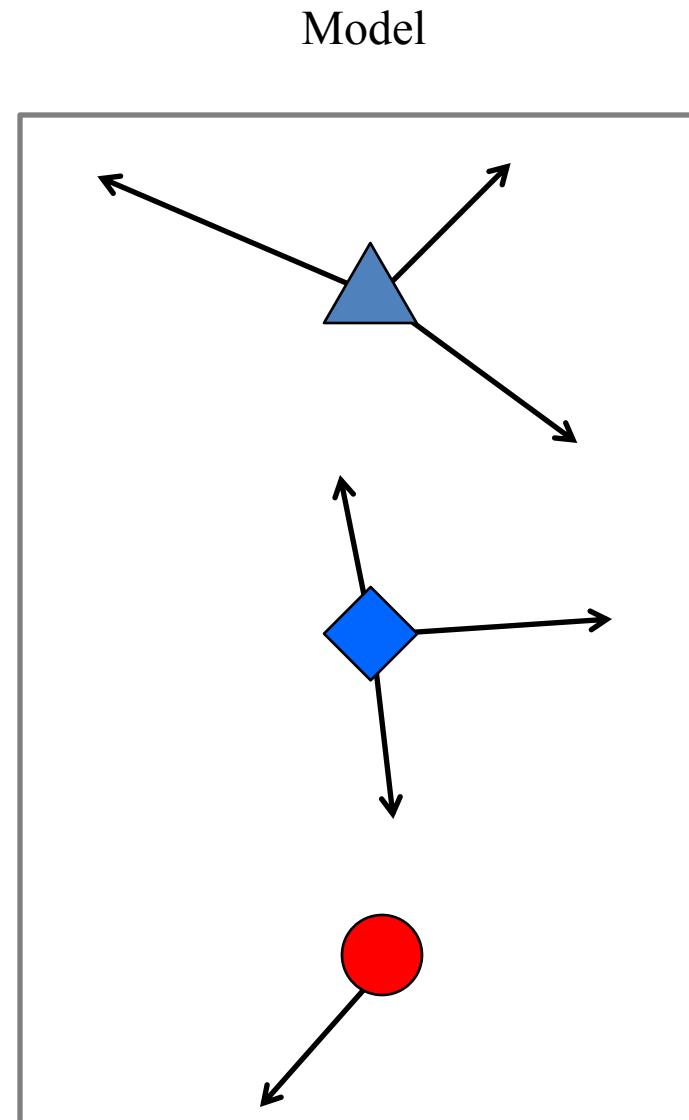
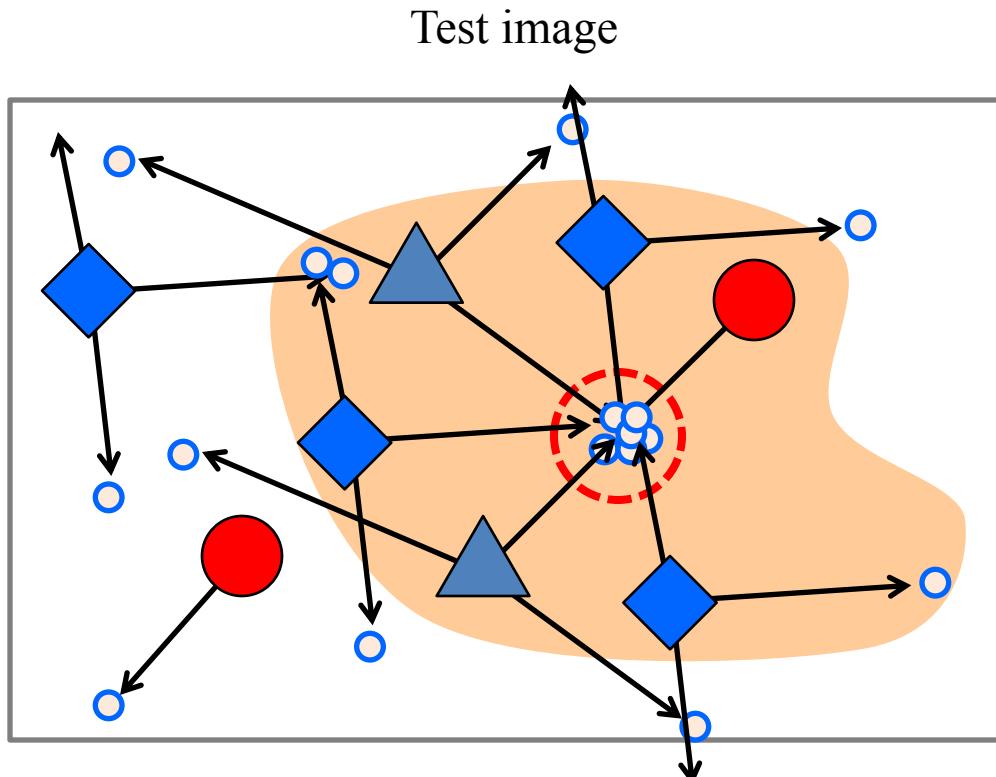
Generalized Hough transform

- Template representation: for each type of landmark point, store all possible displacement vectors towards the center



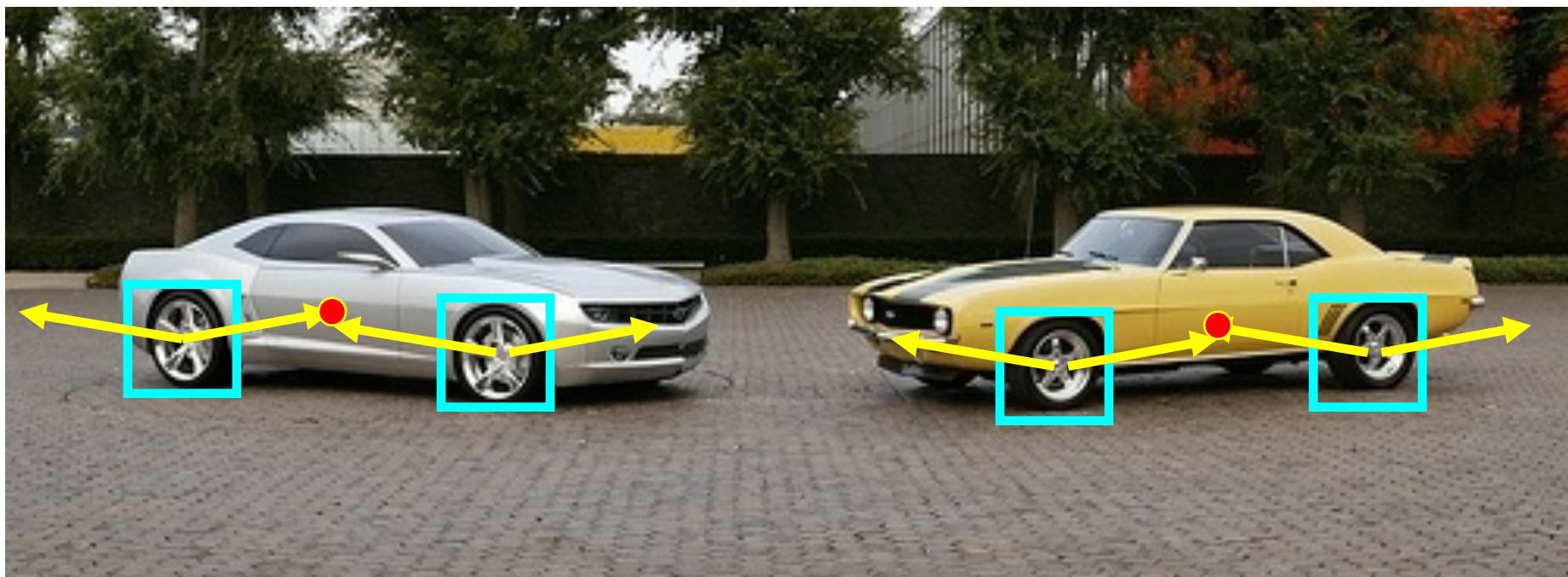
Generalized Hough transform

- Detecting the template:
 - For each feature in a new image, look up that feature type in the model and vote for the possible center locations associated with that type in the model



Application in recognition

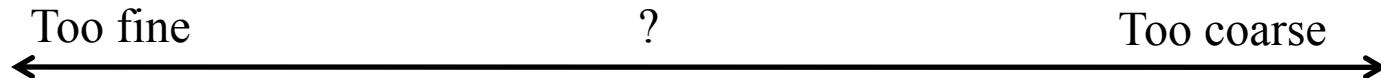
- Index displacements by “visual codeword”



test image

Voting: practical tips

- Minimize irrelevant tokens first
- Choose a good grid / discretization



- Vote for neighbors, also (smoothing in accumulator array)
- Use direction of edge to reduce parameters by 1
- To read back which points voted for “winning” peaks, keep tags on the votes.

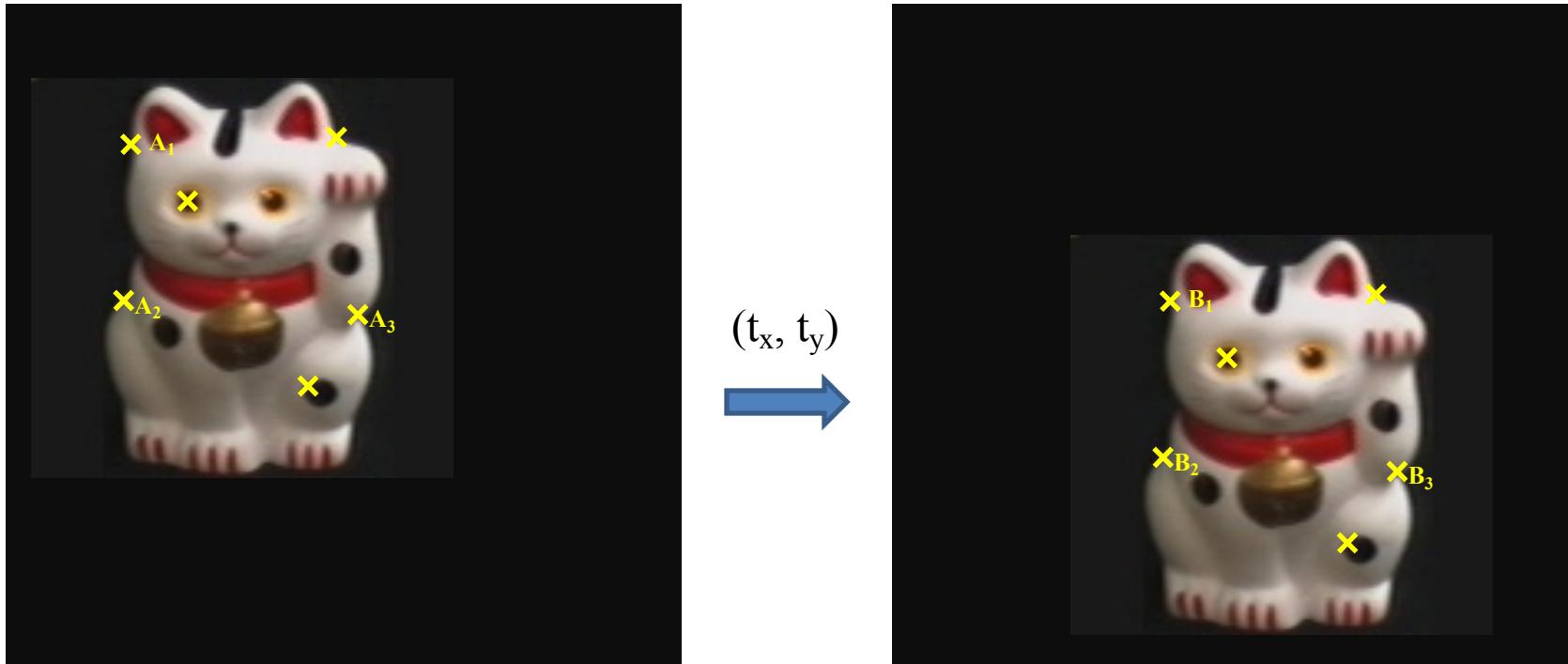
Hough transform: Discussion

- Pros
 - All points processed independently
 - Can deal with occlusion and gaps
 - Can detect multiple instances of a model
 - Some robustness to noise: noise points unlikely to contribute consistently to any single bin
- Cons
 - Complexity of search time increases exponentially with the number of model parameters
 - Non-target shapes can produce spurious peaks in parameter space
 - It's hard to pick a good grid size

Fitting Algorithm Summary

- Least Squares Fit
 - closed form solution
 - robust to noise
 - not robust to outliers
- Robust Least Squares
 - improves robustness to noise
 - requires iterative optimization
- Hough transform
 - robust to noise and outliers
 - can fit multiple models
 - only works for a few parameters (1-4 typically)
- RANSAC
 - robust to noise and outliers
 - works with a moderate number of parameters (e.g, 1-8)

Example: solving for translation



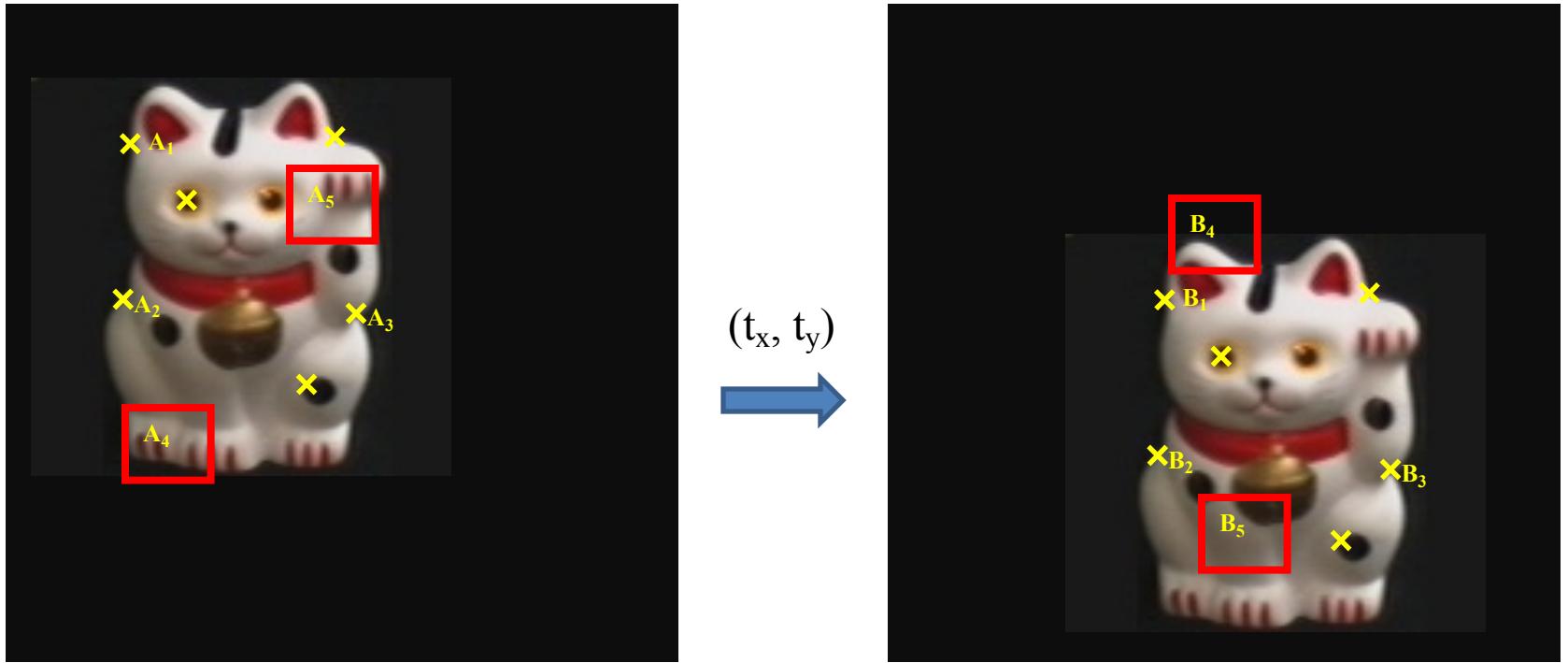
Least squares solution

1. Write down objective function
2. Derived solution
 - a) Compute derivative
 - b) Compute solution
3. Computational solution
 - a) Write in form $Ax=b$
 - b) Solve using pseudo-inverse or eigenvalue decomposition

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_1^B - x_1^A \\ y_1^B - y_1^A \\ \vdots \\ x_n^B - x_n^A \\ y_n^B - y_n^A \end{bmatrix}$$

Example: solving for translation



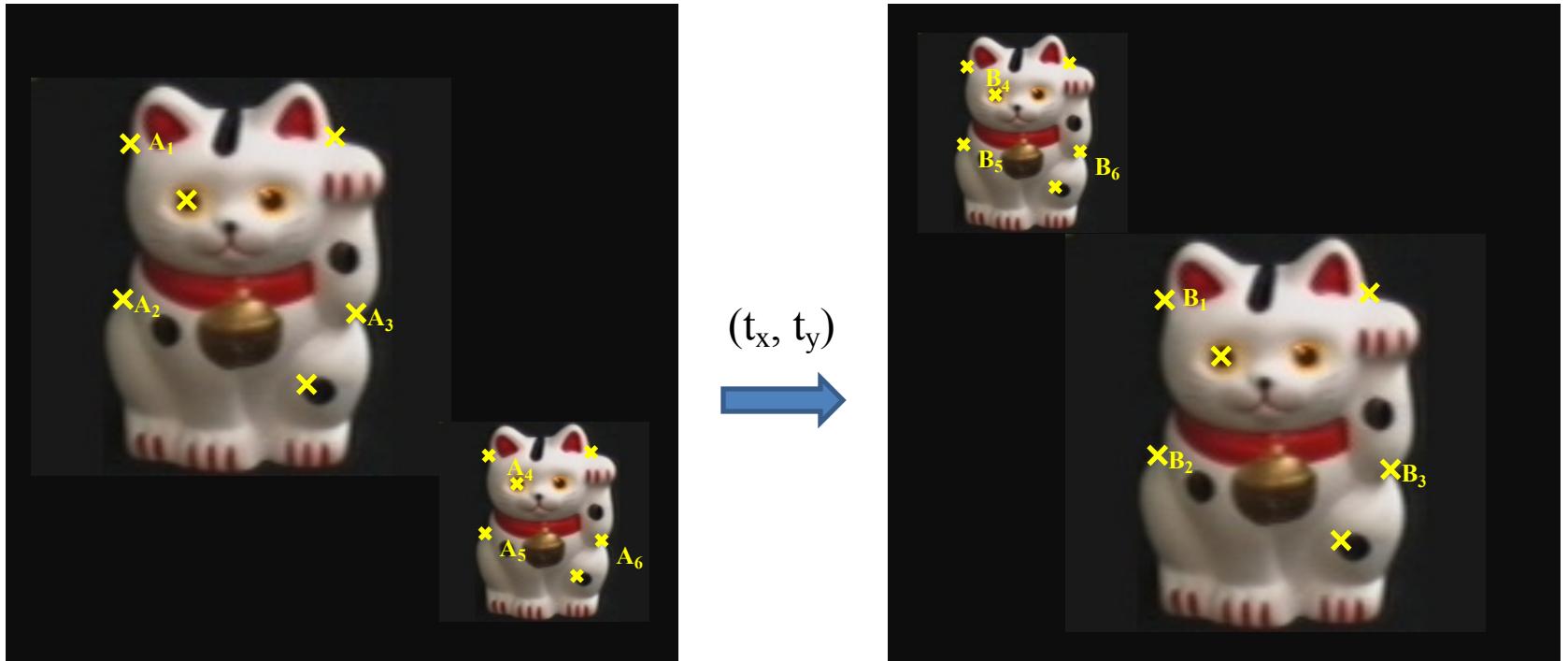
Problem: outliers

RANSAC solution

1. Sample a set of matching points (1 pair)
2. Solve for transformation parameters
3. Score parameters with number of inliers
4. Repeat steps 1-3 N times

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Example: solving for translation



Problem: outliers, multiple objects, and/or many-to-one matches

Hough transform solution

1. Initialize a grid of parameter values
2. Each matched pair casts a vote for consistent values
3. Find the parameters with the most votes
4. Solve using least squares with inliers

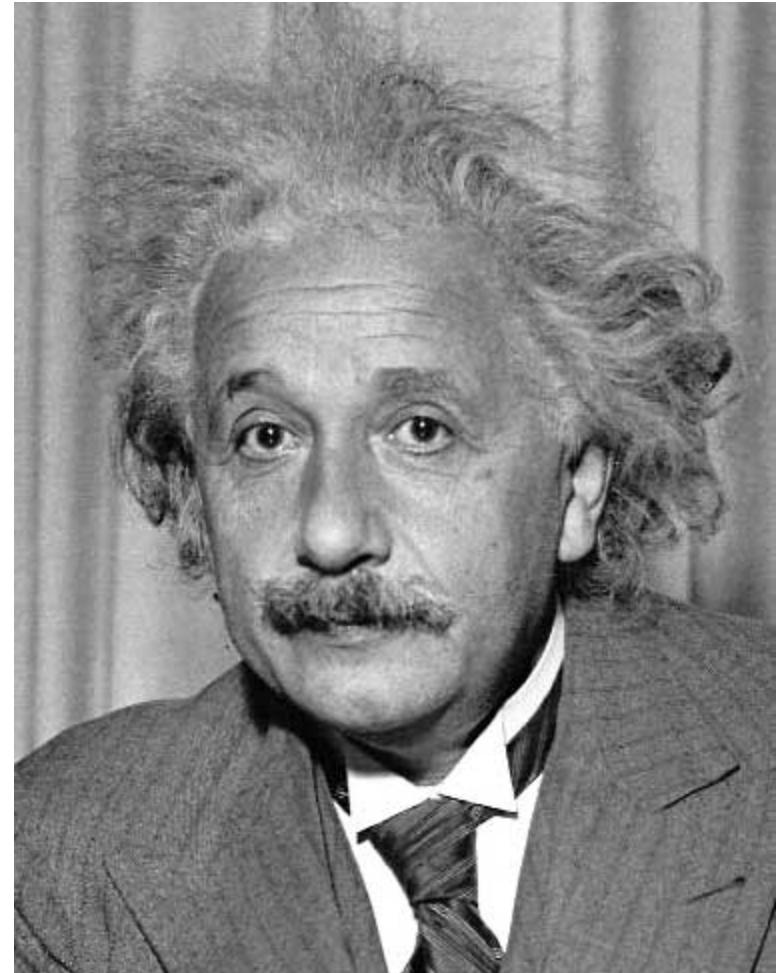
$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Template Matching

Slides based on D. Hoiem's slides

Template matching

- Goal: find  in image
- Main challenge: What is a good similarity or distance measure between two patches?
 - Filtering
 - Zero-mean filtering
 - Sum of Squares Difference
 - Normalized Cross Correlation

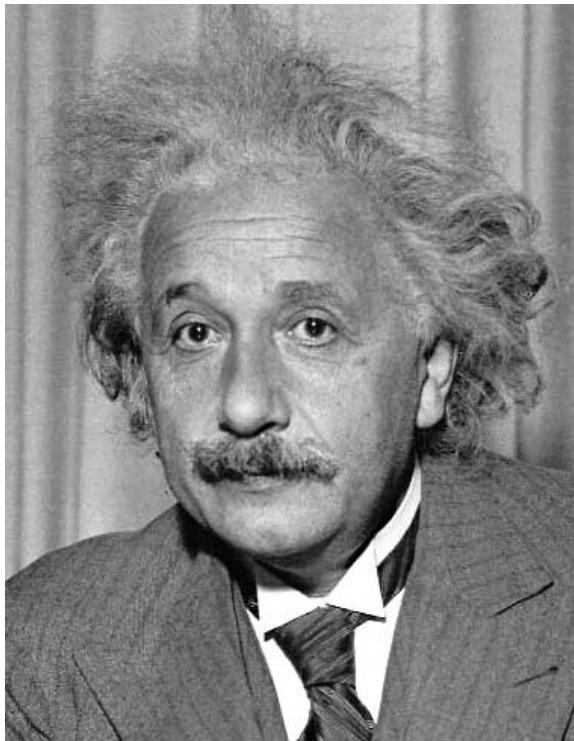


Matching with filters

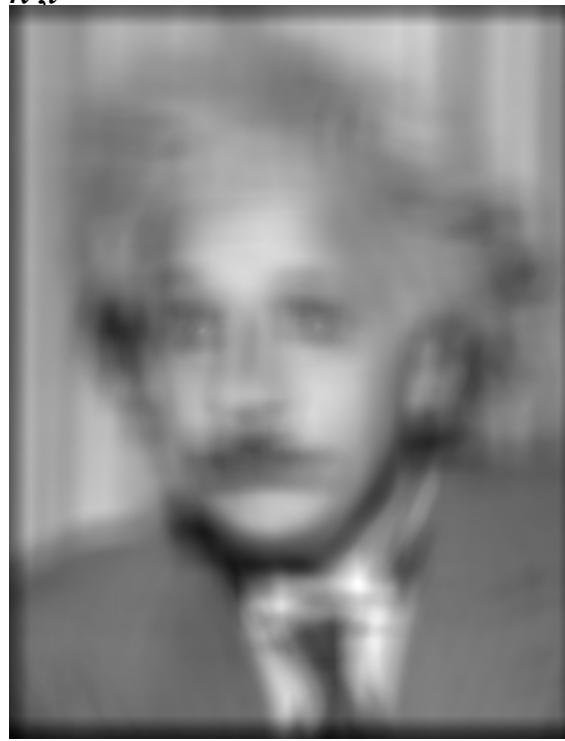
- Goal: find  in image
- Method 0: filter the image with eye patch

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

f = image
g = filter



Input



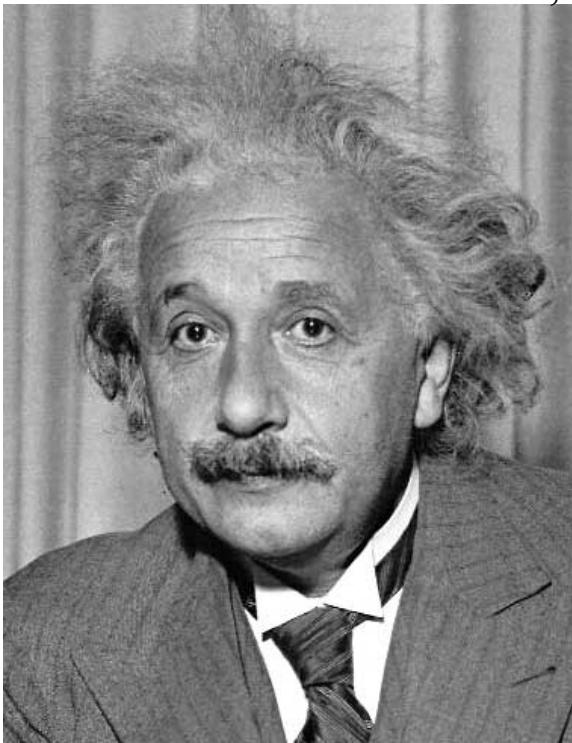
Filtered Image

What went wrong?

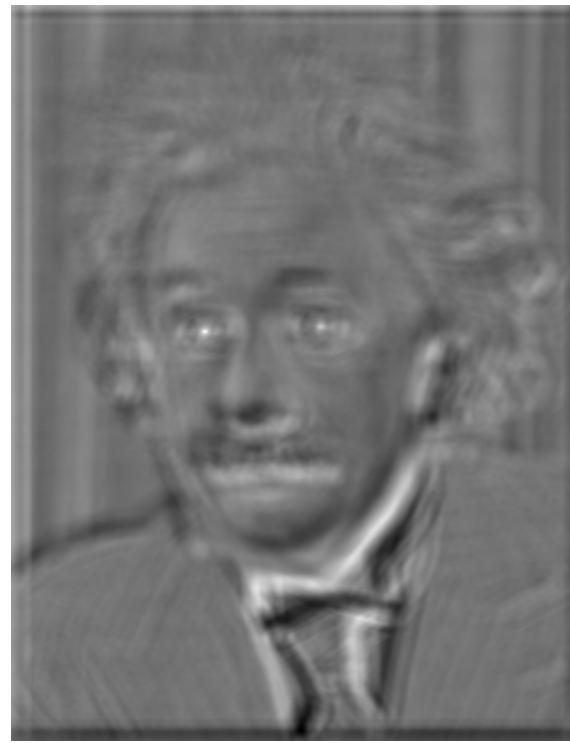
Matching with filters

- Goal: find  in image
- Method 1: filter the image with zero-mean eye

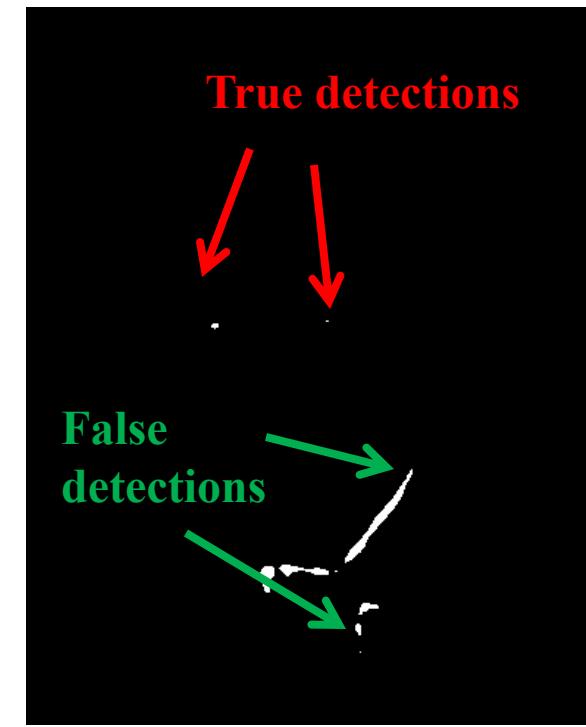
$$h[m, n] = \sum_{k, l} (g[k, l] - \bar{g}) \underbrace{(f[m + k, n + l])}_{\text{mean of template } g}$$



Input



Filtered Image (scaled)

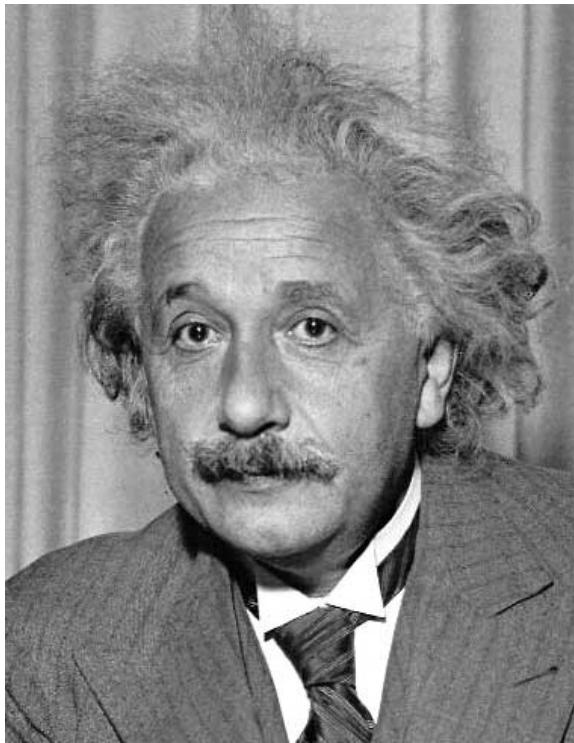


Thresholded Image

SSD

- Goal: find  in image
- Method 2: SSD

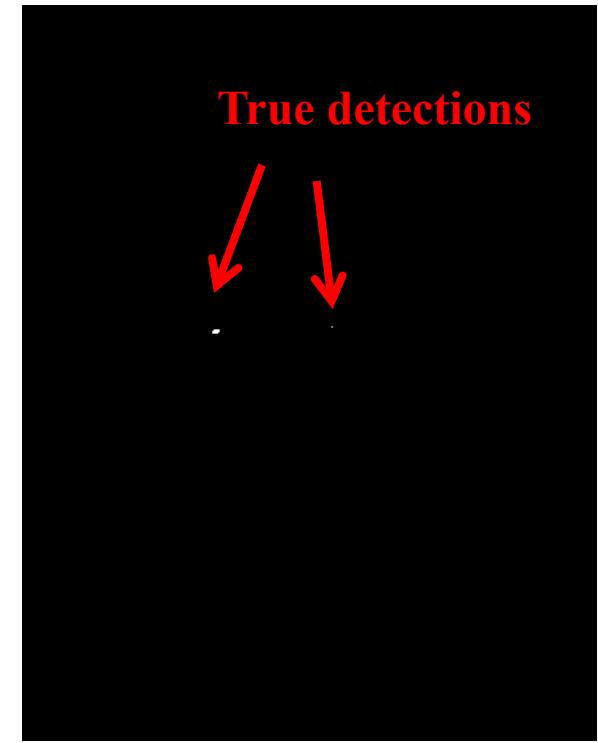
$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$



Input



1 - $\text{sqrt}(\text{SSD})$



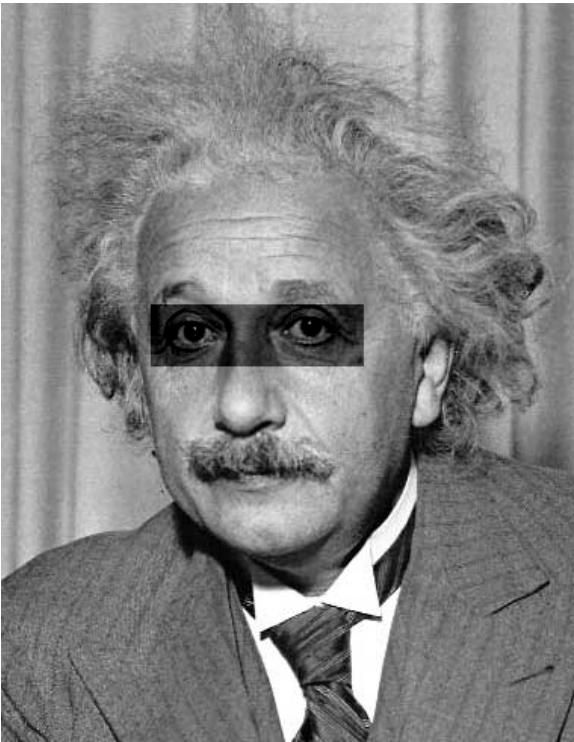
True detections

SSD

- Goal: find  in image
- Method 2: SSD

What's the potential downside of SSD?

$$h[m, n] = \sum_{k, l} (g[k, l] - f[m + k, n + l])^2$$



Input



1 - sqrt(SSD)

NCC

- Goal: find  in image
- Method 3: Normalized cross-correlation

$$h[m, n] = \frac{\sum_{k,l} (g[k, l] - \bar{g})(f[m + k, n + l] - \bar{f}_{m,n})}{\left(\sum_{k,l} (g[k, l] - \bar{g})^2 \sum_{k,l} (f[m + k, n + l] - \bar{f}_{m,n})^2 \right)^{0.5}}$$

mean template mean image patch

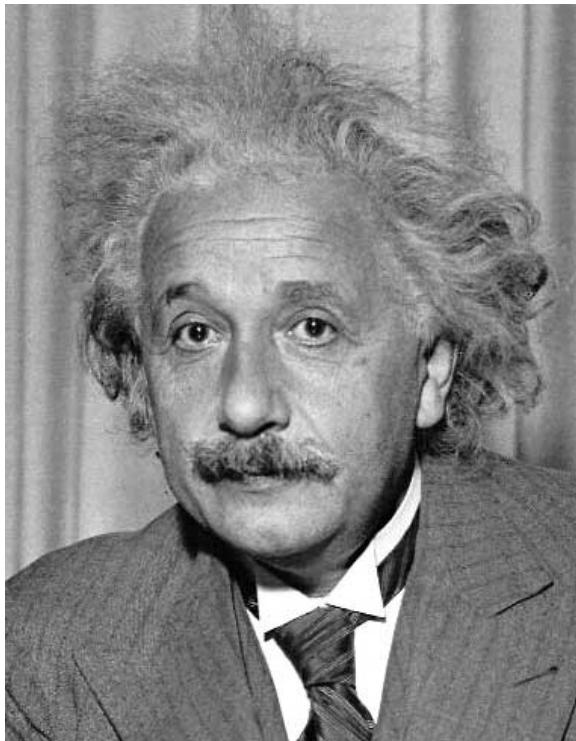
↓ ↓

$$\sum_{k,l} (g[k, l] - \bar{g})(f[m + k, n + l] - \bar{f}_{m,n})$$

Matlab: `normxcorr2(template, im)`

NCC

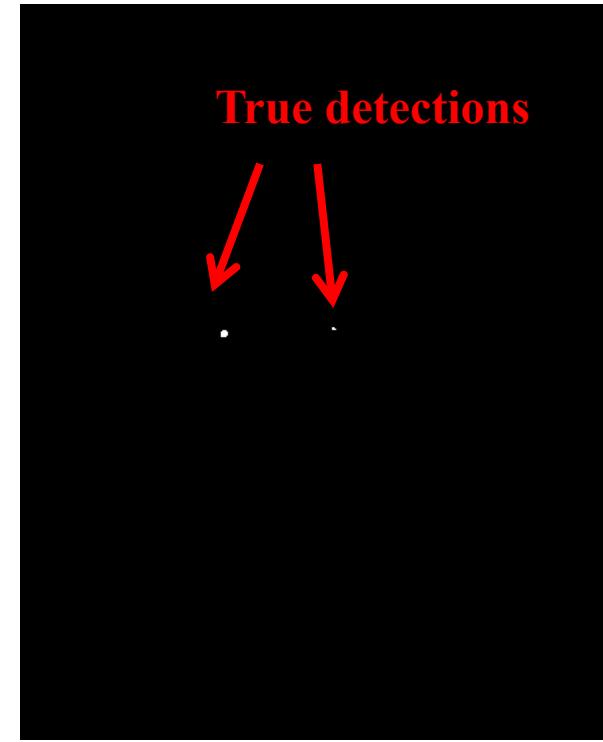
- Goal: find  in image
- Method 3: Normalized cross-correlation



Input



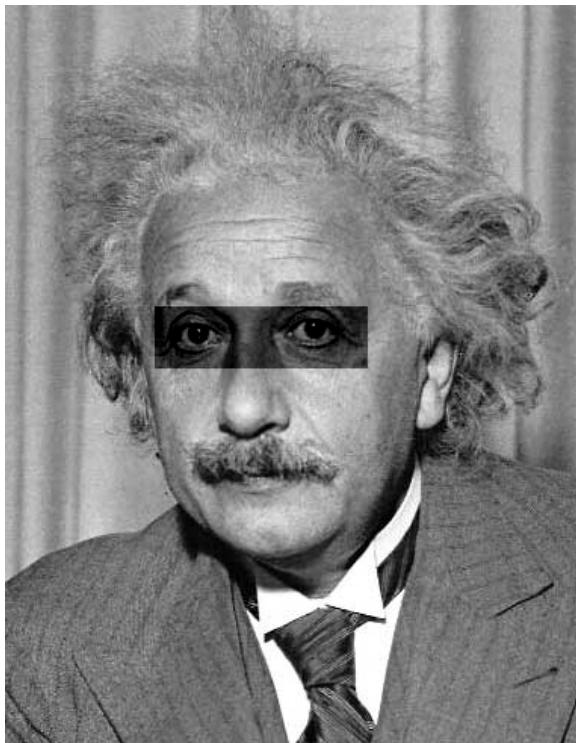
Normalized X-Correlation



True detections

NCC

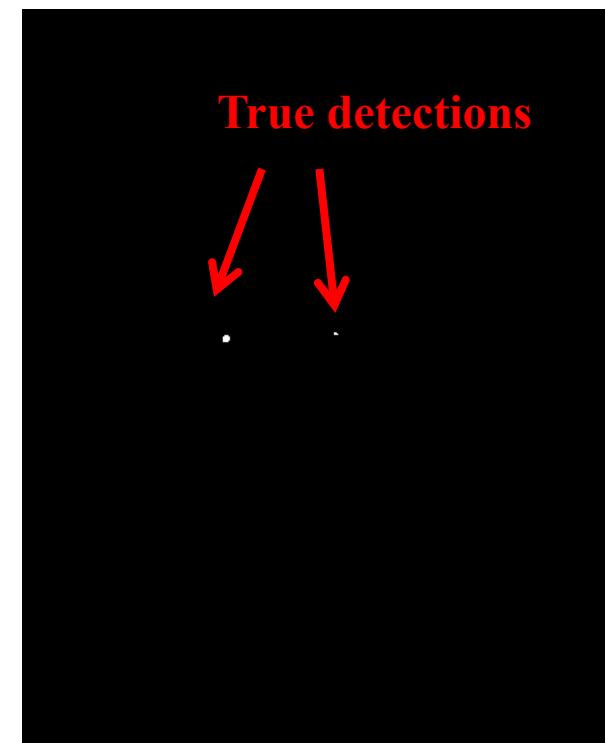
- Goal: find  in image
- Method 3: Normalized cross-correlation



Input



Normalized X-Correlation



Thresholded Image

True detections

Q: What is the best method to use?

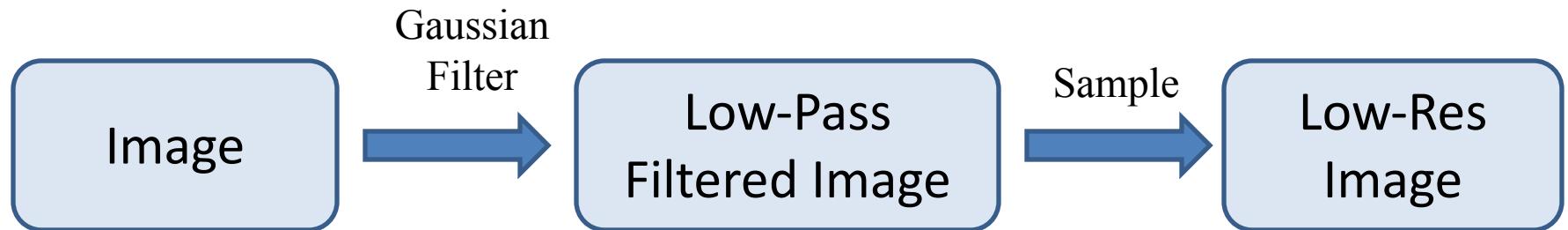
A: Depends

- Zero-mean filter: fastest but not a great matcher
- SSD: next fastest, sensitive to overall intensity
- Normalized cross-correlation: slowest, invariant to local average intensity and contrast

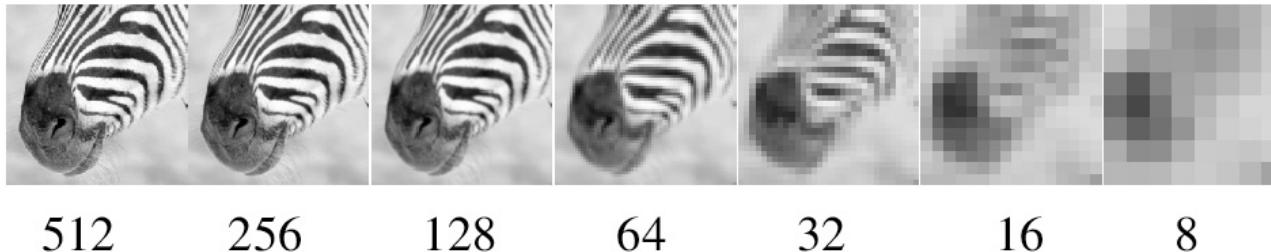
Q: What if we want to find larger or smaller eyes?

A: Image Pyramid

Review of Sampling



Gaussian pyramid



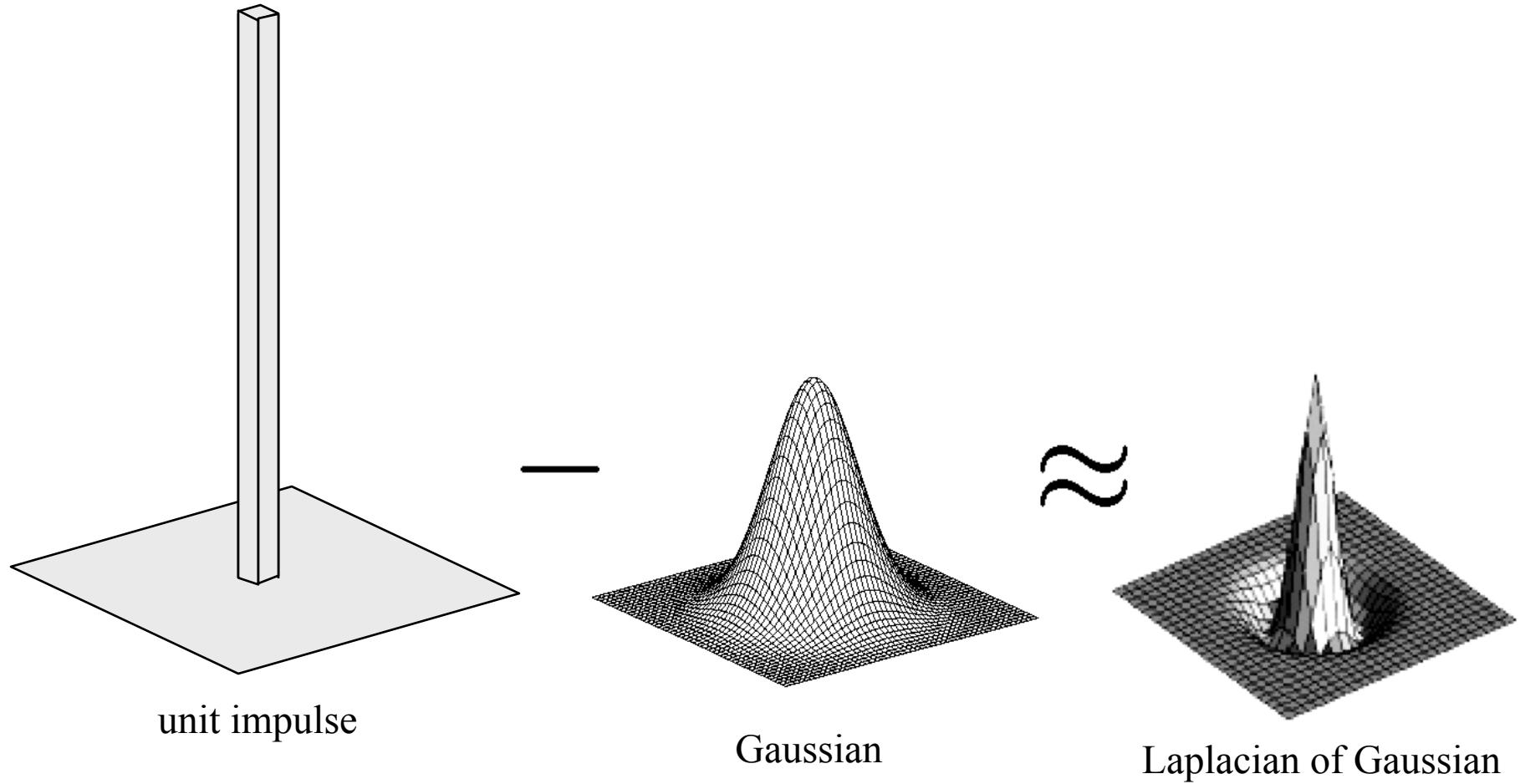
Source: D. Forsyth

Template Matching with Image Pyramids

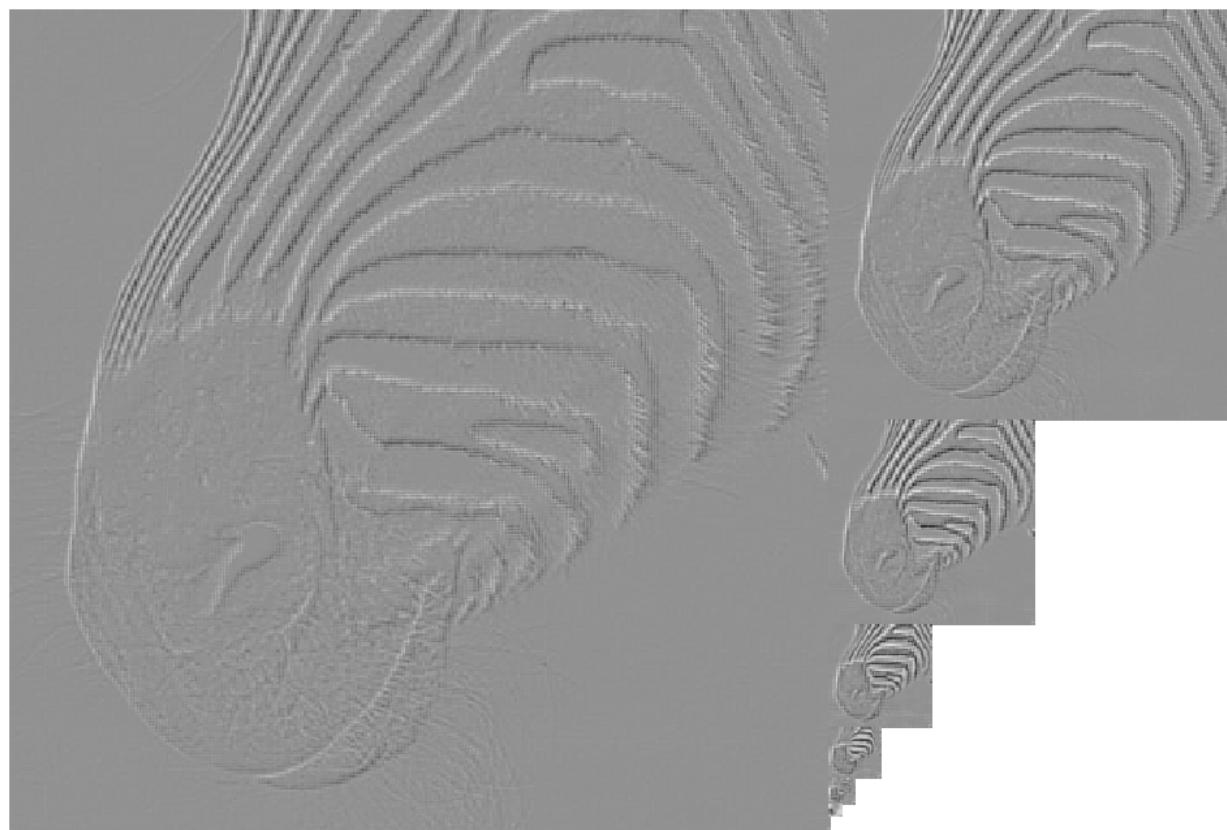
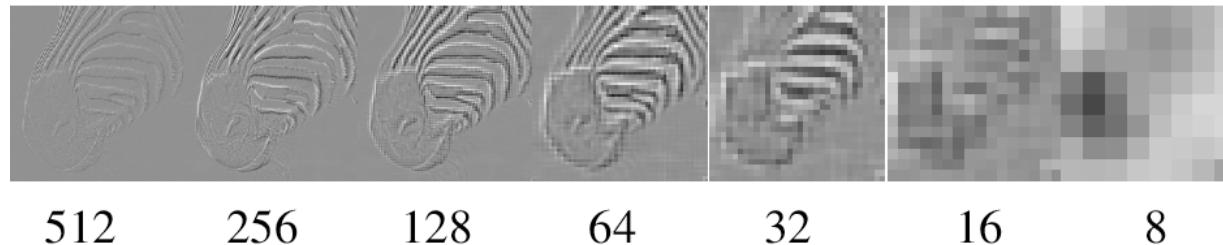
Input: Image, Template

1. Match template at current scale
2. Downsample image
 - In practice, scale step of 1.1 to 1.2
3. Repeat 1-2 until image is very small
4. Take responses above some threshold, perhaps with non-maxima suppression

Laplacian filter

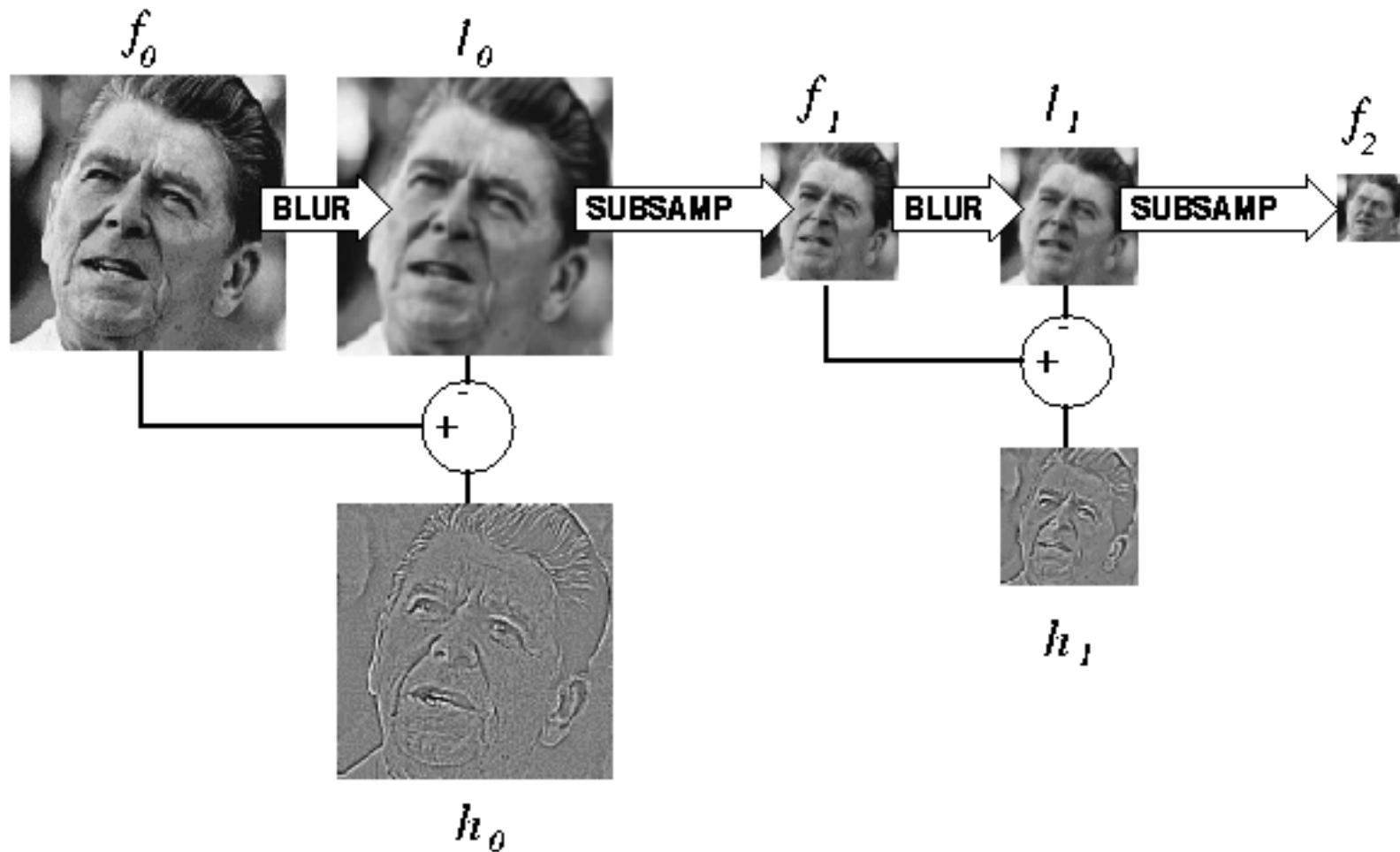


Laplacian pyramid



Source: D. Forsyth

Computing Gaussian/Laplacian Pyramid



Major uses of image pyramids

- Compression
- Object detection
 - Scale search
 - Features
- Detecting stable interest points
- Registration
 - Course-to-fine

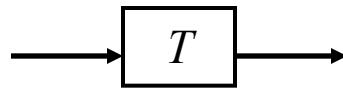
Alignment

Slides based on D. Hoiem's and
S. Lazebnik's slides

Alignment

- Alignment: find parameters of model that maps one set of points to another
- Typically want to solve for a global transformation that accounts for most true correspondences
- Difficulties
 - Noise (perturbation around true features, matches, etc.)
 - Outliers
 - Many-to-one matches or multiple objects

Parametric (global) warping



$$p = (x, y)$$

$$p' = (x', y')$$

Transformation T is a coordinate change

$$p' = T(p)$$

What does it mean that T is global?

- Is the same for any point p
- can be described by just a few numbers (parameters)

For linear transformations, we can represent T as a matrix

$$p' = Tp$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

Common transformations



original

Transformed



translation



rotation



aspect



affine

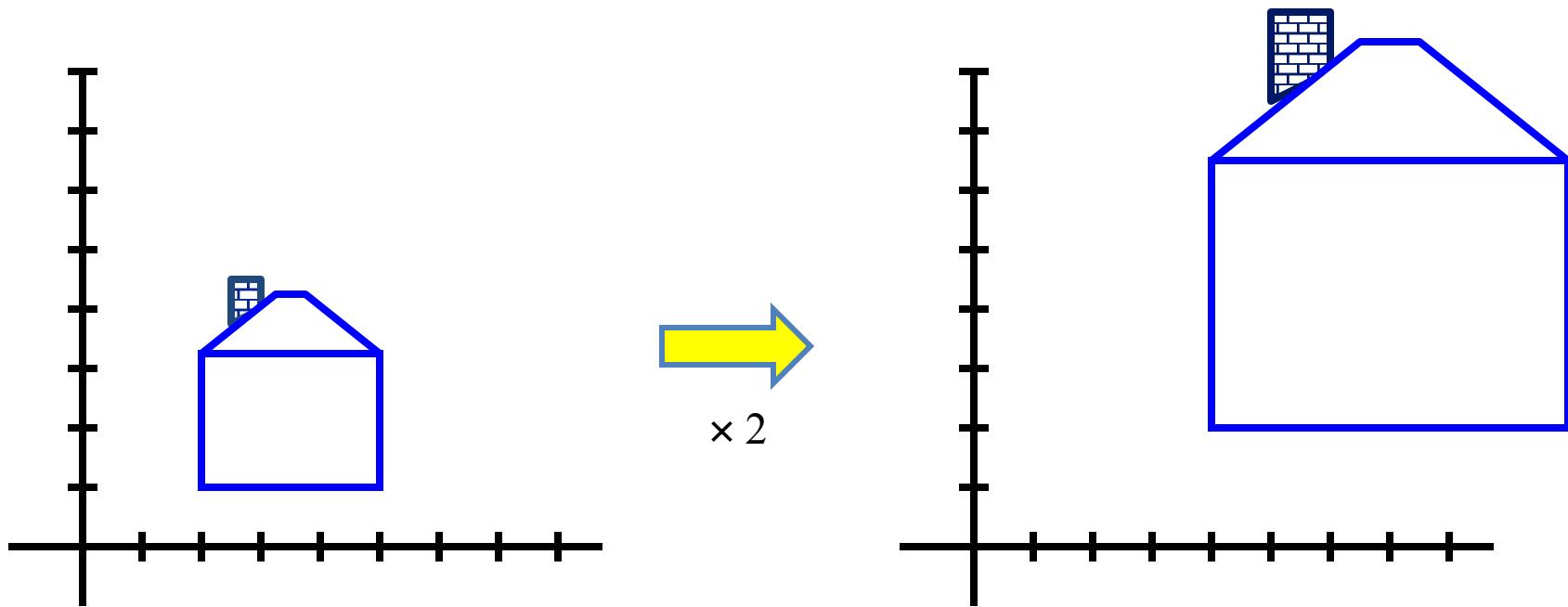


perspective

Slide credit (next few slides): A. Efros and/or S. Seitz

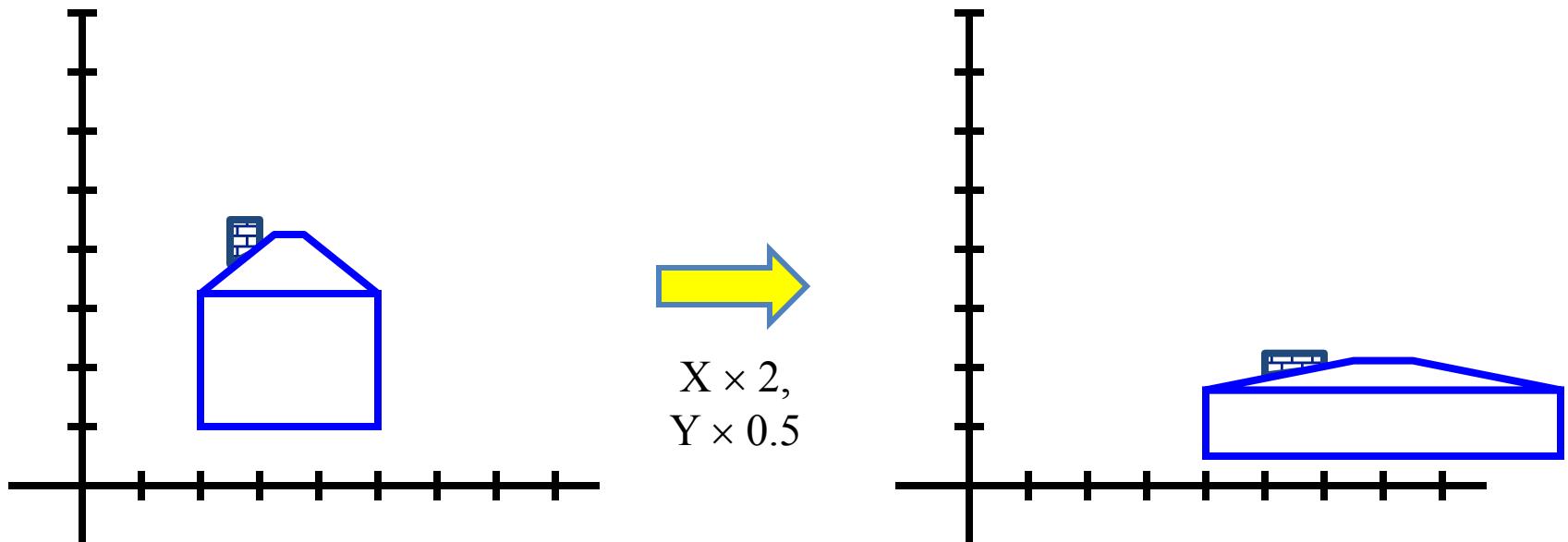
Scaling

- *Scaling* a coordinate means multiplying each of its components by a scalar
- *Uniform scaling* means this scalar is the same for all components:



Scaling

- *Non-uniform scaling*: different scalars per component:



Scaling

- Scaling operation: $x' = ax$

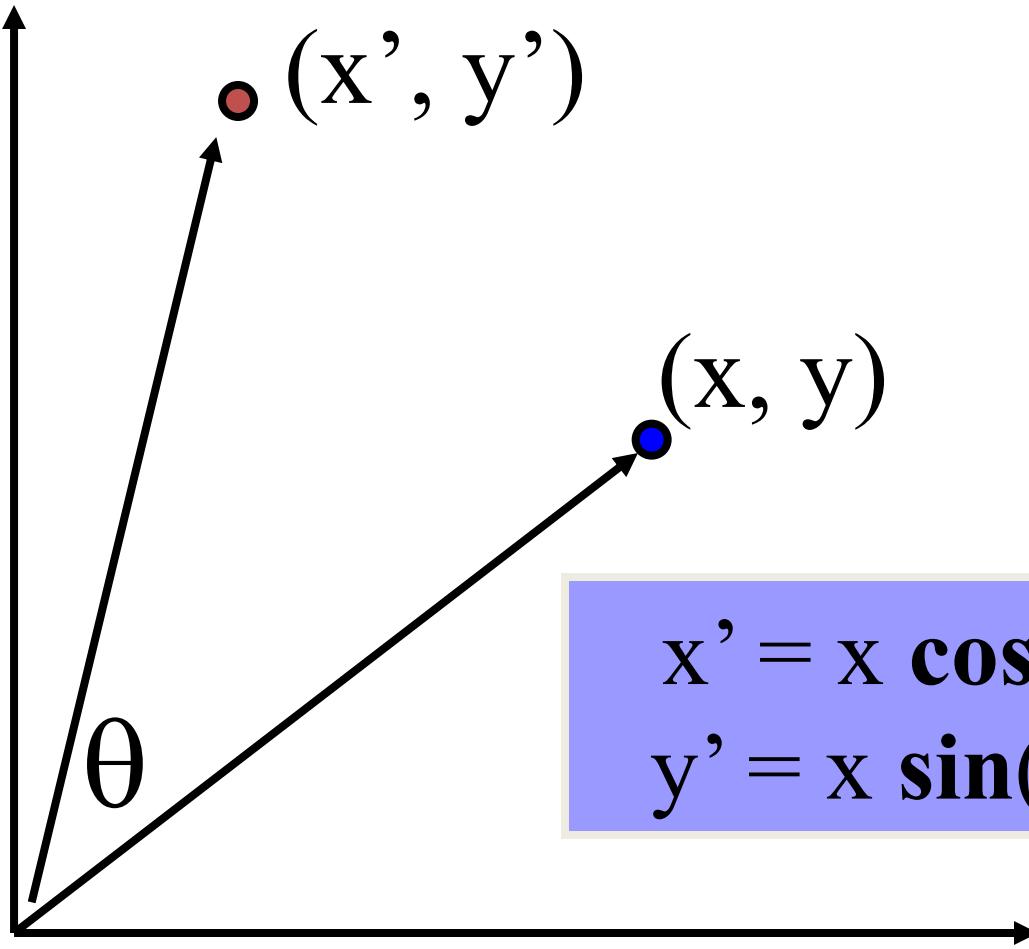
$$y' = by$$

- Or, in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$


scaling matrix S

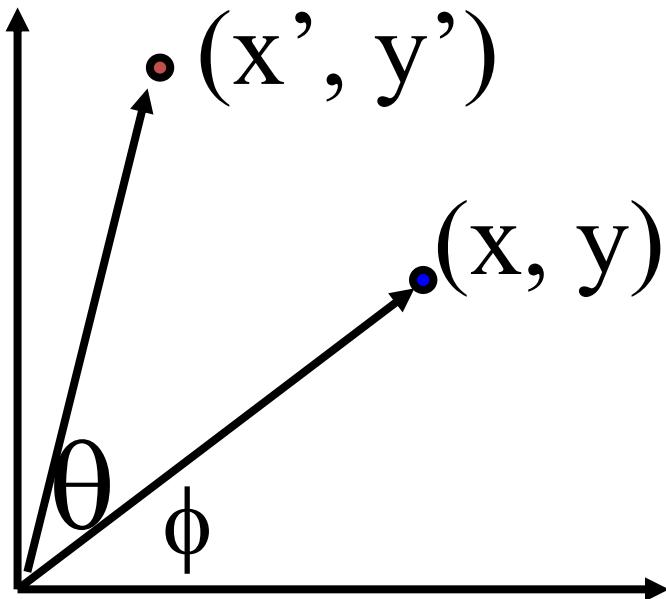
2-D Rotation



$$x' = x \cos(\theta) - y \sin(\theta)$$

$$y' = x \sin(\theta) + y \cos(\theta)$$

2-D Rotation



Polar coordinates...

$$x = r \cos(\phi)$$

$$y = r \sin(\phi)$$

$$x' = r \cos(\phi + \theta)$$

$$y' = r \sin(\phi + \theta)$$

Trig Identity...

$$x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)$$

$$y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta)$$

Substitute...

$$x' = x \cos(\theta) - y \sin(\theta)$$

$$y' = x \sin(\theta) + y \cos(\theta)$$

2-D Rotation

This is easy to capture in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{R} \begin{bmatrix} x \\ y \end{bmatrix}$$

Even though $\sin(\theta)$ and $\cos(\theta)$ are nonlinear functions of θ ,

- x' is a linear combination of x and y
- y' is a linear combination of x and y

What is the inverse transformation?

- Rotation by $-\theta$
- For rotation matrices $\mathbf{R}^{-1} = \mathbf{R}^T$

Basic 2D transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

Affine is any combination of translation, scale, rotation, shearing

Affine Transformations

Affine transformations are combinations of

- Linear transformations, and
- Translations

Properties of affine transformations:

- Lines map to lines
- Parallel lines remain parallel
- Ratios are preserved
- Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

or

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Projective Transformations

Projective transformations are combos of

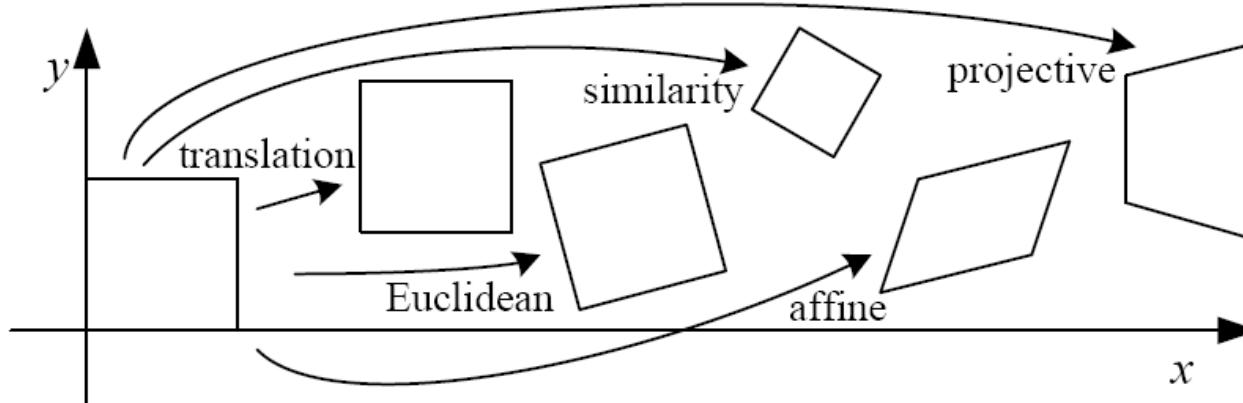
- Affine transformations, and
- Projective warps

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

Properties of projective transformations:

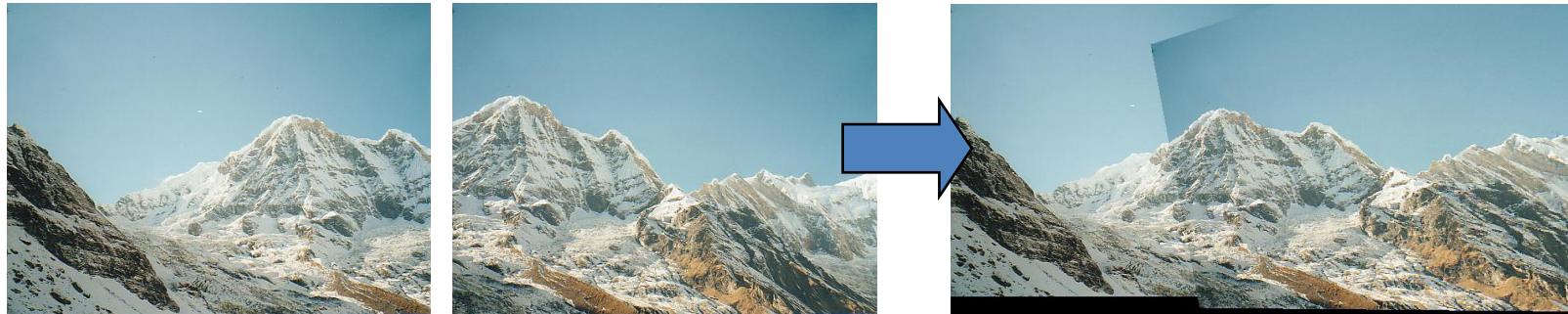
- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Ratios are not preserved
- Closed under composition
- Models change of basis
- Projective matrix is defined up to a scale (8 DOF)

2D image transformations (reference table)



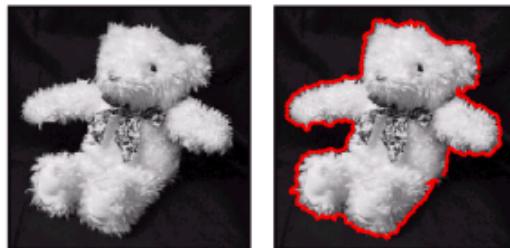
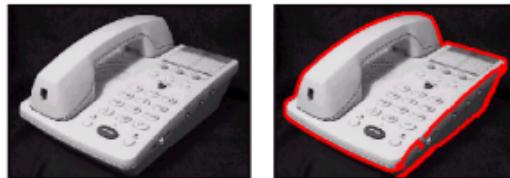
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	2	orientation + ...	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	3	lengths + ...	
similarity	$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	4	angles + ...	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2 \times 3}$	6	parallelism + ...	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3 \times 3}$	8	straight lines	

Image alignment: Challenges



Small degree of overlap

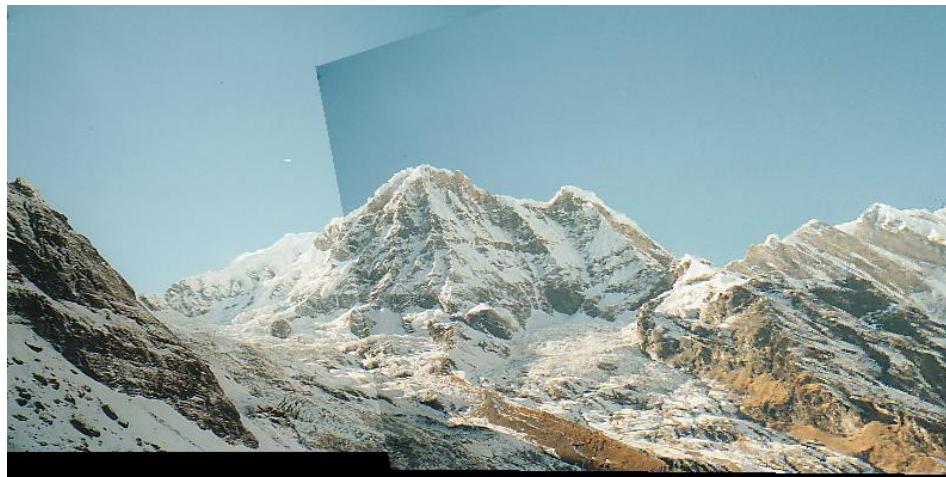
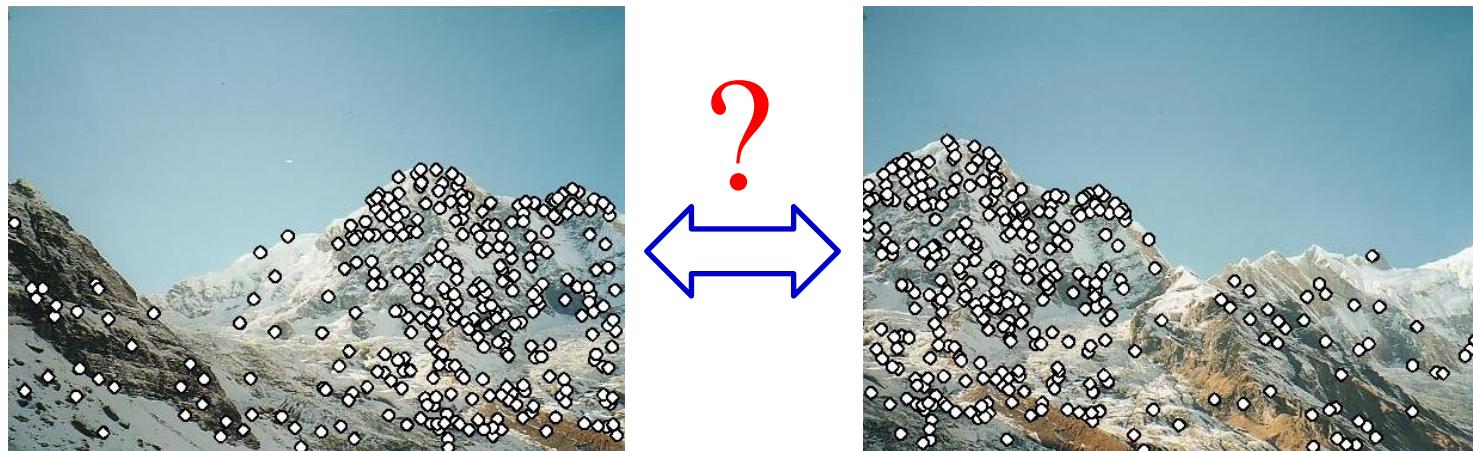
Intensity changes



Occlusion,
clutter

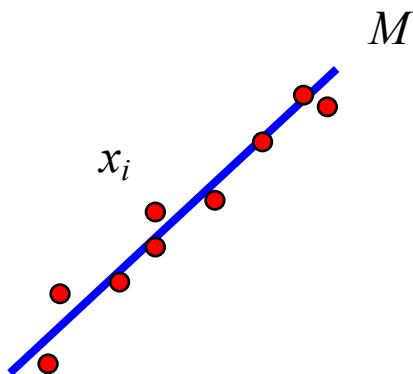
Feature-based alignment

- Search sets of feature matches that agree in terms of:
 - a) Local appearance
 - b) Geometric configuration



Alignment as fitting

- Previously: fitting a model to features in one image

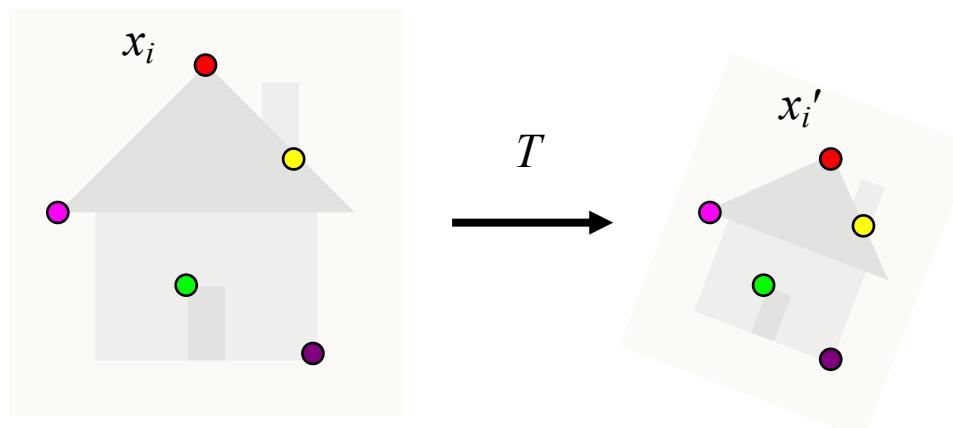


Find model M that minimizes

$$\sum_i \text{residual}(x_i, M)$$

Alignment as fitting

- Alignment: fitting a model to a transformation between pairs of features (*matches*) in two images

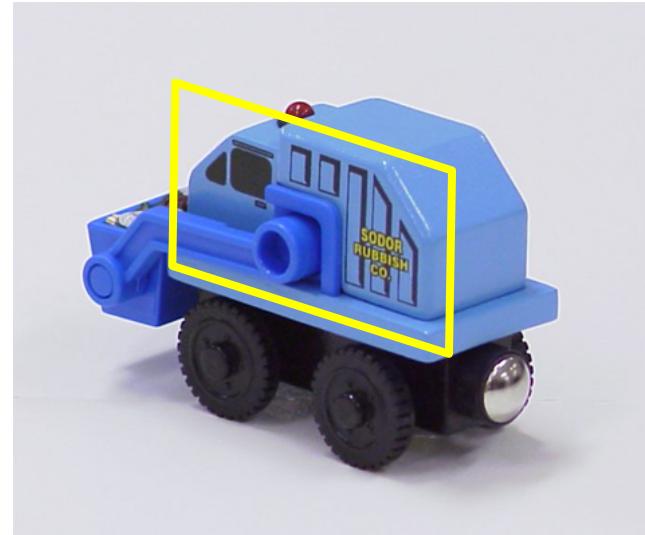


Find transformation T
that minimizes

$$\sum_i \text{residual}(T(x_i), x'_i)$$

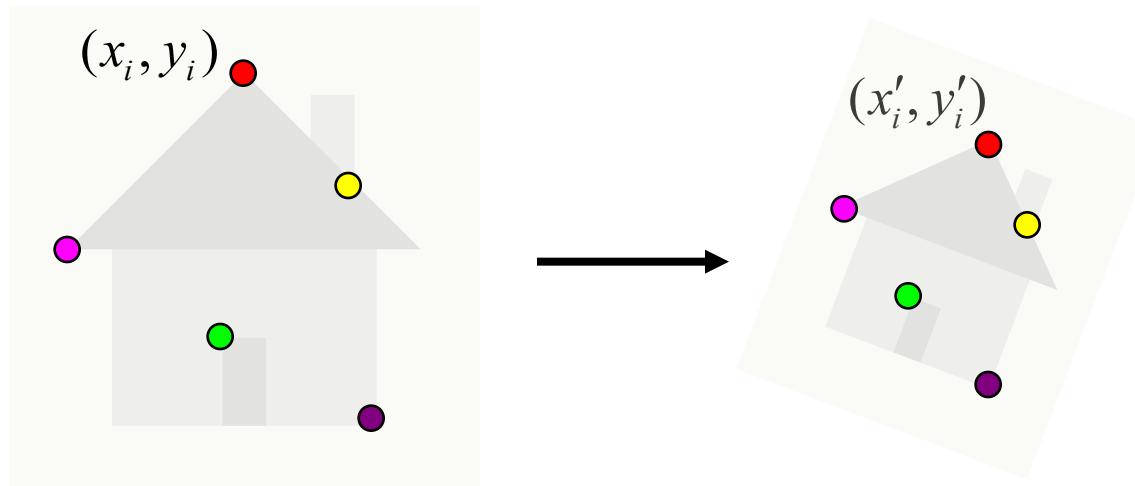
Let's start with affine transformations

- Simple fitting procedure (linear least squares)
- Approximates viewpoint changes for roughly planar objects
- Can be used to initialize fitting for more complex models



Fitting an affine transformation

- Assume we know the correspondences, how do we get the transformation?



$$\mathbf{x}'_i = \mathbf{M}\mathbf{x}_i + \mathbf{t}$$

Want to find \mathbf{M} , \mathbf{t} to minimize

$$\sum_{i=1}^n \| \mathbf{x}'_i - \mathbf{M}\mathbf{x}_i - \mathbf{t} \|^2$$

$$\begin{bmatrix} x'_i \\ y'_i \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

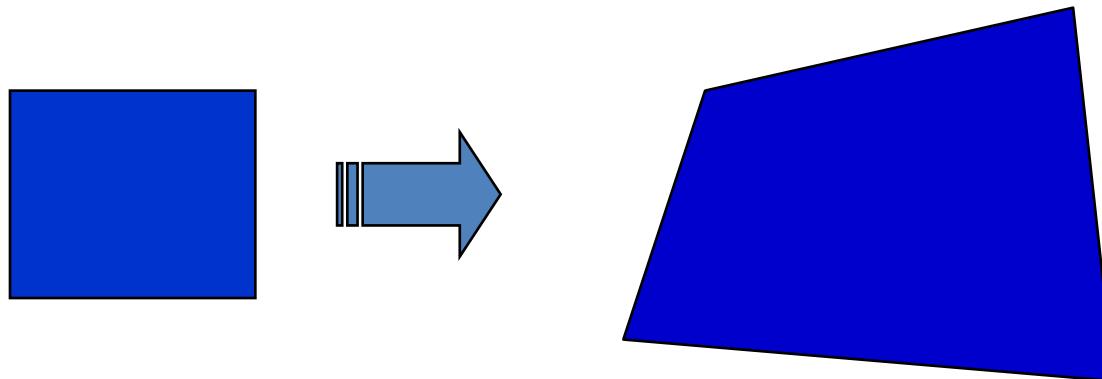
Fitting an affine transformation

$$\begin{bmatrix} & & \cdots \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x'_i \\ y'_i \\ \cdots \end{bmatrix}$$

- Linear system with six unknowns
- Each match gives us two linearly independent equations: need at least three to solve for the transformation parameters

Fitting a plane projective transformation

- Homography: plane projective transformation (transformation taking a quad to another arbitrary quad)

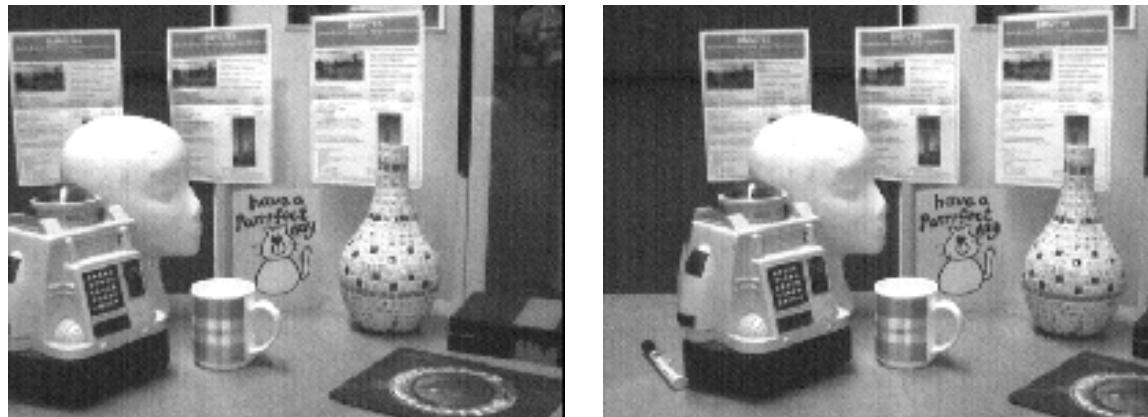


Homography

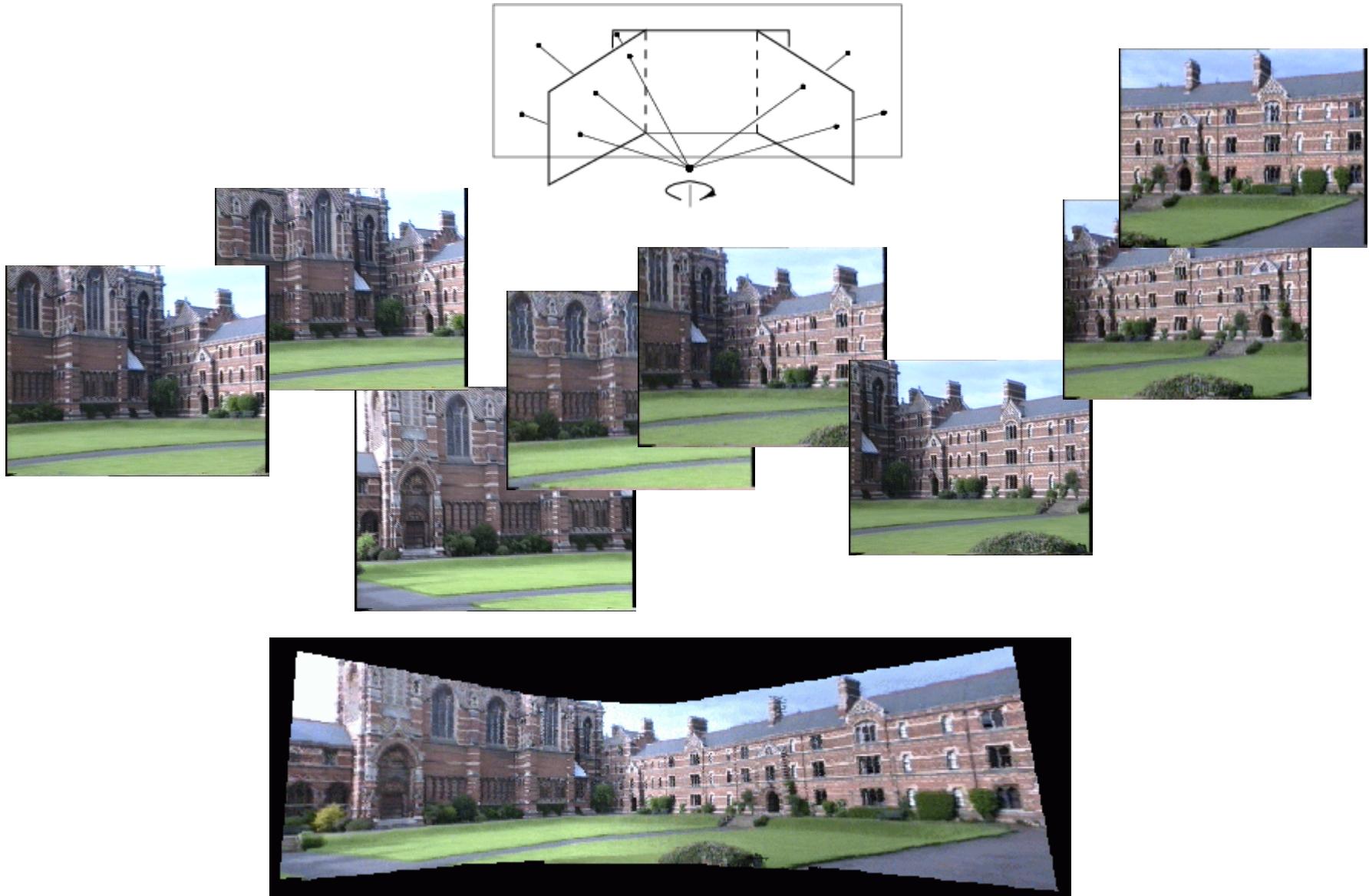
- The transformation between two views of a planar surface



- The transformation between images from two cameras that share the same center



Application: Panorama stitching



Source: Hartley & Zisserman

Fitting a homography

- Homogeneous coordinates (more later)

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Converting *to* homogeneous
image coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Converting *from* homogeneous
image coordinates

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Fitting a homography

- Equation for homography:

$$\lambda \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \quad \lambda \mathbf{x}'_i = \mathbf{H} \mathbf{x}_i$$
$$\mathbf{x}'_i \times \mathbf{H} \mathbf{x}_i = 0$$

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{h}_1^T \mathbf{x}_i \\ \mathbf{h}_2^T \mathbf{x}_i \\ \mathbf{h}_3^T \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} y'_i \mathbf{h}_3^T \mathbf{x}_i - \mathbf{h}_2^T \mathbf{x}_i \\ \mathbf{h}_1^T \mathbf{x}_i - x'_i \mathbf{h}_3^T \mathbf{x}_i \\ x'_i \mathbf{h}_2^T \mathbf{x}_i - y'_i \mathbf{h}_1^T \mathbf{x}_i \end{bmatrix}$$

$$\begin{bmatrix} 0^T & -\mathbf{x}_i^T & y'_i \mathbf{x}_i^T \\ \mathbf{x}_i^T & 0^T & -x'_i \mathbf{x}_i^T \\ -y'_i \mathbf{x}_i^T & x'_i \mathbf{x}_i^T & 0^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0$$

3 equations,
only 2 linearly
independent

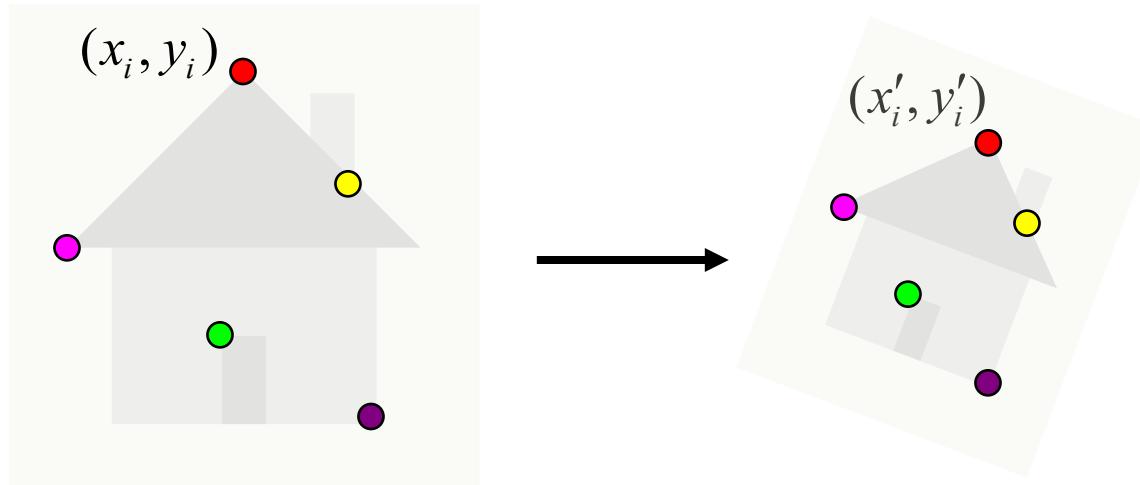
Direct linear transform

$$\begin{bmatrix} 0^T & \mathbf{x}_1^T & -y'_1 \mathbf{x}_1^T \\ \mathbf{x}_1^T & 0^T & -x'_1 \mathbf{x}_1^T \\ \dots & \dots & \dots \\ 0^T & \mathbf{x}_n^T & -y'_n \mathbf{x}_n^T \\ \mathbf{x}_n^T & 0^T & -x'_n \mathbf{x}_n^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0 \quad \mathbf{A h} = 0$$

- \mathbf{H} has 8 degrees of freedom (9 parameters, but scale is arbitrary)
- One match gives us two linearly independent equations
- Homogeneous least squares: find \mathbf{h} minimizing $\|\mathbf{Ah}\|^2$
 - Eigenvector of $\mathbf{A}^T \mathbf{A}$ corresponding to smallest eigenvalue
 - Four matches needed for a minimal solution

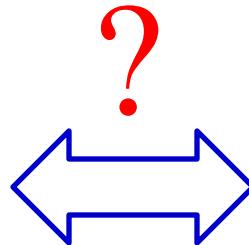
Robust feature-based alignment

- So far, we've assumed that we are given a set of “ground-truth” correspondences between the two images we want to align
- What if we don't know the correspondences?

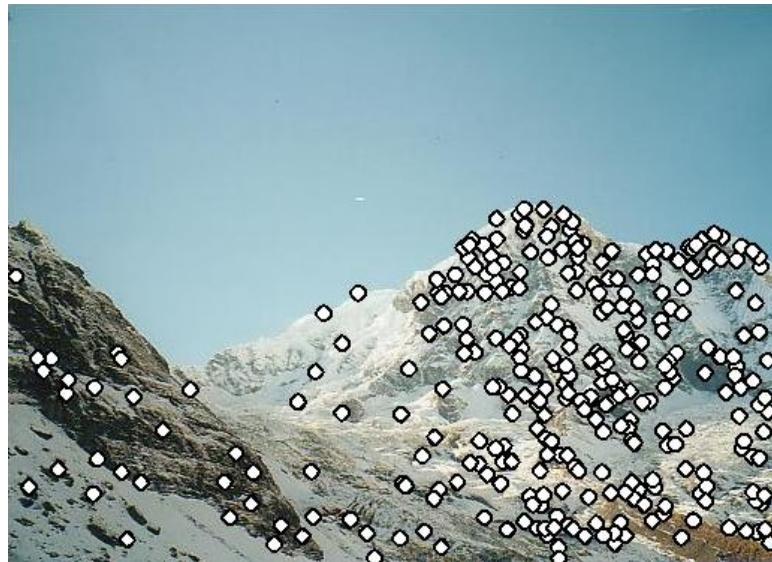


Robust feature-based alignment

- So far, we've assumed that we are given a set of “ground-truth” correspondences between the two images we want to align
- What if we don't know the correspondences?

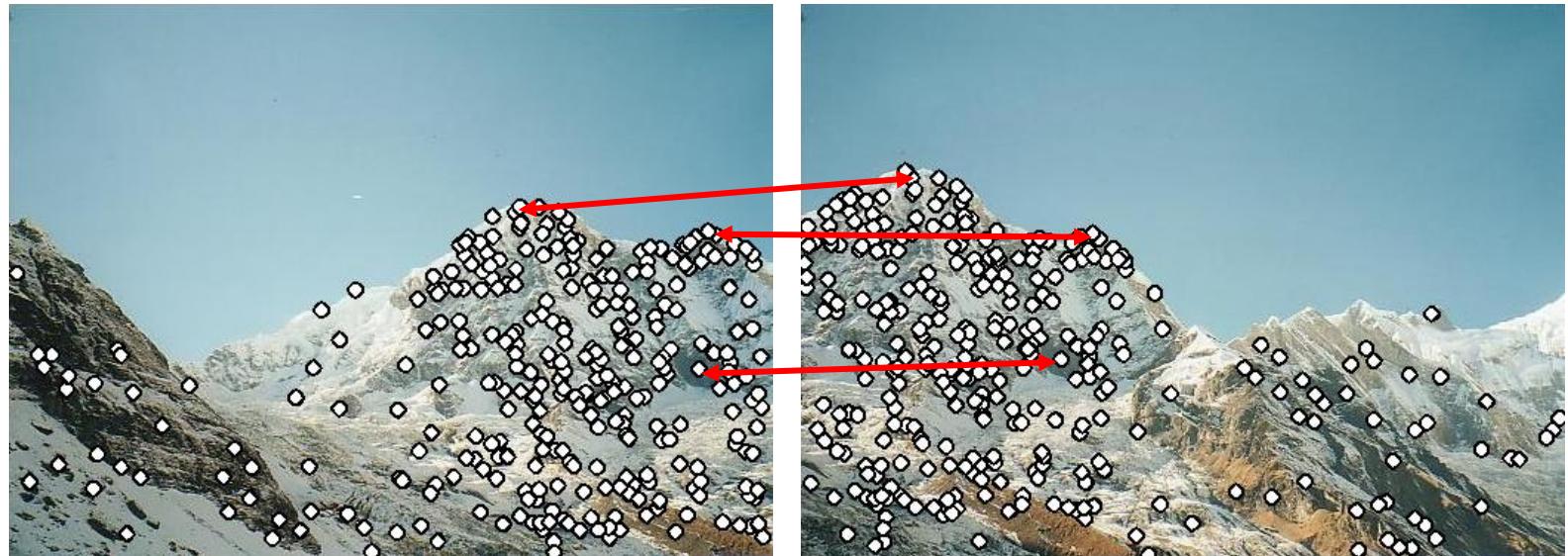


Robust feature-based alignment



- Extract features

Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
 - *Hypothesize* transformation T

Robust feature-based alignment



- Extract features
- Compute *putative matches*
- Loop:
 - *Hypothesize* transformation T
 - *Verify* transformation (search for other matches consistent with T)

Generating putative correspondences



?
↔



feature
descriptor

?
=

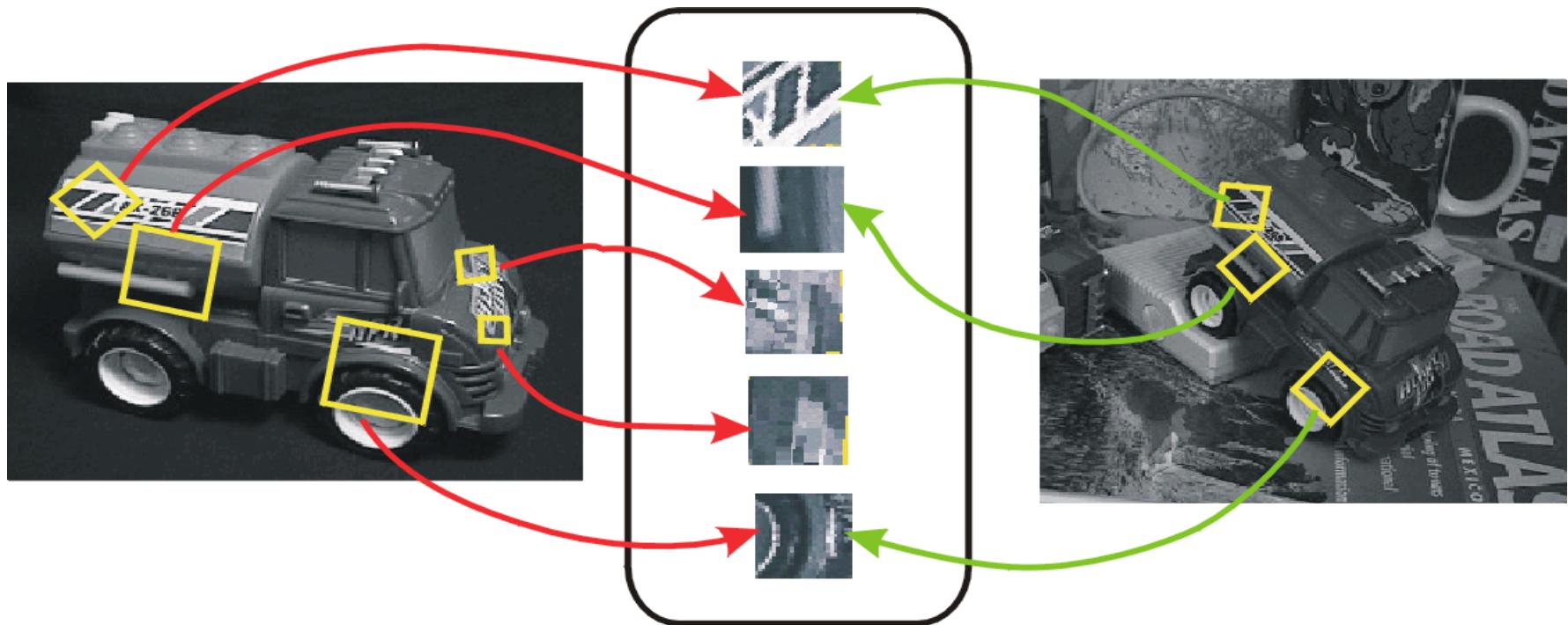


feature
descriptor

- Need to compare *feature descriptors* of local patches surrounding interest points

Feature descriptors

- Recall: feature detection and description



Feature descriptors

- Simplest descriptor: vector of raw intensity values
- How to compare two such vectors?
 - Sum of squared differences (SSD)

$$\text{SSD}(\mathbf{u}, \mathbf{v}) = \sum_i (u_i - v_i)^2$$

- Not invariant to intensity change
- Normalized correlation

$$\rho(\mathbf{u}, \mathbf{v}) = \frac{(\mathbf{u} - \bar{\mathbf{u}}) \cdot (\mathbf{v} - \bar{\mathbf{v}})}{\|\mathbf{u} - \bar{\mathbf{u}}\| \|\mathbf{v} - \bar{\mathbf{v}}\|} = \frac{\sum_i (u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\left(\sum_j (u_j - \bar{u})^2 \right) \left(\sum_j (v_j - \bar{v})^2 \right)}}$$

- Invariant to affine intensity change

Disadvantage of intensity vectors as descriptors

- Small deformations can affect the matching score a lot

