南京大學
NANJING UNIVERSITY

# 计算机视觉表征与识别
# Chapter 9: Alignment and Transformation

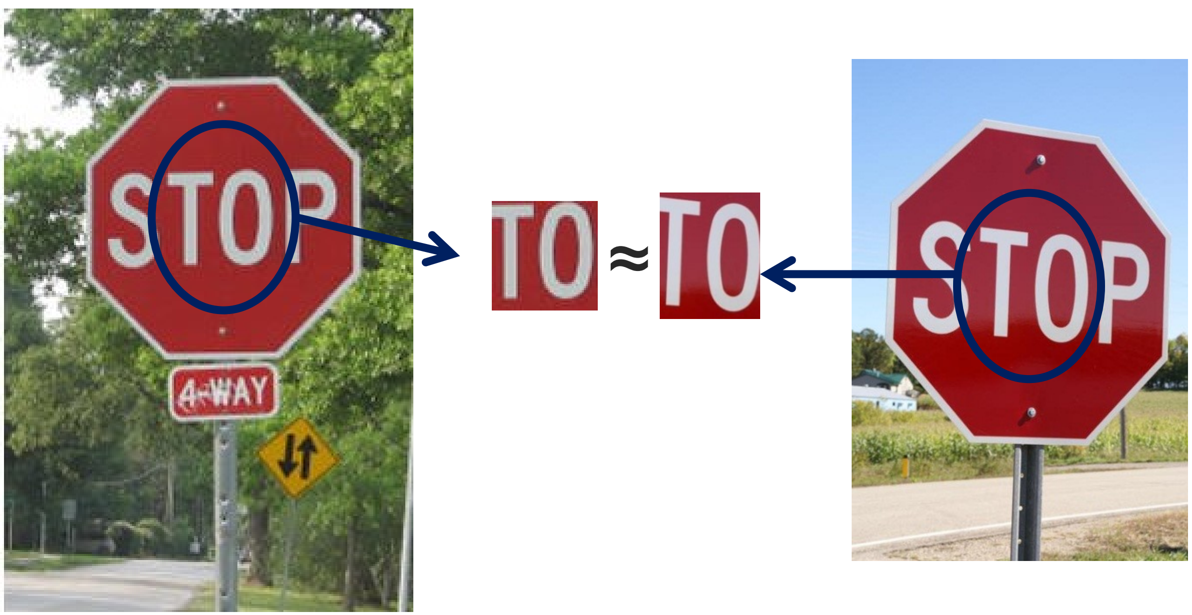
王利民

媒体计算课题组

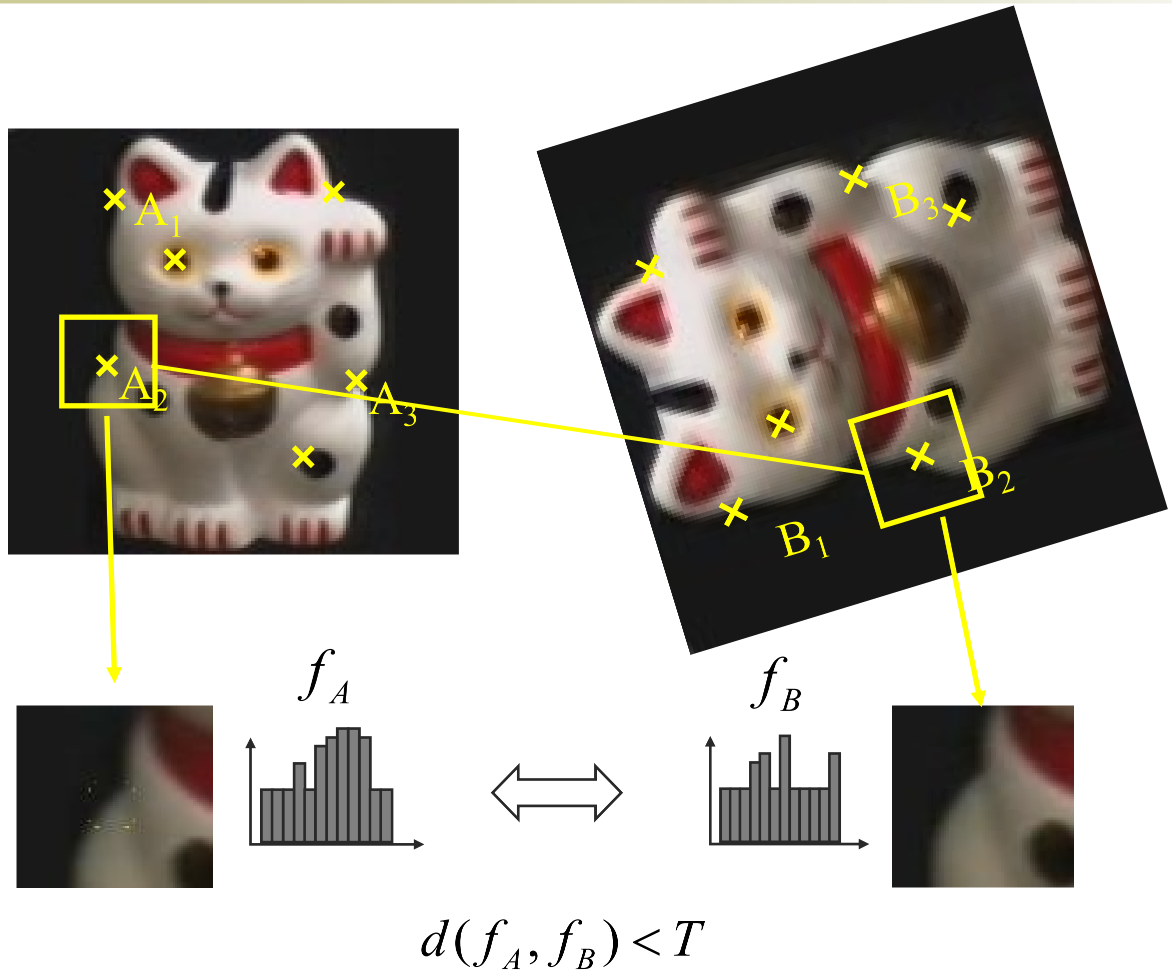
http://mcg.nju.edu.cn/

# Correspondence and alignment

**Correspondence:** matching points, patches, edges, or regions across images

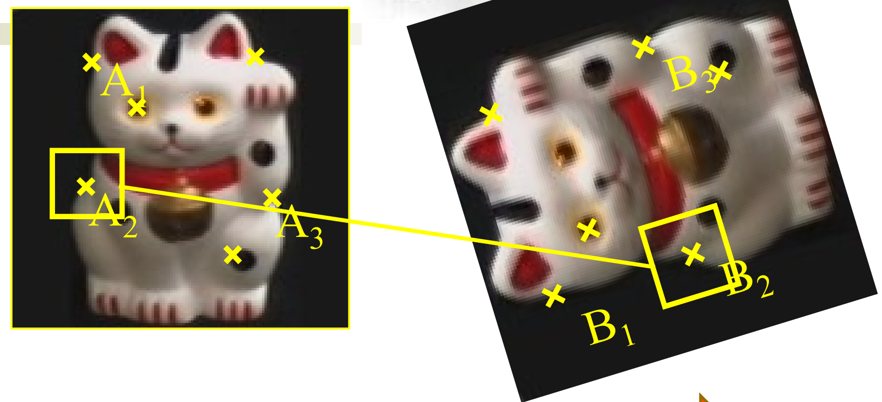


# Recap: Keypoint Matching




1. Find a set of distinctive key-points
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

# Recap: Key trade-offs




## Detection

More Repeatable

Robust detection
Precise localization

More Points

Robust to occlusion
Works with less texture

## Description

More Distinctive

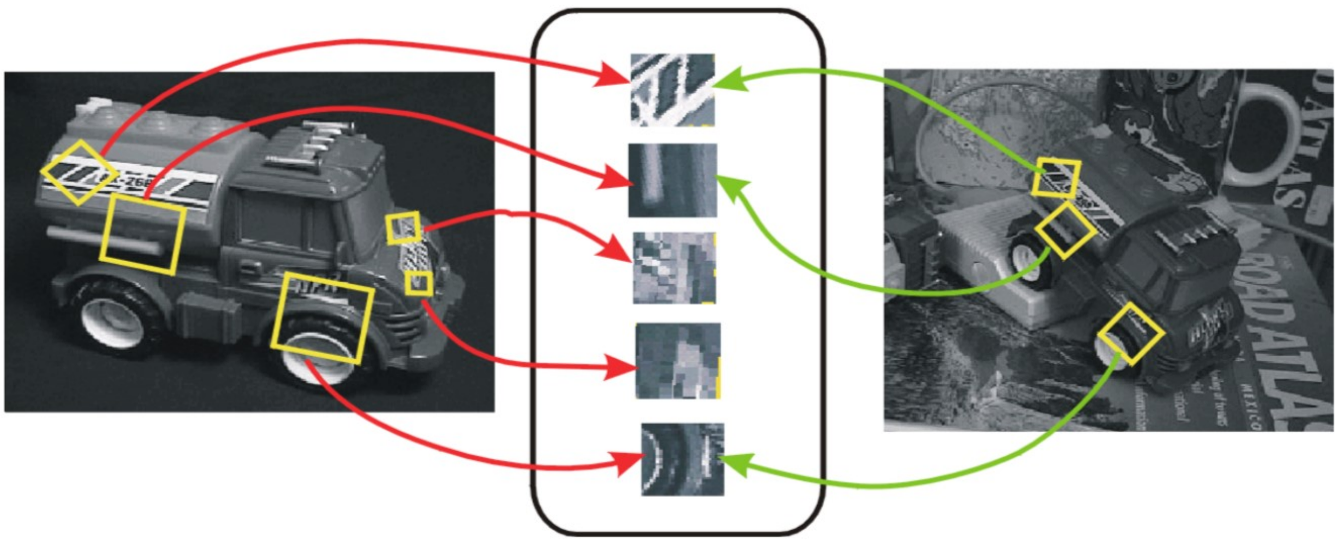
Minimize wrong matches

More Flexible

Robust to expected variations
Maximize correct matches

# Invariance vs. Covariance

- **Invariance:**
  - features(transform(image)) = features(image)
- **Covariance:**
  - features(transform(image)) = transform(features(image))



**Covariant detection ⇒ invariant description**

# Summary: Scale Invariant Detection

- **Given:** Two images of the same scene with a large *scale difference* between them.
- **Goal:** Find *the same* interest points *independently* in each image.
- **Solution:** Search for *maxima* of suitable functions in *scale* and in *space* (over the image).

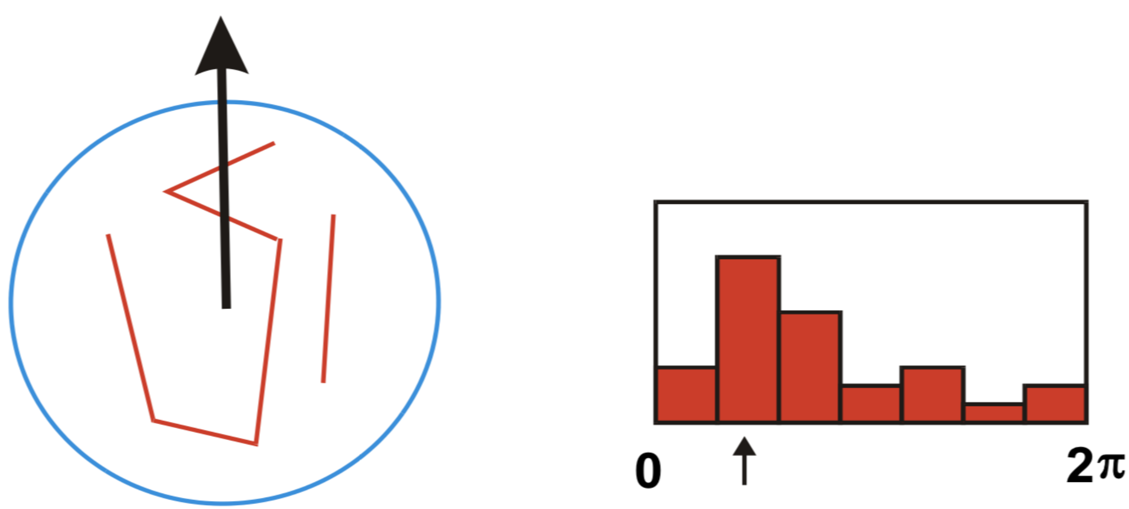
- Two strategies
  - Laplacian-of-Gaussian (LoG)
  - Difference-of-Gaussian (DoG) as a fast approximation

  - *These can be used either on their own, or in combinations with single-scale keypoint detectors (Harris, Hessian).*

# Orientation Normalization

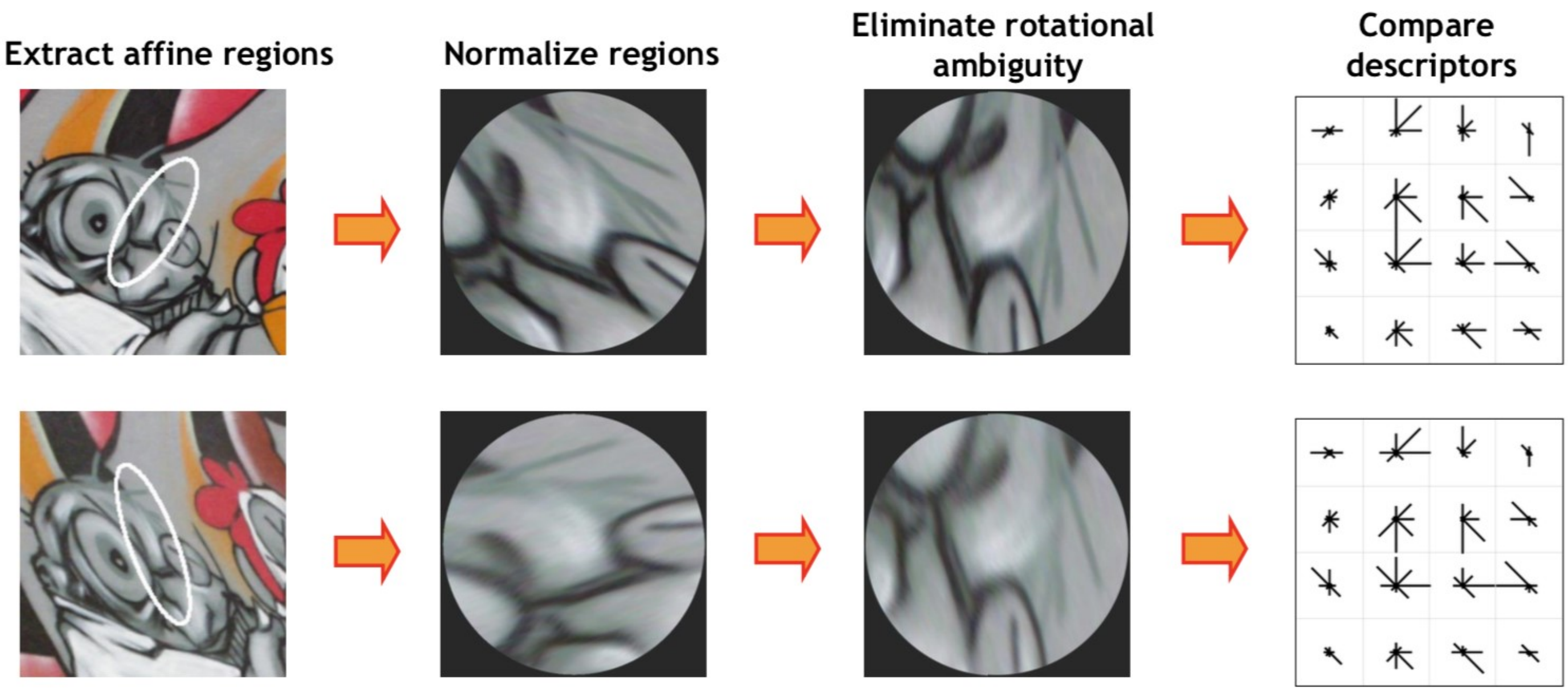
- Compute orientation histogram
- Select dominant orientation
- Normalize: rotate to fixed orientation

[Lowe, SIFT, 1999]



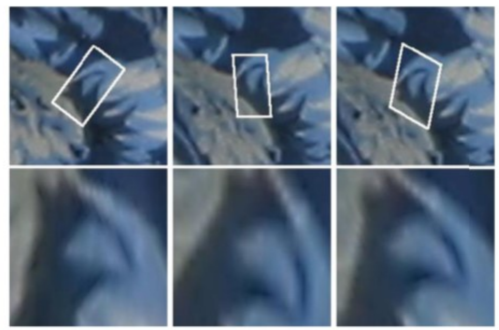

Slide adapted from David Lowe

# Summary：Affine Invariance

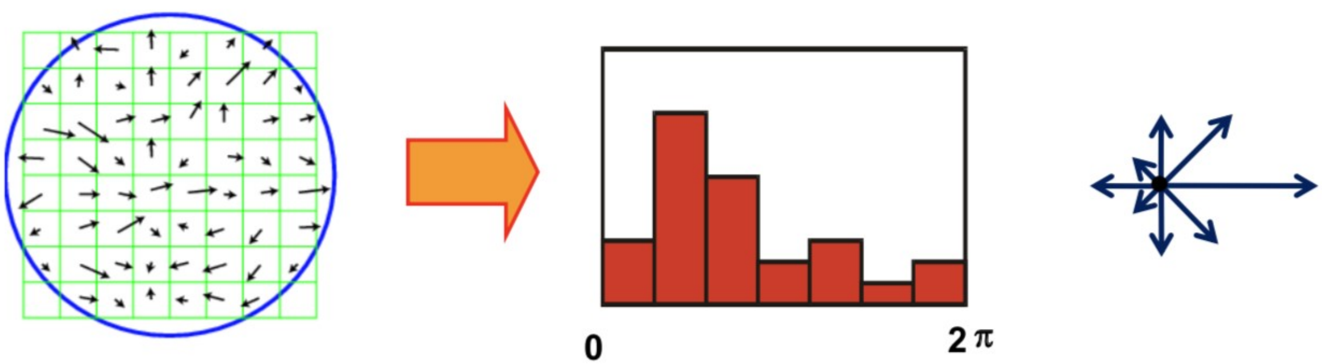


# Feature Descriptor

- **Disadvantage of patches as descriptors:**
  - Small shifts can affect matching score a lot

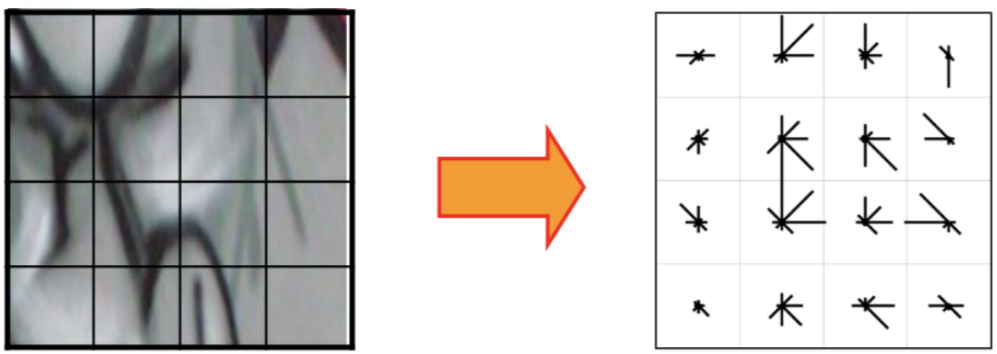


- **Solution: histograms**



# Feature Descriptor：SIFT

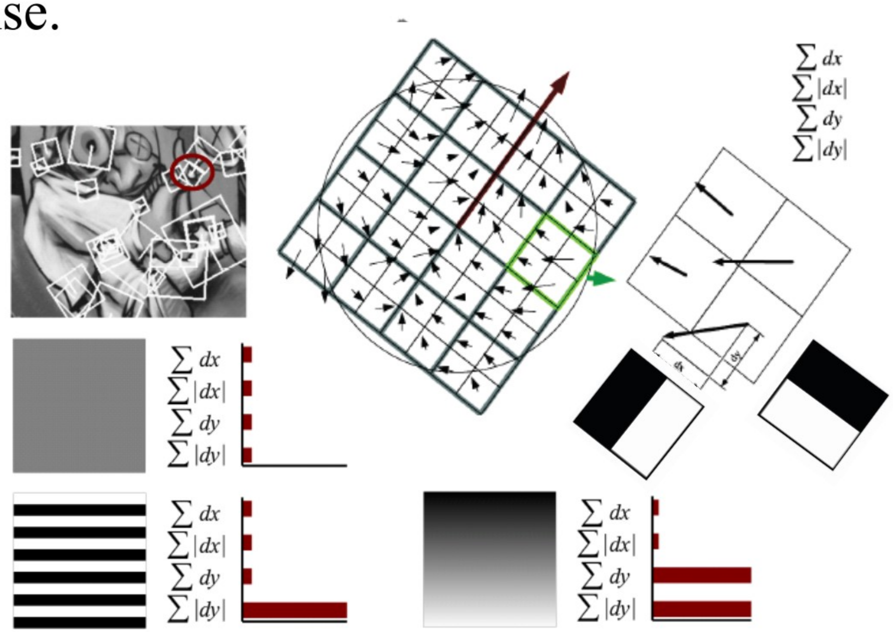
- **Scale Invariant Feature Transform**
- **Descriptor computation:**
  - Divide patch into 4x4 sub-patches: 16 cells
  - Compute histogram of gradient orientations (8 reference angles) for all pixels inside each sub-patch
  - Resulting descriptor: 4x4x8 = 128 dimensions



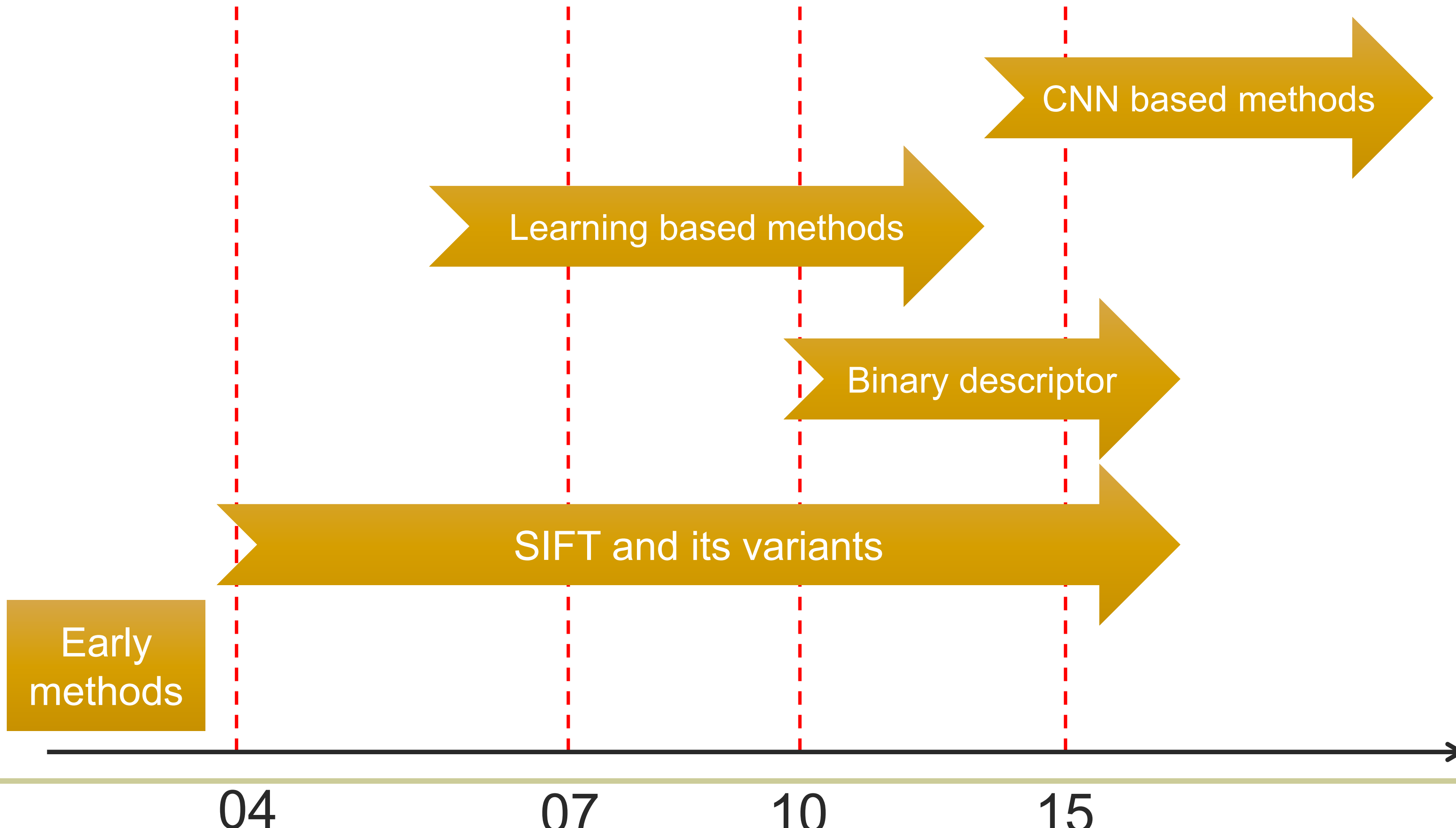
David G. Lowe. "Distinctive image features from scale-invariant keypoints." *IJCV* 60 (2), pp. 91-110, 2004.

# SURF: Descriptor Extraction

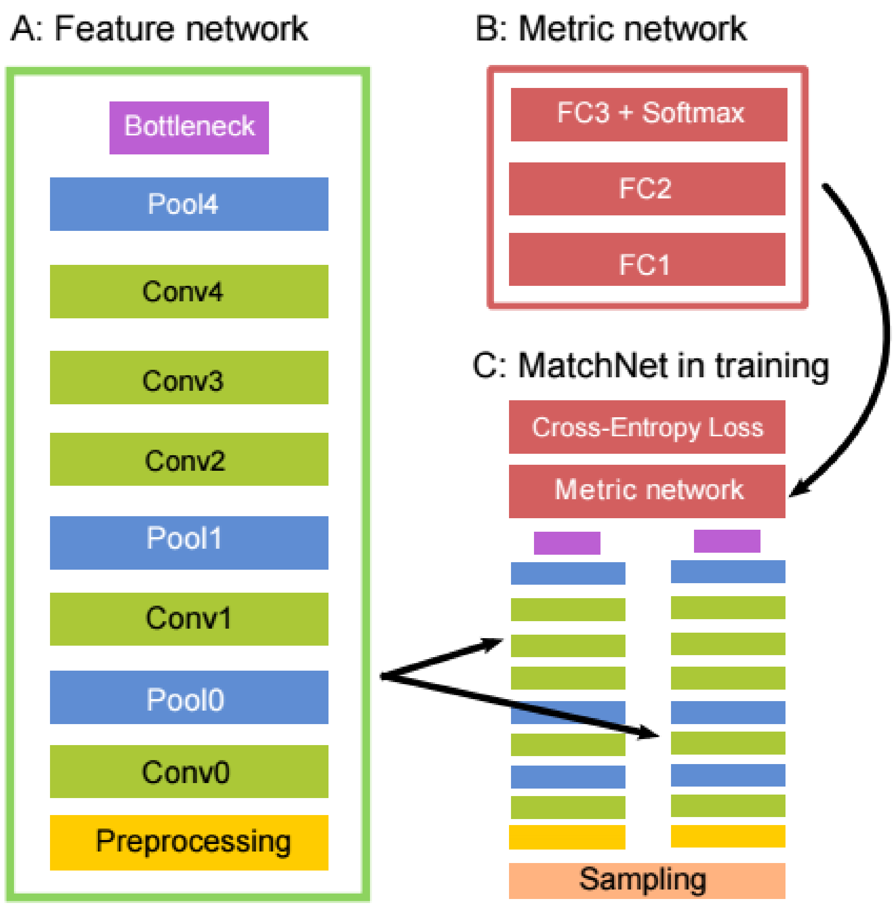
1. Split the interest region (20s x 20s) into 4 x 4 square sub-regions.
2. Calculate Haar wavelet responses dx and dy, and weight the responses with a Gaussian kernel.
3. Sum the response over each sub-region for dx and dy, then sum the absolute value of response.
4. Concatenate summation results in all sub-regions, forming a 64D SURF descriptor.



# Local Descriptors: Trend



# MatchNet



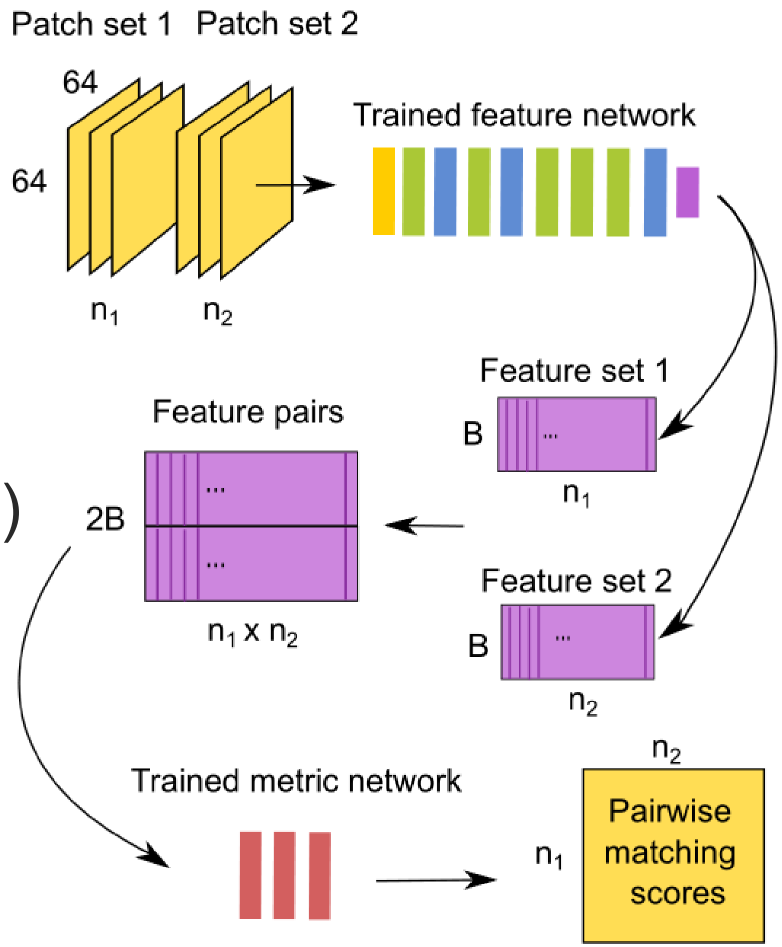

- Simultaneously learn the descriptor and the metric
- Siamese Feature descriptor network
- Metric network on top
- Cross-entropy loss, transfer matching problem to classification problem
- Train time： 1 day – 1 week

# Training MatchNet

- Cross-entropy error

$$E = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

- Stochastic gradient descent (SGD)
- A special reservoir sampler for negative sampling

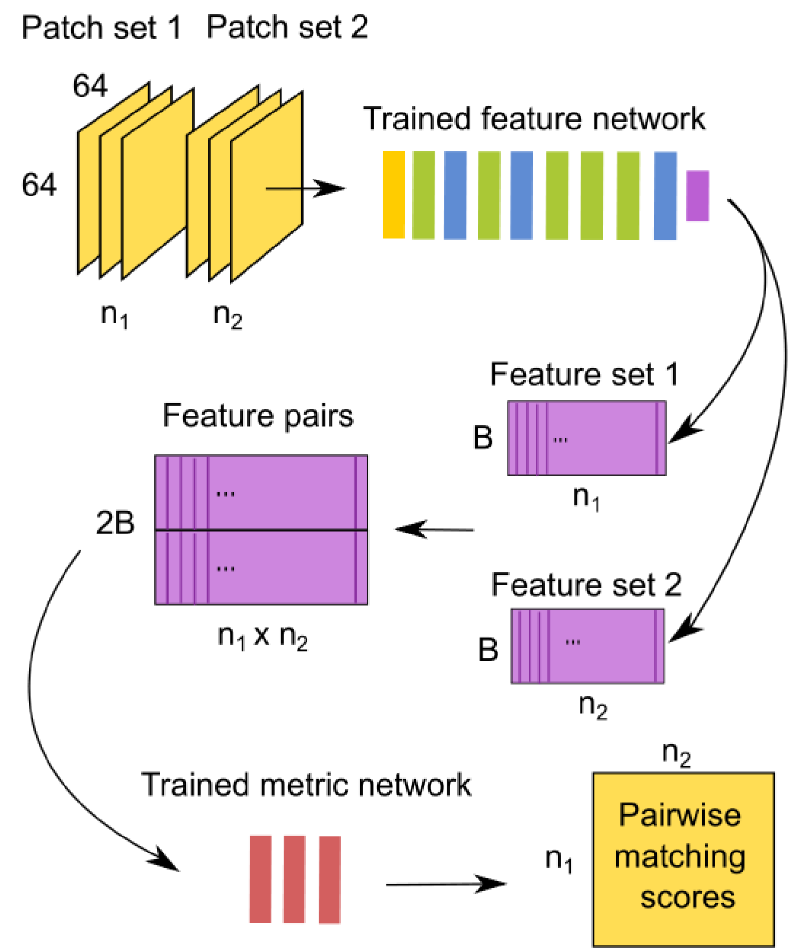


# Testing MatchNet

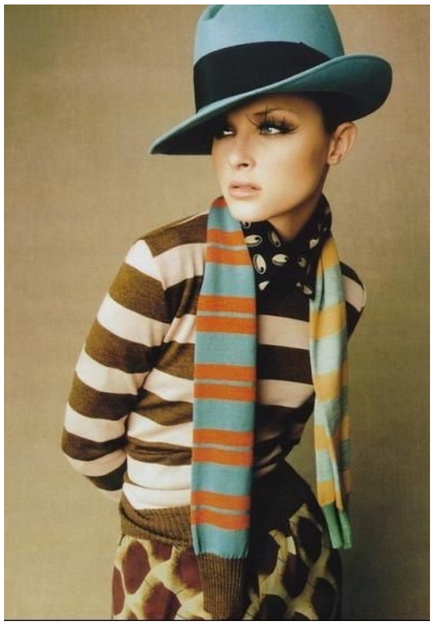
A two-stage prediction pipeline:

1. Generate feature descriptors for all patches.

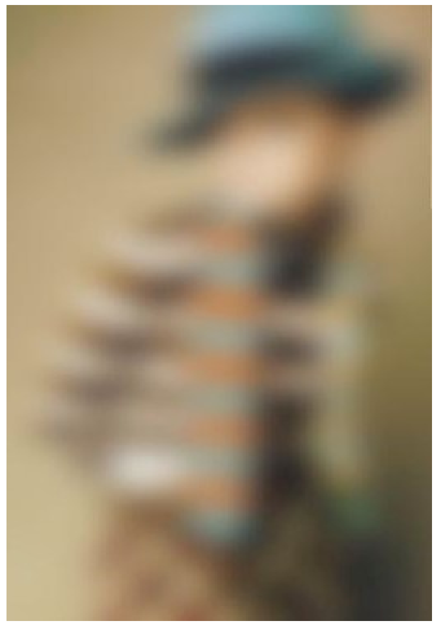
2. Pair the features and push them through the metric network to get the scores.



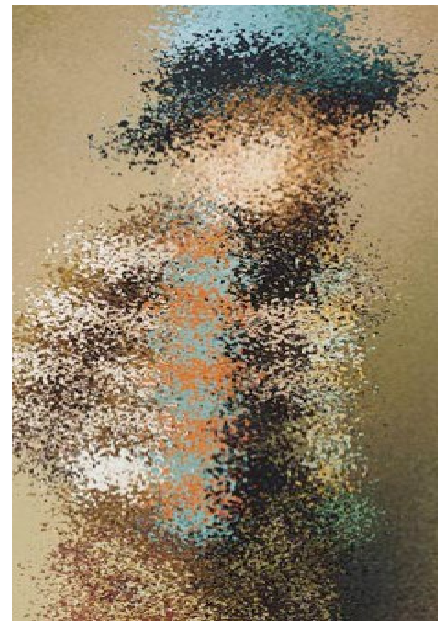
# Discriminative power



Raw pixels



Sampled



Locally orderless

Global histogram

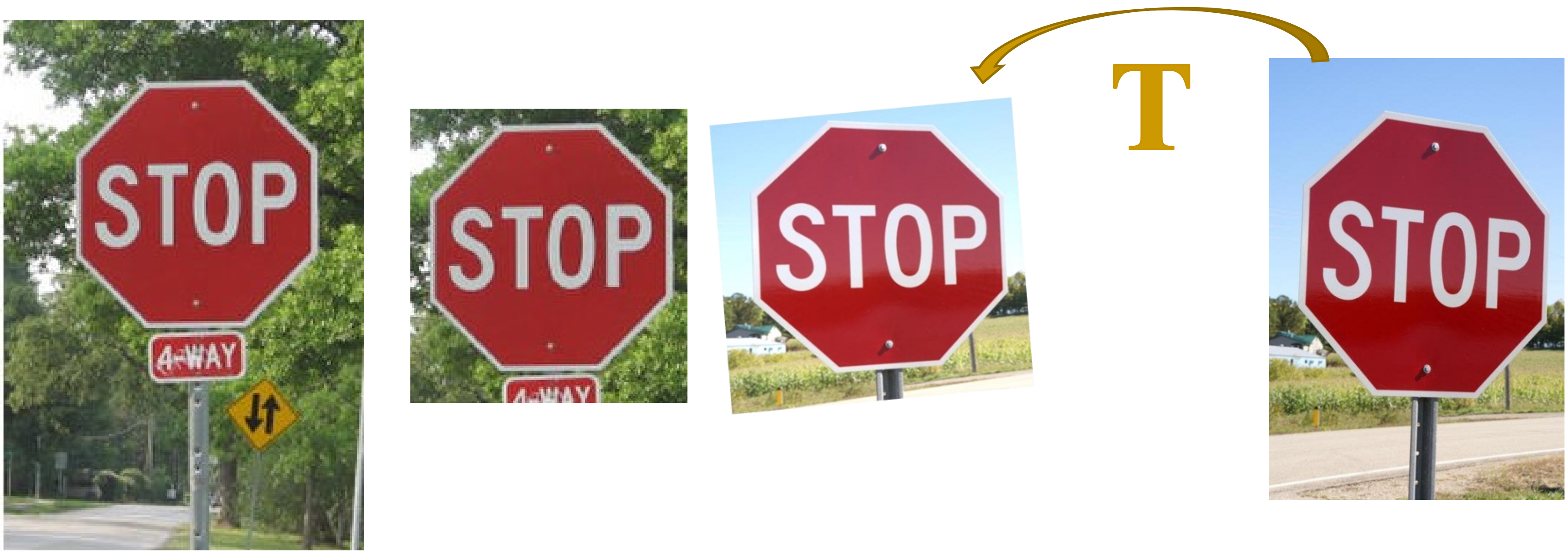
# Generalization power

# Today's class

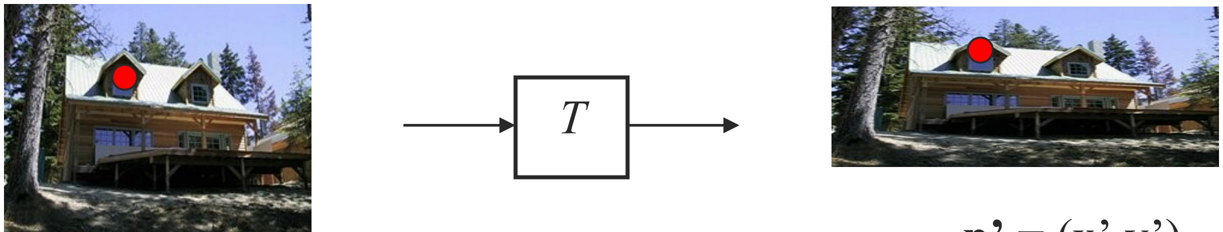
- Introduction to alignment
- Alignment methods
  - Global methods
  - Hypothesize and test
- Image Transformation
  - Common image transformations
  - Examples of solving image alignment
- Homework: Mosaics

# Alignment

**Alignment:** solving the transformation that makes two things match better



# Parametric (global) warping




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

Transformation T is a coordinate-changing machine:

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' = **T**p

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

# Fitting and Alignment

**Fitting**: find the parameters of a model that best fit  the data.

**Alignment**: find the parameters of the transformation that best align matched points.

# Fitting and Alignment

- Design challenges
  - Design a suitable **goodness of fit** measure
    - Similarity should reflect application goals
    - Encode robustness to outliers and noise
  - Design an **optimization** method
    - Avoid local optima
    - Find best parameters quickly

# Fitting and Alignment: Methods

- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Iterative closest point (ICP)

- Hypothesize and test
  - Generalized Hough transform
  - RANSAC

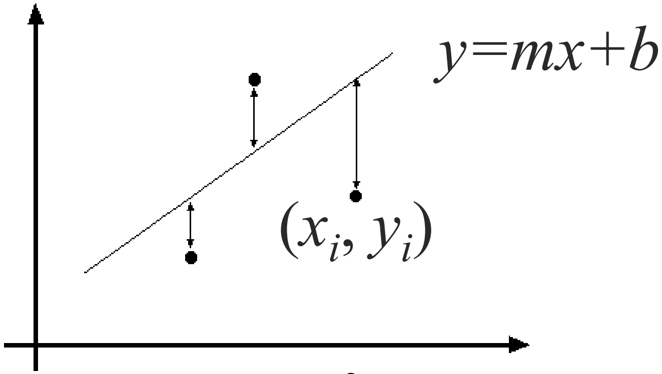
# Today's class

- Introduction to alignment
- Alignment methods
  - **Global methods**
  - Hypothesize and test
- Image Transformation
  - Common image transformations
  - Examples of solving image alignment
- Homework: Mosaics

# Least squares line fitting

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m x_i + b$
- Find $(m, b)$ to minimize

$$\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$$




$$E = \sum_{i=1}^{n} \left( \begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - y_i \right)^2 = \left\| \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\|^2 = \|\mathbf{Ap} - \mathbf{y}\|^2$$

$$= \mathbf{y}^T \mathbf{y} - 2(\mathbf{Ap})^T \mathbf{y} + (\mathbf{Ap})^T (\mathbf{Ap})$$

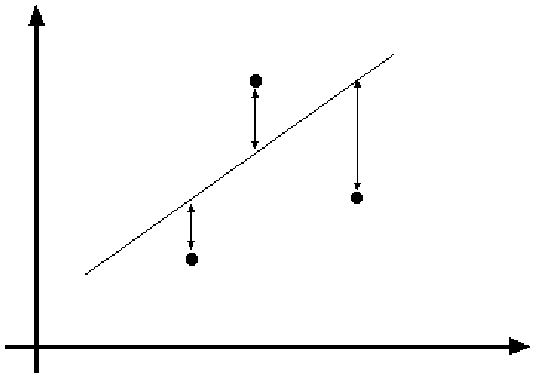
$$\frac{dE}{dp} = 2\mathbf{A}^T \mathbf{Ap} - 2\mathbf{A}^T \mathbf{y} = 0$$

$$\mathbf{A}^T \mathbf{Ap} = \mathbf{A}^T \mathbf{y} \Rightarrow \mathbf{p} = \left(\mathbf{A}^T \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{y}$$

Matlab: `p = A \ y;`

# Problem with “vertical” least squares

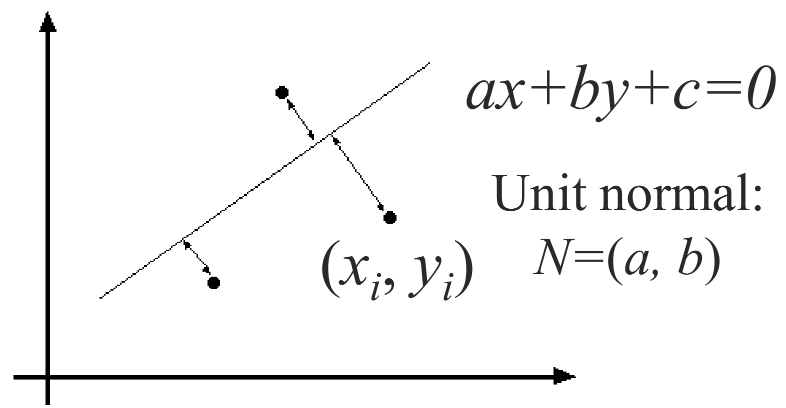
- Not rotation-invariant
- Fails completely for vertical lines



# Total least squares

If ($a^2+b^2=1$) then

Distance between point $(x_i, y_i)$ and line $ax+by+c=0$ is $|ax_i + by_i + c|$

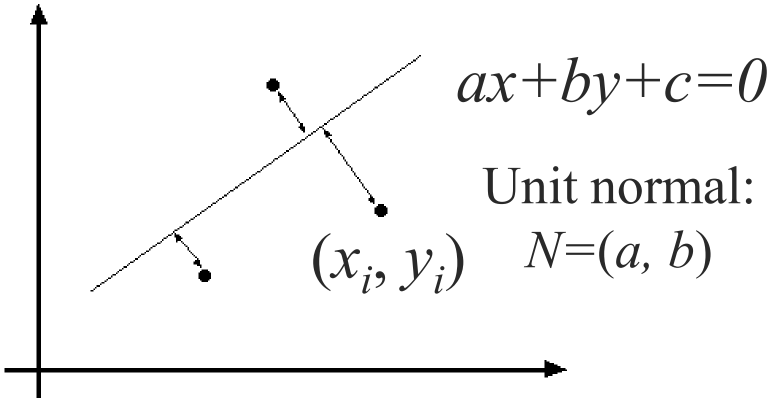



proof:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html

# Total least squares

If ($a^2$+$b^2$=1) then

Distance between point $(x_i, y_i)$ and line $ax+by+c=0$ is $|ax_i + by_i + c|$

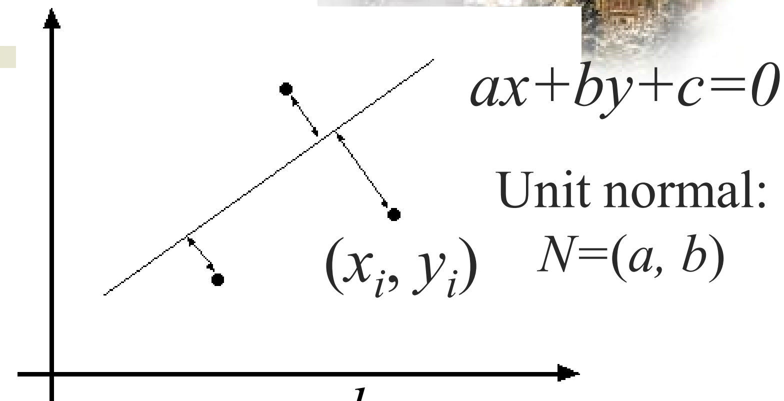



Find $(a, b, c)$ to minimize the sum of squared perpendicular distances

$$E = \sum_{i=1}^{n} (ax_i + by_i + c)^2$$

# Total least squares

Find $(a, b, c)$ to minimize the sum of squared perpendicular distances




$$\boxed{E = \sum_{i=1}^{n} (ax_i + by_i + c)^2}$$

$$\frac{\partial E}{\partial c} = \sum_{i=1}^{n} 2(ax_i + by_i + c) = 0 \qquad c = -\frac{a}{n}\sum_{i=1}^{n} x_i - \frac{b}{n}\sum_{i=1}^{n} y_i = -a\bar{x} - b\bar{y}$$

$$E = \sum_{i=1}^{n} (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = \mathbf{p}^T \mathbf{A}^T \mathbf{A}\mathbf{p}$$

$$\text{minimize } \mathbf{p}^T \mathbf{A}^T \mathbf{A}\mathbf{p} \quad \text{s.t. } \mathbf{p}^T\mathbf{p} = 1 \quad \Rightarrow \quad \text{minimize } \frac{\mathbf{p}^T \mathbf{A}^T \mathbf{A}\mathbf{p}}{\mathbf{p}^T\mathbf{p}}$$

Solution is eigenvector corresponding to smallest eigenvalue of $A^TA$

See details on Raleigh Quotient: http://en.wikipedia.org/wiki/Rayleigh_quotient

# Recap: Two Common Optimization Problems

## Problem statement

minimize $\|\mathbf{Ax}-\mathbf{b}\|^2$

least squares solution to $\mathbf{Ax}=\mathbf{b}$

## Solution

$$\mathbf{x}=\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$$

$\mathbf{x}=\mathbf{A}\backslash\mathbf{b}$ (matlab)

## Problem statement

minimize $\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}$ s.t. $\mathbf{x}^T\mathbf{x}=1$

minimize $\dfrac{\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}}{\mathbf{x}^T\mathbf{x}}$

non - trivial lsq solution to $\mathbf{Ax}=0$

## Solution

$[\mathbf{v},\lambda]=\mathrm{eig}(\mathbf{A}^T\mathbf{A})$

$\lambda_1<\lambda_{2..n}:\mathbf{x}=\mathbf{v}_1$

# Least squares (global) optimization

Good

- Clearly specified objective
- Optimization is easy

Bad

- May not be what you want to optimize
- Sensitive to outliers
  - Bad matches, extra points
- Doesn't allow you to get multiple good fits
  - Detecting multiple objects, lines, etc.

# Other ways to search for parameters

- **Line search**
  1. For each parameter, step through values and choose value that gives best fit
  2. Repeat (1) until no parameter changes
- **Grid search**
  1. Propose several sets of parameters, evenly sampled in the joint set
  2. Choose best (or top few) and sample joint parameters around the current best; repeat
- **Gradient descent**
  1. Provide initial position (e.g., random)
  2. Locally search for better parameters by following gradient

# Today’s class

- Introduction to alignment
- Alignment methods
  - Global methods
  - **Hypothesize and test**
- Image Transformation
  - Common image transformations
  - Examples of solving image alignment
- Homework: Mosaics

# Hypothesize and test

1. Propose parameters
   - Try all possible
   - Each point votes for all consistent parameters
   - Repeatedly sample enough points to solve for parameters
2. Score the given parameters
   - Number of consistent points, possibly weighted by distance
3. Choose from among the set of parameters
   - Global or local maximum of scores
4. Possibly refine parameters using inliers

# Hough Transform: Outline

1. Create a grid of parameter values

2. Each point votes for a set of parameters, incrementing those values in grid

3. Find maximum or local maxima in grid

# Hough Space

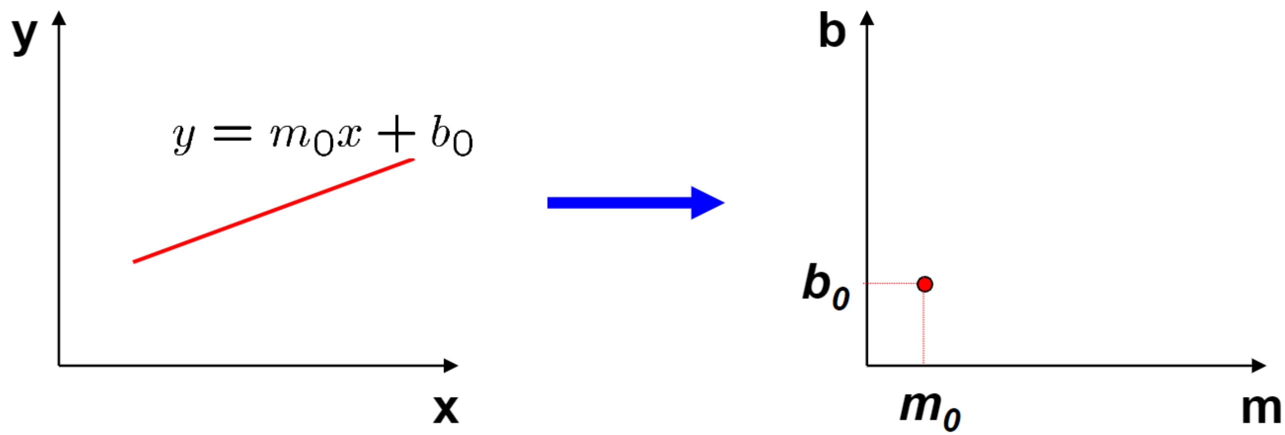
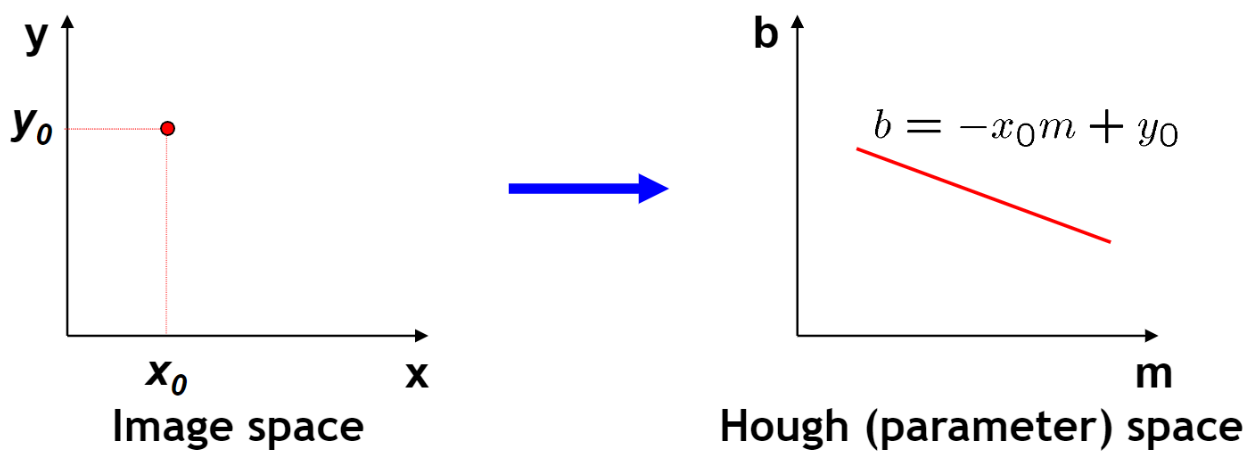



Image space

Hough (parameter) space

- **Connection between image $(x,y)$ and Hough $(m,b)$ spaces**
  - A line in the image corresponds to a point in Hough space.
  - To go from image space to Hough space:
    - Given a set of points $(x,y)$, find all $(m,b)$ such that $y = mx + b$

# Hough Space

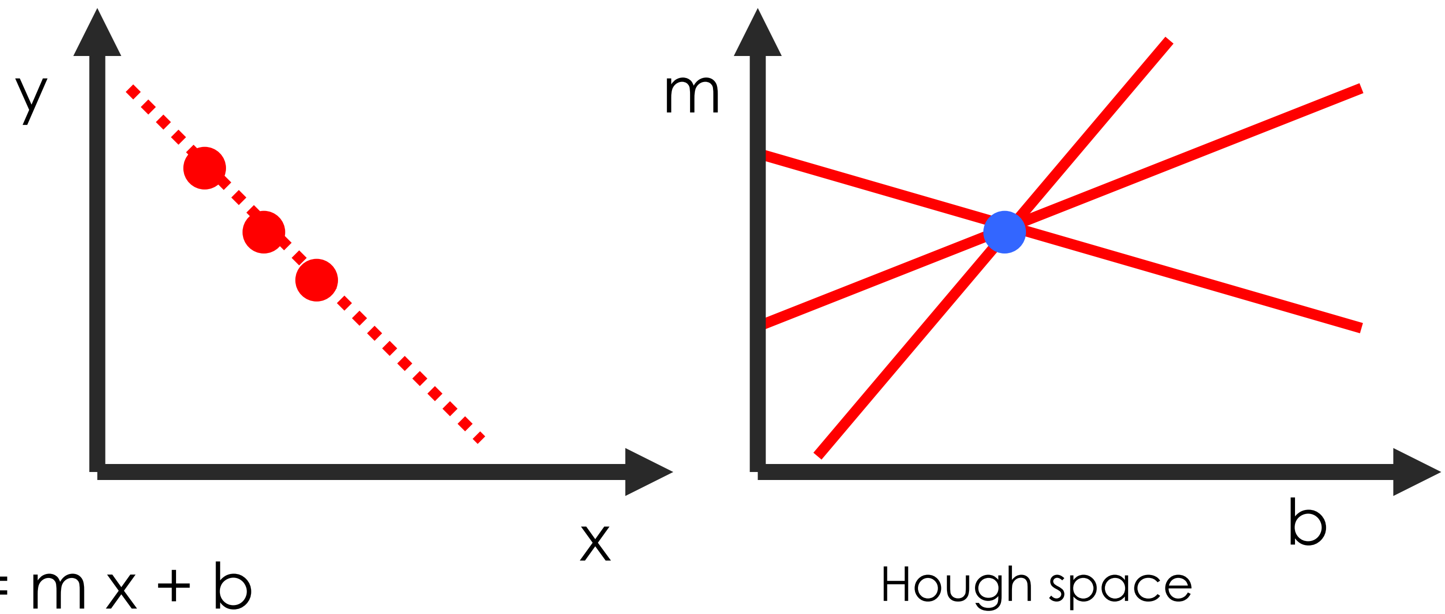



- Connection between image $(x,y)$ and Hough $(m,b)$ spaces
  - A line in the image corresponds to a point in Hough space.
  - To go from image space to Hough space:
    - Given a set of points $(x,y)$, find all $(m,b)$ such that $y = mx + b$
  - What does a point $(x_0, y_0)$ in the image space map to?
    - Answer: the solutions of $b = -x_0 m + y_0$
    - This is a line in Hough space.

# Hough transform

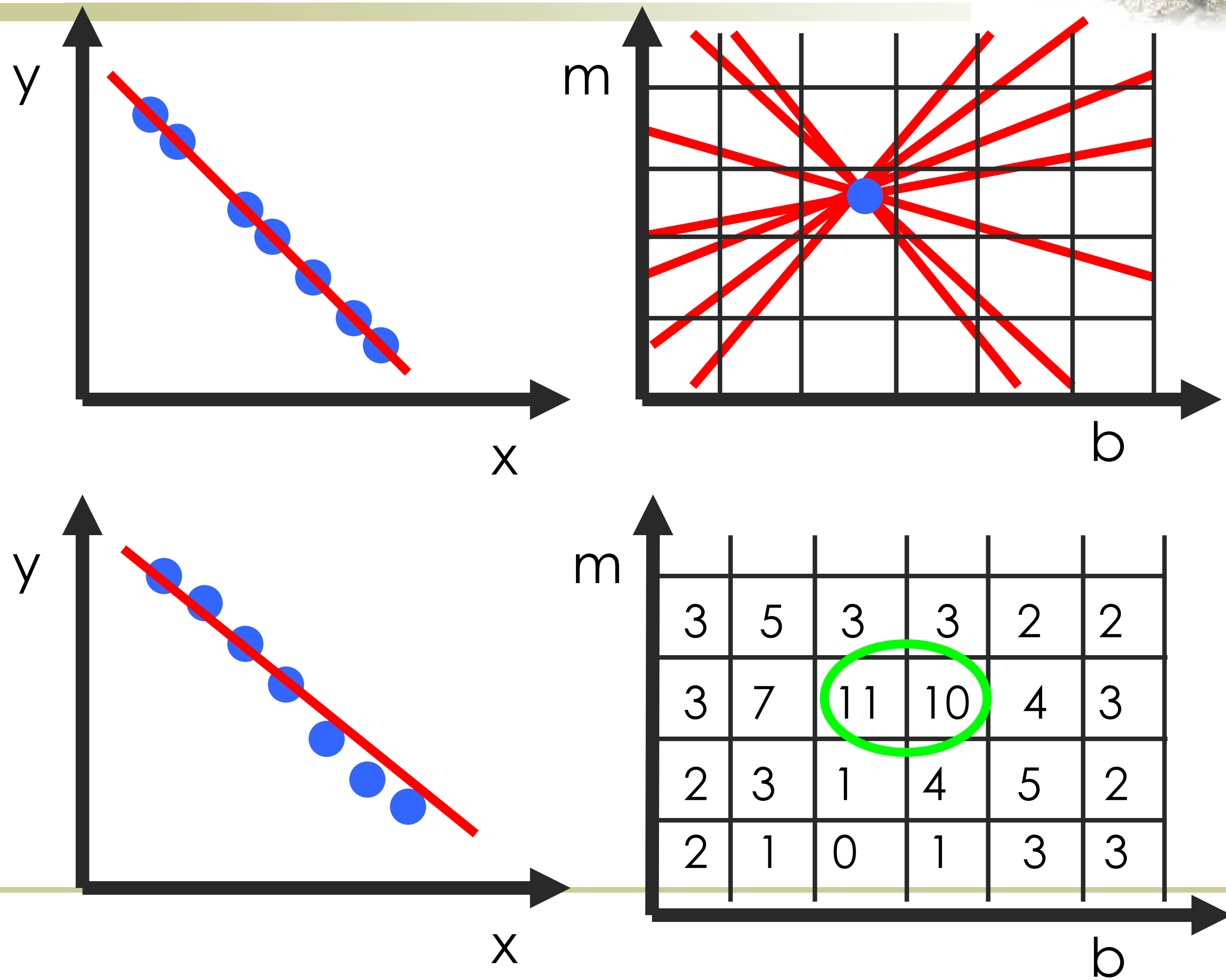
P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Given a set of points, find the curve or line that explains the data points best




$y = m x + b$

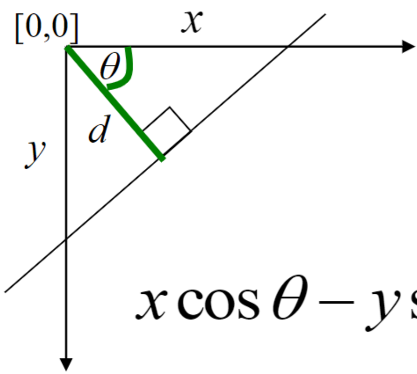
# Hough transform




Slide from S. Savarese

# Polar Representation for Lines

- **Issues with usual $(m,b)$ parameter space: can take on infinite values, undefined for vertical lines.**

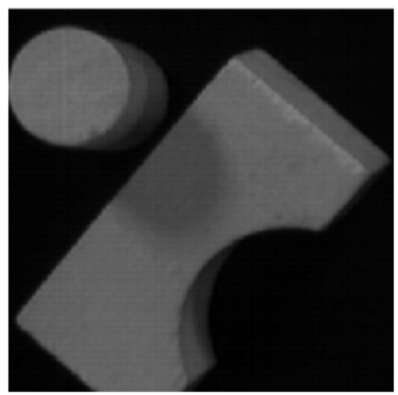


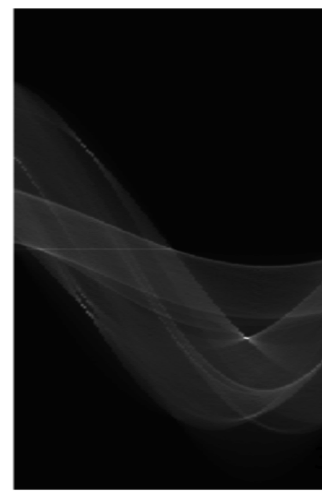

$$x\cos\theta - y\sin\theta = d$$

$d$ **: perpendicular distance from line to origin**

$\theta$ **: angle the perpendicular makes with the x-axis**

- **Point in image space**
  **$\Rightarrow$ Sinusoid segment in Hough space**





Slide adapted from Steve Seitz

# Hough Transform Algorithm

**Using the polar parameterization:**

$$x\cos\theta + y\sin\theta = d$$

**Basic Hough transform algorithm**

1. **Initialize** $H[d,\theta] = 0$.
2. **For each edge point** $(x,y)$ **in the image**

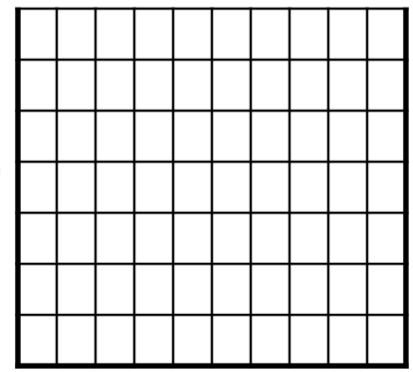
   **for** $\theta = 0$ **to** $180$ // some quantization

   $d = x\cos\theta + y\sin\theta$

   H[d, θ] += 1
3. **Find the value(s) of** $(d,\theta)$ **where** $H[d,\theta]$ **is maximal.**
4. **The detected line in the image is given by** $d = x\cos\theta + y\sin\theta$

$H$: accumulator array (votes)

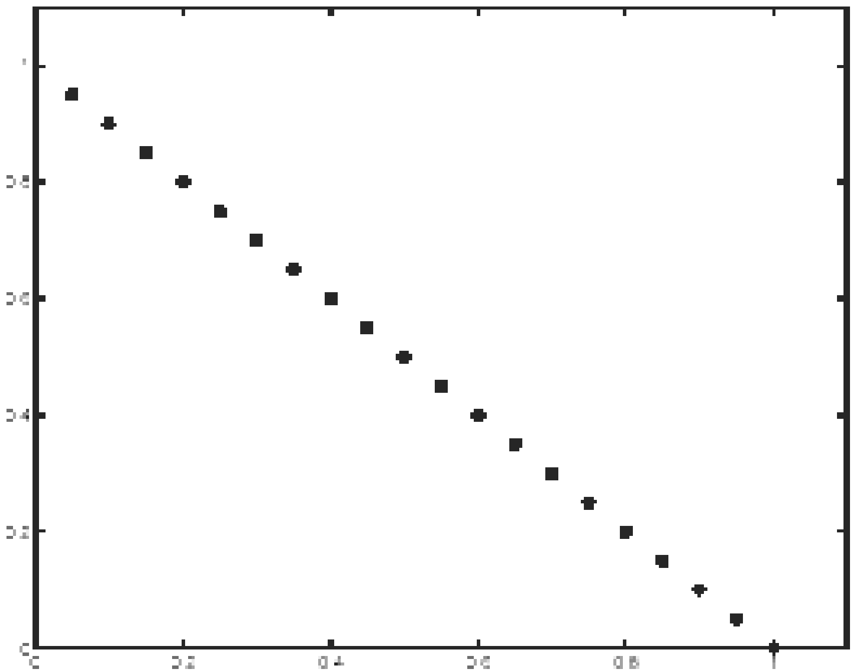
$\theta$



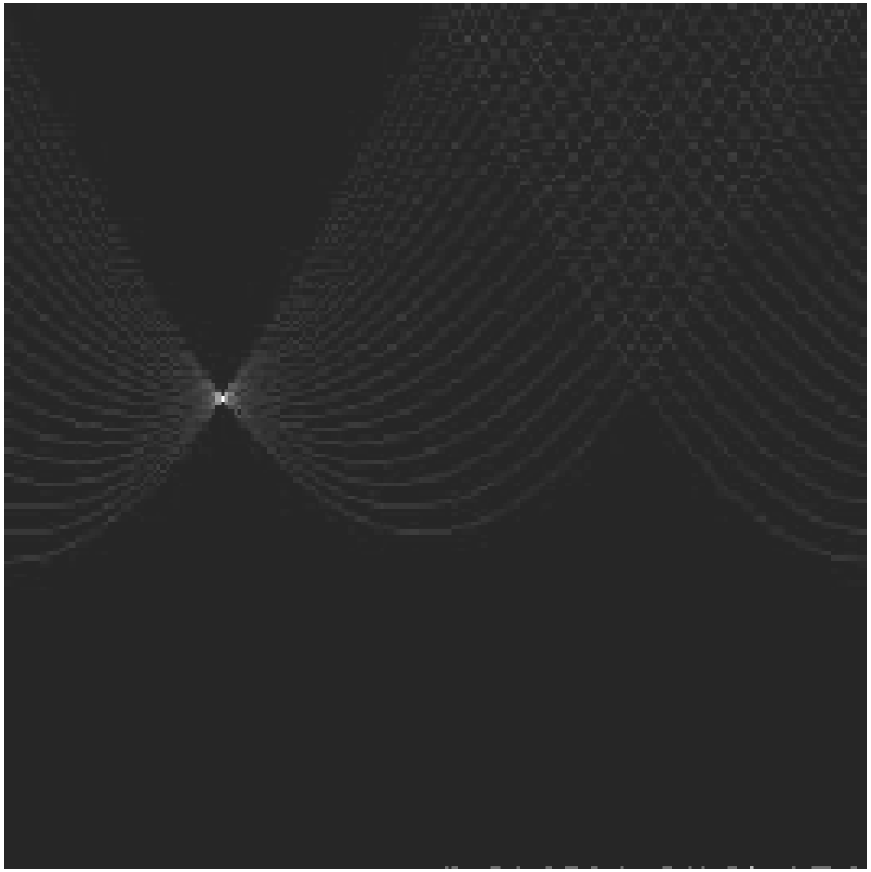
$d$

- **Time complexity (in terms of number of votes)?**

# Hough transform - experiments



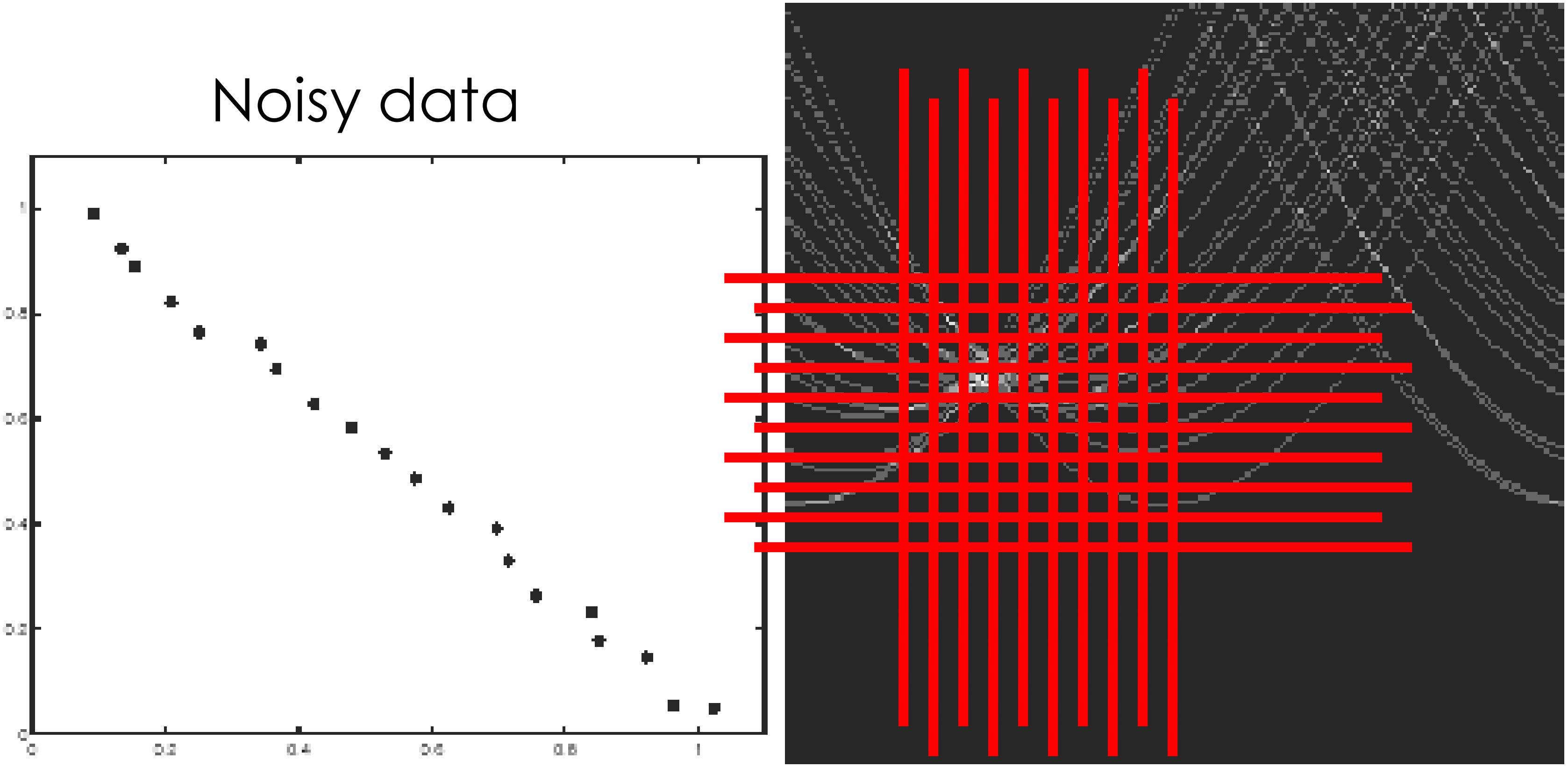
features



votes

# Hough transform - experiments

Noisy data



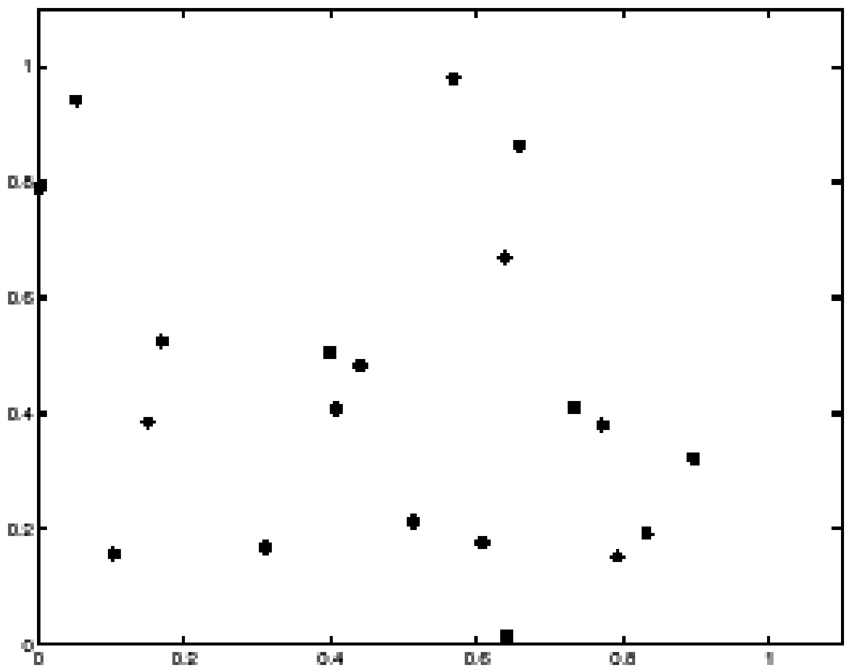
features

votes

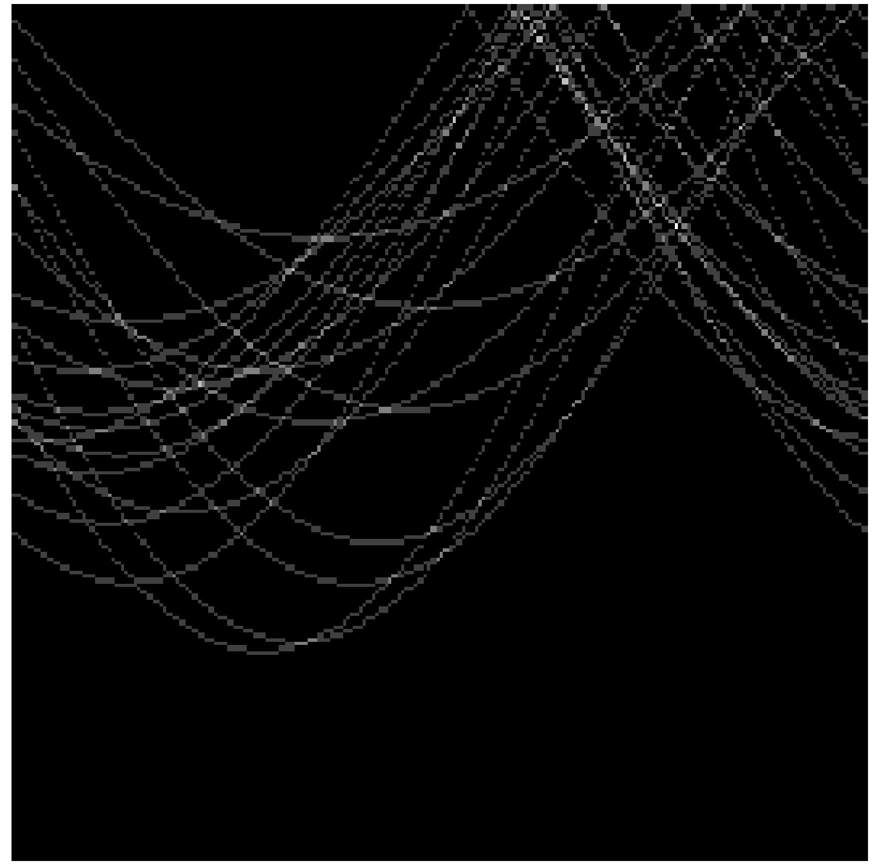
Need to adjust grid size or smooth

Slide from S. Savarese

# Hough transform - experiments




features



votes

Issue: spurious peaks due to uniform noise

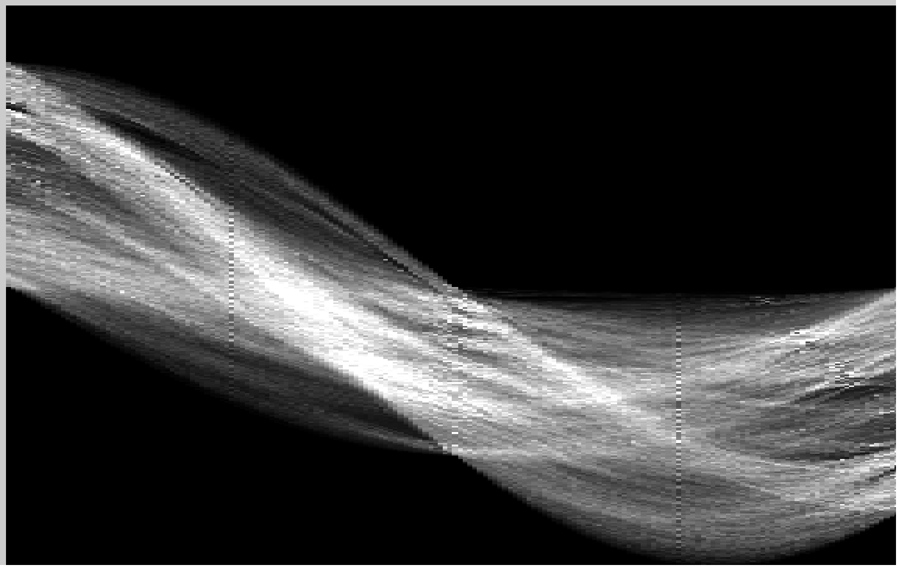
Slide from S. Savarese

# 1. Image → Canny

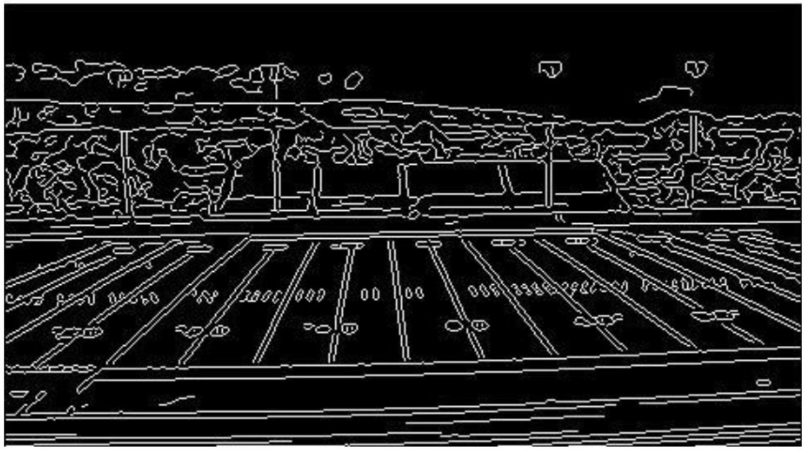
# 2. Canny → Hough votes

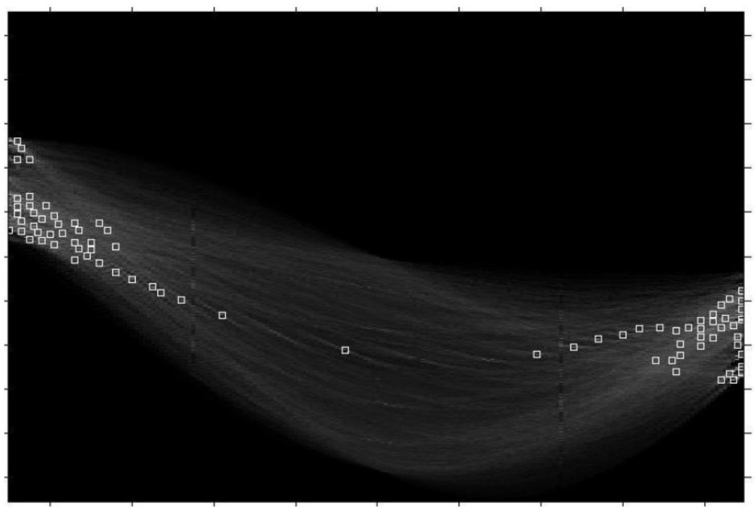
# 3. Hough votes → Edges

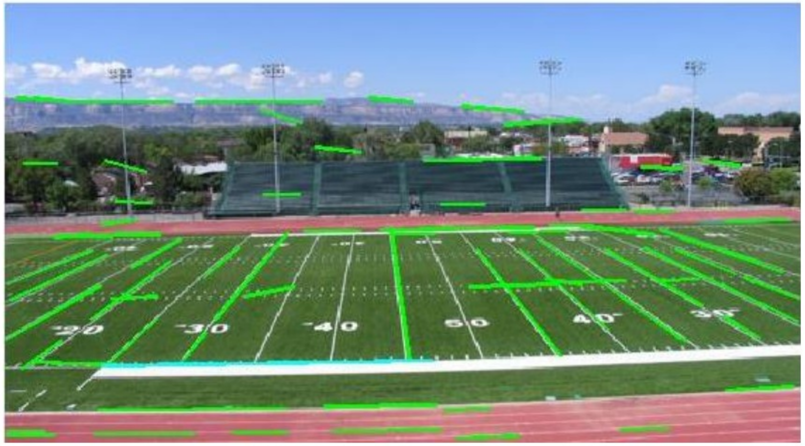
Find peaks and post-process



# Real-World Example







Showing longest segments found

# Hough transform: pros and cons

## Pros

- All points are processed independently, so can cope with occlusion, gaps
- Some robustness to noise: noise points unlikely to contribute *consistently* to any single bin
- Can detect multiple instances of a model in a single pass

## Cons

- Complexity of search time increases exponentially with the number of model parameters
- Non-target shapes can produce spurious peaks in parameter space
- Quantization: can be tricky to pick a good grid size

# Hough transform conclusions

## Common applications

- Line fitting (also circles, ellipses, etc.)
- Object instance recognition (parameters are position/scale/orientation)
- Object category recognition (parameters are position/scale)

# RANSAC

- RANdom Sample Consensus
- **Approach**: we want to avoid the impact of outliers, so let's look for "inliers", and use those only.
- **Intuition**: if an outlier is chosen to compute the current fit, then the resulting line (transformation) won't have much support from rest of the points (matches).

# RANSAC

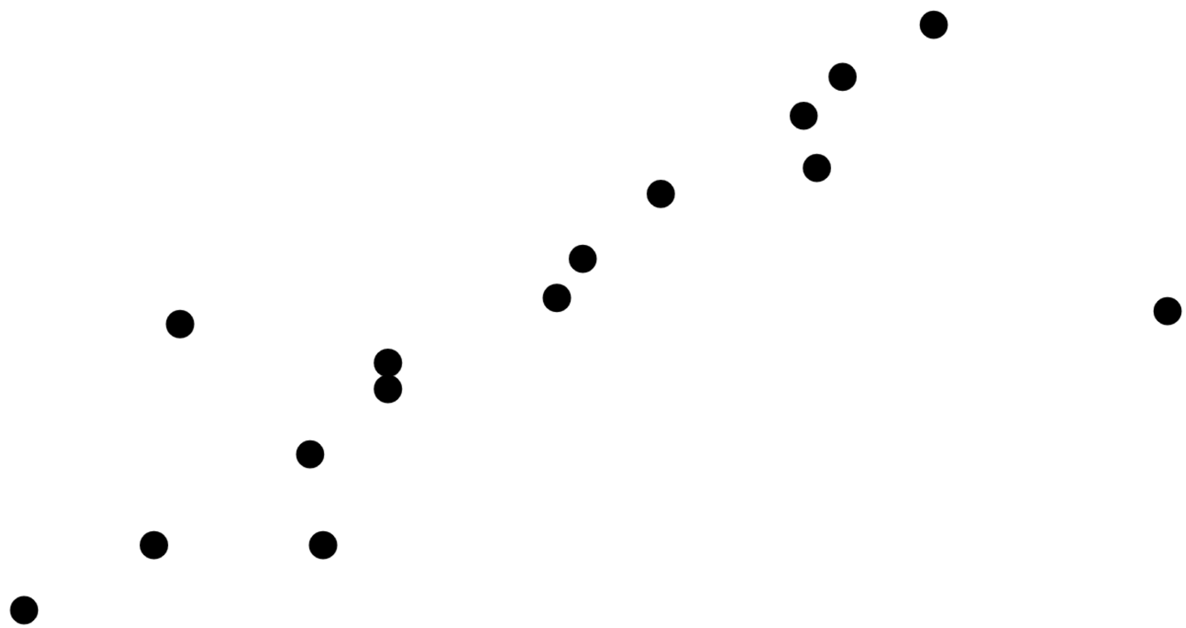
**<u>RANSAC loop</u>:**

1. **Randomly select a *seed group* of points on which to base transformation estimate (e.g., a group of matches)**
2. **Compute transformation from seed group**
3. **Find *inliers* to this transformation**
4. **If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers**

- **Keep the transformation with the largest number of inliers**

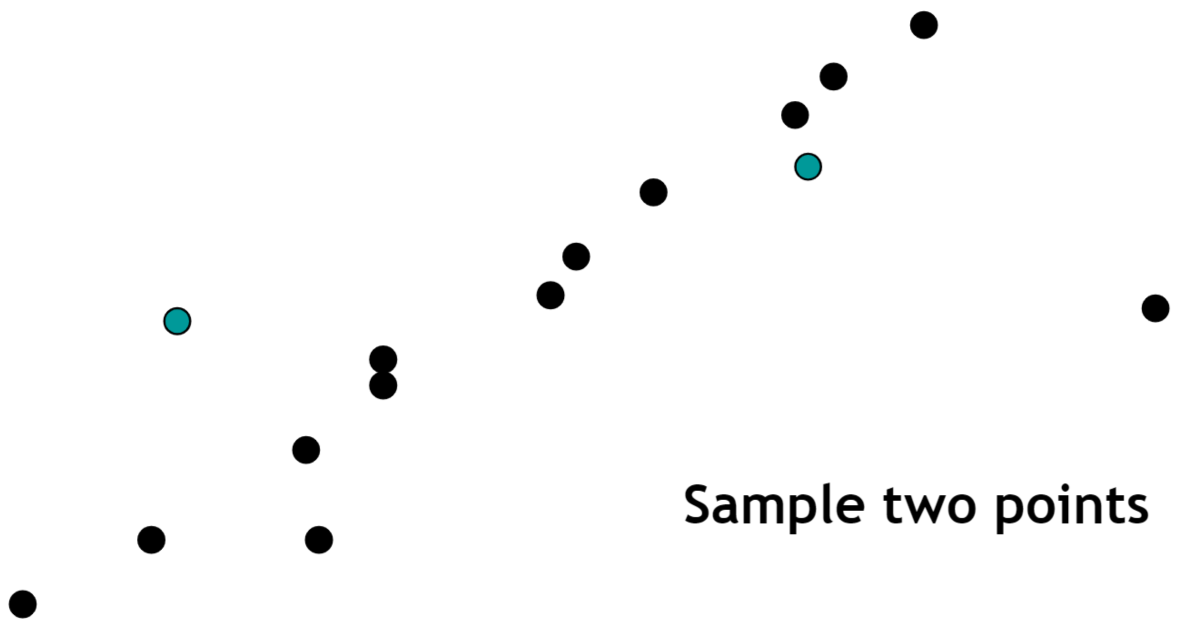
# RANSAC Line Fitting Example

- **Task: Estimate the best line**
  - ***How many points do we need to estimate the line?***



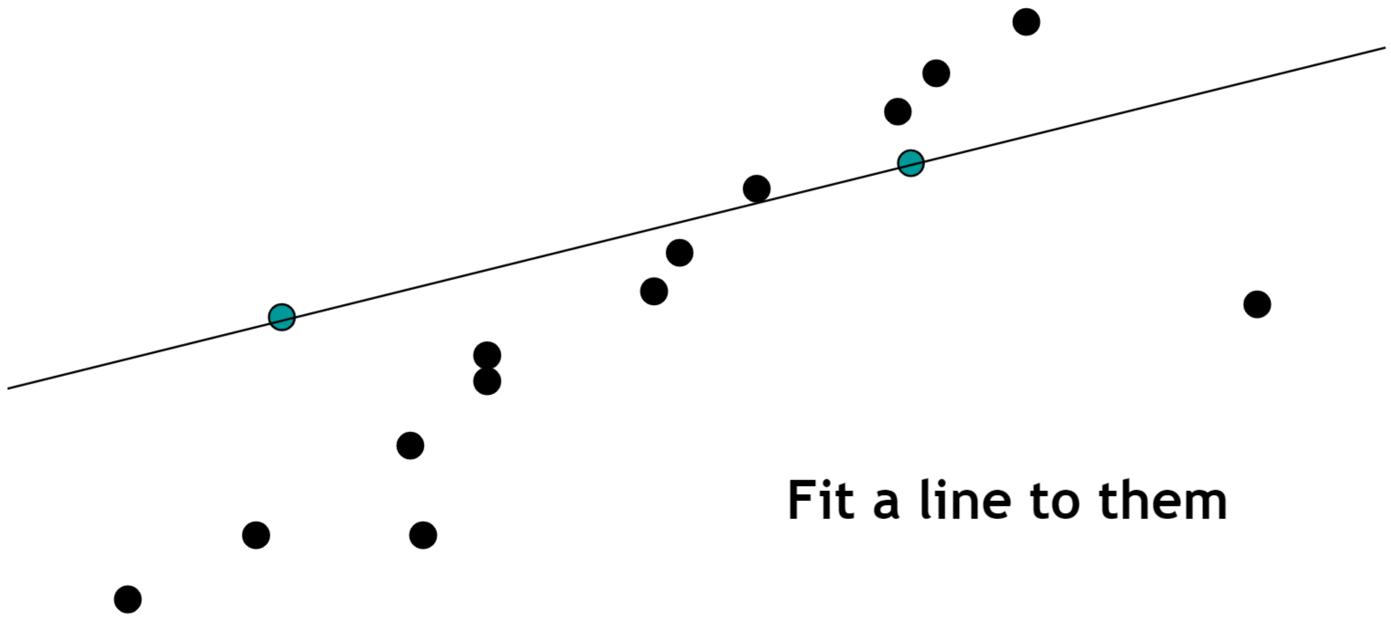
# RANSAC Line Fitting Example

- **Task: Estimate the best line**



# RANSAC Line Fitting Example

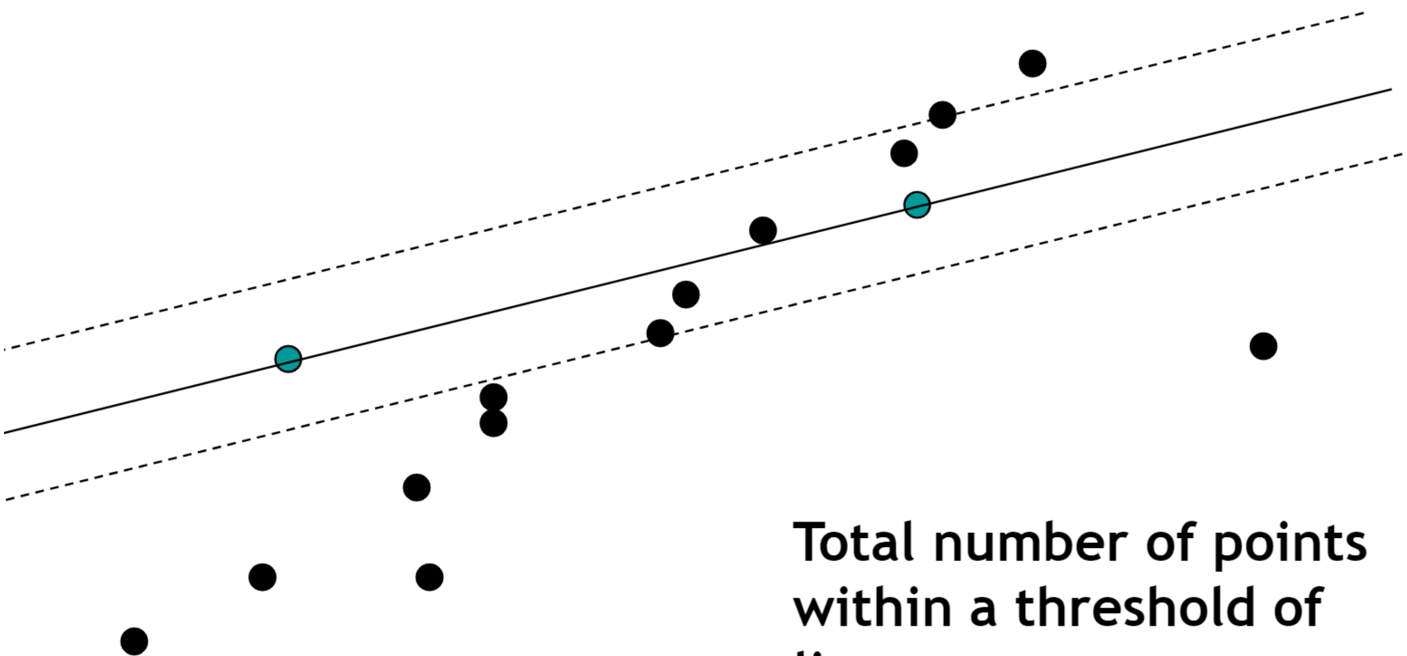
- **Task: Estimate the best line**



**Fit a line to them**

# RANSAC Line Fitting Example

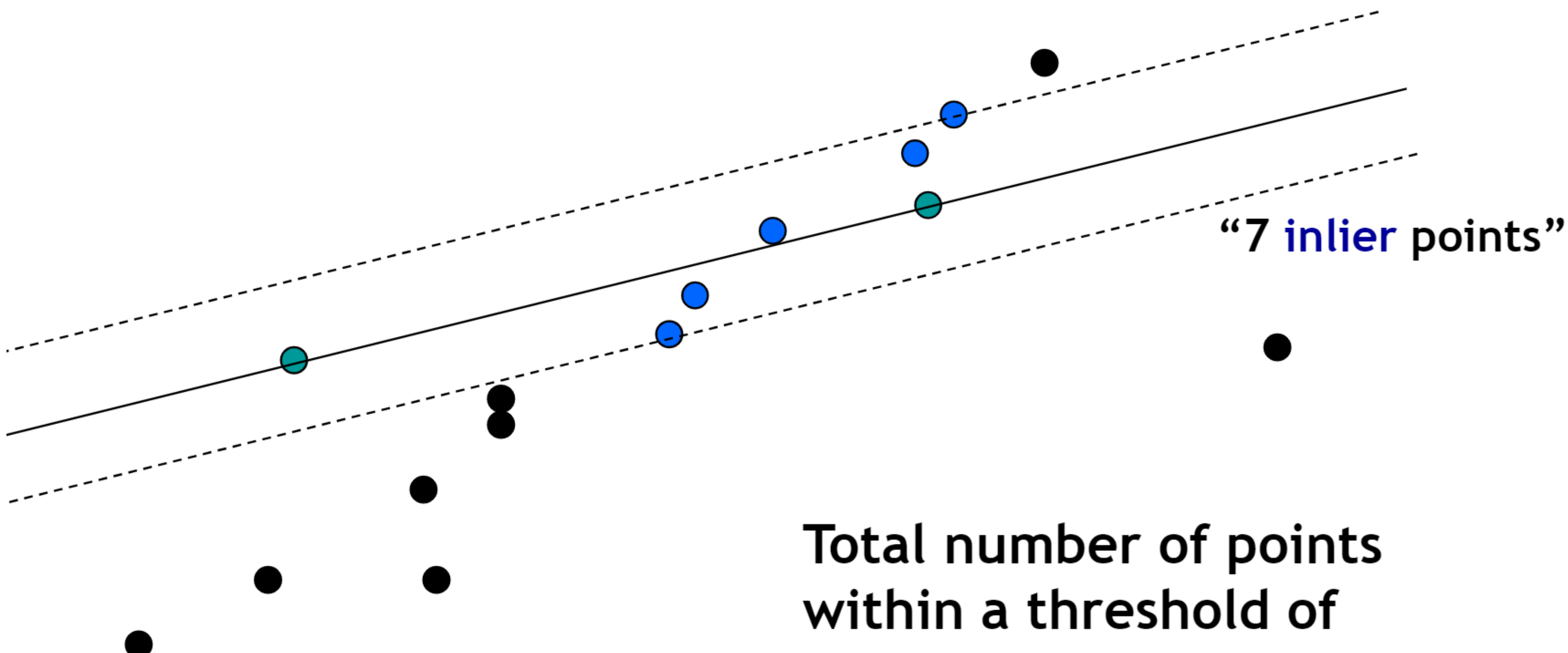
- **Task: Estimate the best line**



**Total number of points within a threshold of line.**

# RANSAC Line Fitting Example

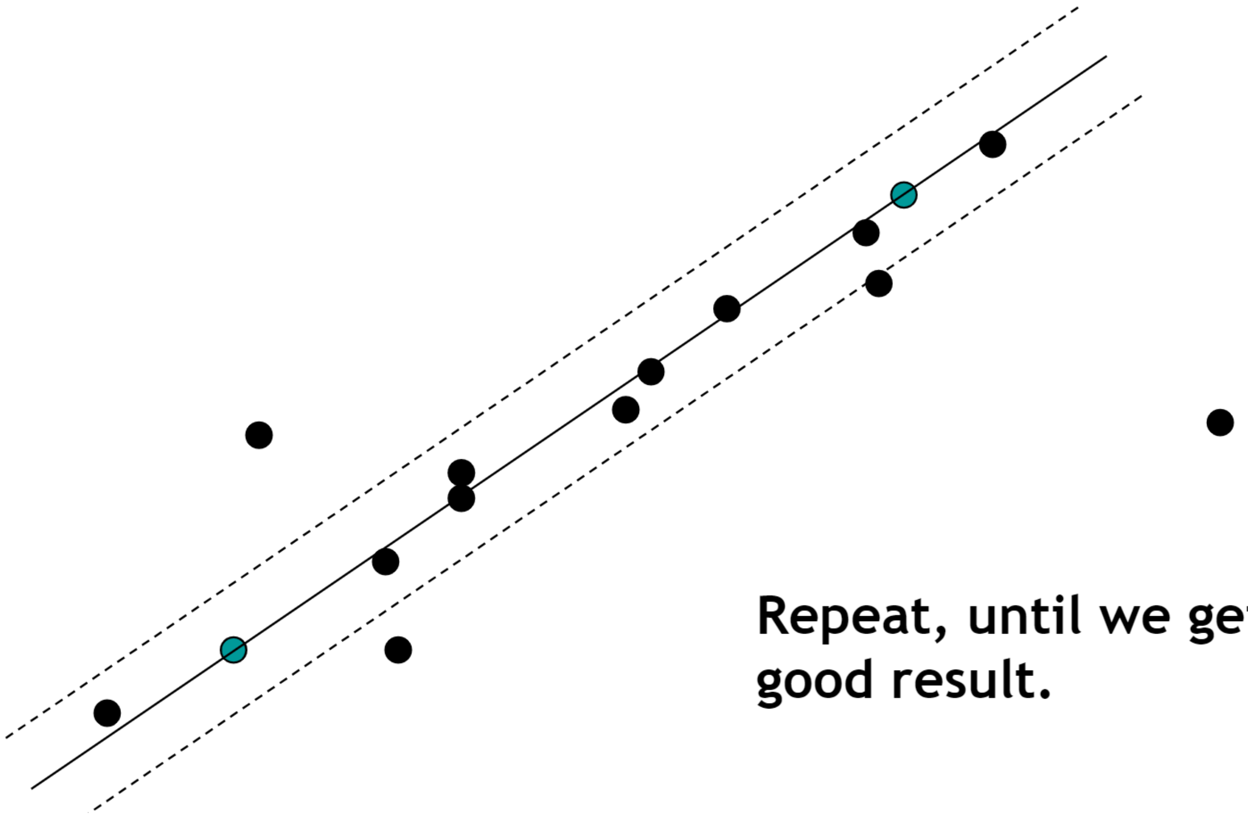
- **Task: Estimate the best line**




**Total number of points within a threshold of line.**

# RANSAC Line Fitting Example

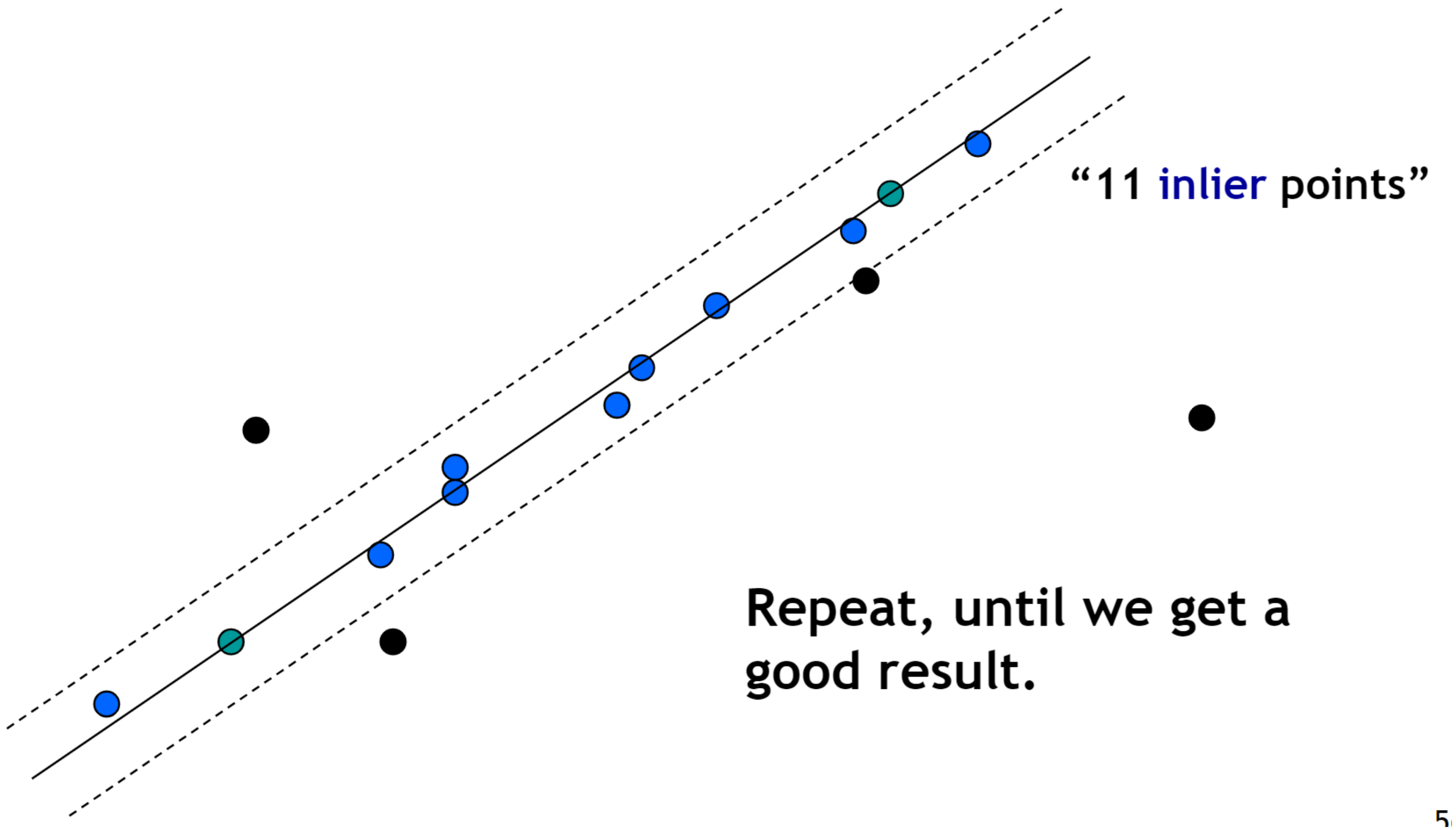
- **Task: Estimate the best line**



**Repeat, until we get a good result.**

# RANSAC Line Fitting Example

- **Task: Estimate the best line**




**Repeat, until we get a good result.**

# RANSAC: How many samples?

- **How many samples are needed?**
  - Suppose $w$ is fraction of inliers (points from line).
  - $n$ points needed to define hypothesis (2 for lines)
  - $k$ samples chosen.
- **Prob. that a single sample of $n$ points is correct:** $w^n$
- **Prob. that all $k$ samples fail is:** $(1-w^n)^k$

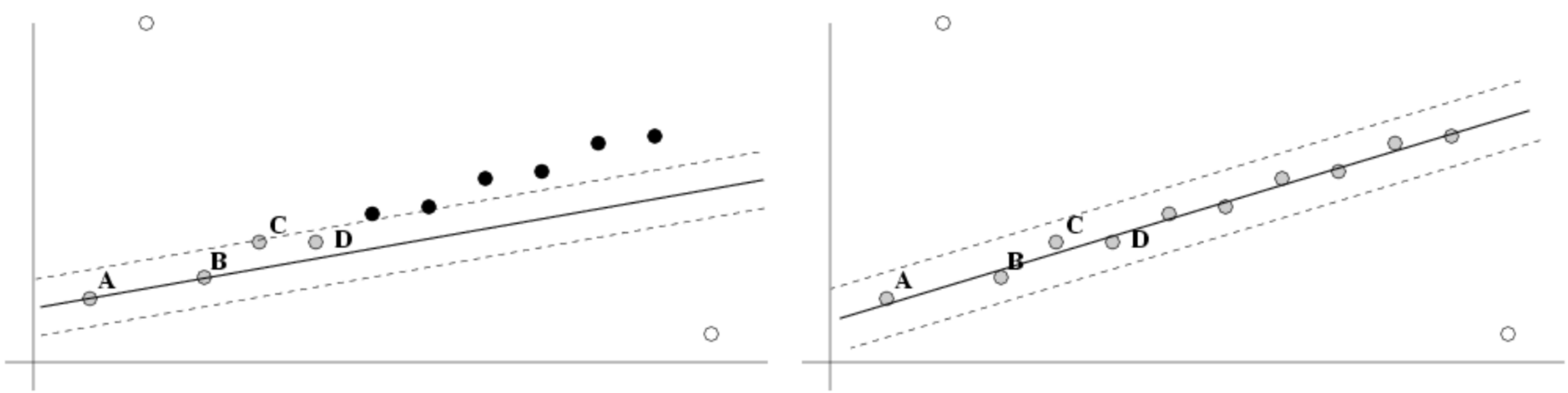
$\Rightarrow$ **Choose $k$ high enough to keep this below desired failure rate.**

# RANSAC: Computed k (p=0.99)

| Sample size n | Proportion of outliers 5% | 10% | 20% | 25% | 30% | 40% | 50% |
|---|---|---|---|---|---|---|---|
| 2 | 2 | 3 | 5 | 6 | 7 | 11 | 17 |
| 3 | 3 | 4 | 7 | 9 | 11 | 19 | 35 |
| 4 | 3 | 5 | 9 | 13 | 17 | 34 | 72 |
| 5 | 4 | 6 | 12 | 17 | 26 | 57 | 146 |
| 6 | 4 | 7 | 16 | 24 | 37 | 97 | 293 |
| 7 | 4 | 8 | 20 | 33 | 54 | 163 | 588 |
| 8 | 5 | 9 | 26 | 44 | 78 | 272 | 1177 |

# After RANSAC

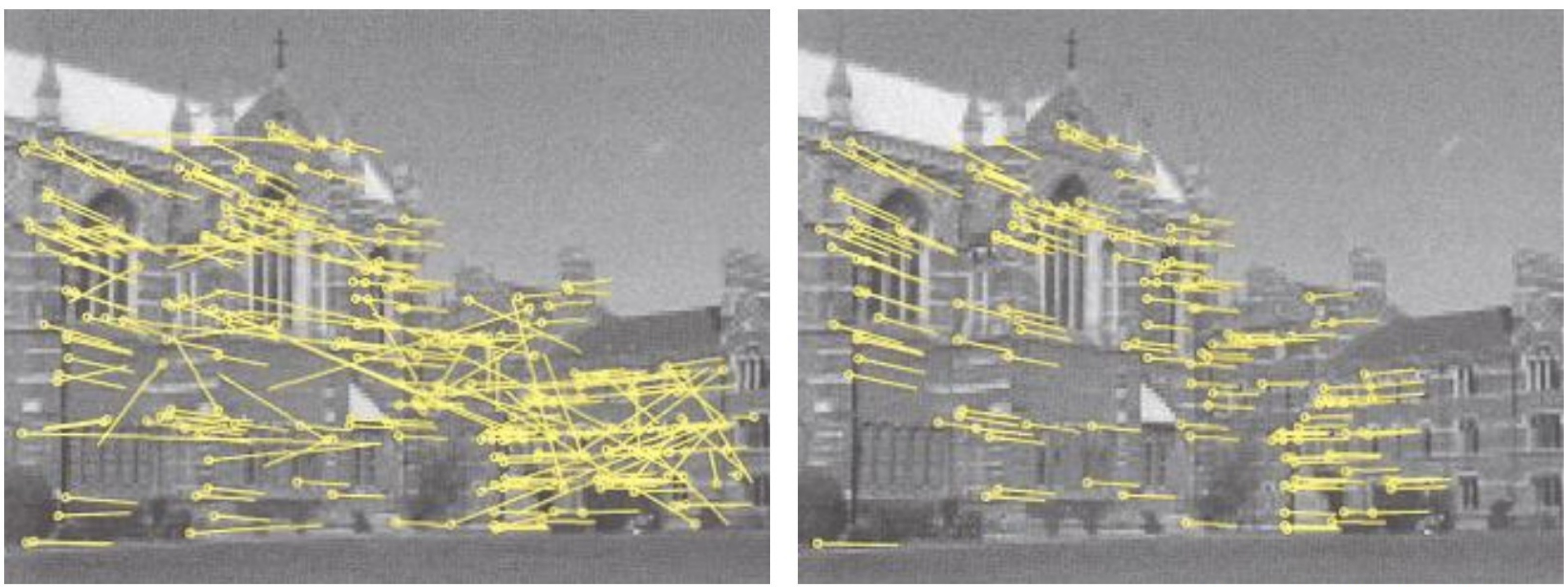
- RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers.
- Improve this initial estimate with estimation over all inliers (e.g. with standard least-squares minimization).
- But this may change inliers, so alternate fitting with re-classification as inlier/outlier.



# RANSAC Example

- Find best stereo match within a square search window (here 300 pixels$^2$)
- Global transformation model: epipolar geometry

before RANSAC after RANSAC



Images from Hartley & Zisserman

# RANSAC conclusions

Good

- Robust to outliers
- Applicable for larger number of objective function parameters than Hough transform
- Optimization parameters are easier to choose than Hough transform

Bad

- Computational time grows quickly with fraction of outliers and number of parameters
- Not as good for getting multiple fits (though one solution is to remove inliers after each fit and repeat)

Common applications

- **Computing a homography (e.g., image stitching)**
- Estimating fundamental matrix (relating two views)

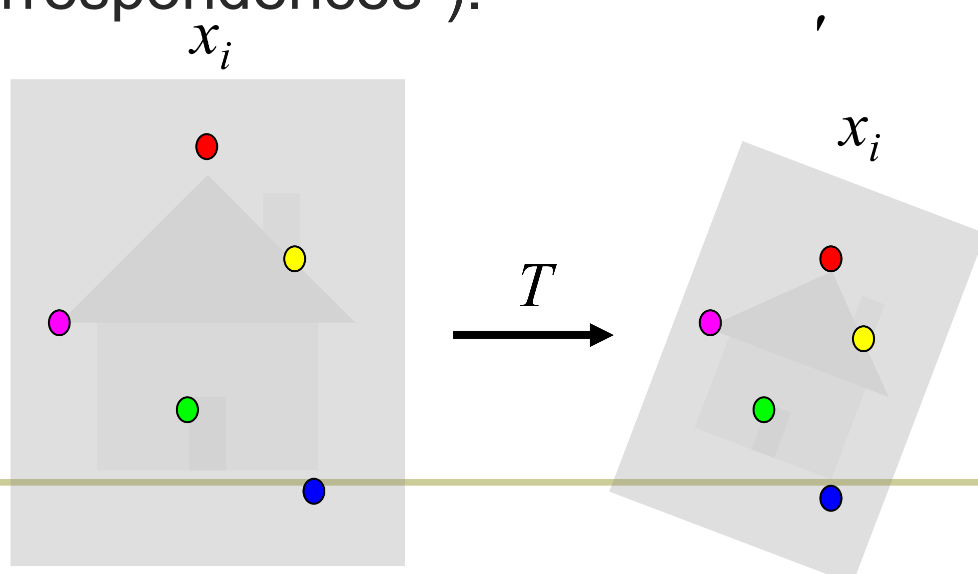
# Today's class

- Introduction to alignment
- Alignment methods
  - Global methods
  - Hypothesize and test
- Image Transformation
  - **Common image transformations**
  - Examples of solving image alignment
- Homework: Mosaics

# Alignment problem

- We have previously considered how to **fit a model to image evidence**
  - e.g., a line to edge points

- In alignment, we will fit the parameters of some **transformation** according to a set of matching feature pairs (“correspondences”).



# Common transformations

original

**Transformed**

translation

rotation

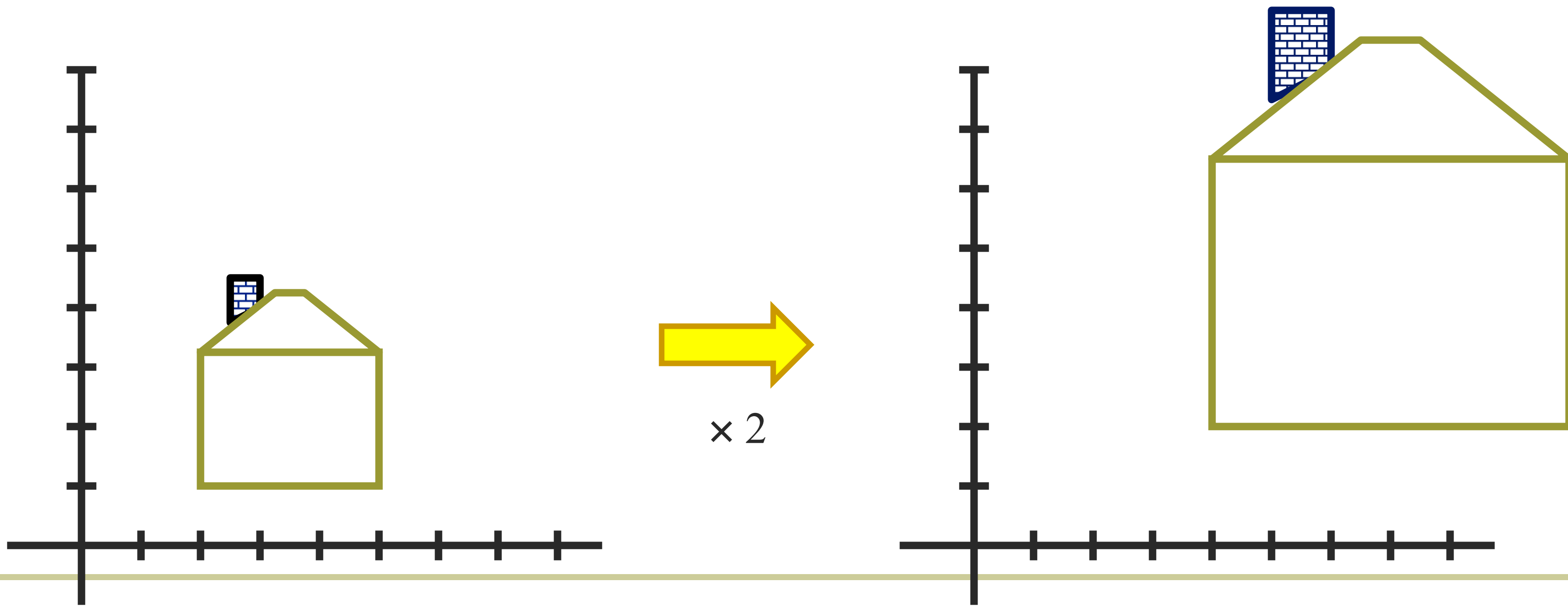
aspect

affine

perspective

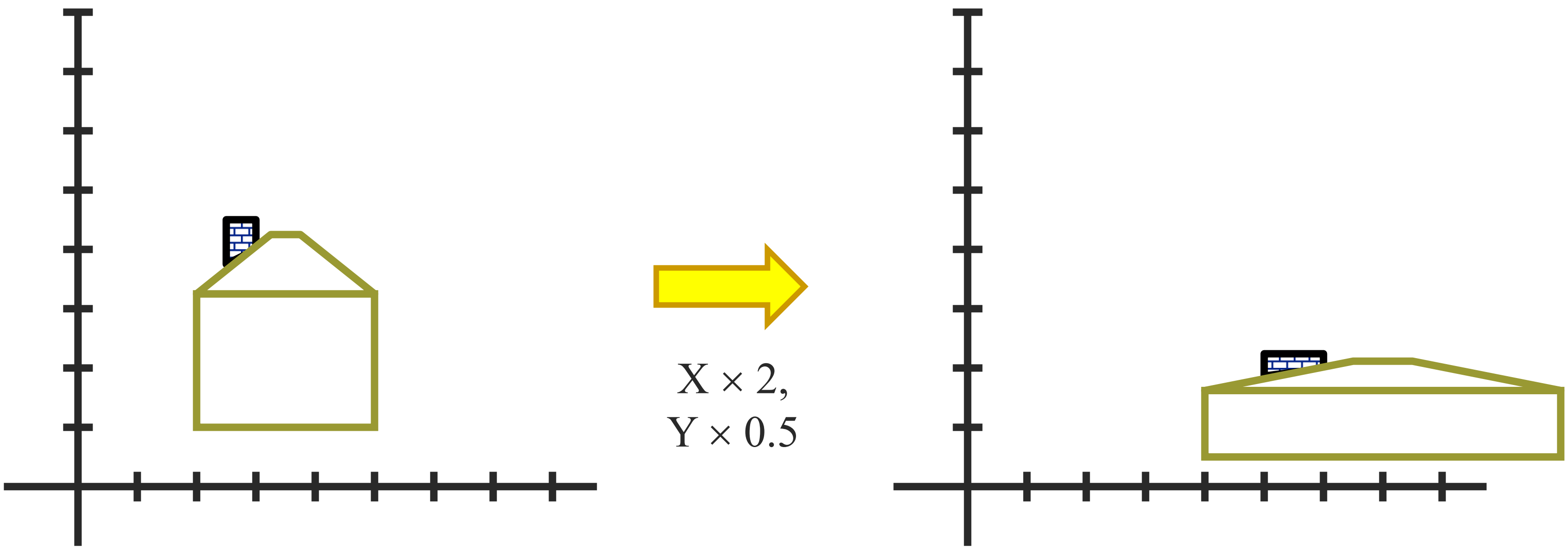
# Scaling

- *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

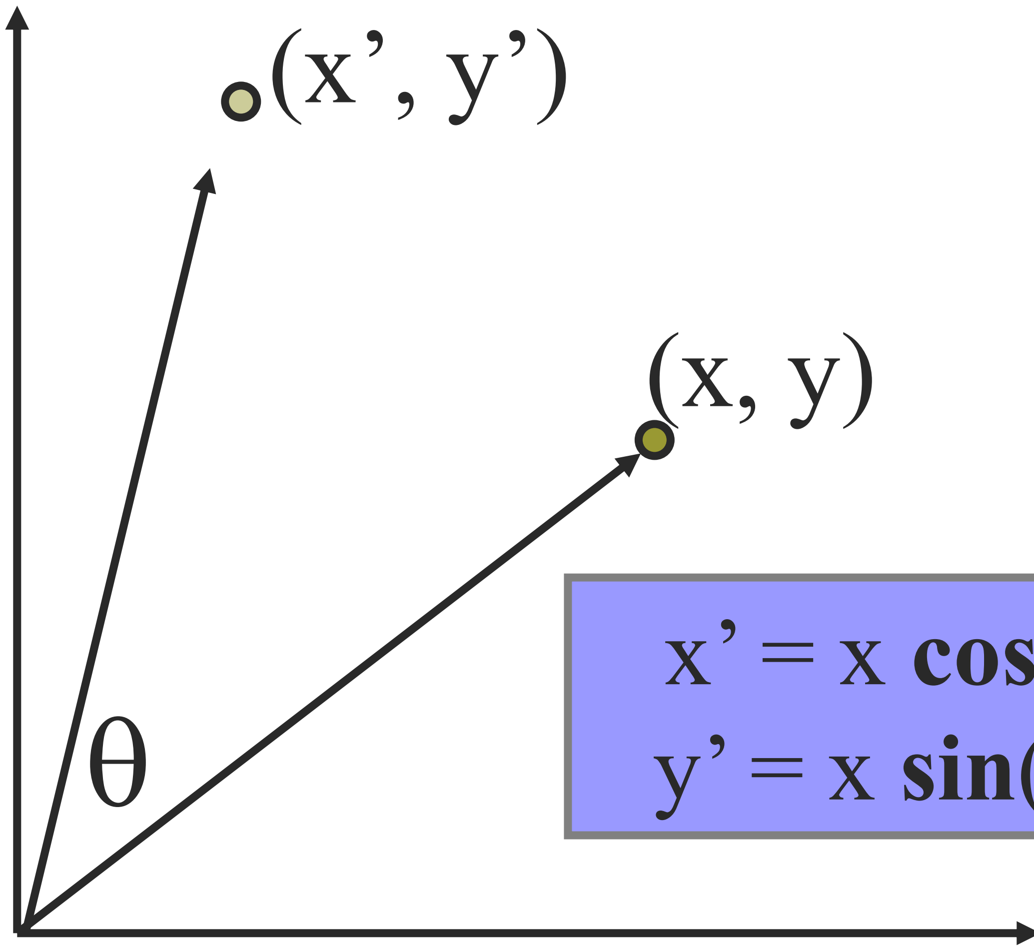
- Scaling operation: $x' = ax$

  $y' = by$

- Or, in matrix form:

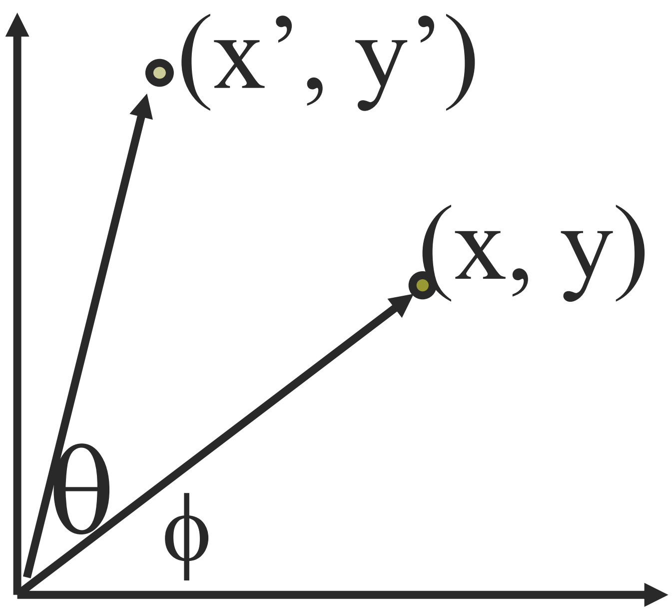
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\text{scaling matrix } S} \begin{bmatrix} x \\ y \end{bmatrix}$$

# 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…

$$x' = x\,\mathbf{cos}(\theta) - y\,\mathbf{sin}(\theta)$$
$$y' = x\,\mathbf{sin}(\theta) + y\,\mathbf{cos}(\theta)$$

# 2-D Rotation

This is easy to capture in matrix form:

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

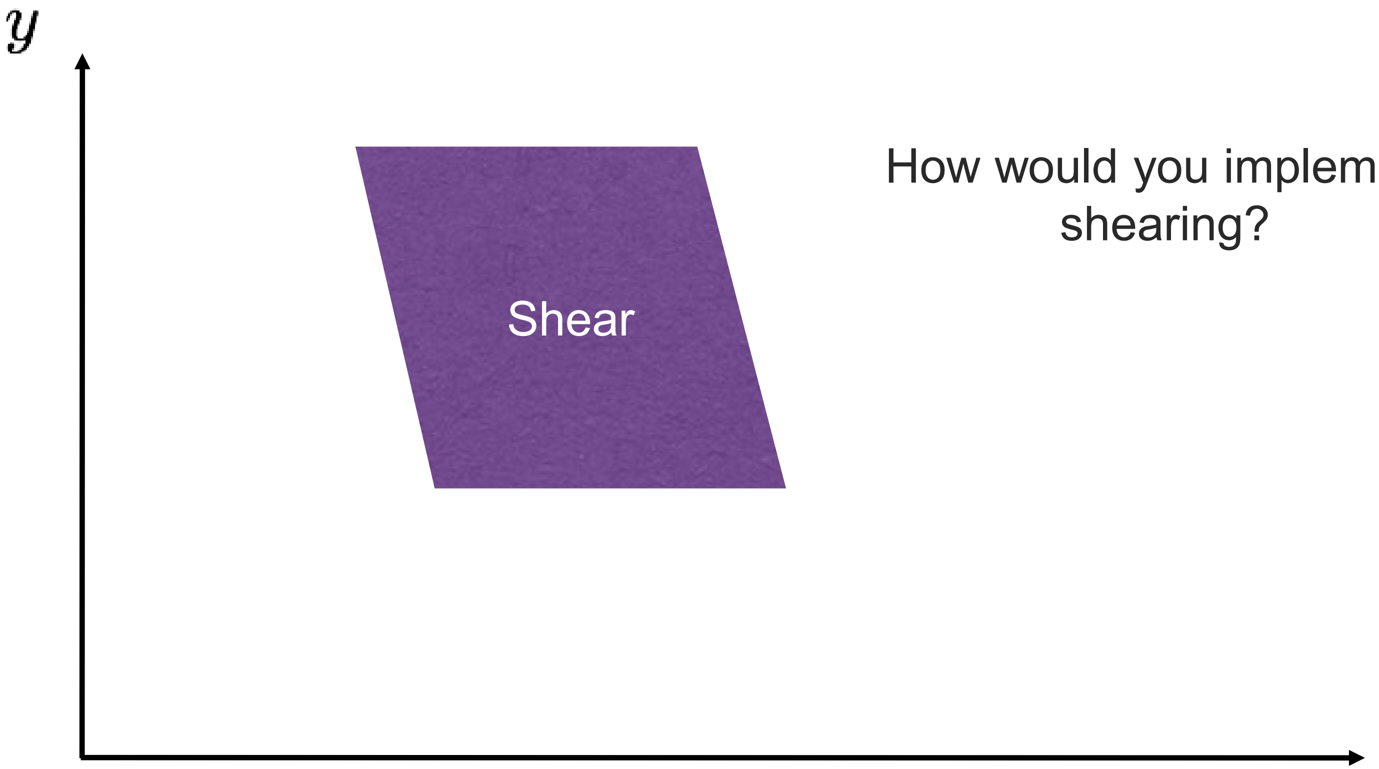
Even though sin(θ) and cos(θ) are nonlinear functions of θ,

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

What is the inverse transformation?

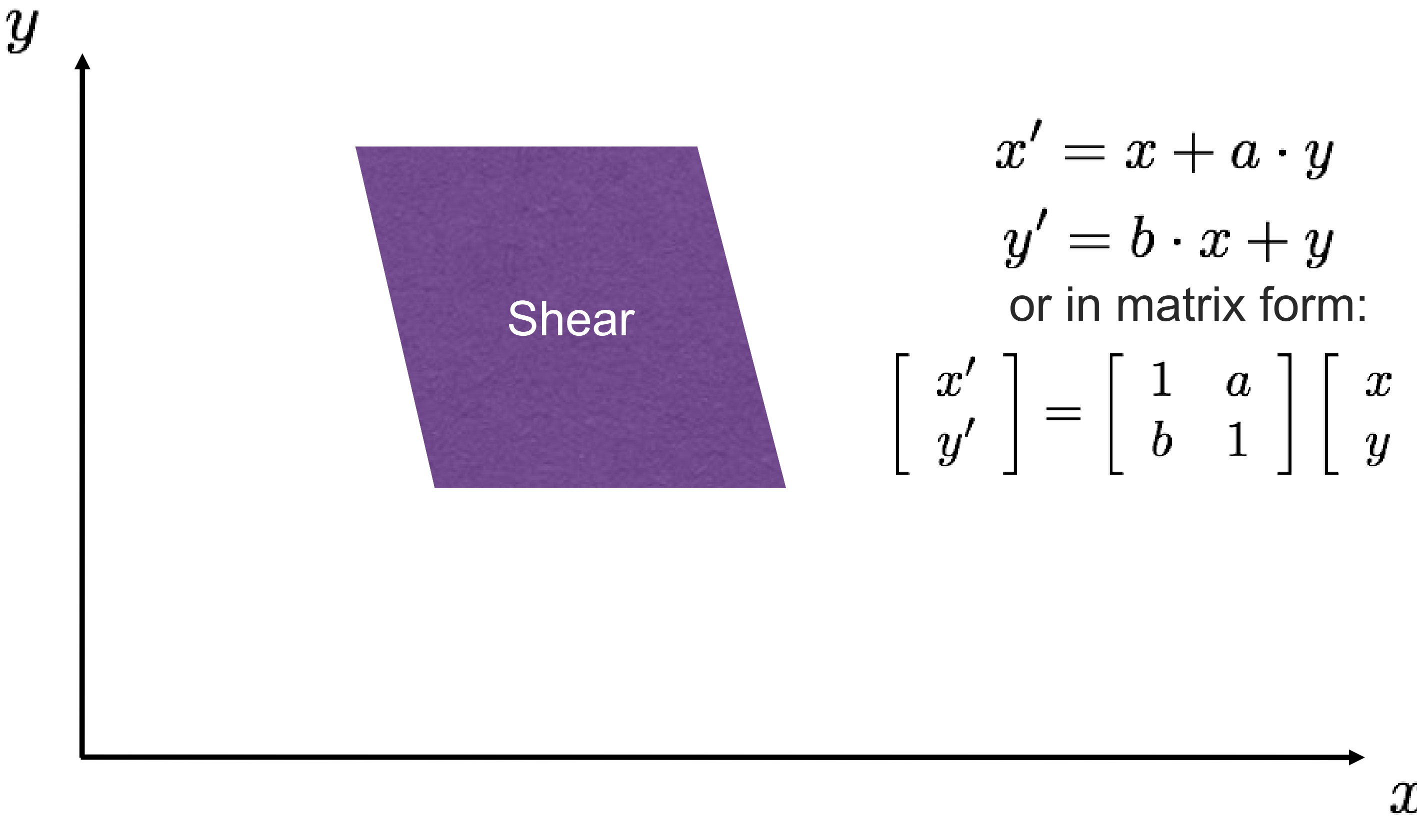
- Rotation by –θ
- For rotation matrices $\mathbf{R}^{-1} = \mathbf{R}^{T}$

# 2-D Shearing




How would you implement shearing?

# 2-D Shearing




$$x' = x + a \cdot y$$

$$y' = b \cdot x + y$$

or in matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & a \\ b & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

# What transformations can be represented with a 2x2 matrix?

2D Scaling?

$$x' = s_x * x$$

$$y' = s_y * y$$

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

2D Rotate around (0,0)?

$$x' = \cos\Theta * x - \sin\Theta * y$$
$$y' = \sin\Theta * x + \cos\Theta * y$$

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

2D Shear?

$$x' = x + sh_x * y$$

$$y' = sh_y * x + y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

# What transformations can be represented with a 2x2 matrix?

2D Mirror about Y axis?

$$x' = -x$$
$$y' = y$$

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

2D Mirror over (0,0)?

$$x' = -x$$
$$y' = -y$$

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

2D Translation?

$$\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}_x$$
$$\boldsymbol{y}' = \boldsymbol{y} + \boldsymbol{t}_y$$

**NO!**

Source: Alyosha E

# 2D Linear Transformations

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

- Only linear 2D transformations can be represented with a 2x2 matrix.
- Linear transformations are combinations of …
  - Scale,
  - Rotation,
  - Shear, and
  - Mirror

Source: Alyosha Ef

# Homogeneous coordinates

To convert to homogeneous coordinates:

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

homogeneous image
coordinates

Converting *from* homogeneous coordinates

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

# Homogeneous Coordinates

- Q: How can we represent 2d translation as a 3x3 matrix using homogeneous coordinates?

$$x' = x + t_x$$

$$y' = y + t_y$$

- A: Using the rightmost column:

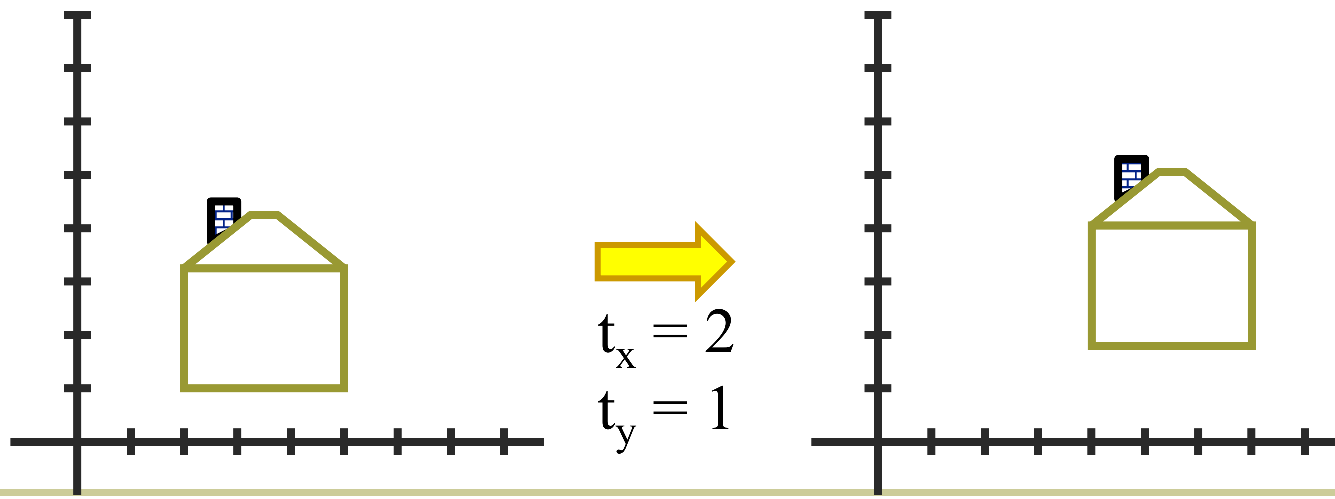
$$\mathbf{T}\boldsymbol{ranslation} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Source: Alyosha E

# Translation

Homogeneous Coordinates

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




Source: Alyosha E

# Basic 2D Transformations

- Basic 2D transformations as 3x3 matrices

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

Translate

$$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s_x} & 0 & 0 \\ 0 & \boldsymbol{s_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

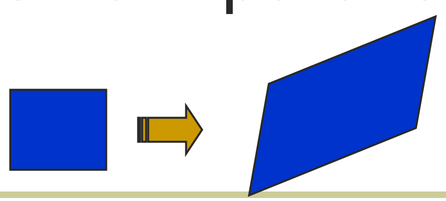
$$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh_x} & 0 \\ \boldsymbol{sh_y} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Shear

# 2D Affine Transformations

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

- Affine transformations are combinations of …
  - Linear transformations, and
  - Translations
- Parallel lines remain parallel



# 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

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

Projective transformations are combos of

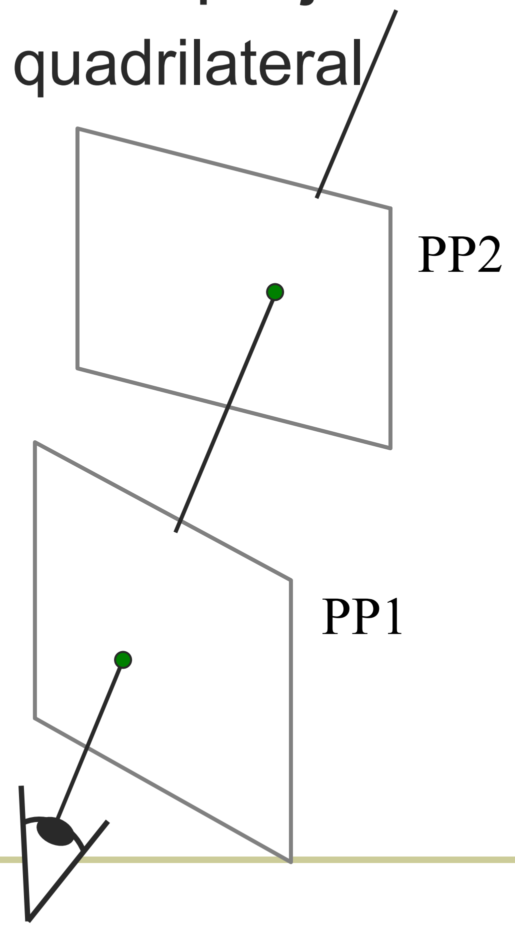
- Affine transformations, and
- Projective warps

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)

# Image reprojection: Homography

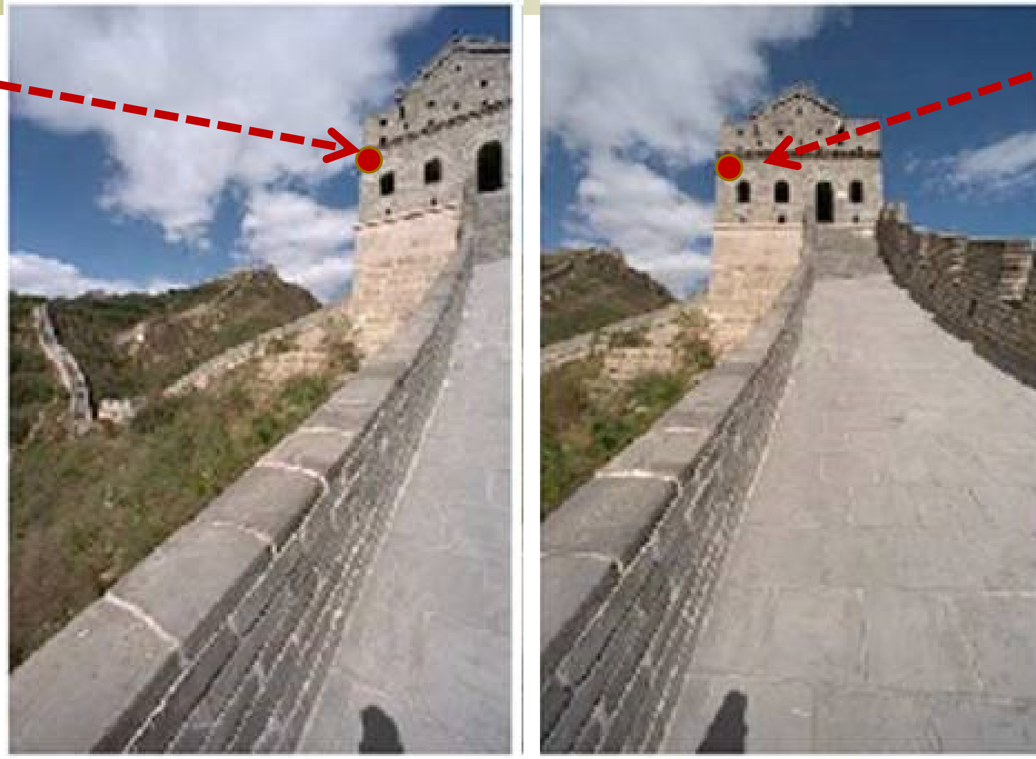
- A projective transform is a mapping between any two PPs with the same center of projection
  - rectangle should map to arbitrary quadrilateral
  - parallel lines aren't
  - but must preserve straight lines
- called **Homography**




$$\underbrace{\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}}_{\mathbf{p'}} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathbf{p}}$$

# Homography

$$(x, y)$$



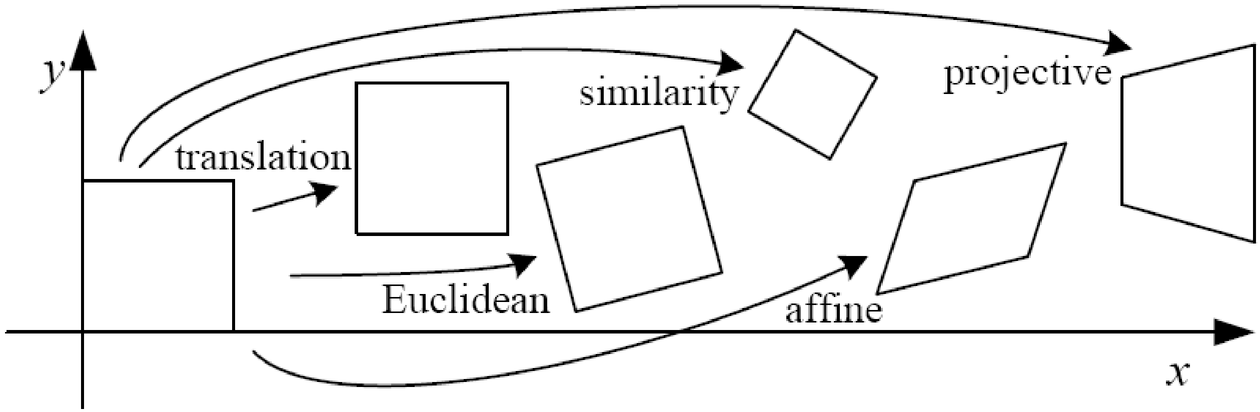
$$\left( {}^{wx'}\!/_{w}, {}^{wy'}\!/_{w} \right) = (x', y')$$

To **apply** a given homography **H**

- Compute **p'** = **Hp** (regular matrix multiply)
- Convert **p'** from homogeneous to image coordinates

$$\underbrace{\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}}_{\mathbf{p'}} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathbf{p}}$$

# 2D image transformations

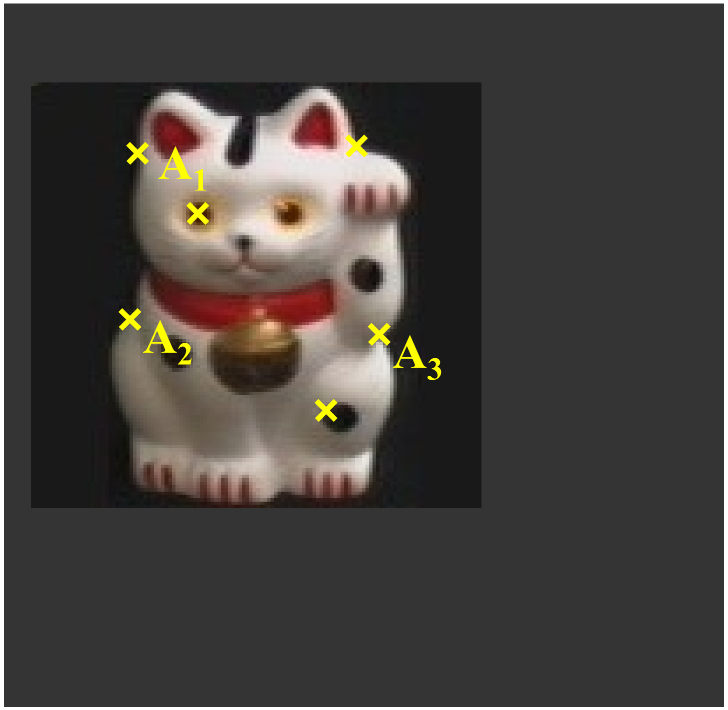



| Name | Matrix | # D.O.F. | Preserves: | Icon |
|---|---|---|---|---|
| translation | $\begin{bmatrix} \boldsymbol{I} & \vert & \boldsymbol{t} \end{bmatrix}_{2\times 3}$ | 2 | orientation $+\cdots$ | |
| rigid (Euclidean) | $\begin{bmatrix} \boldsymbol{R} & \vert & \boldsymbol{t} \end{bmatrix}_{2\times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

# Today's class

- Introduction to alignment
- Alignment methods
  - Global methods
  - Hypothesize and test
- Image Transformation
  - Common image transformations
  - **Examples of solving image alignment**
- Homework: Mosaics

# Example: solving for translation

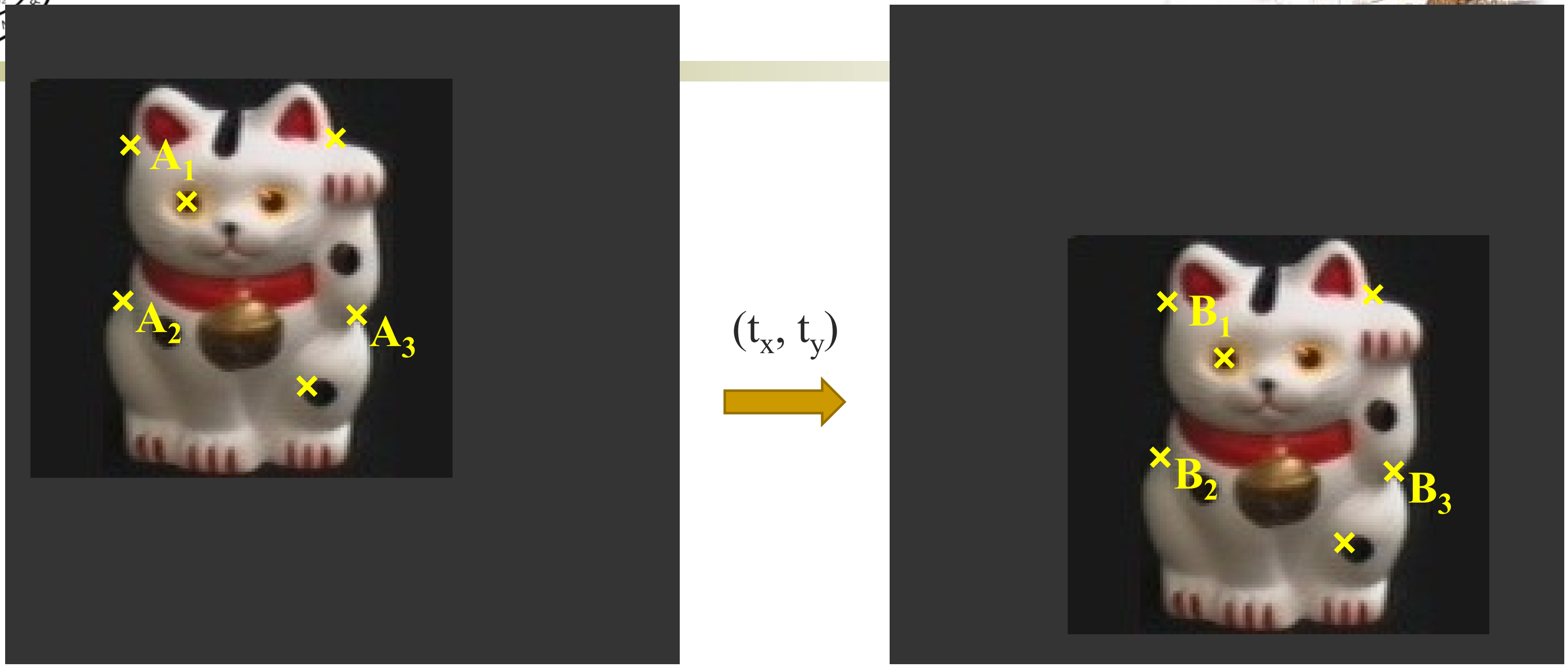




Given matched points in {A} and {B}, estimate the translation of the object

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

# Example: solving for translation



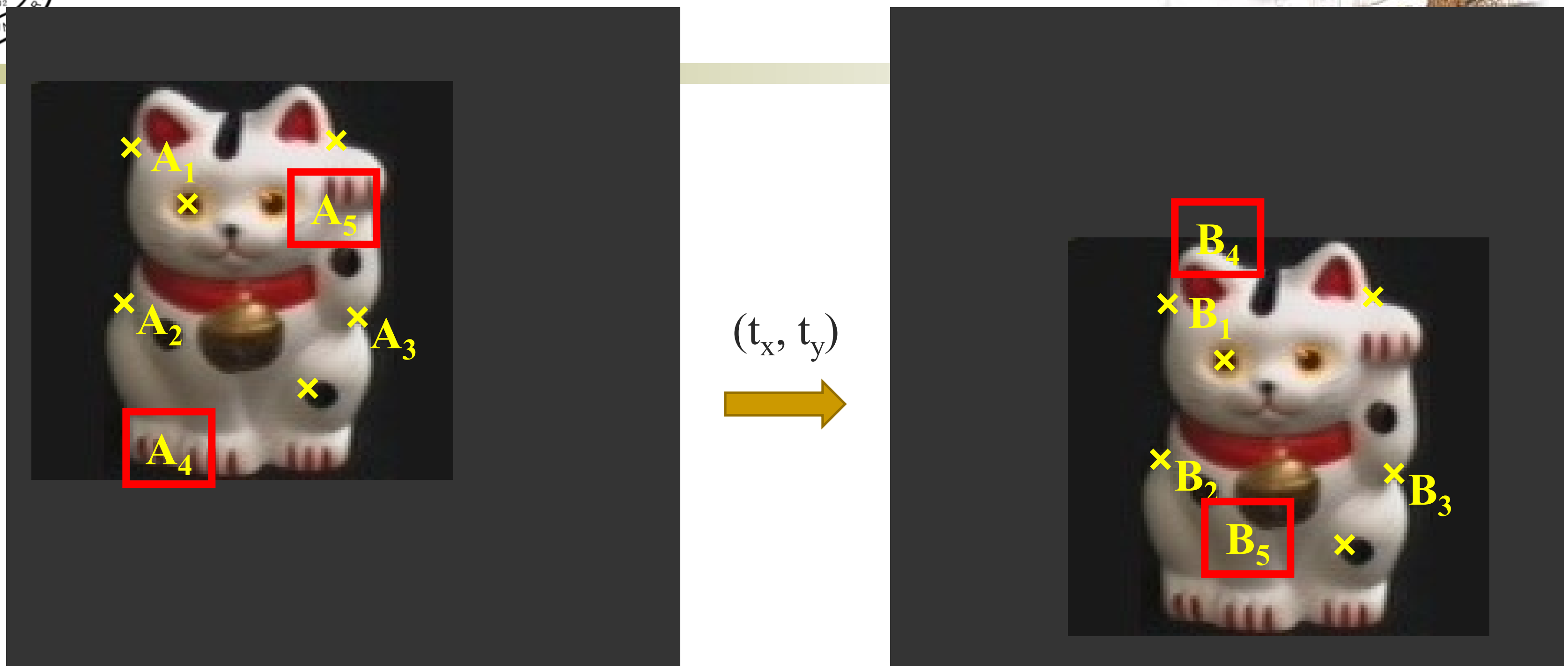

## Least squares solution

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



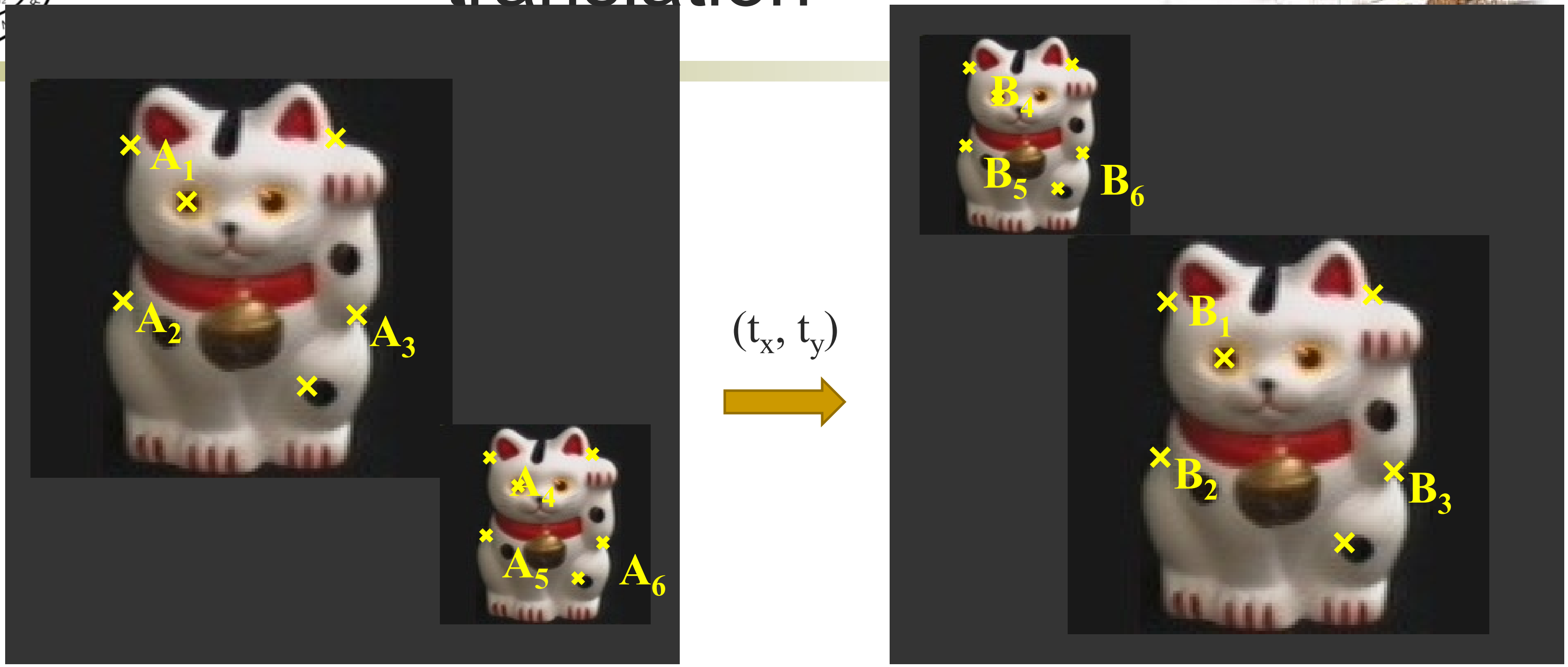

**Problem: outliers**

## 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**

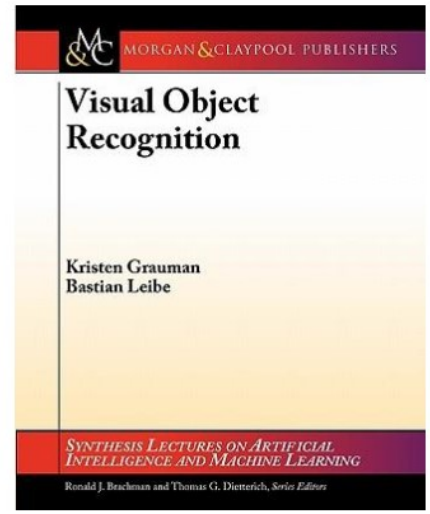
## Hough transform solution

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

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

# References

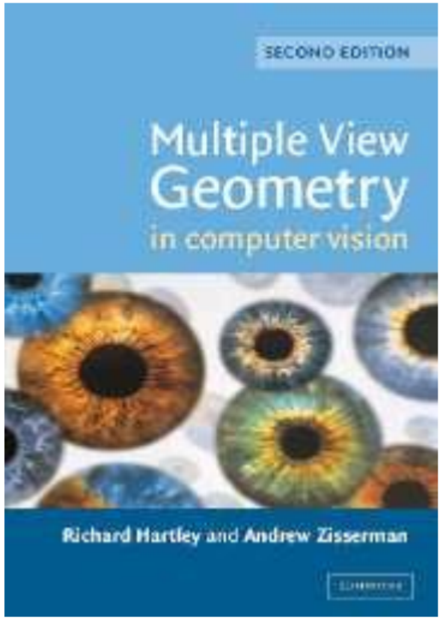
- We've created a script… for the part of the lecture on object recognition & categorization
  - K. Graumann, B. Leibe
    Visual Object Recognition
    Morgan & Claypool publishers, 2011




- Chapter 3: Local Feature Extraction (Last lecture)
- Chapter 5: Geometric Verification (Today)

# References

- More details on homography estimation can be found in Chapter 4.7 of
  - R. Hartley, A. Zisserman
    Multiple View Geometry in Computer Vision
    2nd Ed., Cambridge Univ. Press, 2004




- Details about the DoG detector and the SIFT descriptor can be found in
  - D. Lowe, Distinctive image features from scale-invariant keypoints,
    *IJCV* 60(2), pp. 91-110, 2004
- Try the available local feature detectors and descriptors
  - http://www.robots.ox.ac.uk/~vgg/research/affine/detectors.html#binaries

# Today's class

- Introduction to alignment
- Alignment methods
  - Global methods
  - Hypothesize and test
- Image Transformation
  - Common image transformations
  - Examples of solving image alignment
- **Homework: Mosaics**

# Homework: Mosaics

. . .

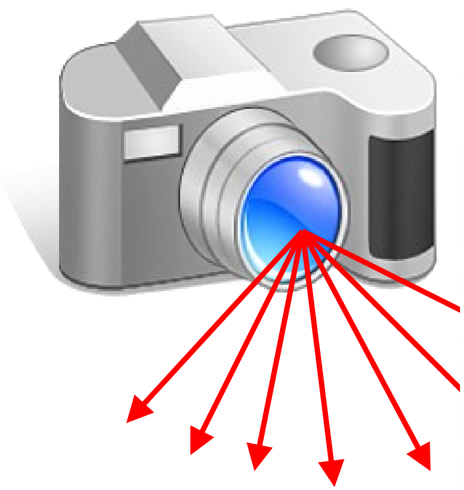
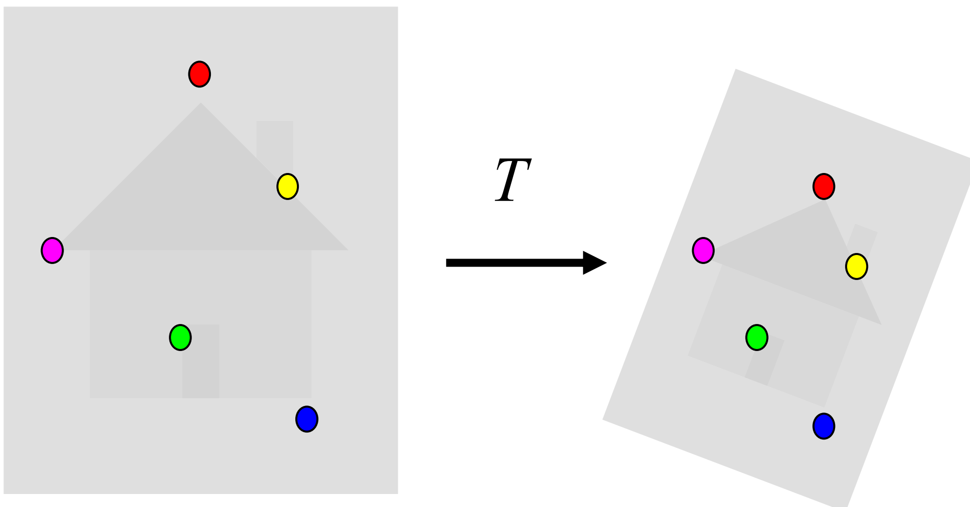


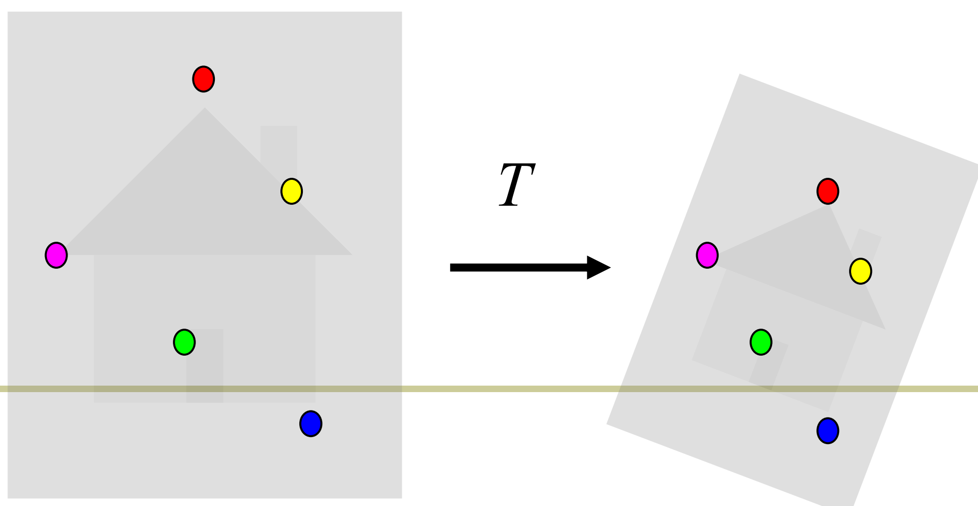
image from S. Seitz

Obtain a wider angle view by combining multiple images.

# Main questions




**Alignment**: Given two images, what is the transformation between them?




**Warping**: Given a source image and a transformation, what does the transformed output look like?

# How to stitch together a panorama (a.k.a. mosaic)?

- Basic Procedure
  - Take a sequence of images from the same position
    - Rotate the camera about its optical center
  - Compute transformation (homography) between second image and first using corresponding points.
  - Transform the second image to overlap with the first.
  - Blend the two together to create a mosaic.
  - (If there are more images, repeat)

# 本科生大作业

- 综述性研究报告：针对一个专门话题，进行深入调研分析，形成一篇高质量的综述。
- 培养：查阅文献，阅读文献，撰写文献的能力，可能启发对某个方向深入研究的兴趣。
- 提交格式：CVPR论文格式，建议英文撰写。
- 调研范围：顶级会议和顶级期刊论文，以及最新的arXiv论文，
- **不要仅仅去看一些公众号、知乎、中文博客**
- **提交时间：6月20号之前（毕业班）**

# 话题

- Low-level vision: denoise, super resolution etc.
- Edge detection
- Grouping and segmentation
- Local descriptor and image matching
- Deep learning for vision
- Tracking, video analysis
- Vision and language
- Weakly/Self supervised learning
- Etc.