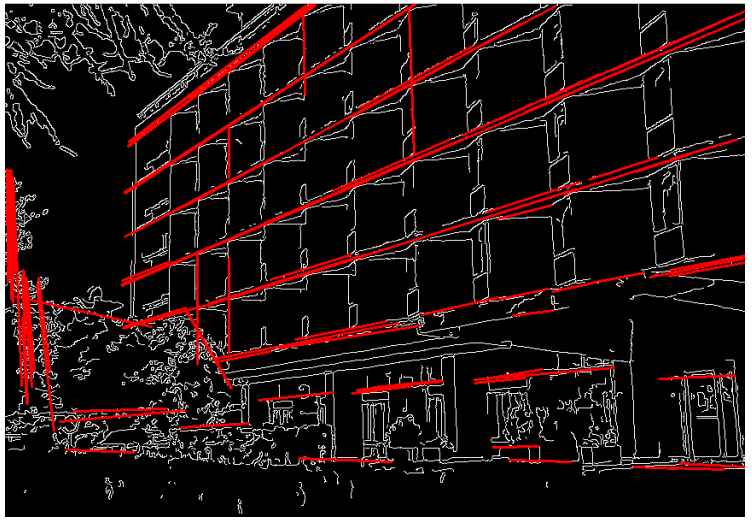
Fitting and Alignment

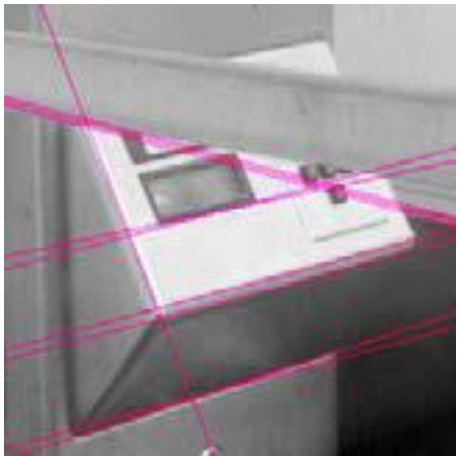


Fitting

- We've learned how to detect edges, corners, blobs. Now what?
- We would like to form a higher-level, more compact representation of the features in the image by grouping multiple features according to a simple model

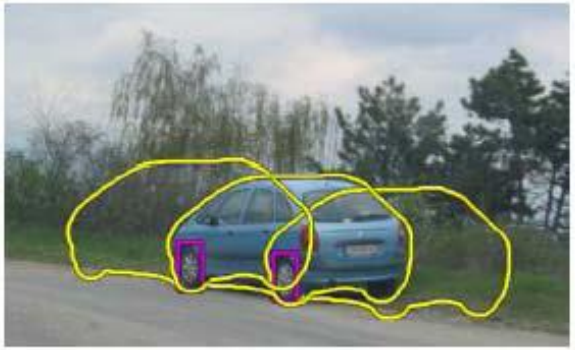
Fitting

- Choose a parametric model to represent a set of features



simple model: lines

simple model: circles



complicated model: car

Source: K. Grauman

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

Hypothesize and test

- RANSAC
- Hough transform

Simple example: Fitting a line

Fitting: Issues

Case study: Line detection

- **Noise** in the measured feature locations
- **Extraneous data:** clutter (outliers), multiple lines
- **Missing data:** occlusions

Fitting: Overview

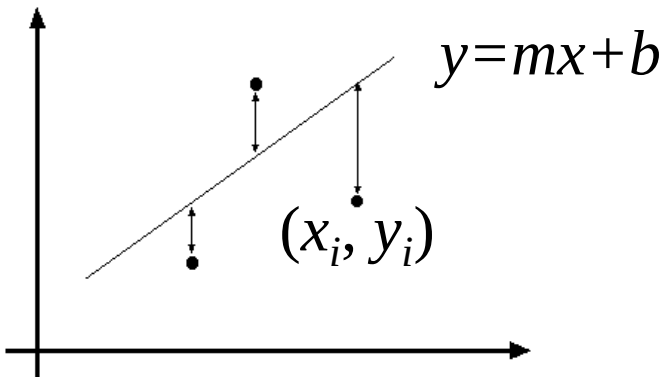
- If we know which points belong to the line, how do we find the “optimal” line parameters?
  - Least squares
- What if there are outliers?
  - Robust fitting, RANSAC
- What if there are many lines?
  - Voting methods: RANSAC, Hough transform
- What if we’re not even sure it’s a line?
  - Model selection

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



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

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \qquad B = \begin{bmatrix} m \\ b \end{bmatrix}$$

$$E = \left\| Y - XB \right\|^2 = (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB)$$

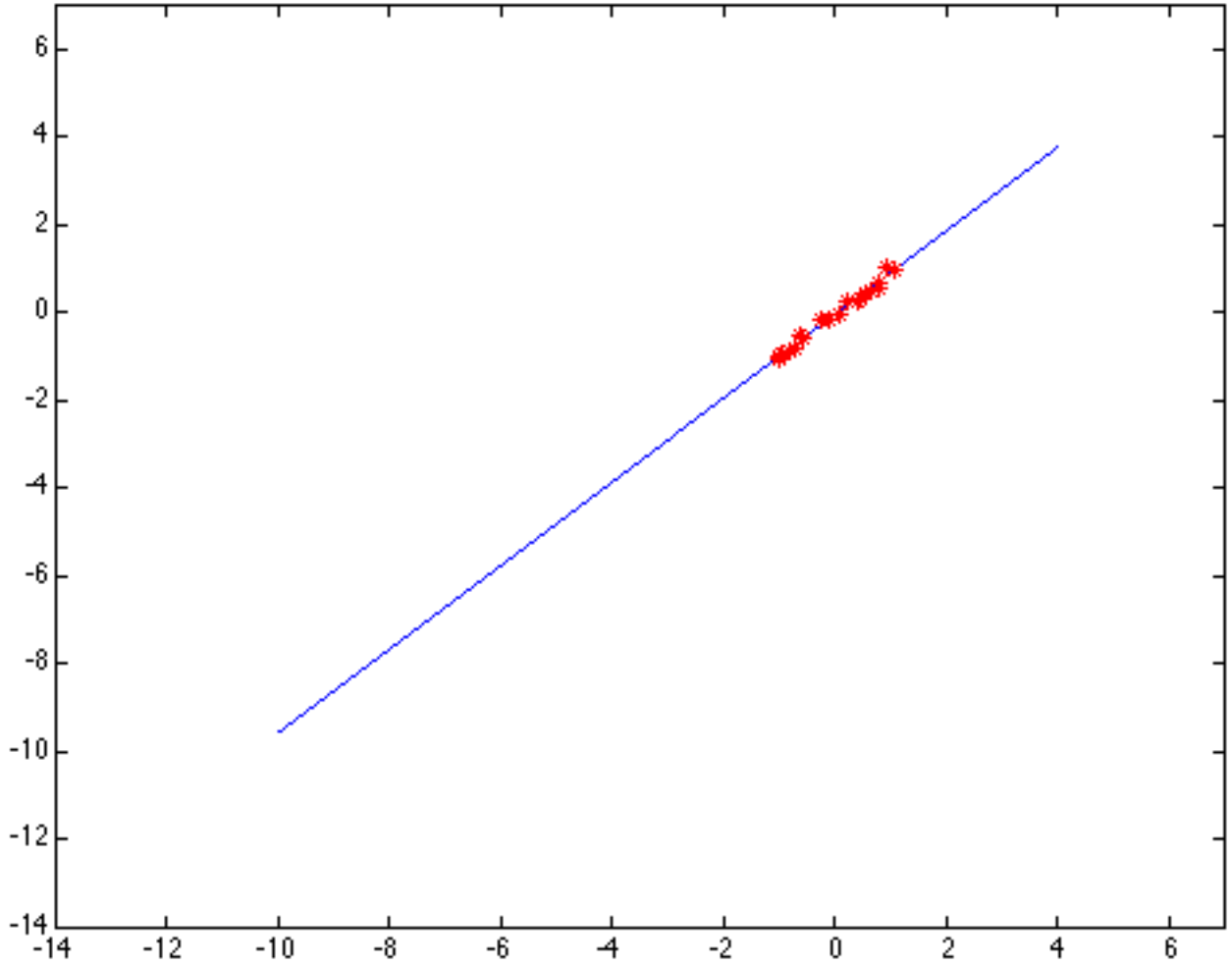
$$\frac{dE}{dB} = 2X^T XB - 2X^T Y = 0$$

$$X^T XB = X^T Y$$

Normal equations: least squares solution to $XB = Y$

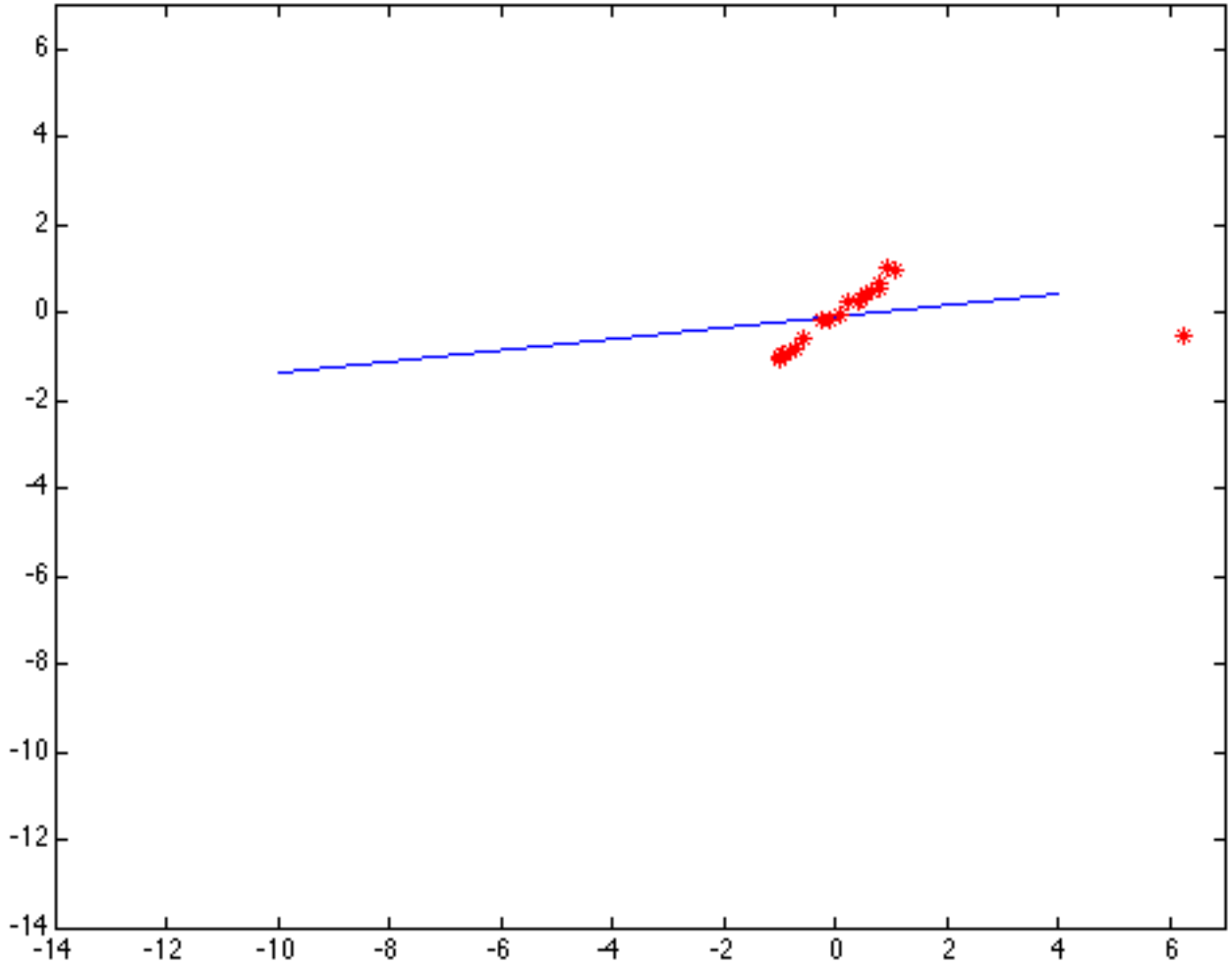
Least squares: Robustness to noise

Least squares fit to the red points:



Least squares: Robustness to noise

Least squares fit with an outlier:



Problem: squared error heavily penalizes outliers

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.

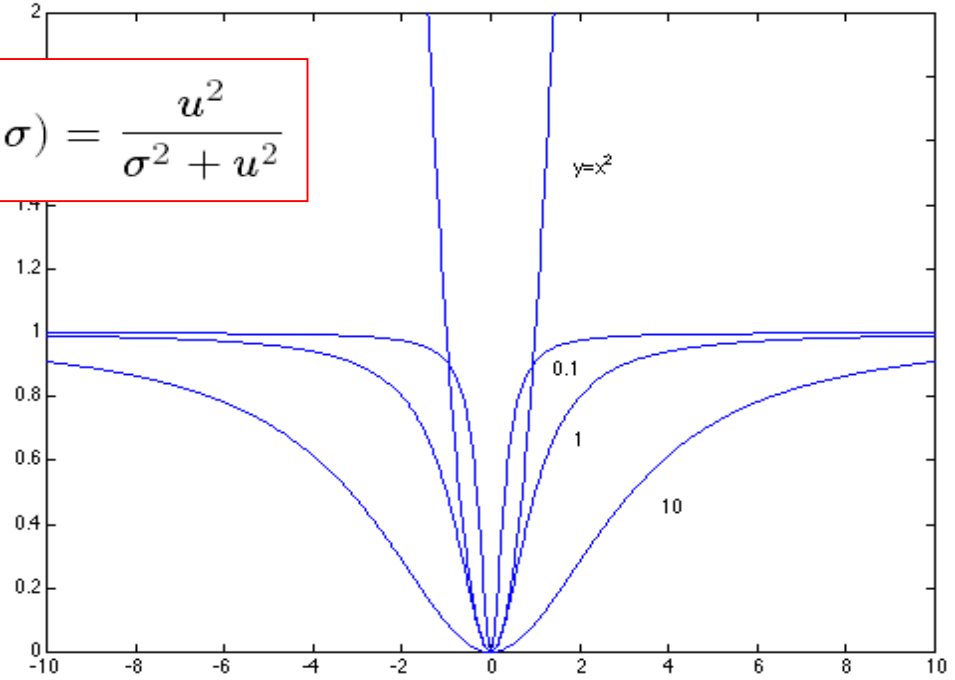
Robust estimators

- General approach: find model parameters θ that minimize

$$\sum_i \rho\left(r_i\left(x_i, \theta\right); \sigma\right) \qquad r^2 = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

$r_i(x_i, \theta)$ – residual of ith point w.r.t. model parameters θ
ρ – robust function with scale parameter σ

$$\rho(u;\sigma) = \frac{u^2}{\sigma^2 + u^2}$$



The robust function ρ behaves like squared distance for small values of the residual *u* but saturates for larger values of *u*

Robust Estimator

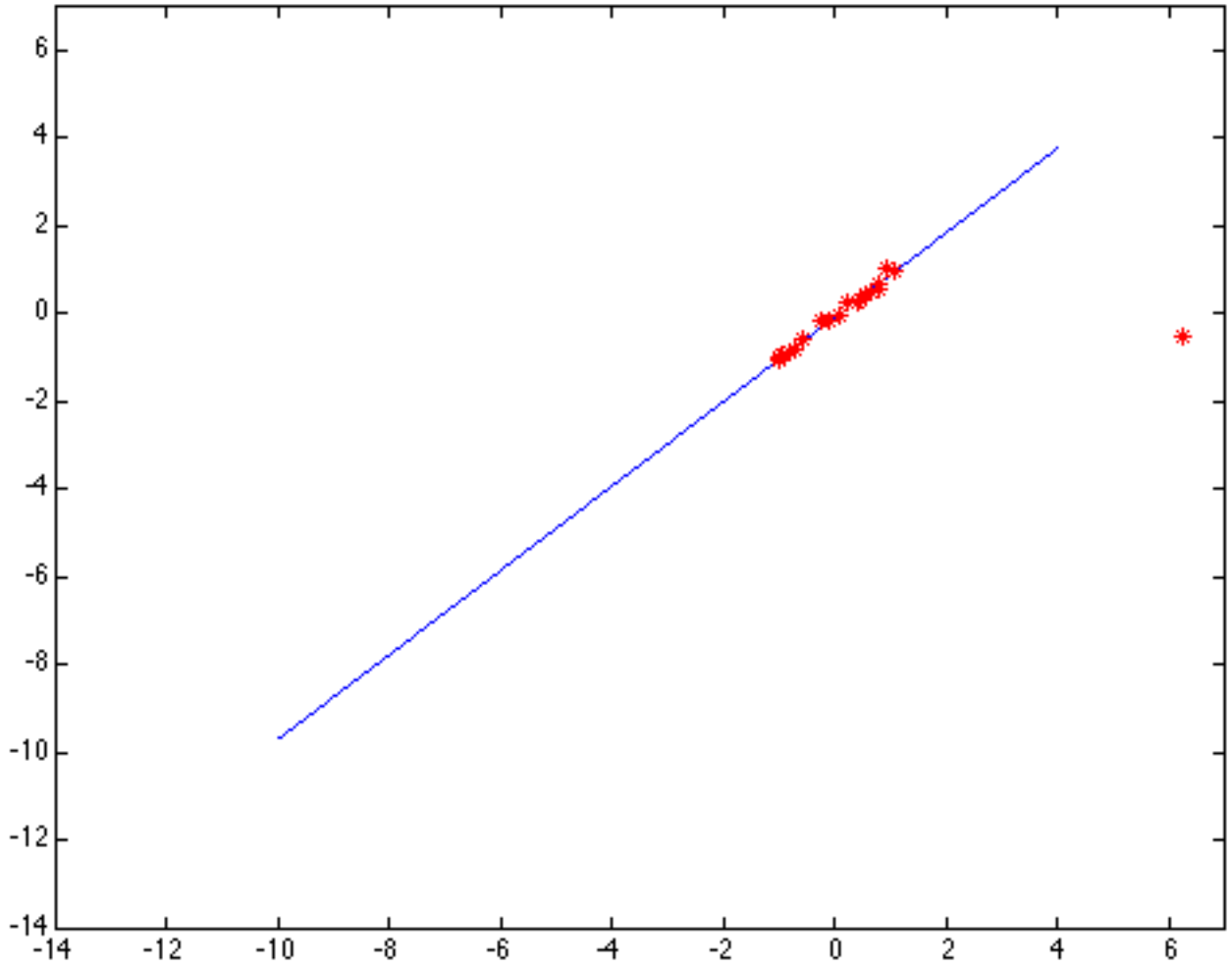
1. Initialize: e.g., choose θ by least squares fit and $\sigma = 1.5 \cdot \text{median}(error)$

2. Choose params to minimize: $\sum_i \frac{error(\theta, data_i)^2}{\sigma^2 + error(\theta, data_i)^2}$
   - E.g., numerical optimization

3. Compute new $\sigma = 1.5 \cdot \text{median}(error)$

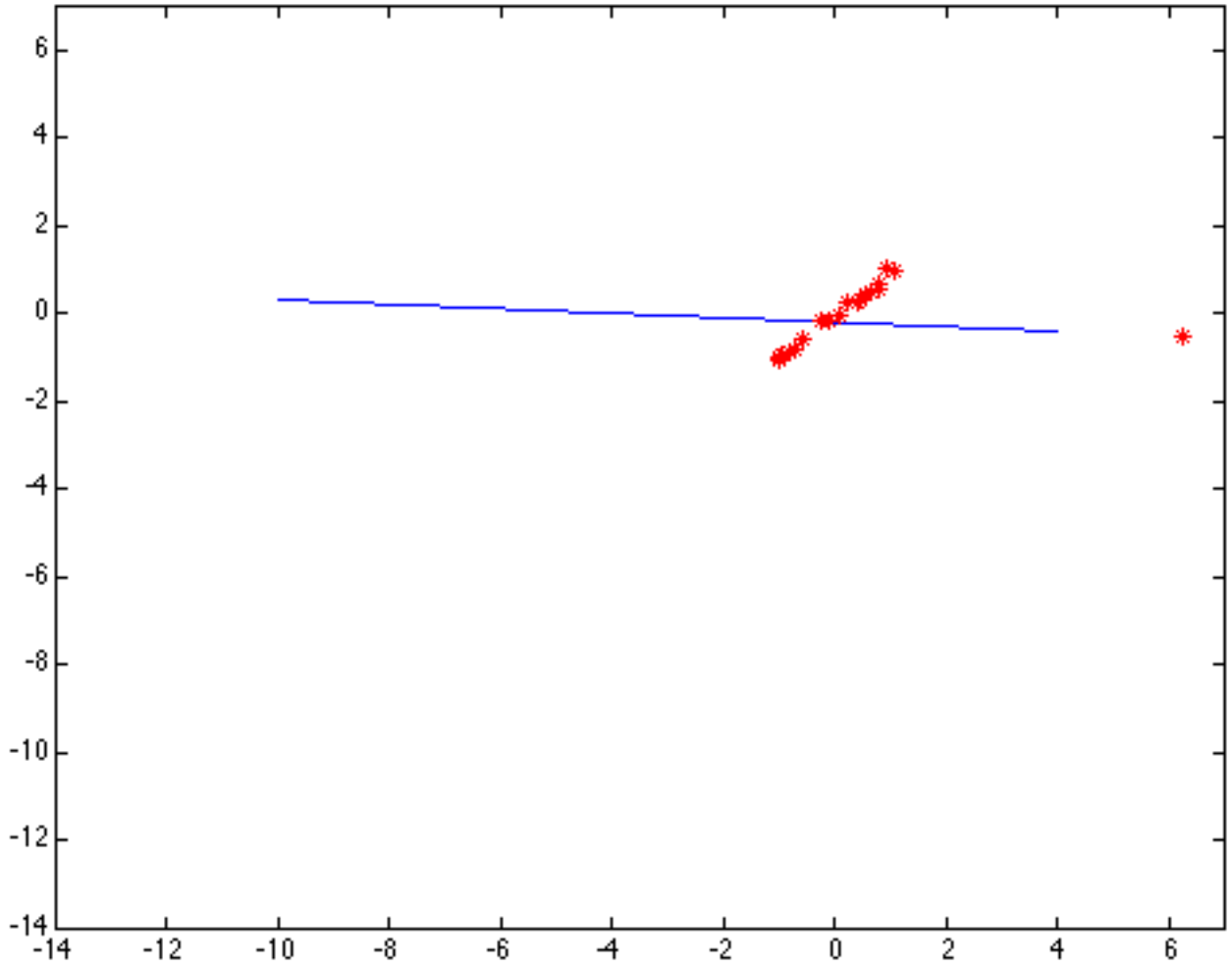
4. Repeat (2) and (3) until convergence

Choosing the scale: Just right



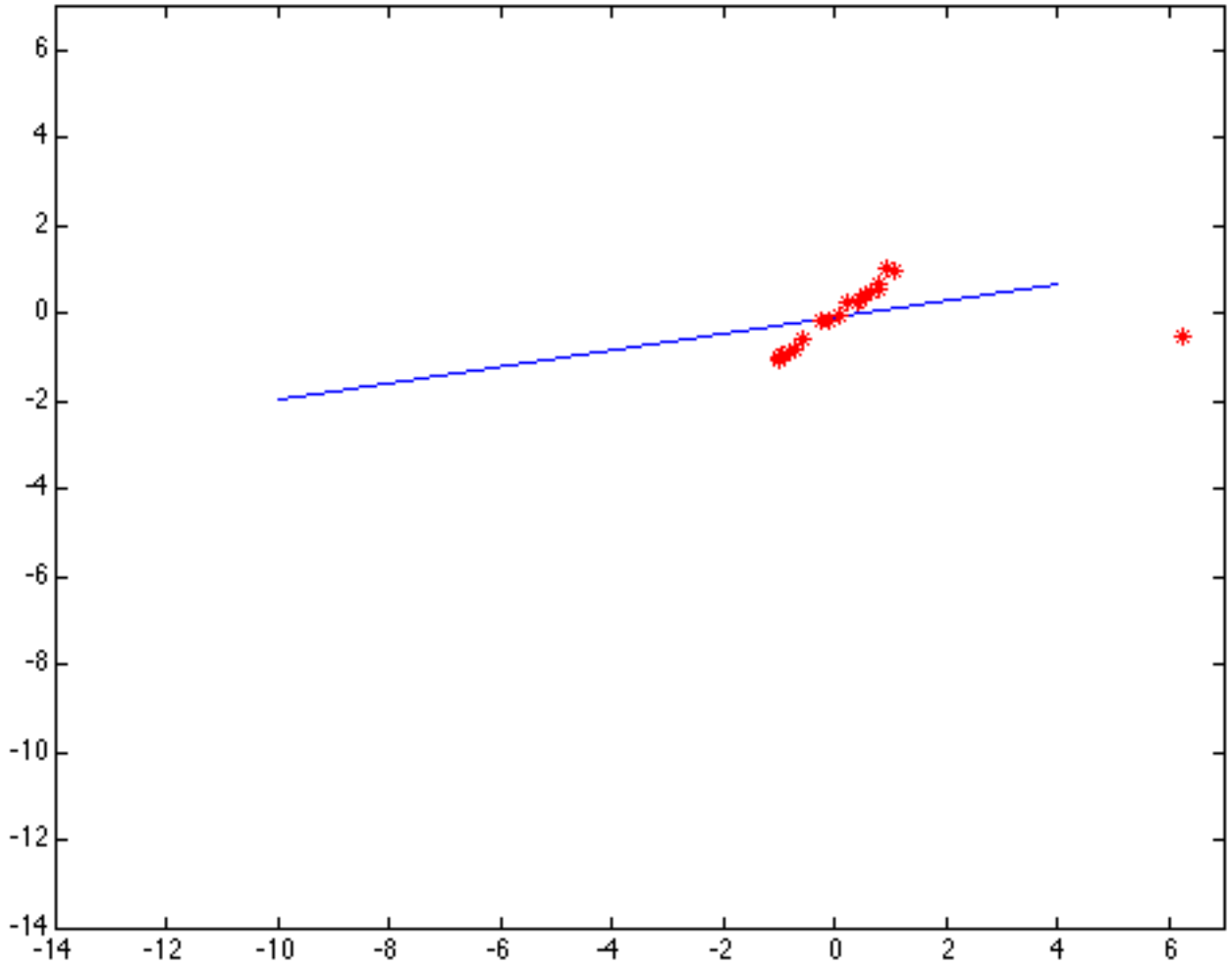
The effect of the outlier is minimized

Choosing the scale: Too small



The error value is almost the same for every point and the fit is very poor

Choosing the scale: Too large



Behaves much the same as least squares

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

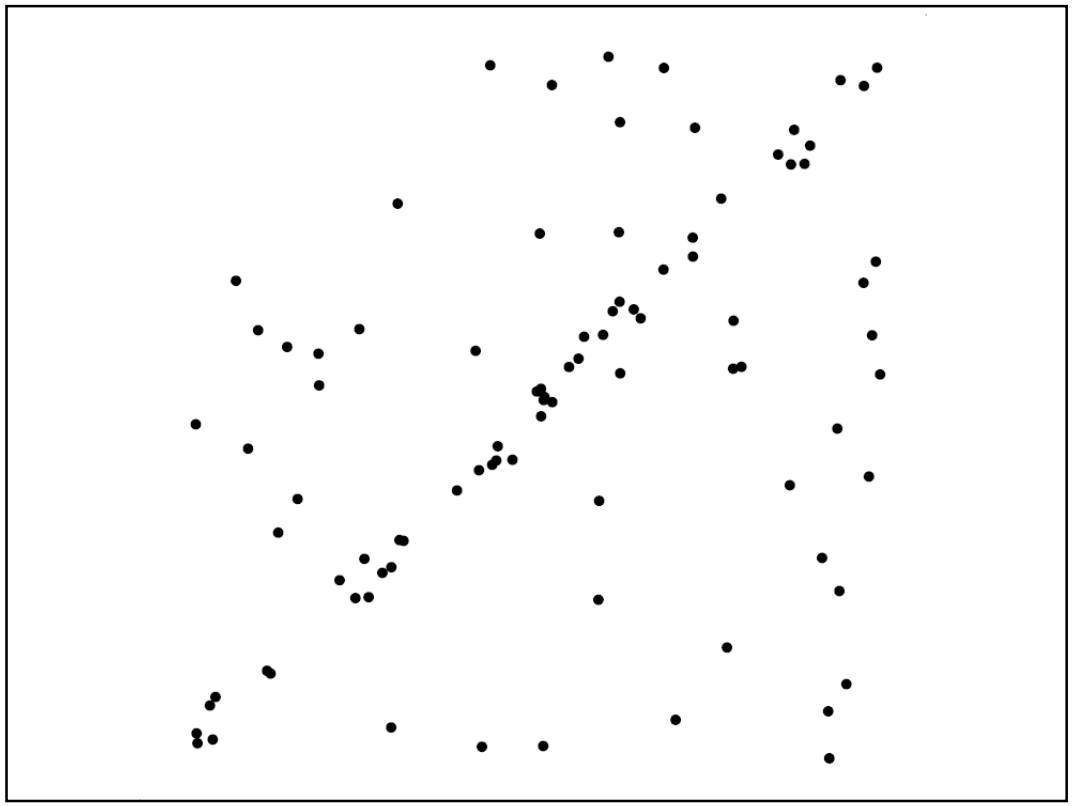
4. Possibly refine parameters using inliers

RANSAC

- Robust fitting can deal with a few outliers – what if we have very many?
- Random sample consensus (RANSAC): Very general framework for model fitting in the presence of outliers
- Outline
  - Choose a small subset of points uniformly at random
  - Fit a model to that subset
  - Find all remaining points that are “close” to the model and reject the rest as outliers
  - Do this many times and choose the best model

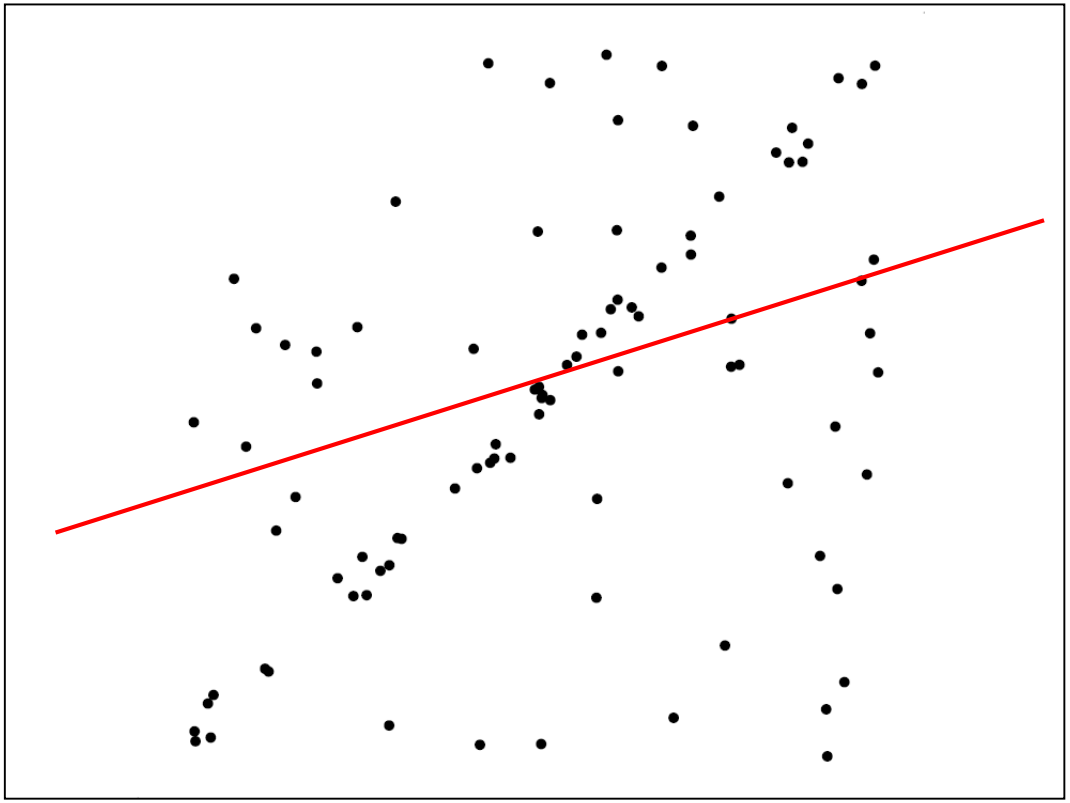
M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

RANSAC for line fitting example



Source: R. Raguram

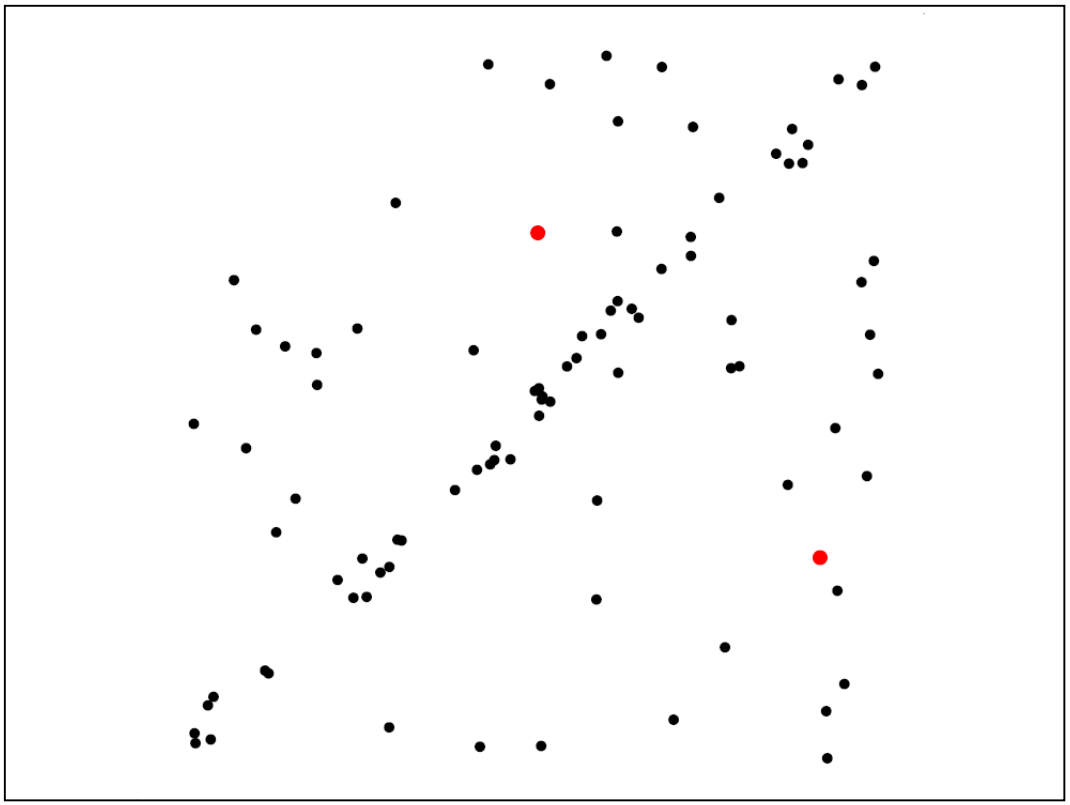
RANSAC for line fitting example



Least-squares fit

Source: R. Raguram

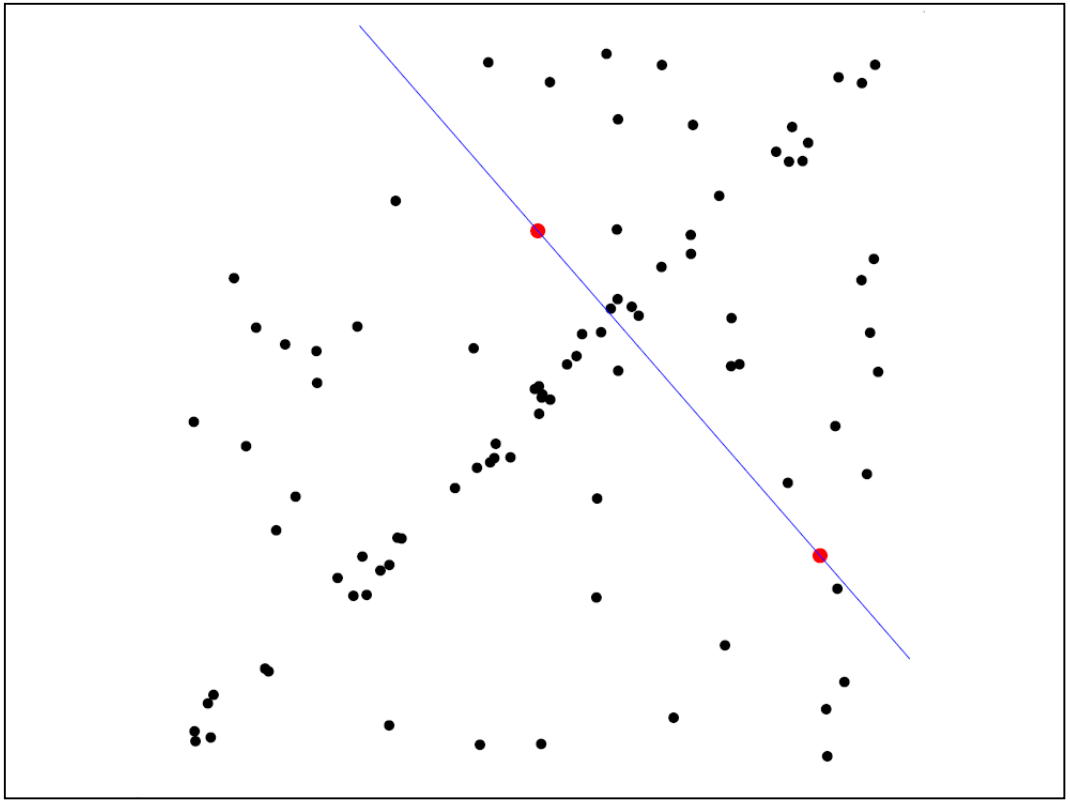
RANSAC for line fitting example



1. Randomly select minimal subset of points

Source: R. Raguram

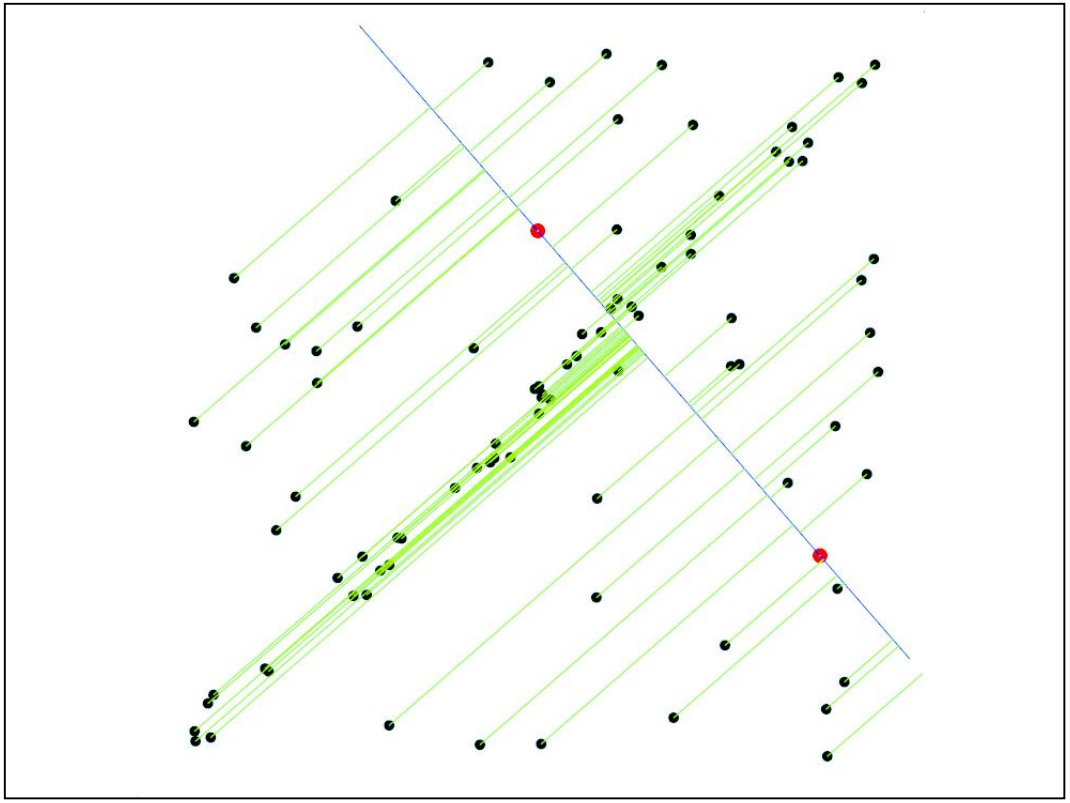
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model

Source: R. Raguram

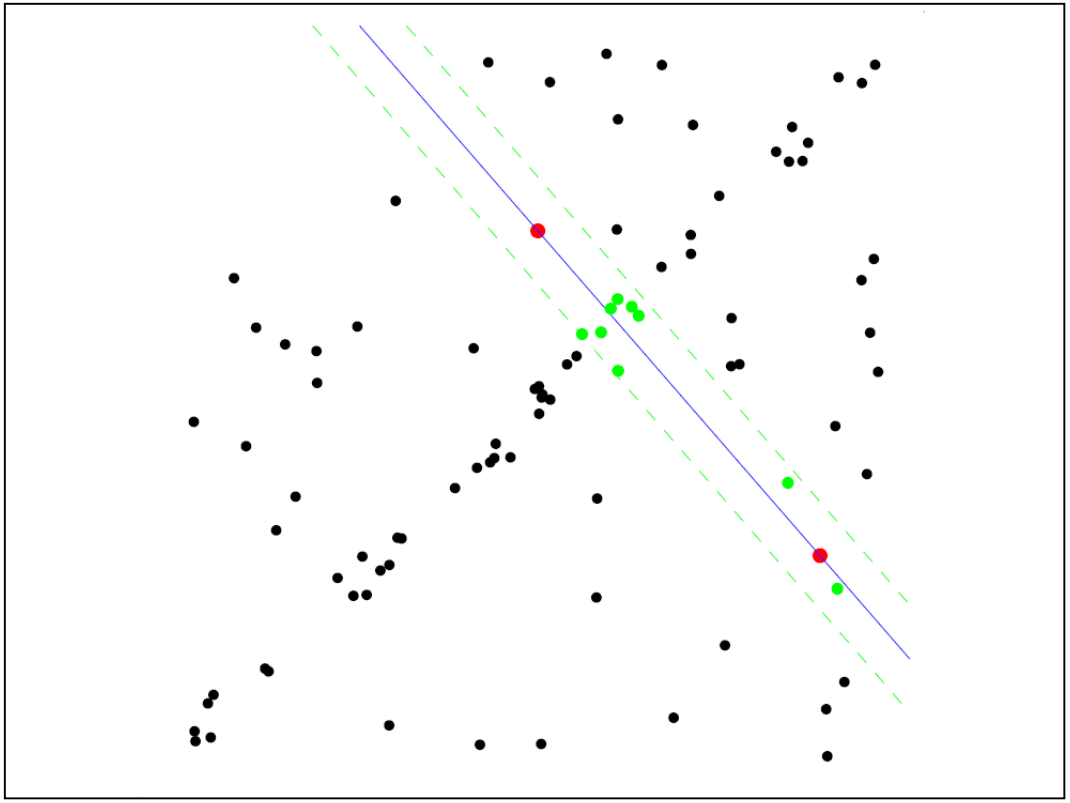
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

Source: R. Raguram

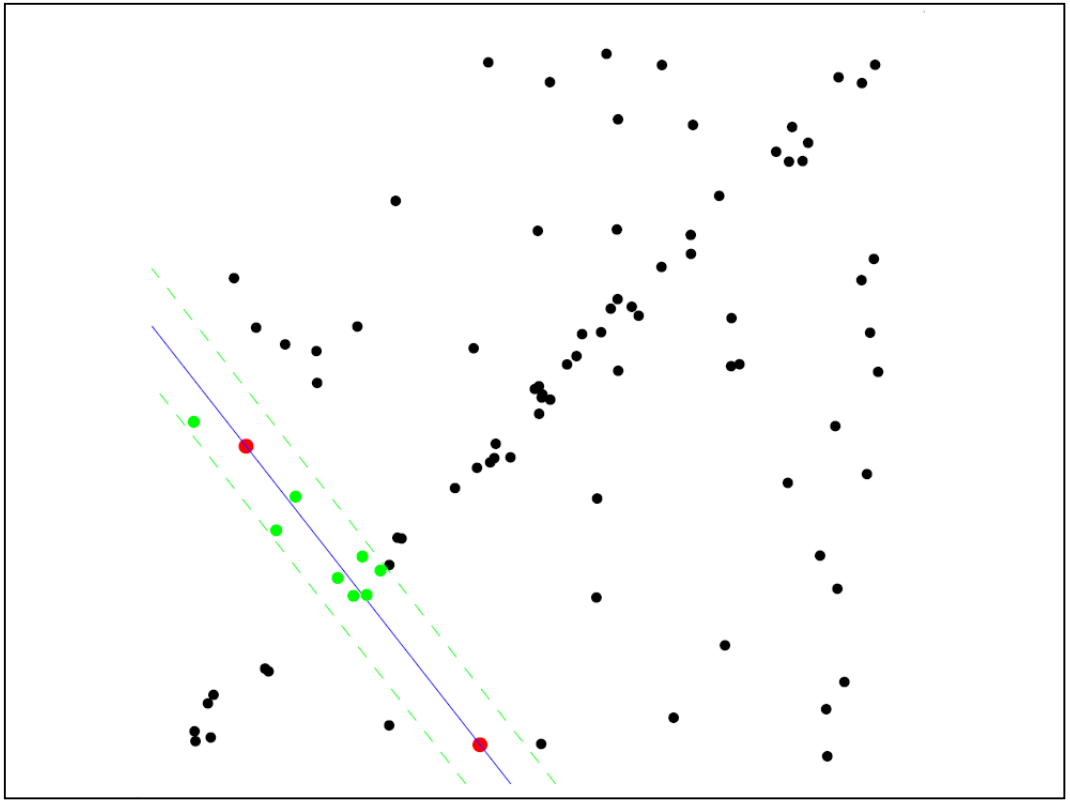
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

Source: R. Raguram

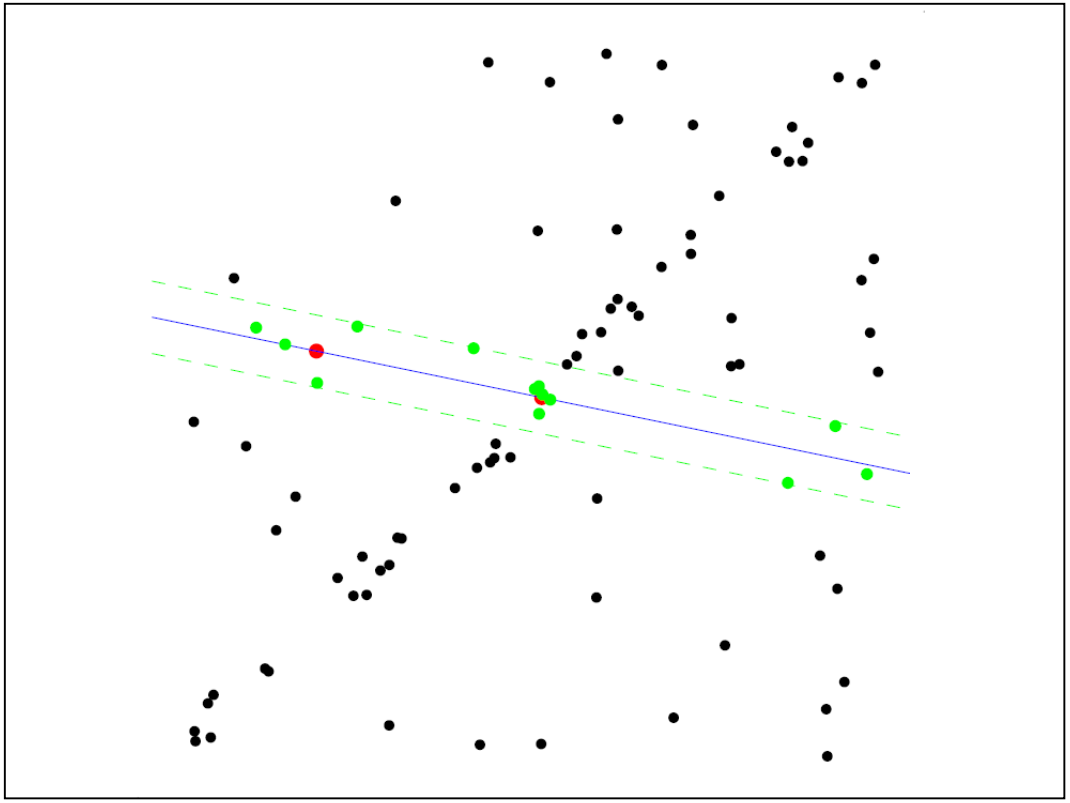
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example

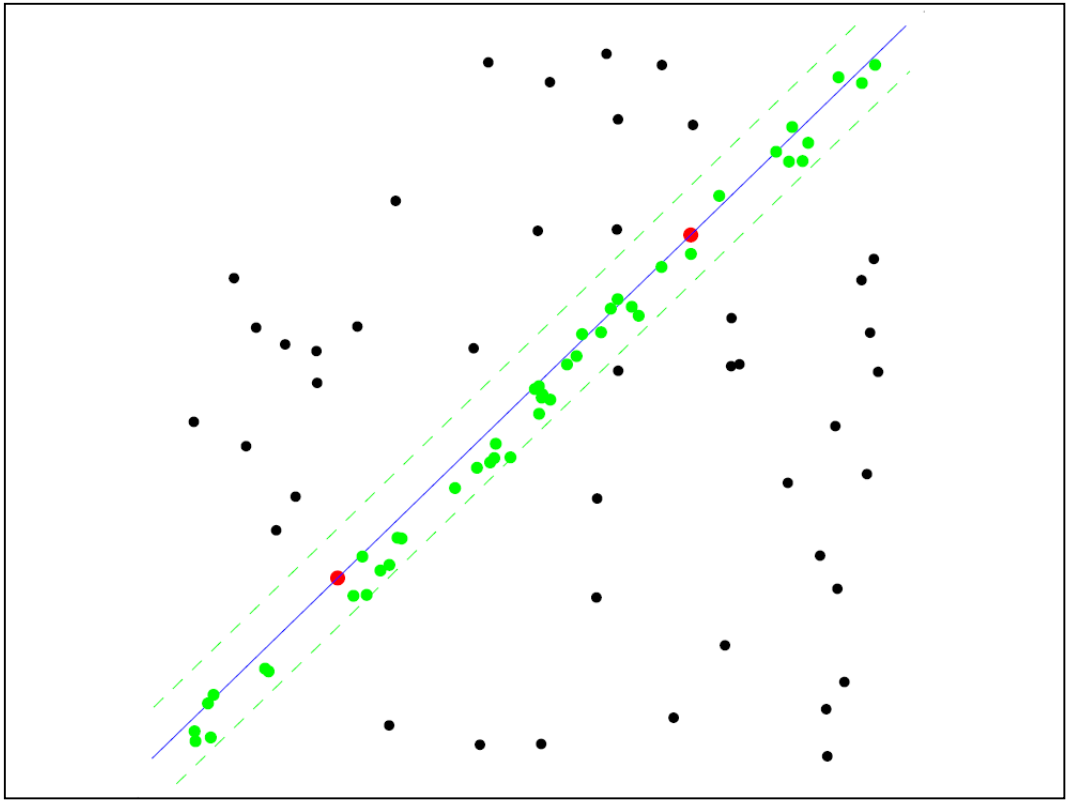


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example

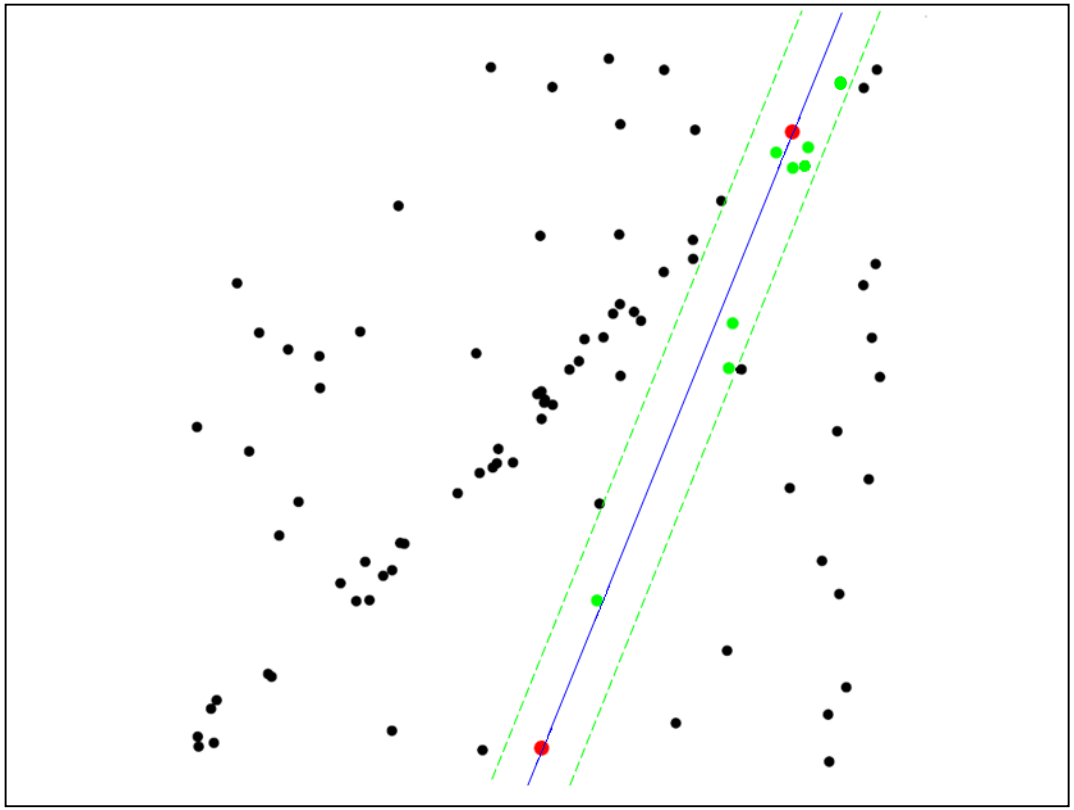
Uncontaminated sample



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting

Repeat ***N*** times:

- Draw ***s*** points uniformly at random
- Fit line to these ***s*** points
- Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
- If there are ***d*** or more inliers, accept the line and refit using all inliers

Choosing the parameters

- Initial number of points *s*
  - Typically minimum number needed to fit the model
- Distance threshold *t*
  - Choose *t* so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)

Source: M. Pollefeys

Choosing the parameters

- Initial number of points *s*
  - Typically minimum number needed to fit the model
- Distance threshold *t*
  - Choose *t* so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)

$$\left(1-\left(1-e\right)^s\right)^N = 1-p$$

$$N = \log\left(1-p\right) / \log\left(1-\left(1-e\right)^s\right)$$

	proportion of outliers *e*						
s	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

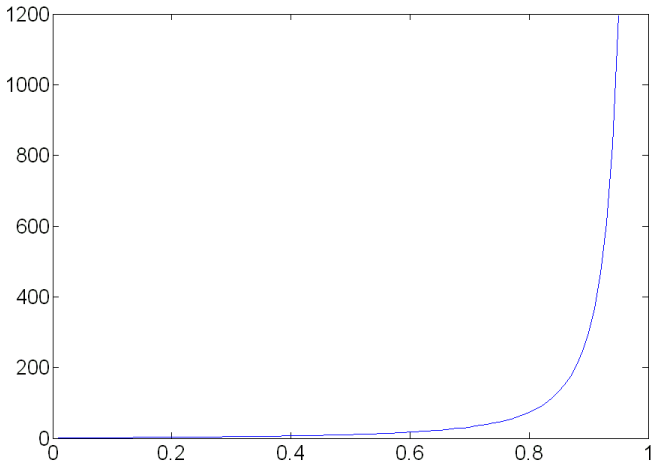
Source: M. Pollefeys

Choosing the parameters

- Initial number of points *s*
  - Typically minimum number needed to fit the model
- Distance threshold *t*
  - Choose *t* so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)

$$\left(1-\left(1-e\right)^s\right)^N = 1-p$$

$$N = \log\left(1-p\right)/\log\left(1-\left(1-e\right)^s\right)$$



Source: M. Pollefeys

Choosing the parameters

- Initial number of points *s*
  - Typically minimum number needed to fit the model
- Distance threshold *t*
  - Choose *t* so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)
- Consensus set size *d*
  - Should match expected inlier ratio

Source: M. Pollefeys

Adaptively determining the number of samples

- Inlier ratio *e* is often unknown a priori, so pick worst case, e.g. 50%, and adapt if more inliers are found, e.g. 80% would yield *e*=0.2
- Adaptive procedure:
  - *N*=∞, *sample_count* =0
  - While *N* >*sample_count*
    - Choose a sample and count the number of inliers
    - Set e = 1 – (number of inliers)/(total number of points)
    - Recompute *N* from *e:*

$$N = \log(1-p) / \log\left(1-(1-e)^s\right)$$

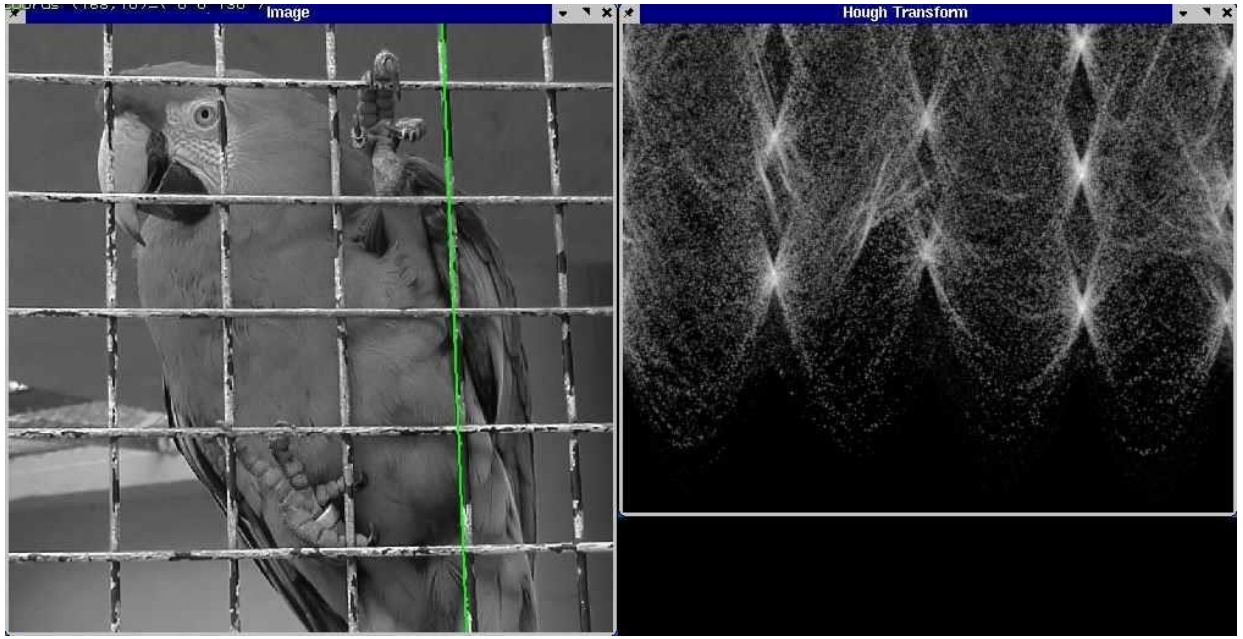
    - Increment the *sample_count* by 1

Source: M. Pollefeys

RANSAC conclusion

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
  - Robust to outliers
- Cons
  - Lots of parameters to tune
  - Doesn’t work well for low inlier ratios (too many iterations, or can fail completely)
  - Can’t always get a good initialization of the model based on the minimum number of samples
- Common applications
  - Computing a homography (e.g., image stitching)
  - Estimating fundamental matrix (relating two views)

The Hough transform



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
- Missing data doesn’t matter as long as there are enough features remaining to agree on a good 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

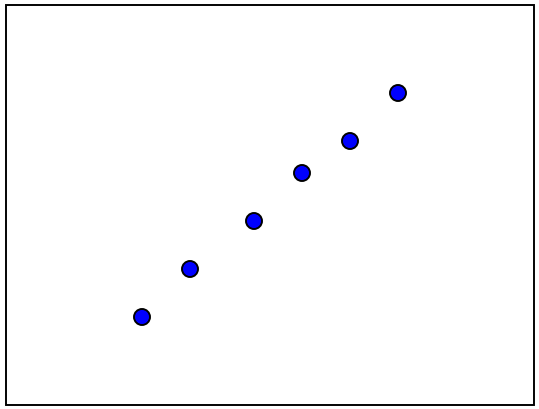
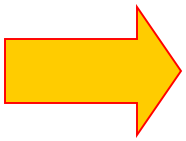
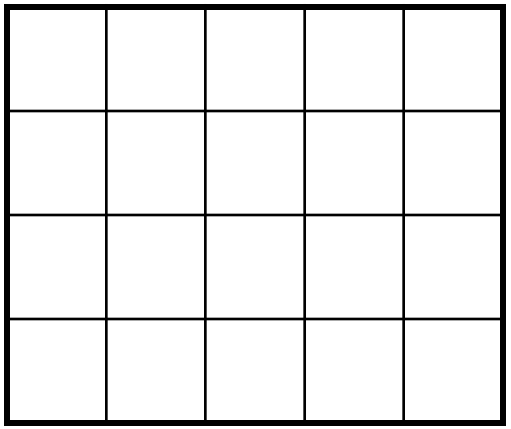


Image space

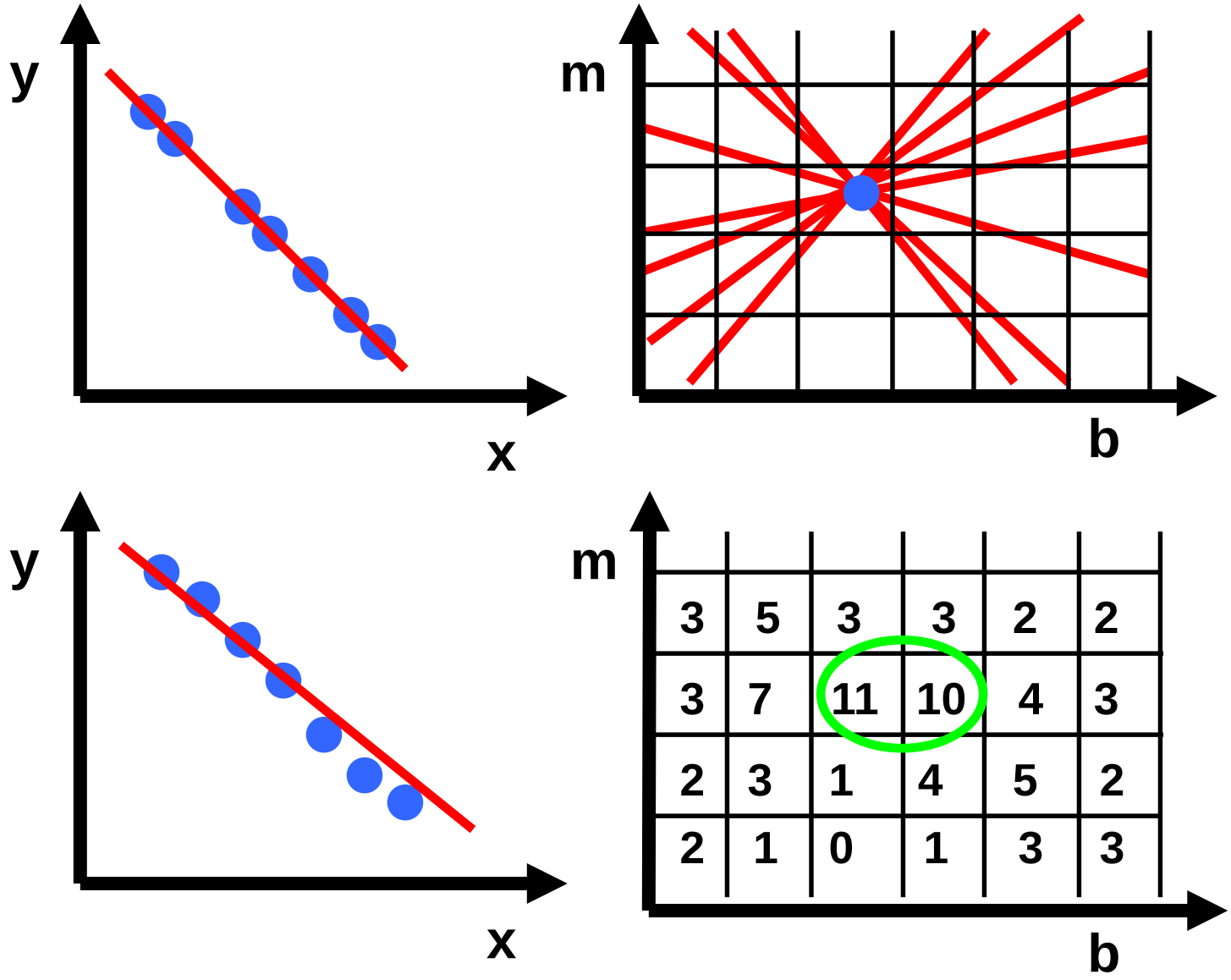




Hough parameter space

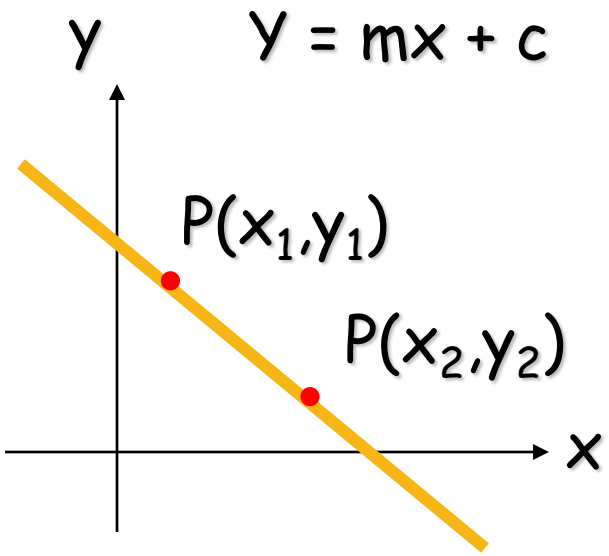
P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Hough transform



Hough Transform

- Mathematical model of a line:



$$y_1 = m\,x_1 + c$$

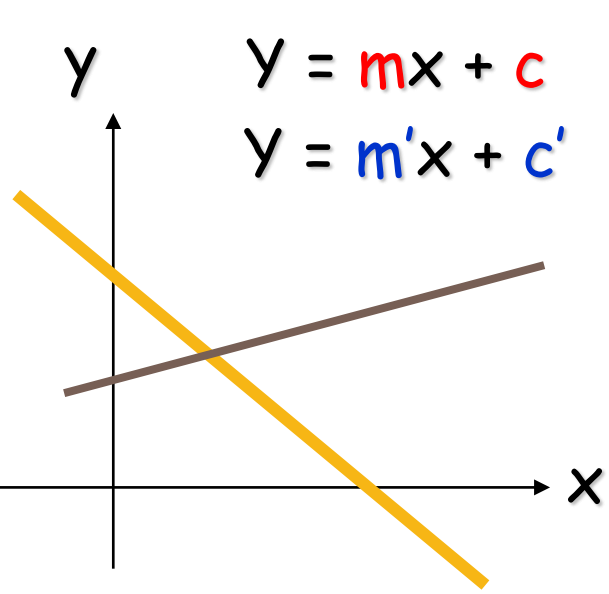
$$y_2 = m\,x_2 + c$$

$$\Downarrow$$

$$y_N = m\,x_N + c$$

Hough Transform

- Image and Parameter Spaces



$y = mx + c$

$y = m'x + c'$

$y_1 = m\,x_1 + c$

$y_2 = m\,x_2 + c$

$y_N = m\,x_N + c$

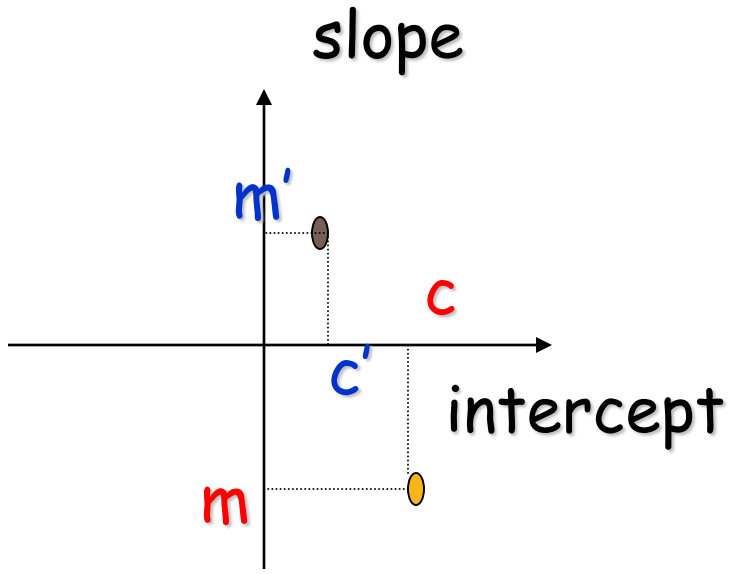


Image Space

Parameter Space

Line in Img. Space ~ Point in Param. Space

Hough Transform

- **Image and Parameter Spaces**

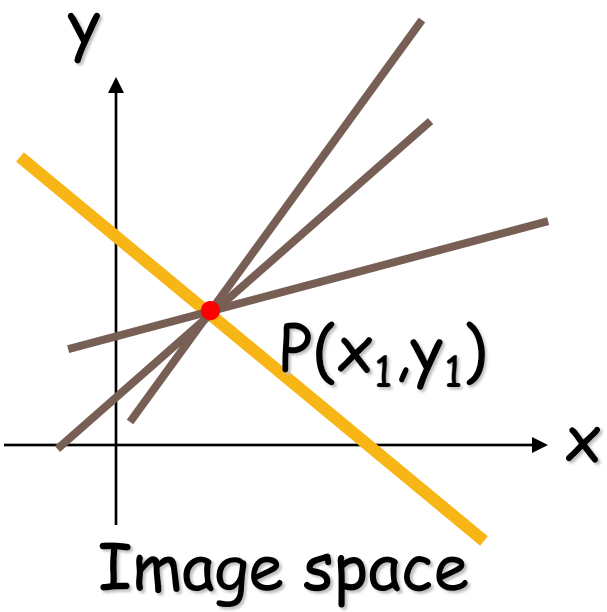
Image space

$y = mx + c$

Fix (m,c), Vary (x,y) - Line

Fix (x_1,y_1), Vary (m,c) – Lines thru a Point

$y_1 = m\,x_1 + c$



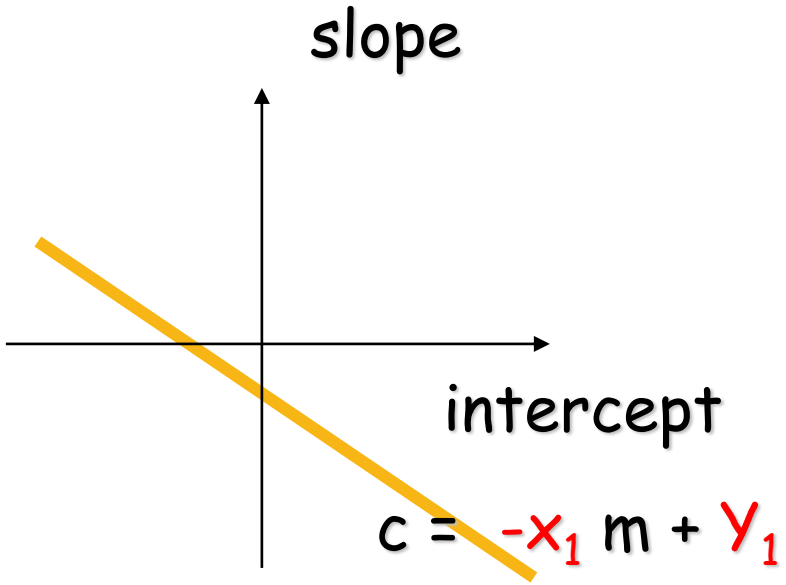
Hough Transform

- **Image and Parameter Spaces**

Parameter space

$y_1 = m\,x_1 + c$ Can be re-written as: $c = -x_1\,m + y_1$

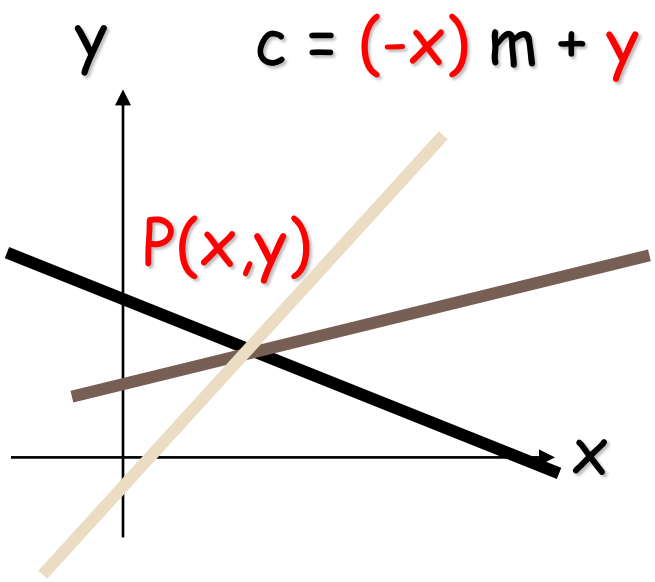
Fix $(-x_1, y_1)$, Vary (m,c) - Line $c = -x_1\,m + y_1$

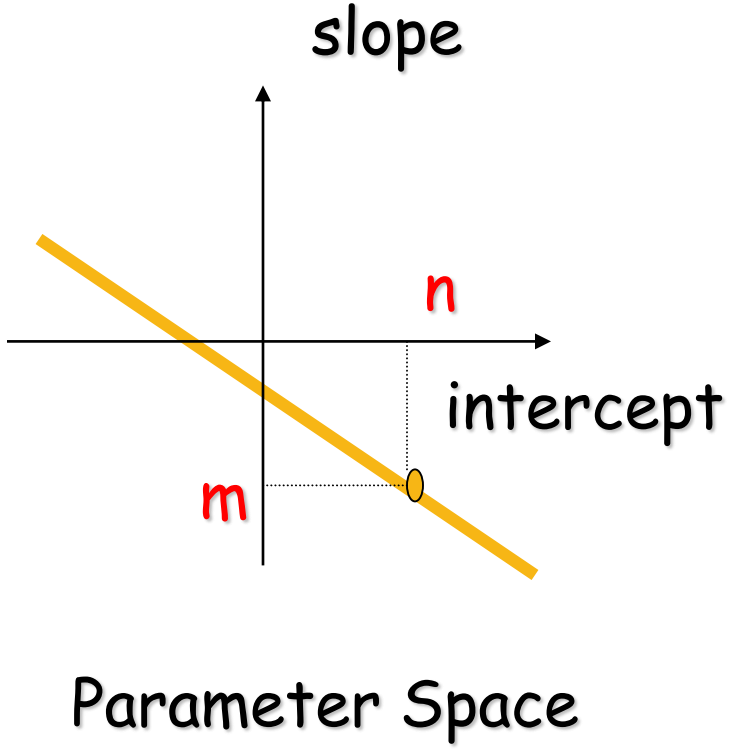


Parameter Space

Hough Transform

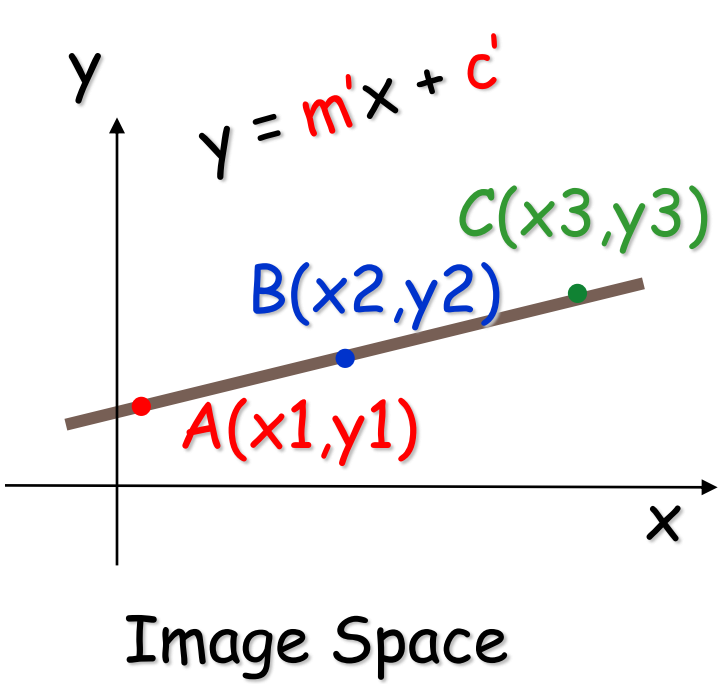
- **Image and Parameter Spaces**
  - Given an edge point, there is an infinite number of lines passing through it (Vary m and c).
    - These lines can be represented as a single line in parameter space.

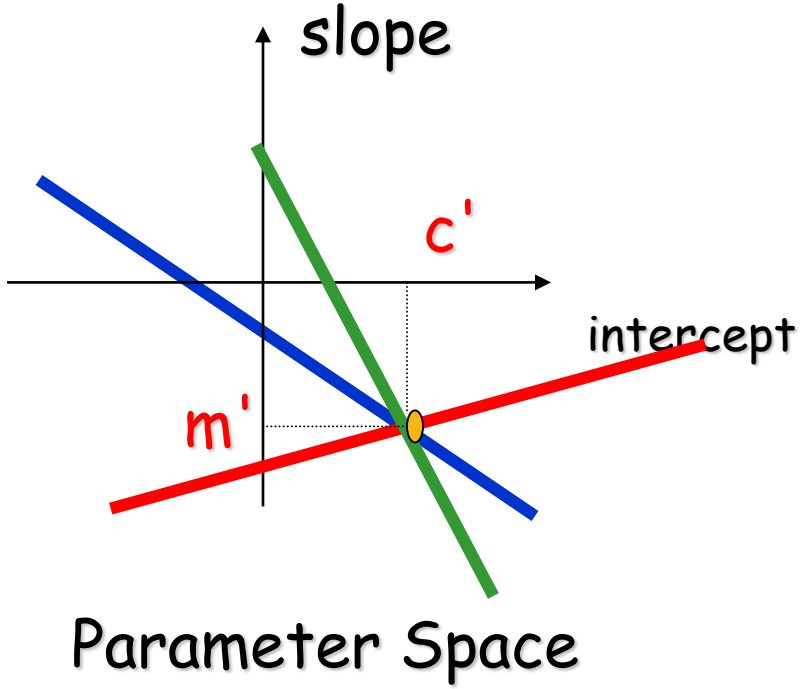




Hough Transform

- **Image and Parameter Spaces**
  - Given a set of collinear edge points, each of them have associated a line in parameter space.
    - These lines intersect at the point (m′,c′) corresponding to the parameters of the line in the image space.





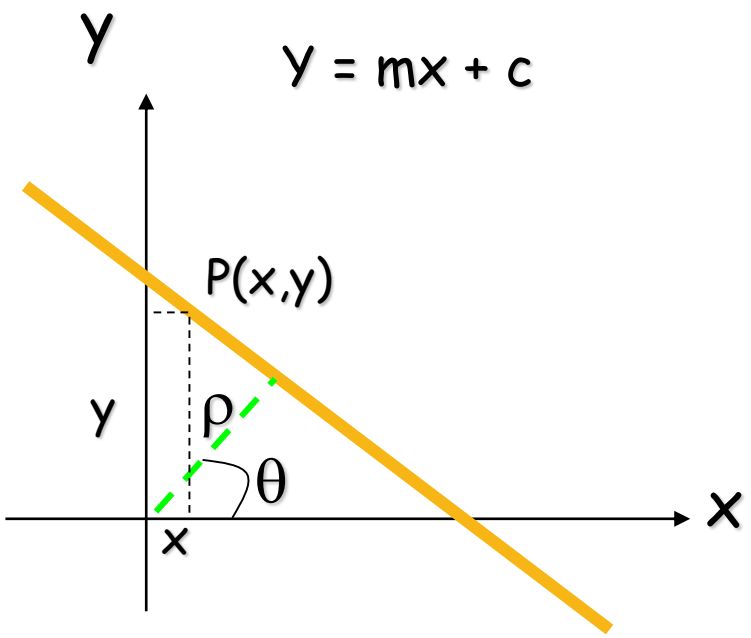
Hough Transform

Image and Parameter Spaces

- Image Space
  - Lines
  - Points
  - Collinear points
- Parameter Space
  - Points
  - Lines
  - Intersecting lines

Hough Parameterization

- **Practical Issues**
  - The slope of the line is $-\infty<m<\infty$, parameter space is INFINITE
  - The representation $y = mx + c$ does not express lines of the form $x = k$
- **Solution:** Use the "Normal" equation of a line:



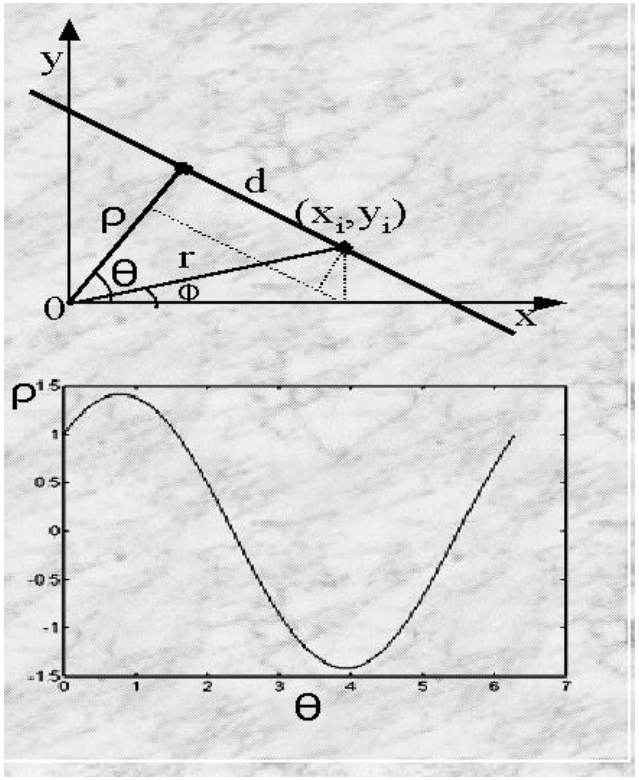
$\rho = x\cos\theta + y\sin\theta$

θ Is the line orientation

ρ Is the distance between the origin and the line

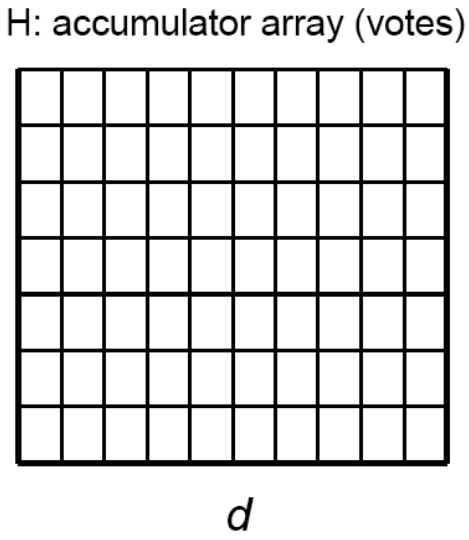
Consequence:

- A Point in Image Space is now represented as a sinusoid
  - $\rho = x\cos\theta + y\sin\theta$
- Use the parameter space (ρ, θ)
- The new space is FINITE
  - $0 < \rho < D$, where D is the image diagonal.
  - $0 < \theta < \pi$
- The new space can represent all lines
  - Y = k is represented with $\rho = k$, $\theta = 90$
  - X = k is represented with $\rho = k$, $\theta = 0$



Hough Transform Algorithm

Input is an edge image (E(i,j)=1 for edgels)



1. Discretize θ and ρ in increments of $d\theta$ and $d\rho$. Let A(D,T) be an array of integer accumulators, initialized to 0.

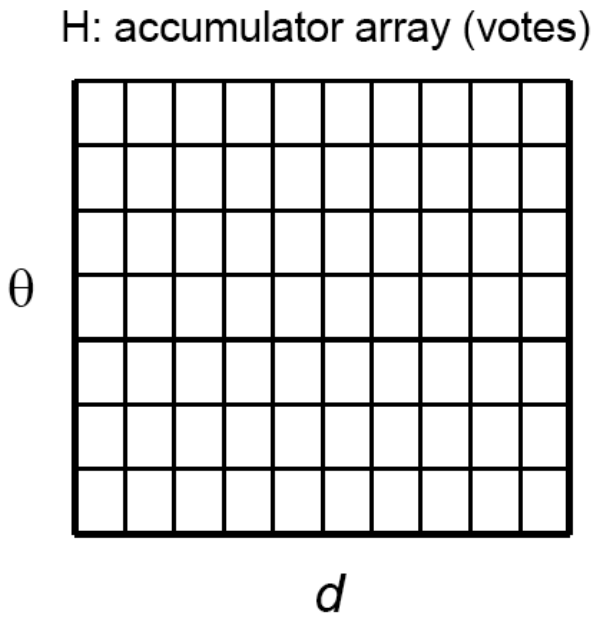
2. For each pixel E(i,j)=1 and h=1,2,…T do
   1. $\rho = i \cos(h * d\theta) + j \sin(h * d\theta)$
   2. Find closest integer d corresponding to ρ
   3. Increment counter A(d,h) by one i.e. A(d,k) = A(d,k) +1

Hough Transform Algorithm

3. Find the value(s) of (d, θ) where H[d, θ] is maximum

4. The detected line in the image is given by

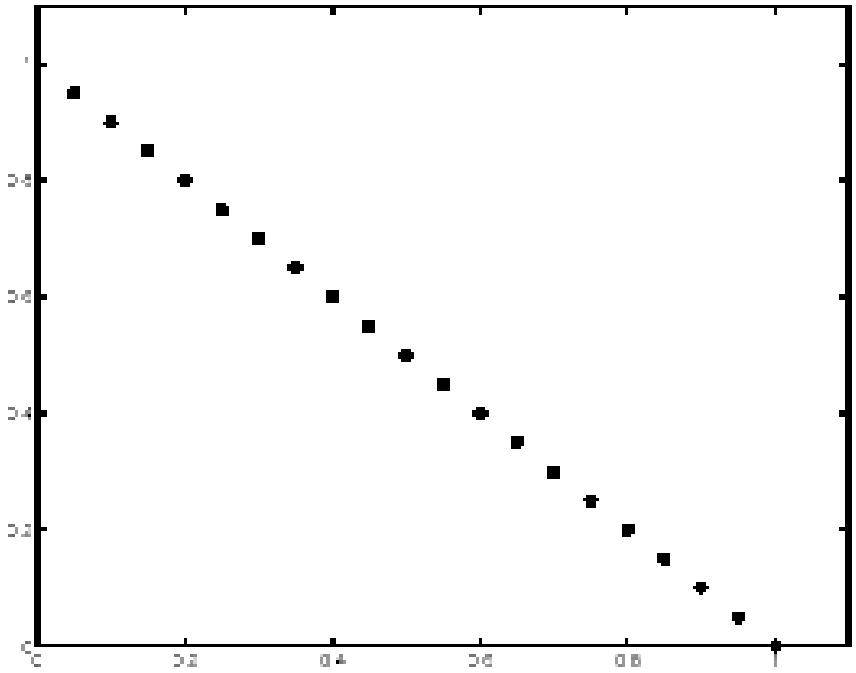
$$d = x \cos\theta + y \sin\theta$$



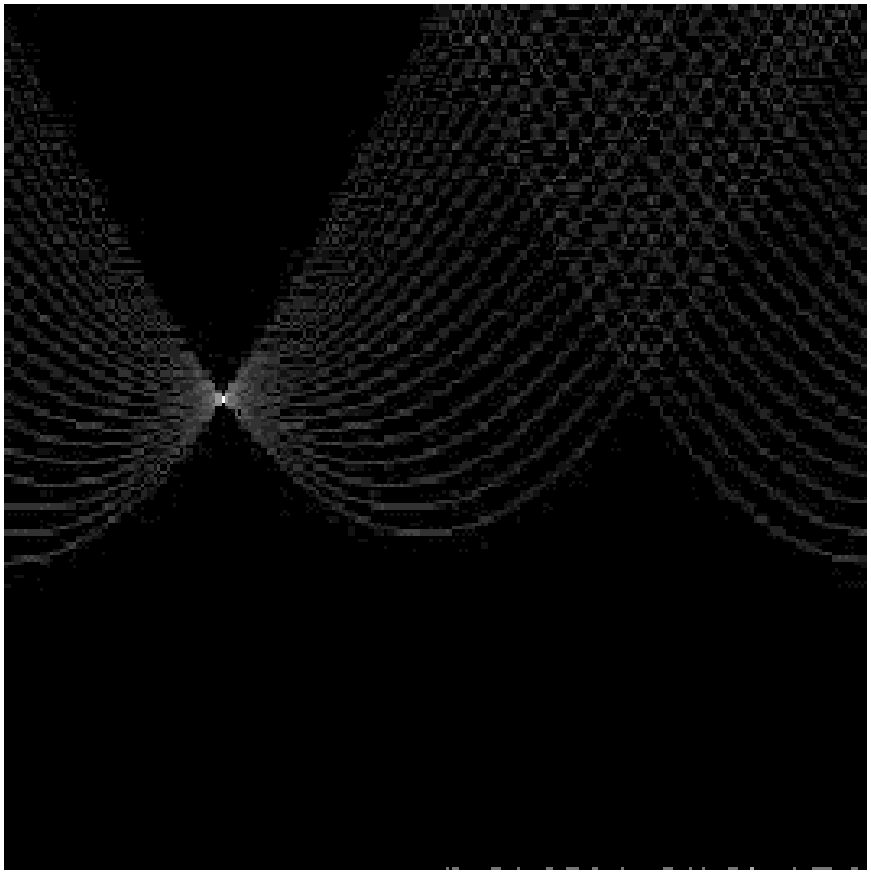
Hough Transform Speed Up

- If we know the orientation of the edge – usually available from the edge detection step
  - We fix theta in the parameter space and increment **only one** counter!
  - We can allow for orientation uncertainty by incrementing a *few* counters around the "nominal" counter.

Hough Transform - Basic illustration



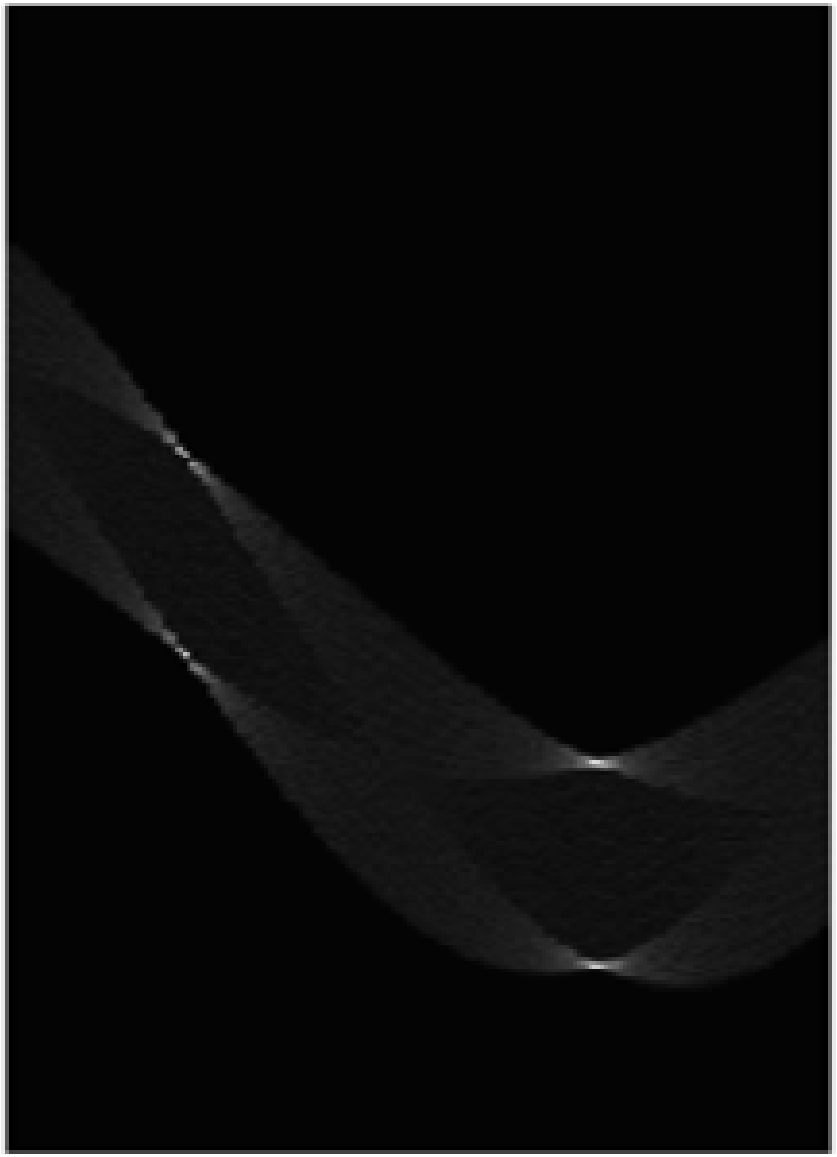
features



votes

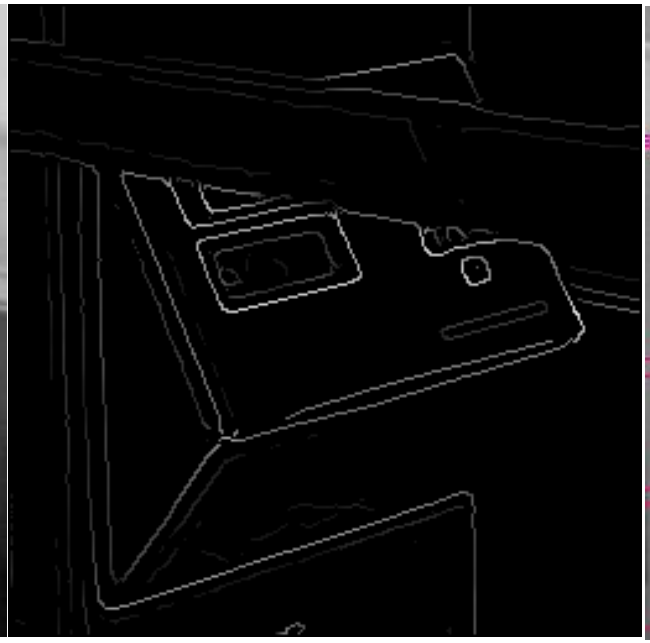
Hough Transform- Other shapes

Square

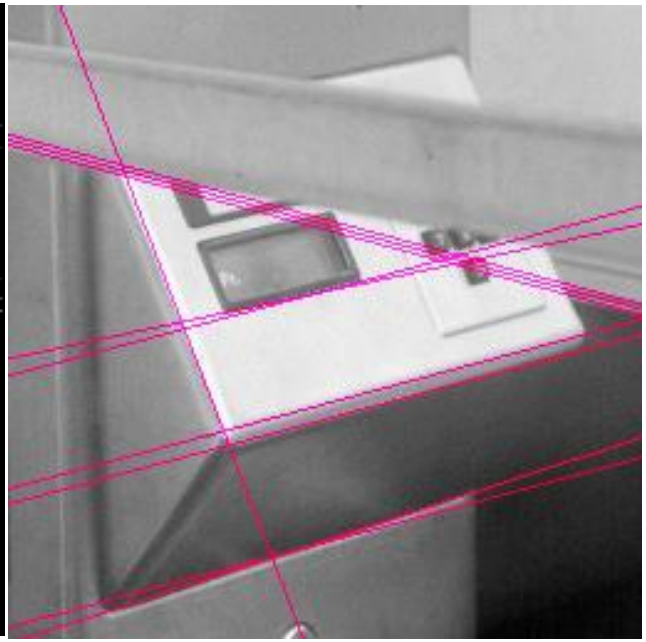


Hough Transform- Real World Example

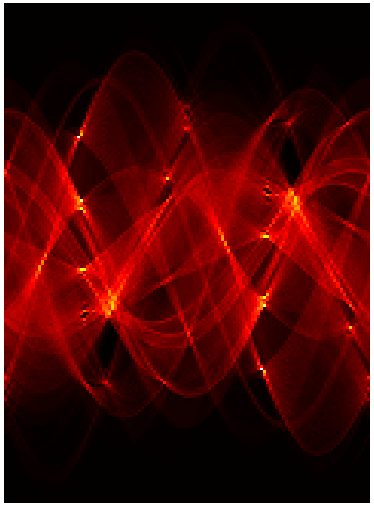
Original



Edge Detection

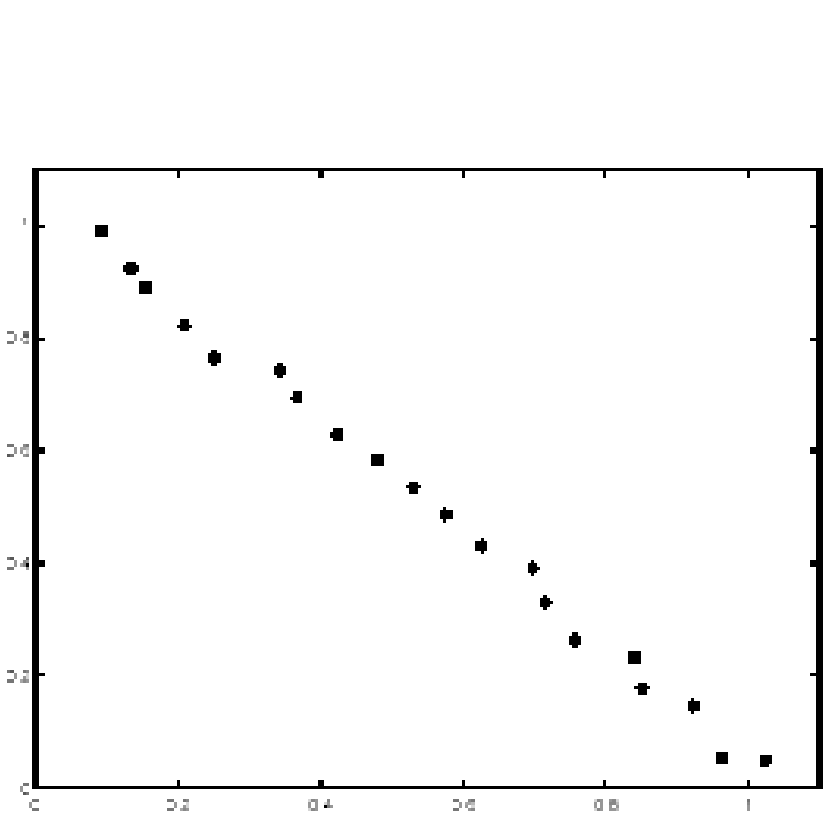


Found Lines

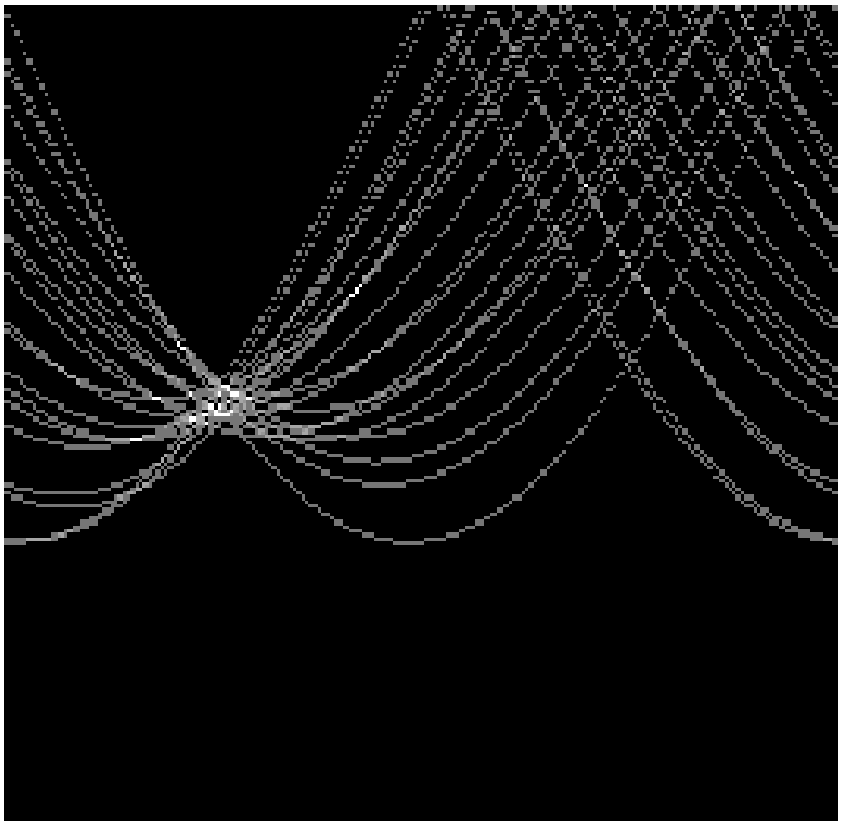


Parameter Space

Hough Transform- Effect of noise



features

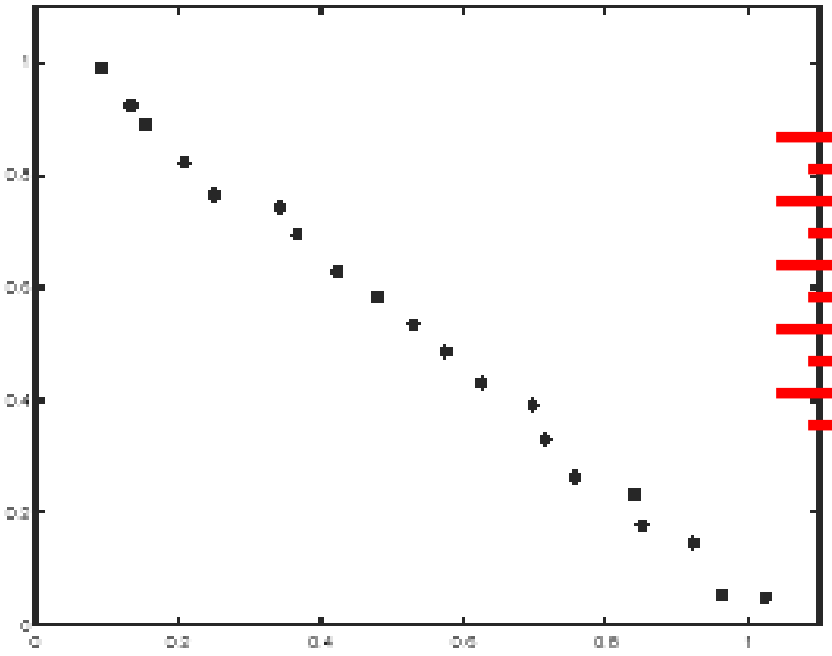


votes

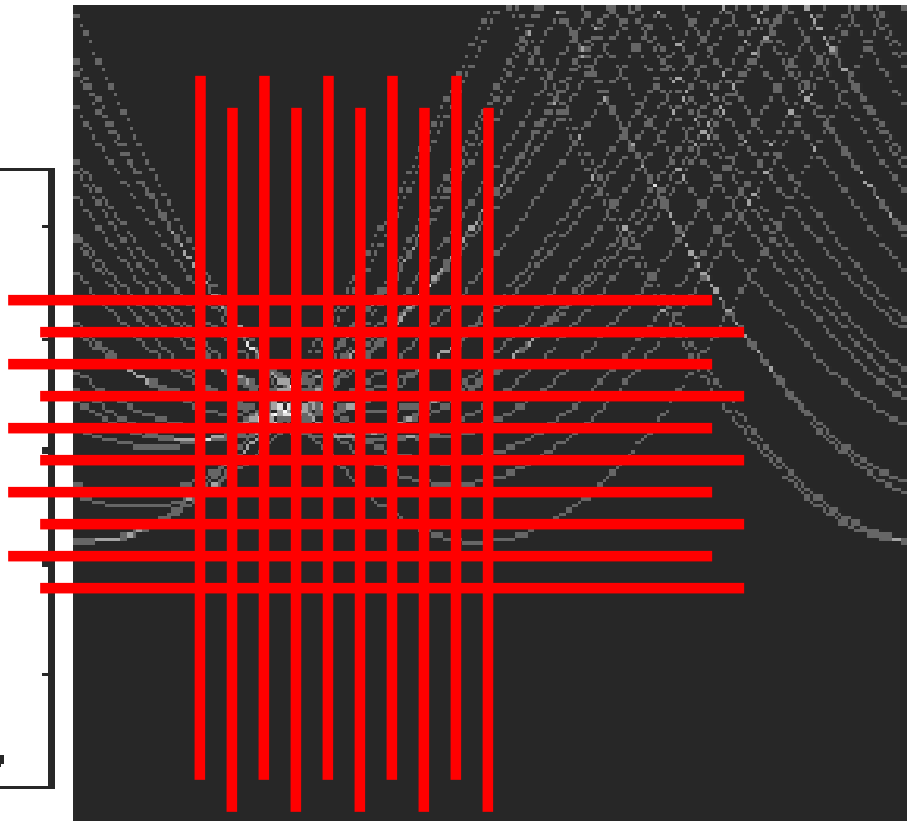
- Peak gets fuzzy and hard to locate

Hough transform - Effect of noise

Noisy data



features



votes

Need to adjust grid size or smooth

Hough transform: Discussion

- Pros
  - Can deal with non-locality and occlusion
  - 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
- Hough transform vs. RANSAC

Practical details

- Try to get rid of irrelevant features
  - Take only edge points with significant gradient magnitude
- Choose a good grid / discretization
  - Too coarse: large votes 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)
- Who belongs to which line?
  - Tag the votes

Image alignment

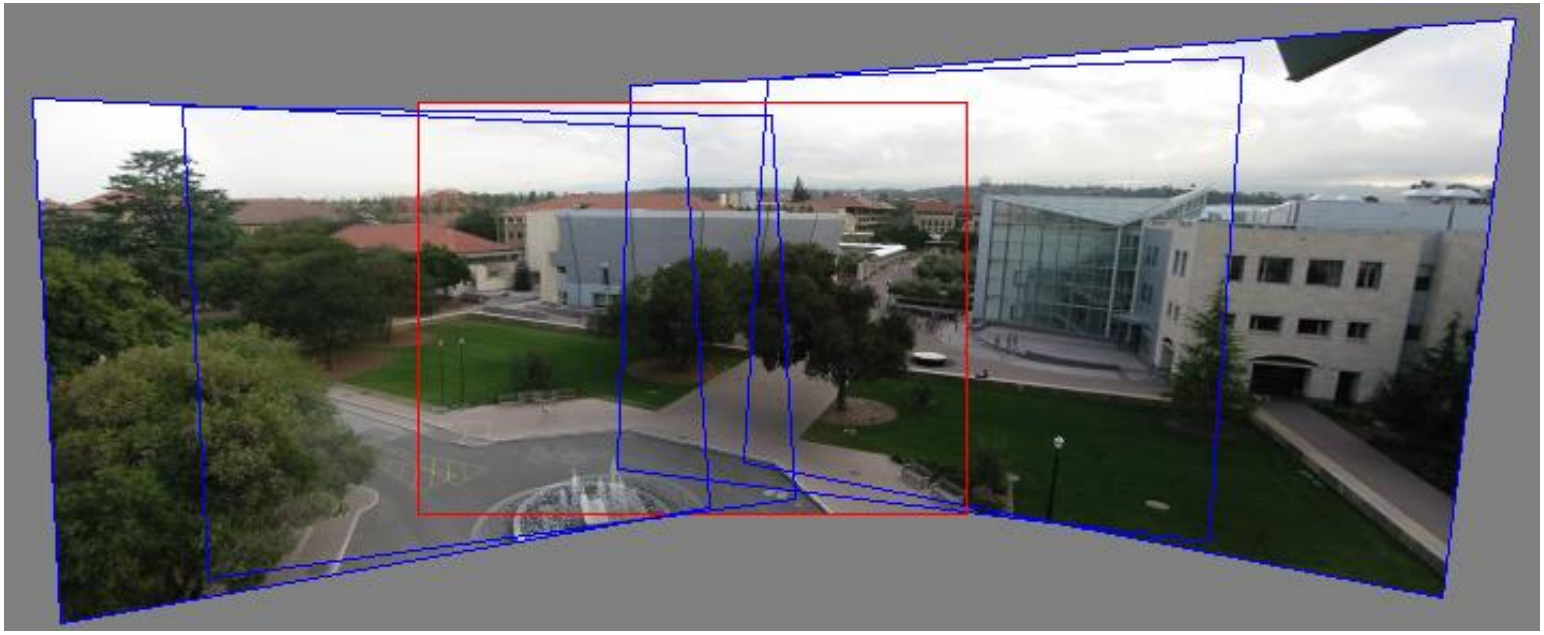
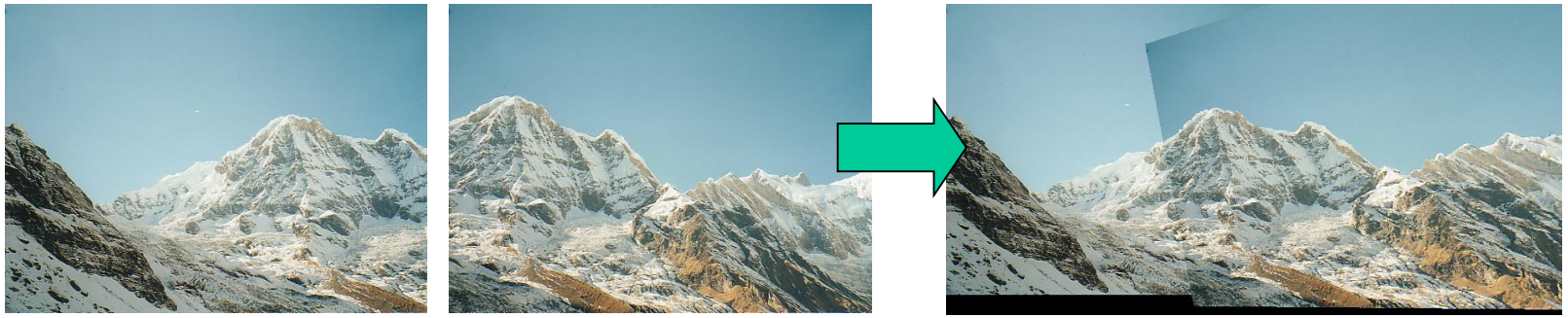
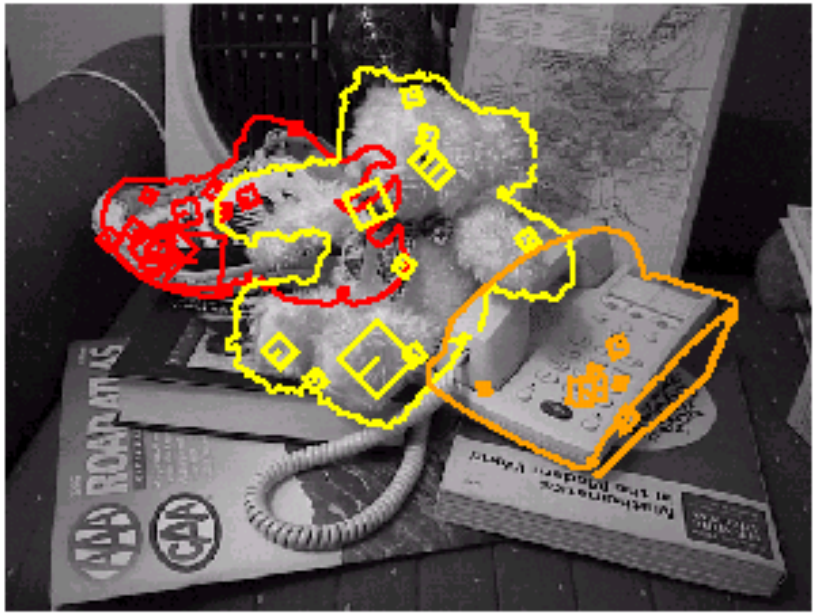


Image from http://graphics.cs.cmu.edu/courses/15-463/2010_fall/

Image alignment: Applications

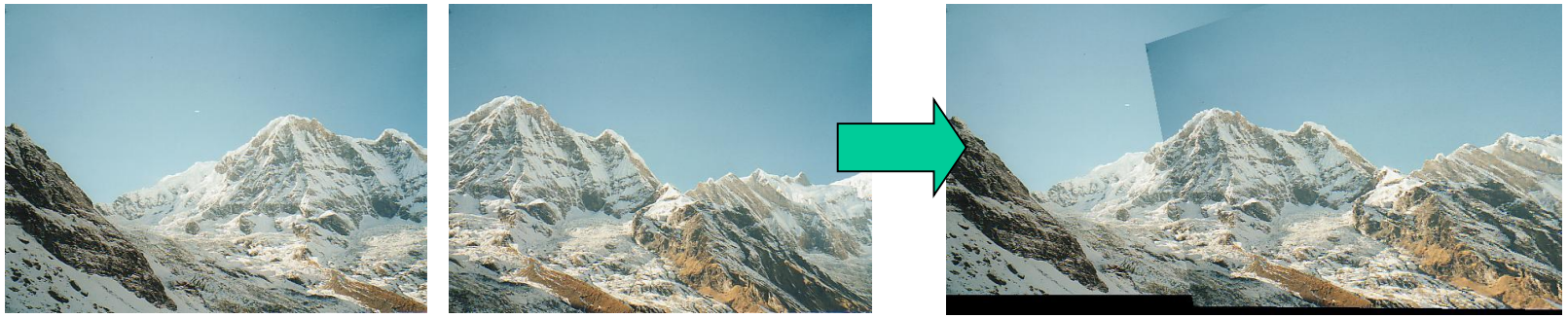


Panorama stitching



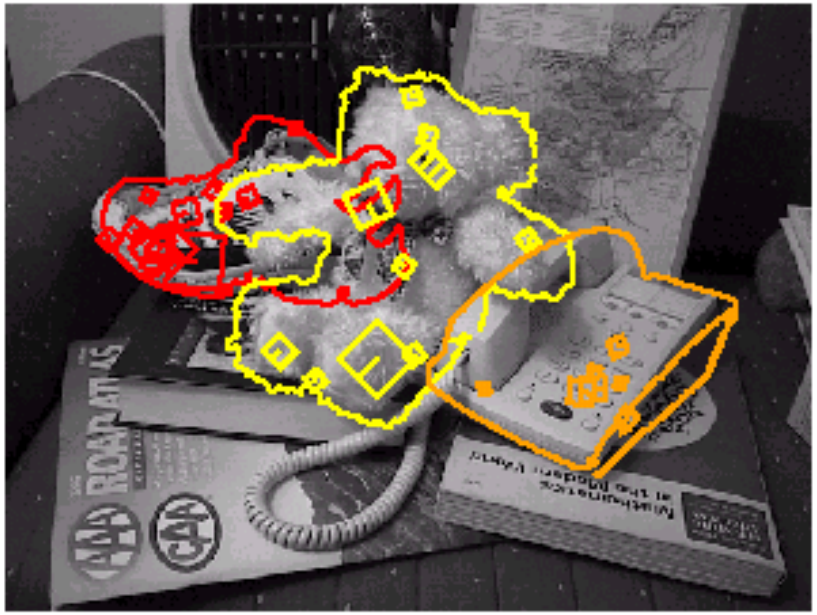
Recognition of object instances

Image alignment: Challenges



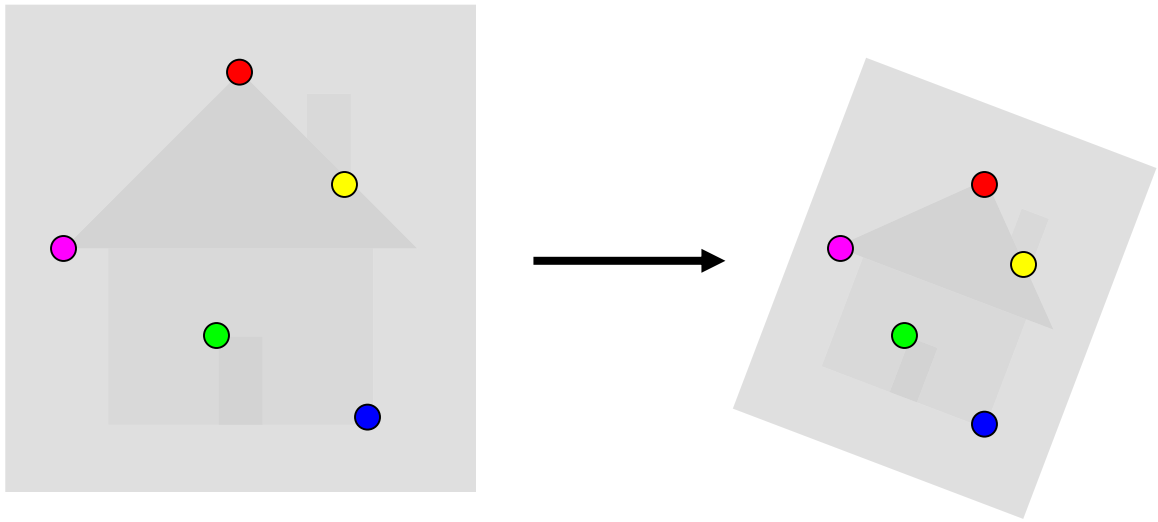
Small degree of overlap

Intensity changes



Occlusion, clutter

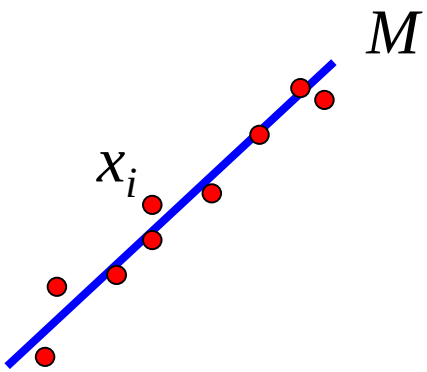
Image alignment



- Two families of approaches:
  - **Direct (pixel-based) alignment**
    - Search for alignment where most pixels agree
  - **Feature-based alignment**
    - Search for alignment where *extracted features* agree
    - Can be verified using pixel-based alignment

Alignment as fitting

- Previous lectures: fitting a model to features in one image

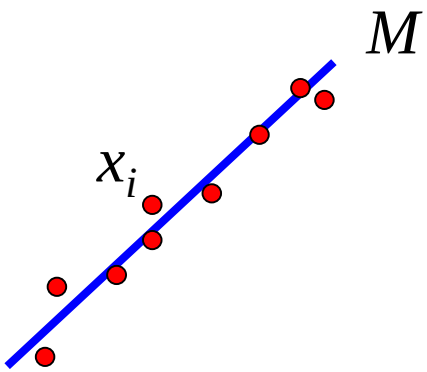


Find model *M* that minimizes

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

Alignment as fitting

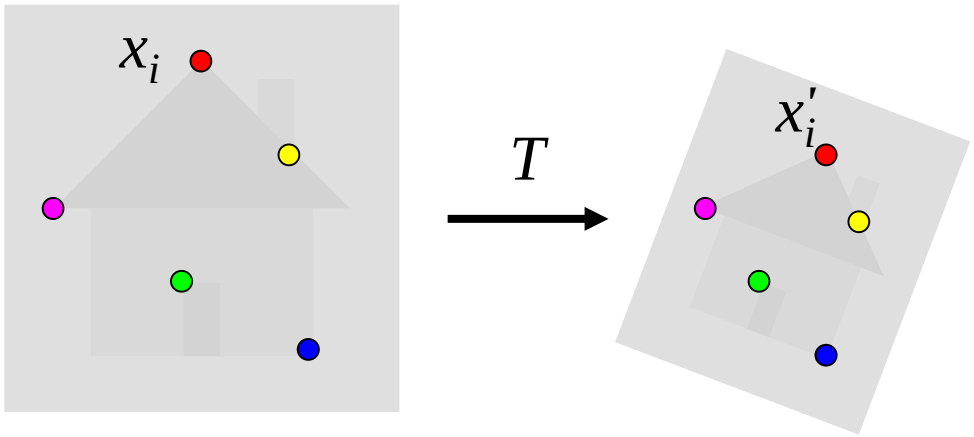
- Previous lectures: fitting a model to features in one image



Find model *M* that minimizes

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

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

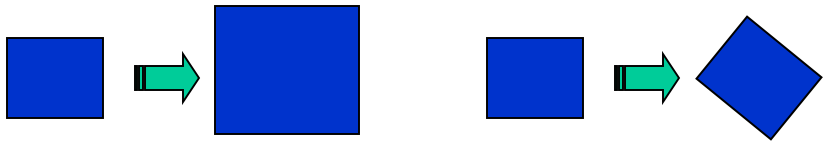


Find transformation *T* that minimizes

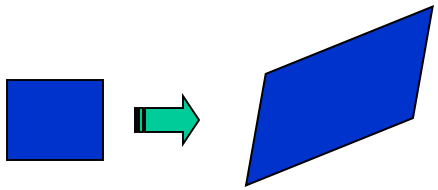
$$\sum_i \text{residual}(T(x_i), x_i')$$

2D transformation models

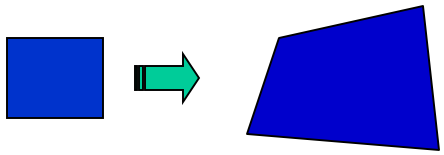
- Similarity (translation, scale, rotation)



- Affine



- Projective (homography)



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

Source: Alyosha Efros

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

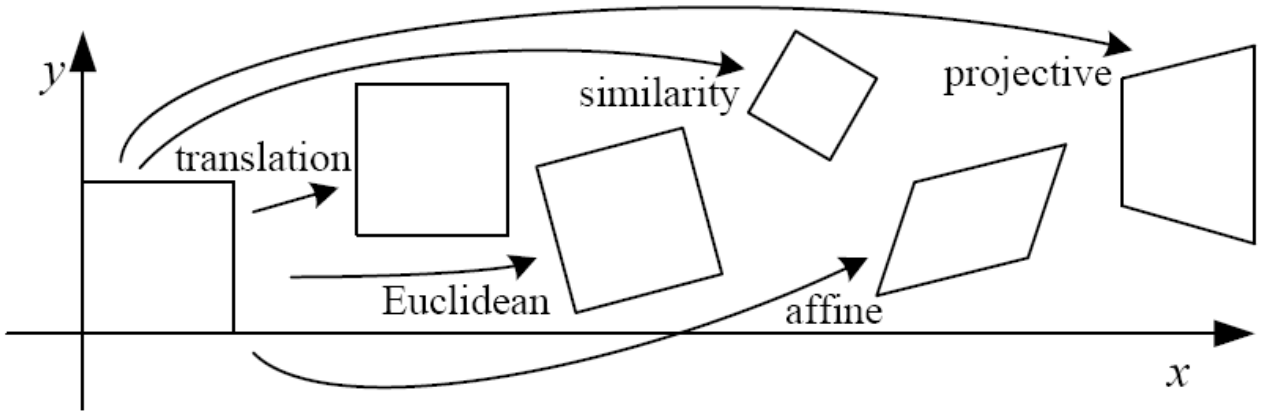
Projective Transformations

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

Projective transformations:

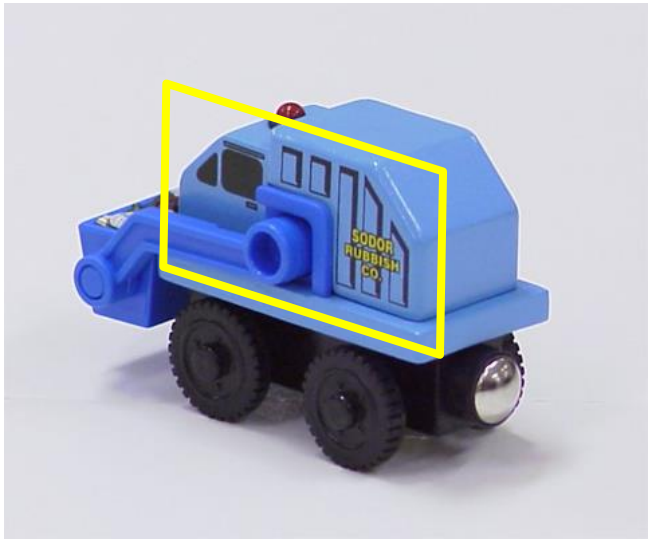
- Affine transformations, and
- Projective warps

Parallel lines do not necessarily remain parallel



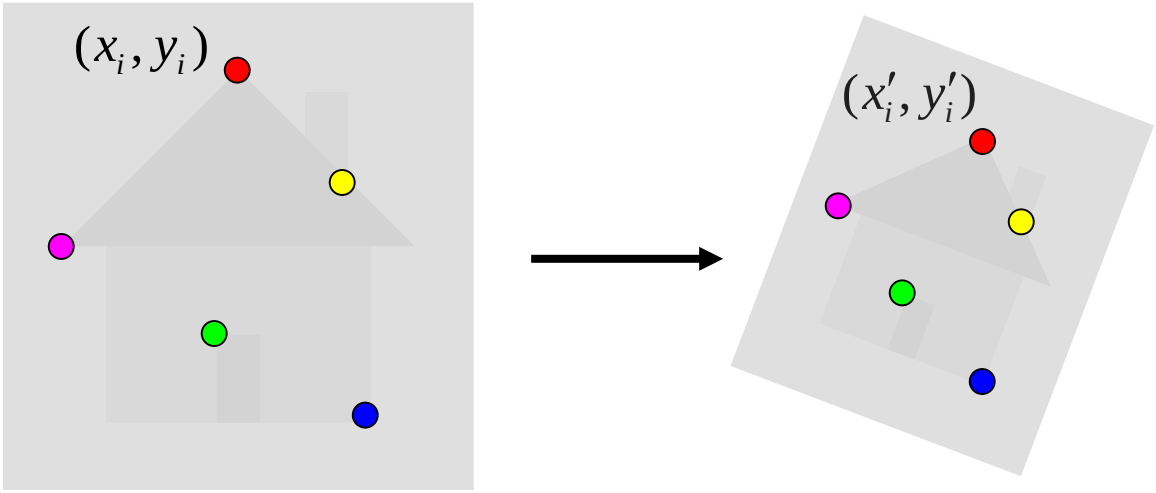
Let’s start with affine transformations

- Simple fitting procedure (linear least squares)
- Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras
- 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?



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

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

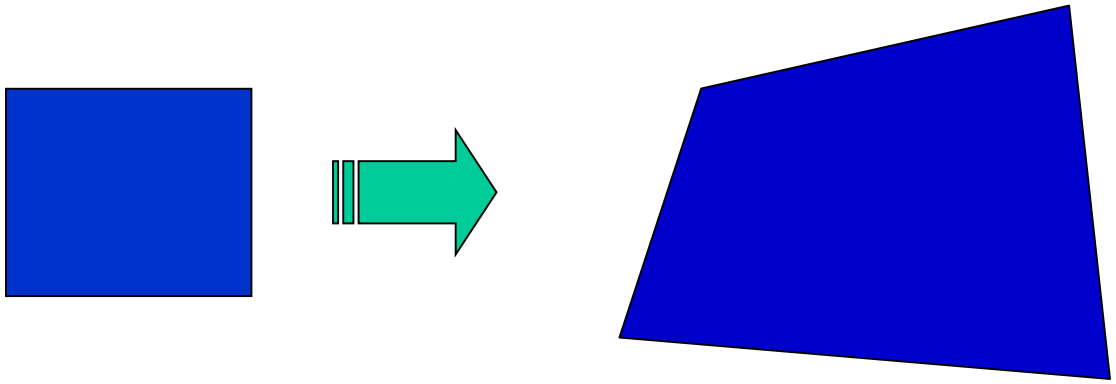
Fitting an affine transformation

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

- 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

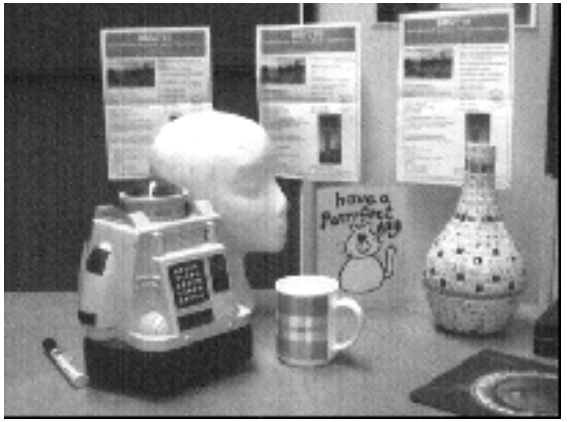
- **Homography:** plane projective transformation (transformation taking a quad to another arbitrary quad)



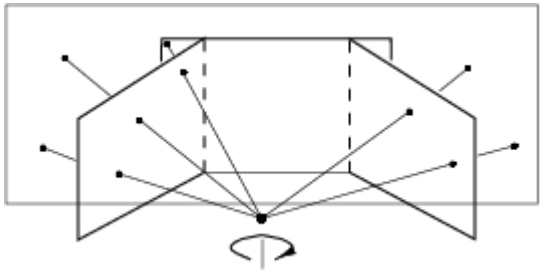
Homography

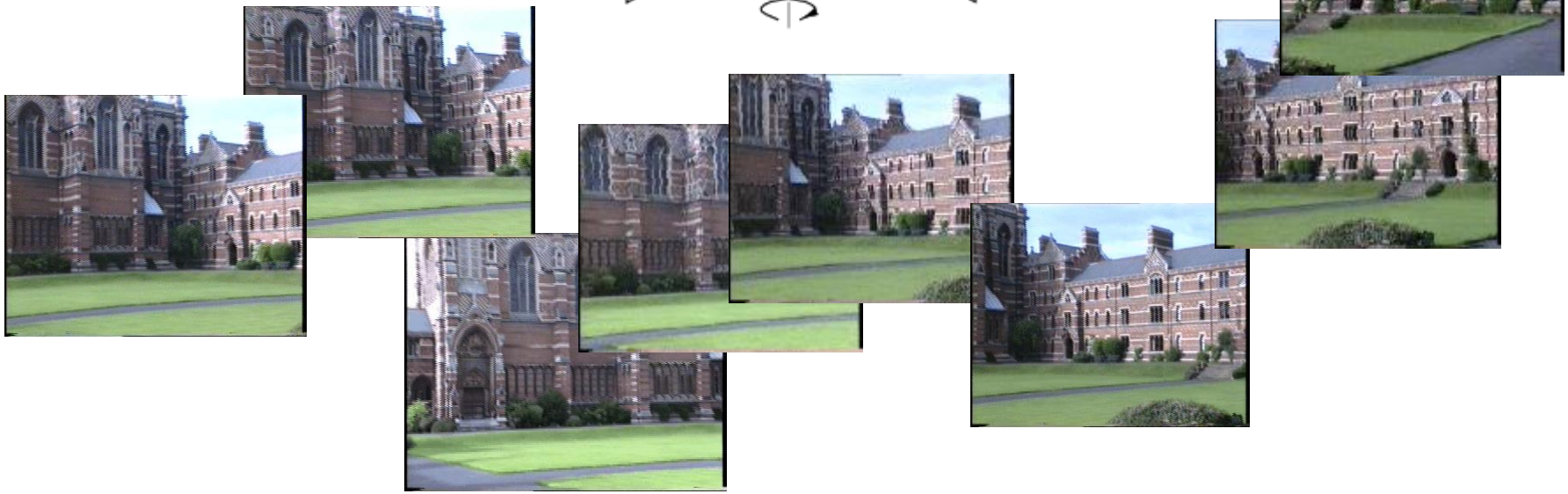
- The transformation between two views of a planar surface

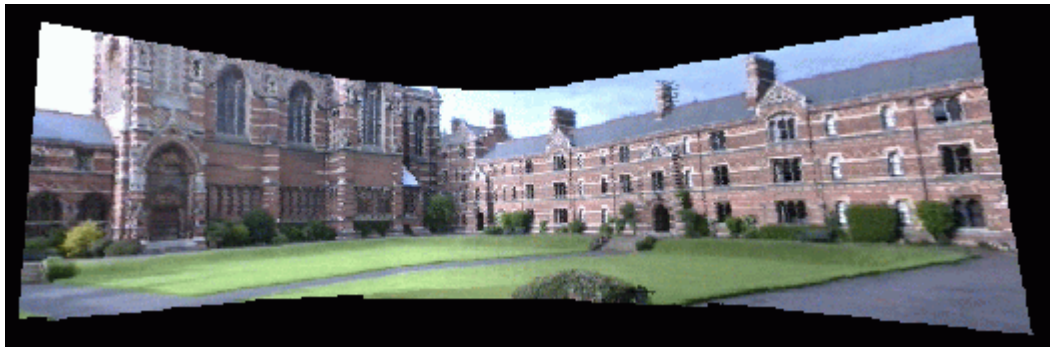

- The transformation between images from two cameras that share the same center

Application: Panorama stitching







Source: Hartley & Zisserman

Fitting a homography

- Recall: homogeneous coordinates

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

Converting *to* homogeneous image coordinates

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

Converting *from* homogeneous image coordinates

Fitting a homography

- Recall: homogeneous coordinates

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

Converting *to* homogeneous image coordinates

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

Converting *from* homogeneous image coordinates

- Equation for homography:

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

Solving for homographies

$$\mathbf{p' = Hp}$$

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

Can set scale factor $h_{33}=1$. So, there are 8 unknowns.

Set up a system of linear equations:

$$\mathbf{Ah = b}$$

where vector of unknowns $h = [h_{11}, h_{12}, \ldots, h_{32}]^T$

Need at least 8 eqs, but the more the better…

Solve for h. If overconstrained, solve using least-squares:

$$\min \left\| Ah - b \right\|^2$$

Fitting a homography

- Equation for homography:

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

$$\mathbf{x}_i' \times \mathbf{H}\mathbf{x}_i = 0$$

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{h_1}^T \mathbf{x}_i \\ \mathbf{h_2}^T \mathbf{x}_i \\ \mathbf{h_3}^T \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} y_i' \mathbf{h}_3^T \mathbf{x}_i - \mathbf{h}_2^T \mathbf{x}_i \\ \mathbf{h}_1^T \mathbf{x}_i - x_i' \mathbf{h}_3^T \mathbf{x}_i \\ x_i' \mathbf{h}_2^T \mathbf{x}_i - y_i' \mathbf{h}_1^T \mathbf{x}_i \end{bmatrix}$$

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

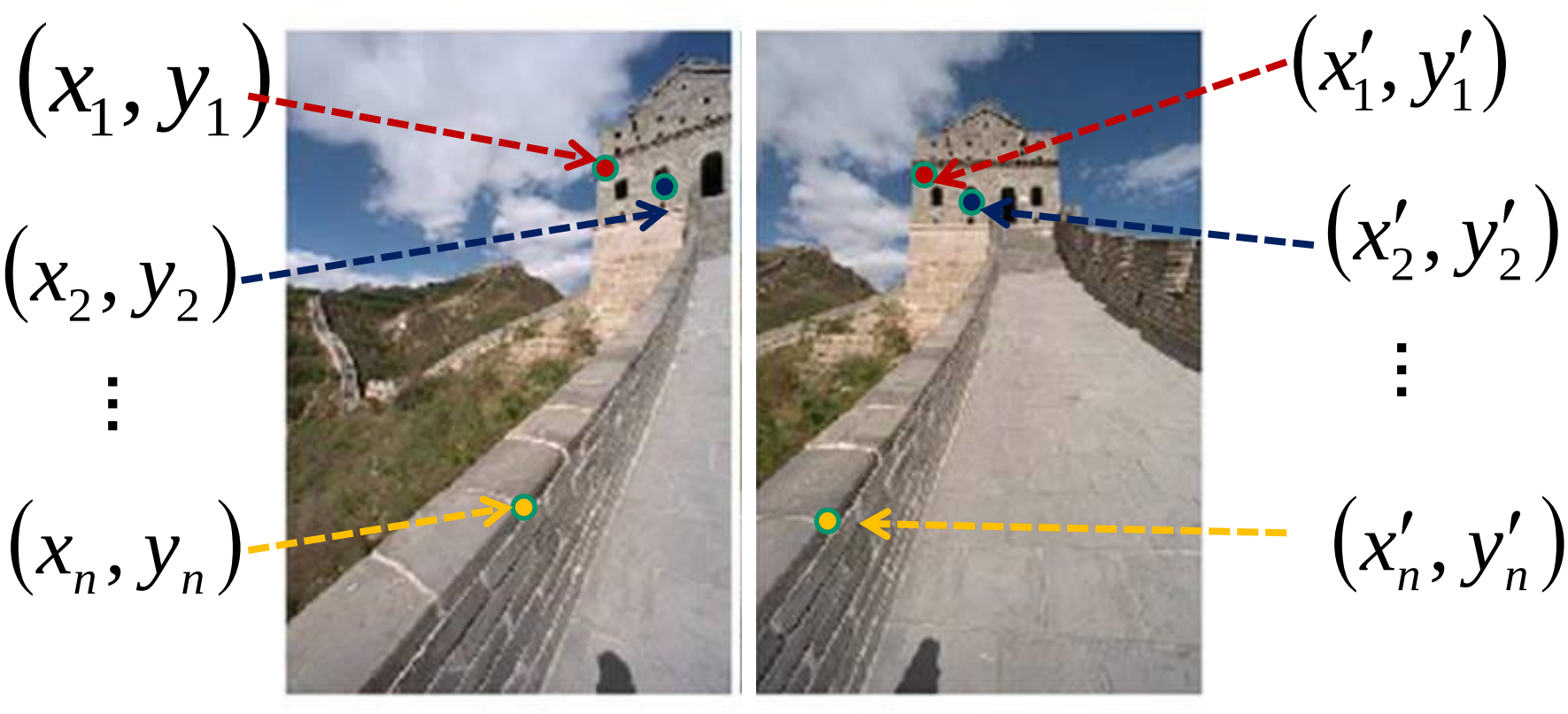
3 equations, only 2 linearly independent

Direct linear transform

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

- H has 8 degrees of freedom (9 parameters, but scale is arbitrary)
- One match gives us two linearly independent equations
- Four matches needed for a minimal solution
- More than four: homogeneous least squares

Homography



To compute the homography given pairs of corresponding points in the images, we need to set up an equation where the parameters of H are the unknowns…