CSCI 4830 / 5722
Computer Vision

University of Colorado **Boulder**

CSCI 4830 / 5722
Computer Vision

Dr. Ioana Fleming
Spring 2019
Lecture 13

Reminders

Submissions:

- Homework 2: due Wed 2/13 at 11 pm

Readings:

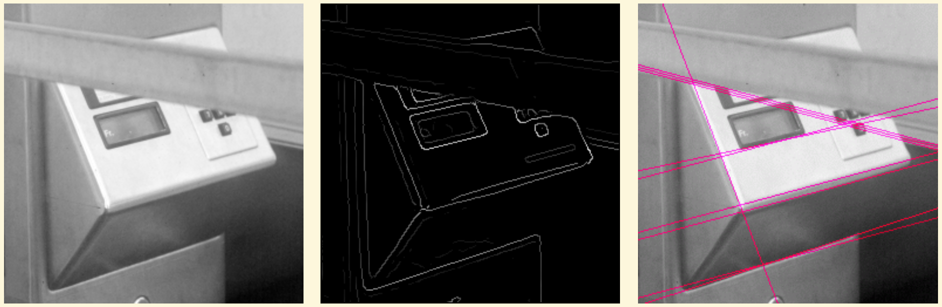
- Szeliski:
  - chapter 6.1 (Least squares, ITP, RANSAC)
- P&F:
  - chapter 10.2 (Least quares), 10.4 (RANSAC)

Today

- Fitting
- Least squares – again
- RANSAC

Fitting

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



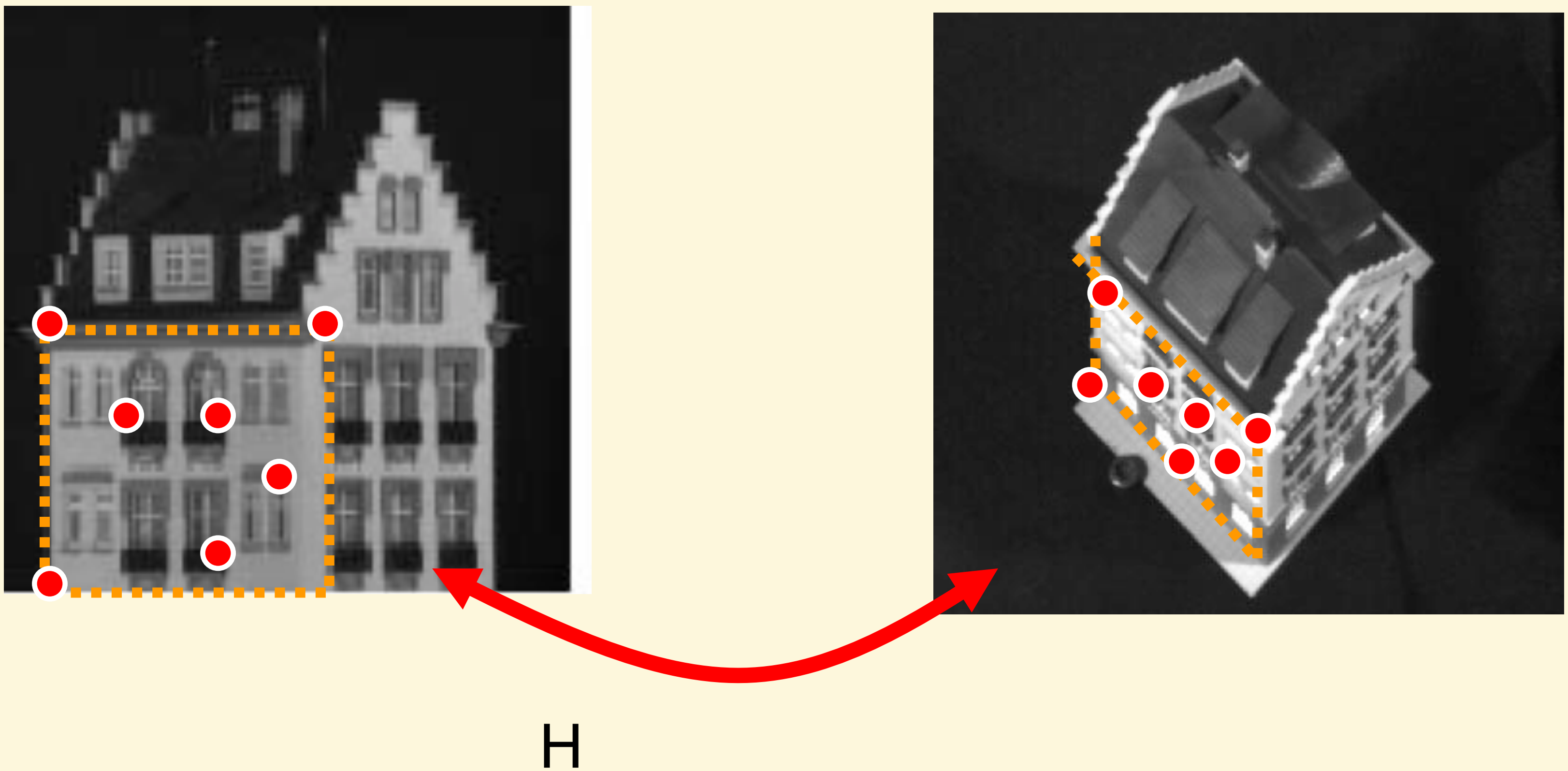
Alignment: find the parameters of the transformation that best align matched points

[Fig from Marszalek & Schmid, 2007]

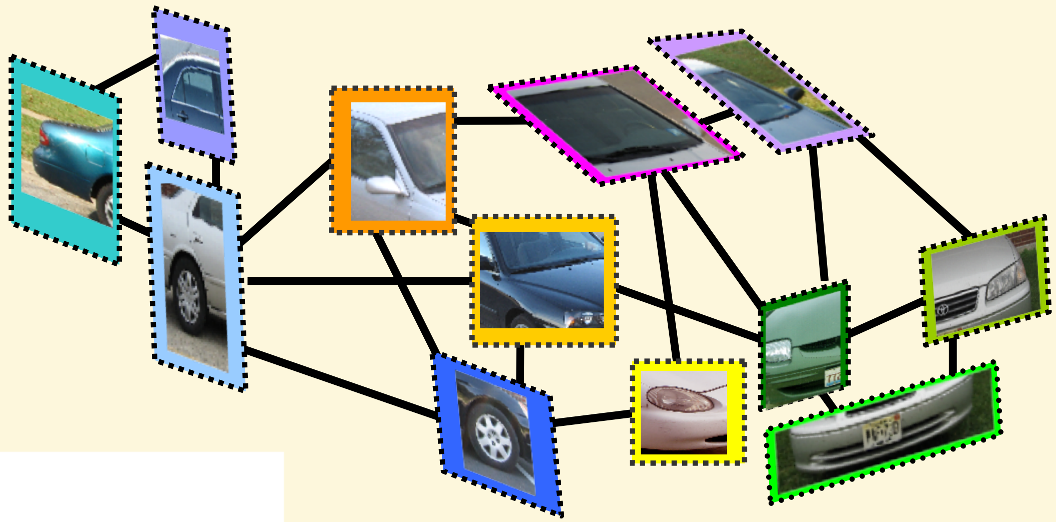
Example: Fitting lines

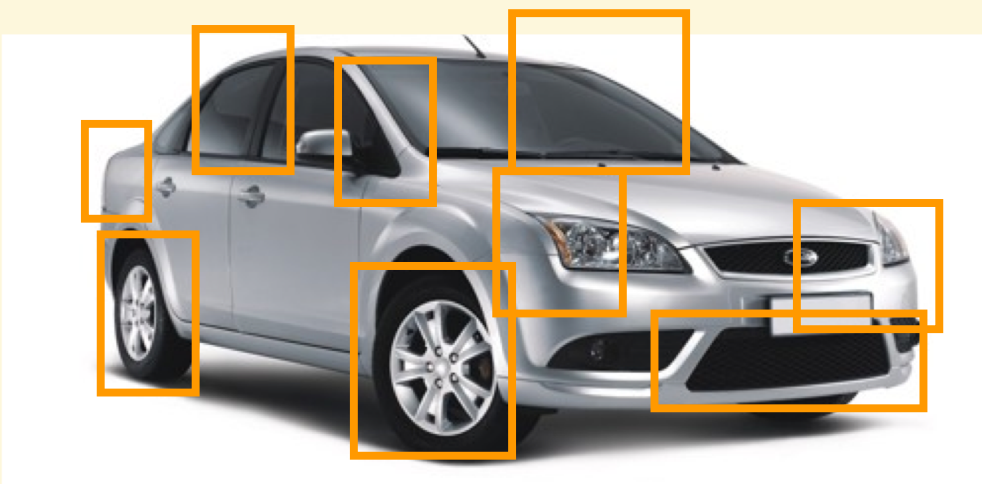
Slide from Silvio Savarese

Example: Estimating an homographic transformation



Example: fitting a 3D object model





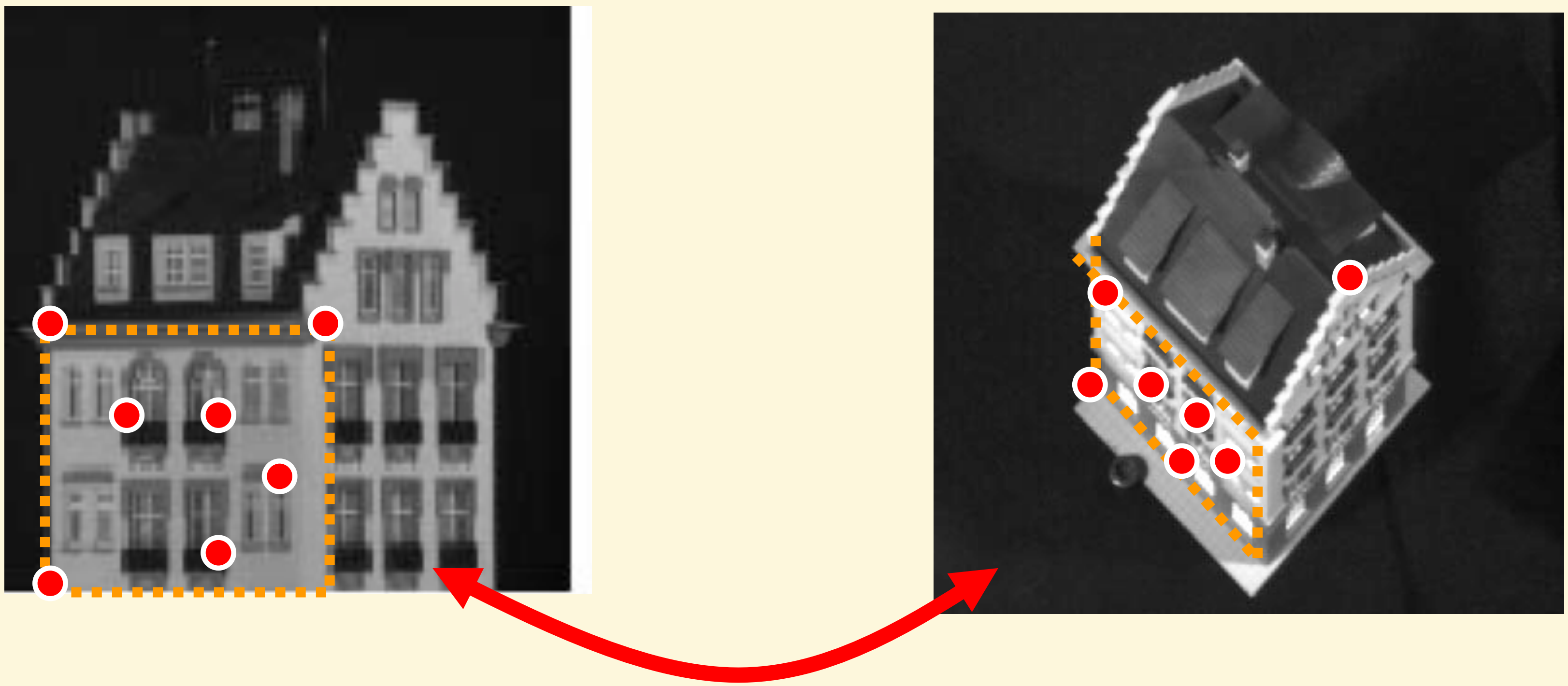
University of Colorado **Boulder**

Slide from Silvio Savarese

Critical issues: noisy data

Slide from Silvio Savarese

Critical issues: outliers

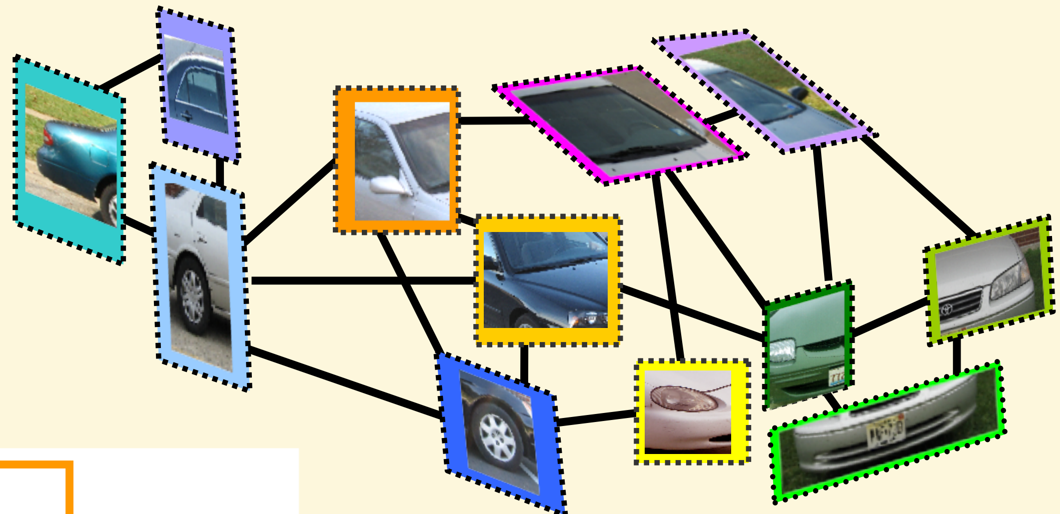


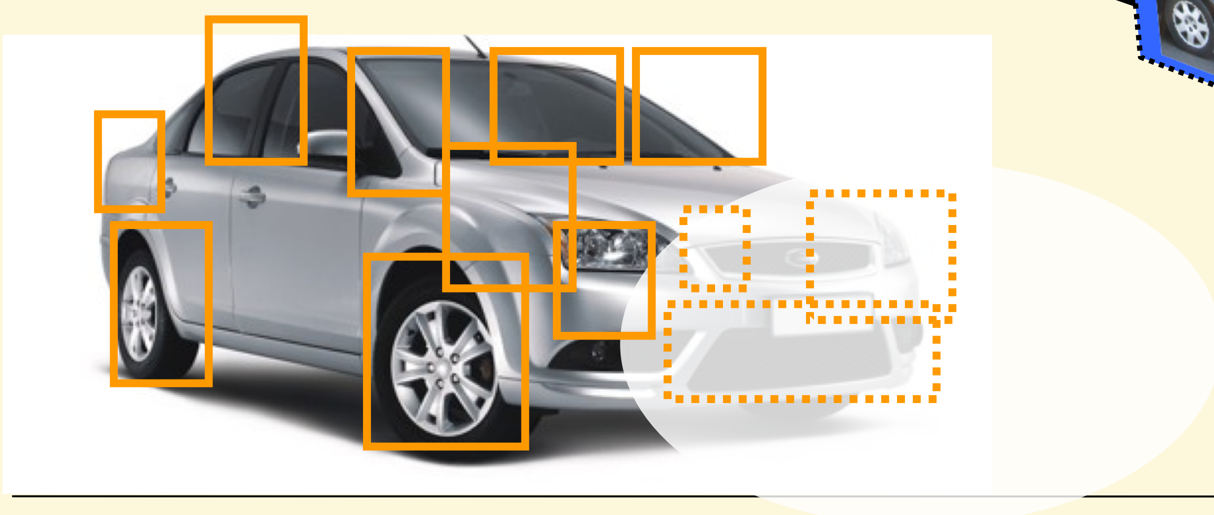
H

University of Colorado **Boulder**

Slide from Silvio Savarese

Critical issues: missing data (occlusions)





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

Slide from Derek Hoiem

A Quick Aside: Least Squares

$\boxed{\text{Algebraically, find } x \text{ s.t. } Ax = b.}$

1. If $A \in \mathbb{R}^{n\times n}$ and invertible, $x = A^{-1}b$
2. If $A \in \mathbb{R}^{m\times n}$ the solution is not unique if $m < n$. The solution is always of the form $x = x_0 + \text{nullspace}(A)$.
3. If $A \in \mathbb{R}^{m\times n}$ and $m > n$, then the solution might or might not exist. We look for a solution s.t. $\|Ax - b\|^2$ is minimized.

$$f(x) = \|Ax-b\|^2 \tag{1a}$$
$$= (Ax-b)^\top (Ax-b) \tag{1b}$$
$$= x^\top A^\top Ax - x^\top A^\top b - b^\top Ax + b^\top b. \tag{1c}$$
$$\nabla f(x) = A^\top(Ax-b) + A^\top(Ax-b) \tag{1d}$$
$$= 2A^\top(Ax-b) \tag{1e}$$
$$\Rightarrow A^\top Ax = A^\top b \tag{1f}$$
$$x = (A^\top A)^{-1}A^\top b \tag{1g}$$

This might not work all the time, only if $\text{rank}(A) = n$.

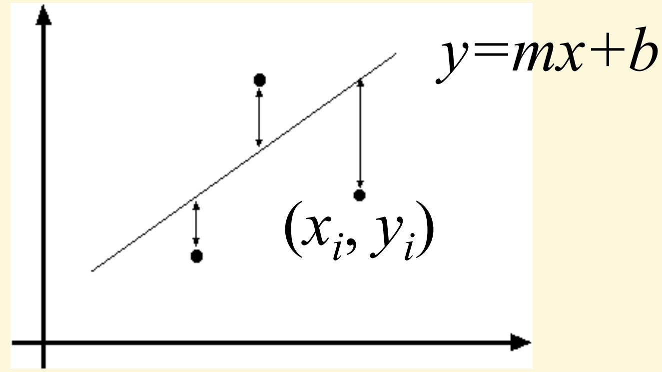
$(A^\top A)^{-1}A^\top$ **= the *pseudoinverse* of *A***

`Ainv = pinv(A)` in Matlab

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

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



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

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

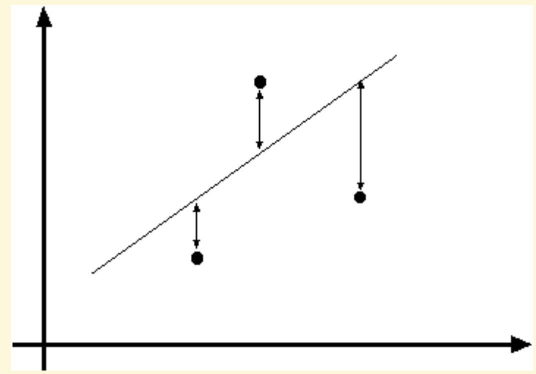
$$\frac{dE}{dy} = 2\mathbf{A}^T \mathbf{Ap} - 2\mathbf{A}^T \mathbf{y} = 0$$

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

Matlab: `p = A \ y;`

Problem with "vertical" least squares

- Not rotation-invariant
- Fails completely for vertical lines



Slide from S. Lazebnik

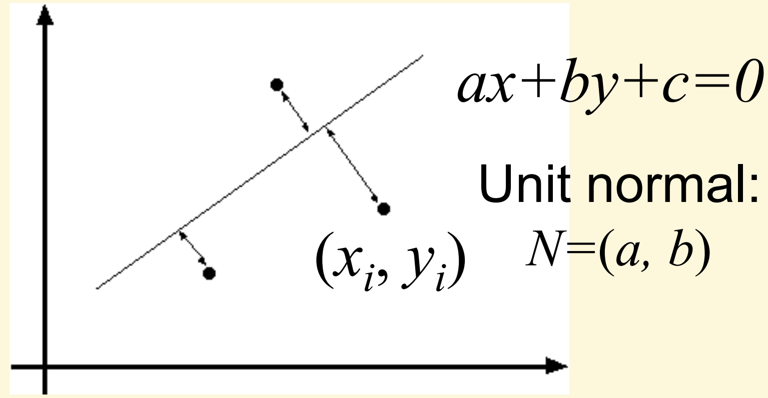
Total least squares

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

Distance between point (x_i, y_i) is

$$|ax_i + by_i + c|$$

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



Total least squares

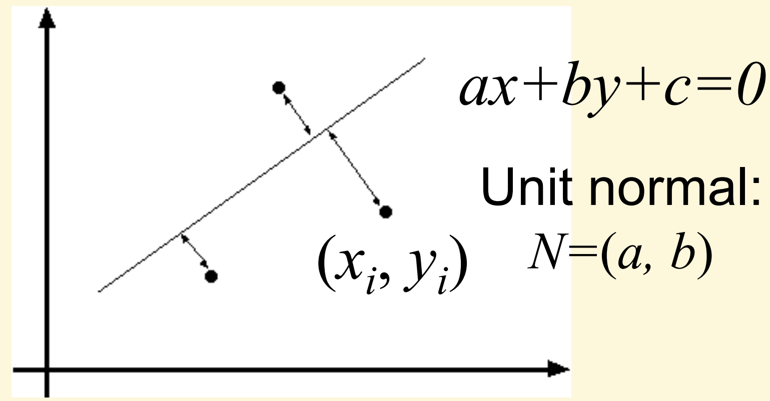
If (a^2+b^2=1) then

Distance between point (x_i, y_i) is

$$|ax_i + by_i + c|$$

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

$$\boxed{E = \sum_{i=1}^{n} (a x_i + b y_i + c)^2}$$

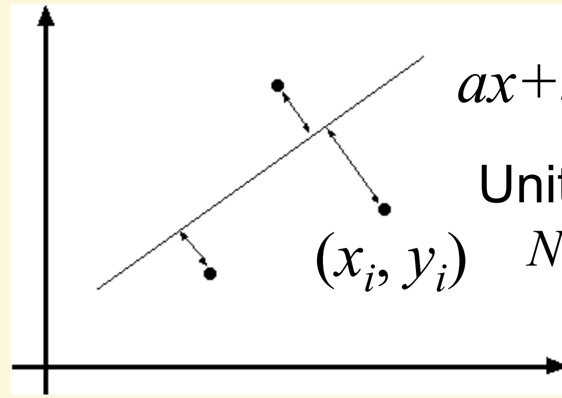


Slide modified from S. Lazebnik

Total least squares

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

$$\boxed{E = \sum_{i=1}^{n} (a x_i + b y_i + c)^2}$$



$ax+by+c=0$

Unit normal: $N=(a, b)$

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

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

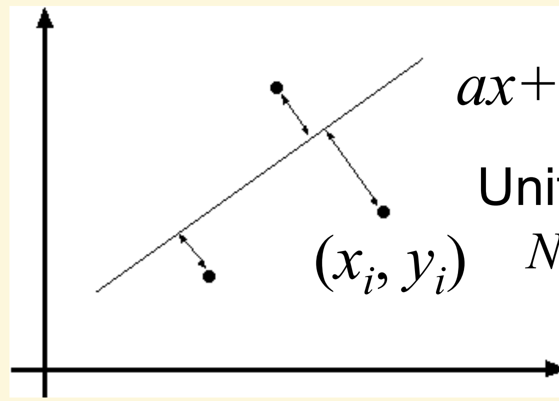
$$\text{minimize}\, \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p} \quad \text{s.t. } \mathbf{p}^T \mathbf{p} = 1 \quad \Rightarrow \quad \text{minimize} \frac{\mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}}{\mathbf{p}^T \mathbf{p}}$$

Solution is eigenvector corresponding to smallest eigenvalue of A^TA

Fitting techniques: Total least squares

- Distance between point (x_i, y_i) and line $ax+by=d$ $(a^2+b^2=1)$: $|ax_i + by_i - d|$
- Find (a, b, d) to minimize the sum of squared perpendicular distances

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

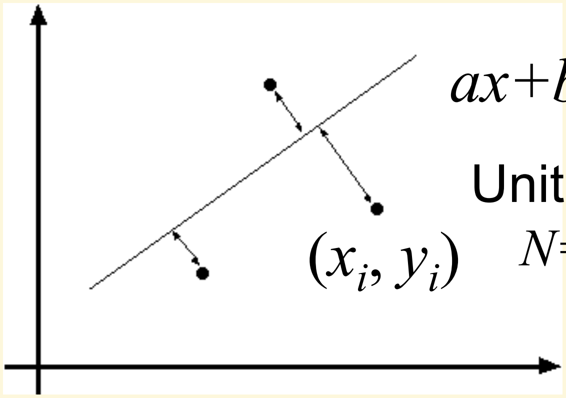


$ax+by=d$

Unit normal:
$N=(a, b)$

Total least squares

- Distance between point (x_i, y_i) and line $ax+by=d$ ($a^2+b^2=1$): $|ax_i + by_i - d|$
- Find (a, b, d) to minimize the sum of squared perpendicular distances



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

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

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

$$\frac{dE}{dN} = 2(U^T U)N = 0$$

Solution to $(U^T U)N = 0$, subject to $\|N\|^2 = 1$: eigenvector of $U^T U$ associated with the smallest eigenvalue (least squares solution to *homogeneous linear system* $UN = 0$)

Slide: S. Lazebnik

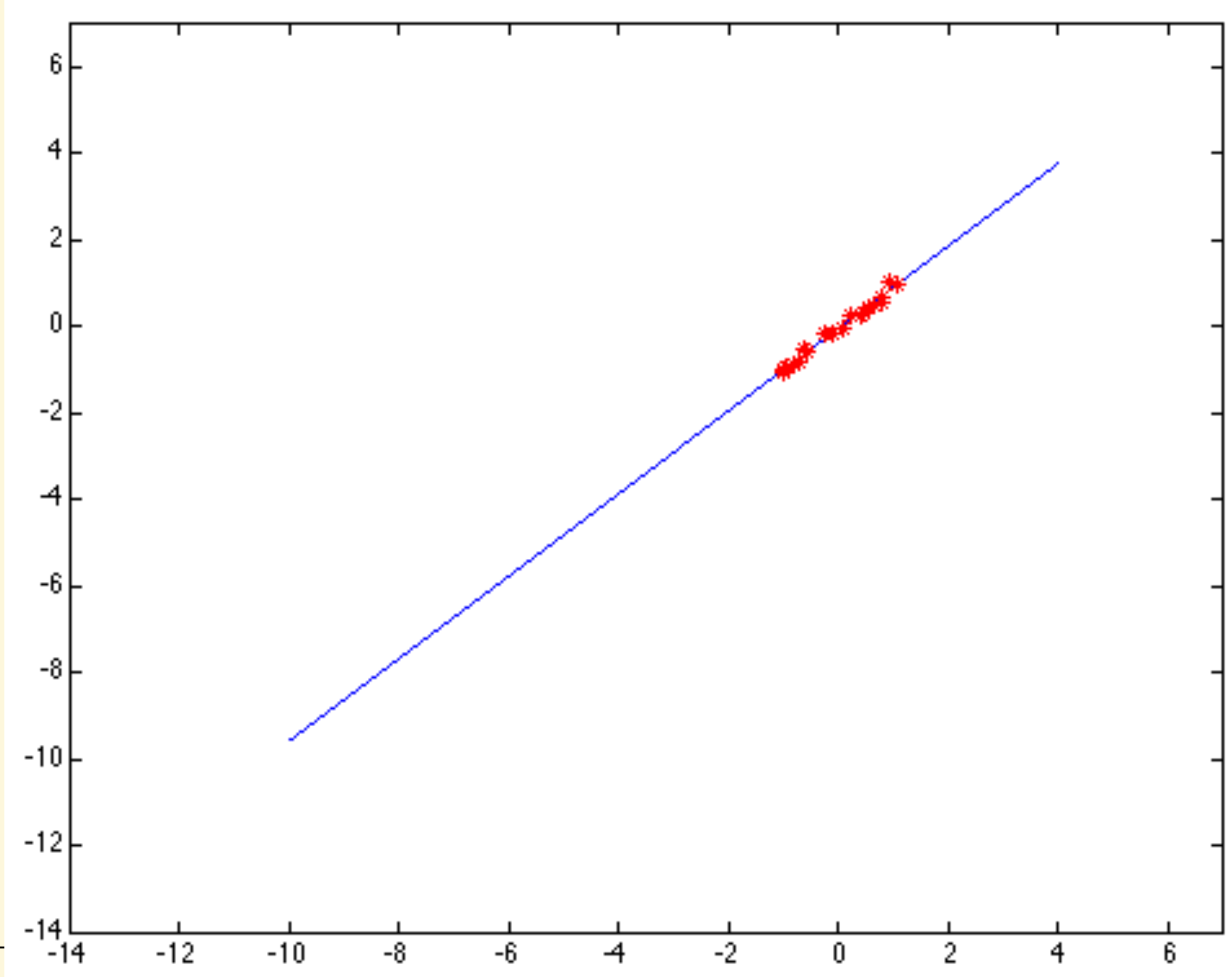
Recap: Two Common Optimization Problems

Problem statement	Solution
minimize $\lVert \mathbf{Ax} - \mathbf{b} \rVert^2$	$\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$
least squares solution to $\mathbf{Ax} = \mathbf{b}$	$\mathbf{x} = \mathbf{A} \backslash \mathbf{b}$ (matlab)

Problem statement	**Solution**
minimize $\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}$ s.t. $\mathbf{x}^T\mathbf{x} = 1$	
minimize $\dfrac{\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}}{\mathbf{x}^T\mathbf{x}}$	$[\mathbf{v}, \lambda] = \mathrm{eig}(\mathbf{A}^T\mathbf{A})$ $\lambda_1 < \lambda_{2..n} : \mathbf{x} = \mathbf{v}_1$
non - trivial lsq solution to $\mathbf{Mx} = 0$	

Least squares: Robustness to noise

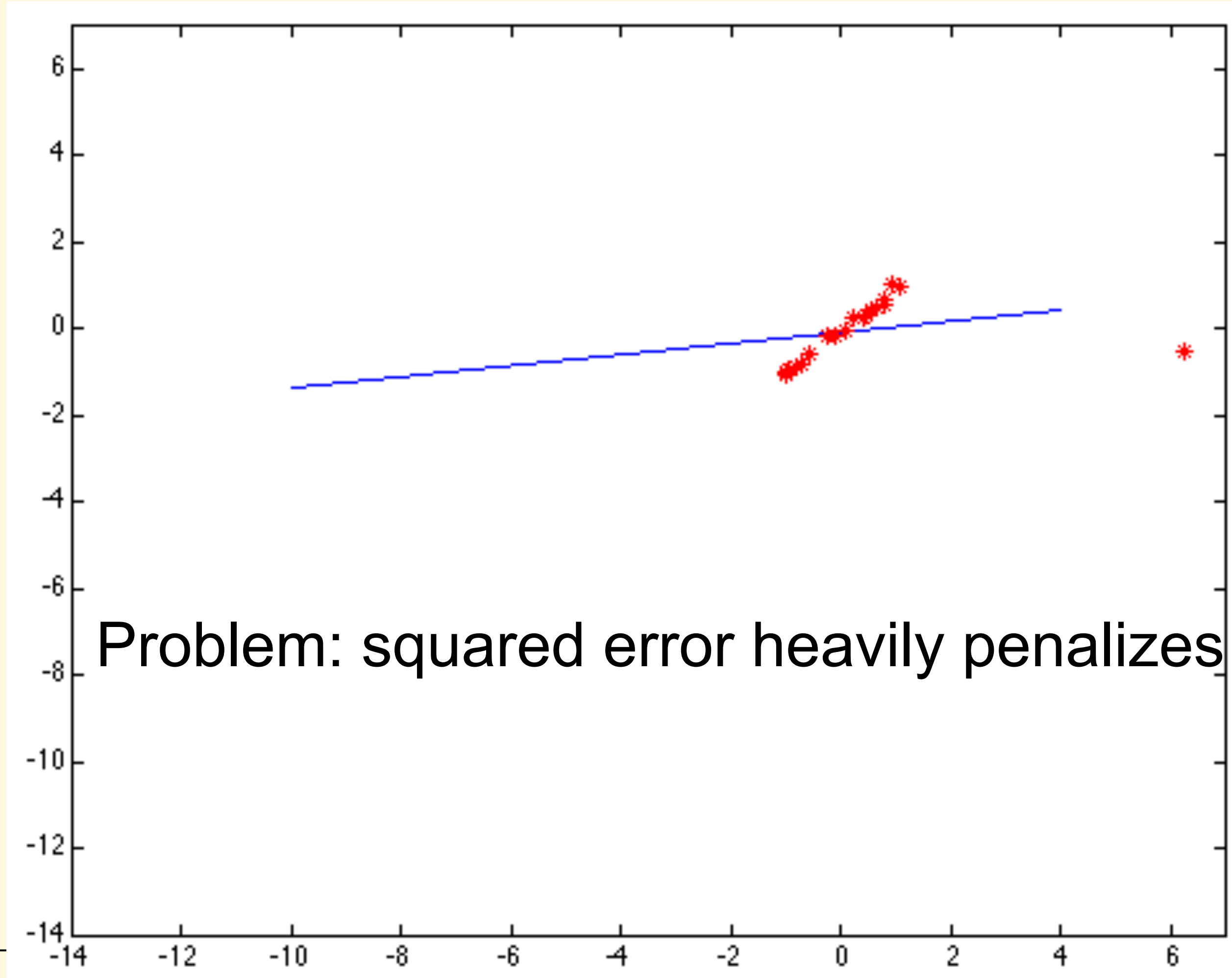
- Least squares fit to the red points:



Slides from Svetlana Lazebnik

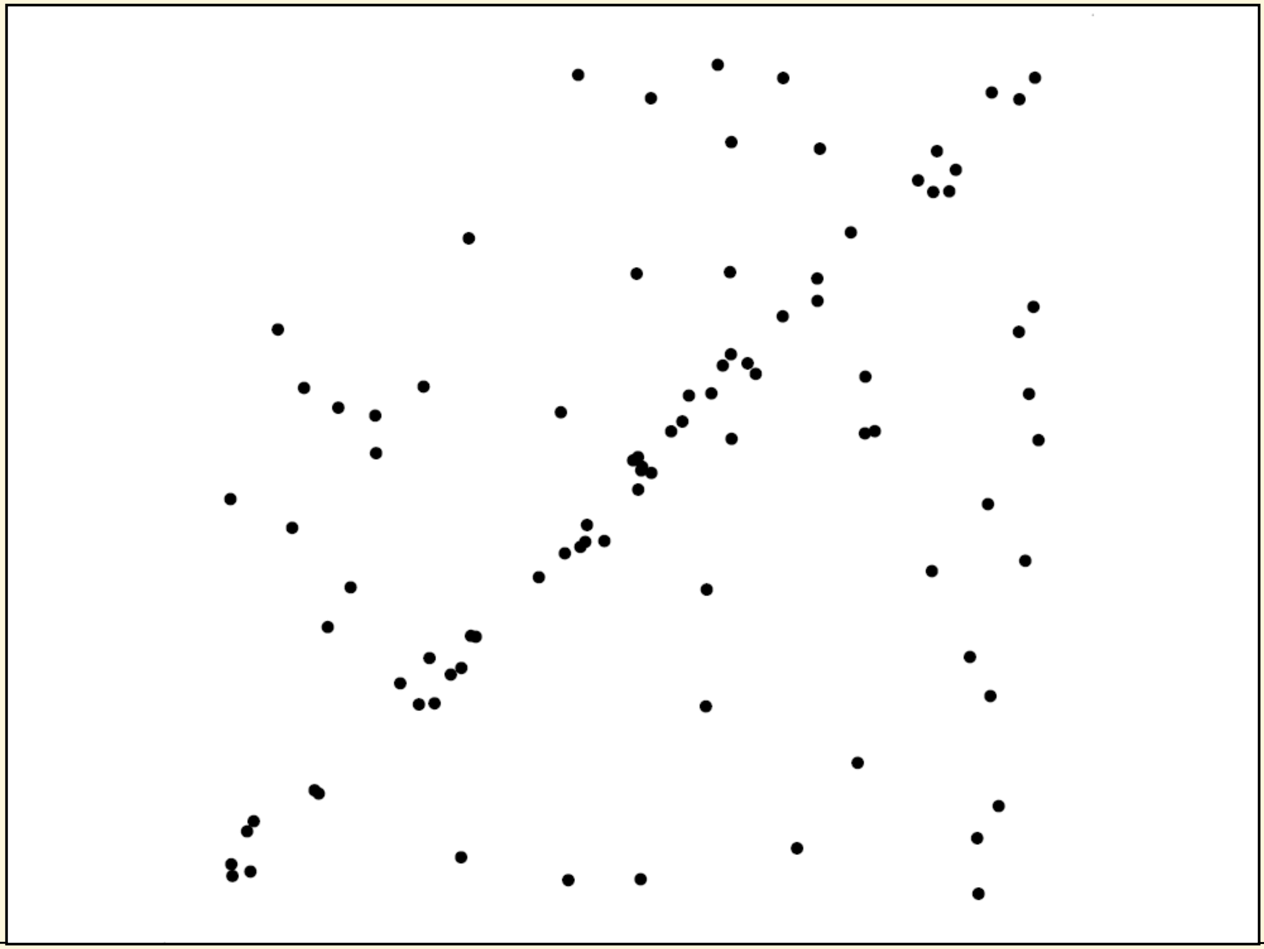
Least squares: Robustness to noise

- Least squares fit with an outlier:

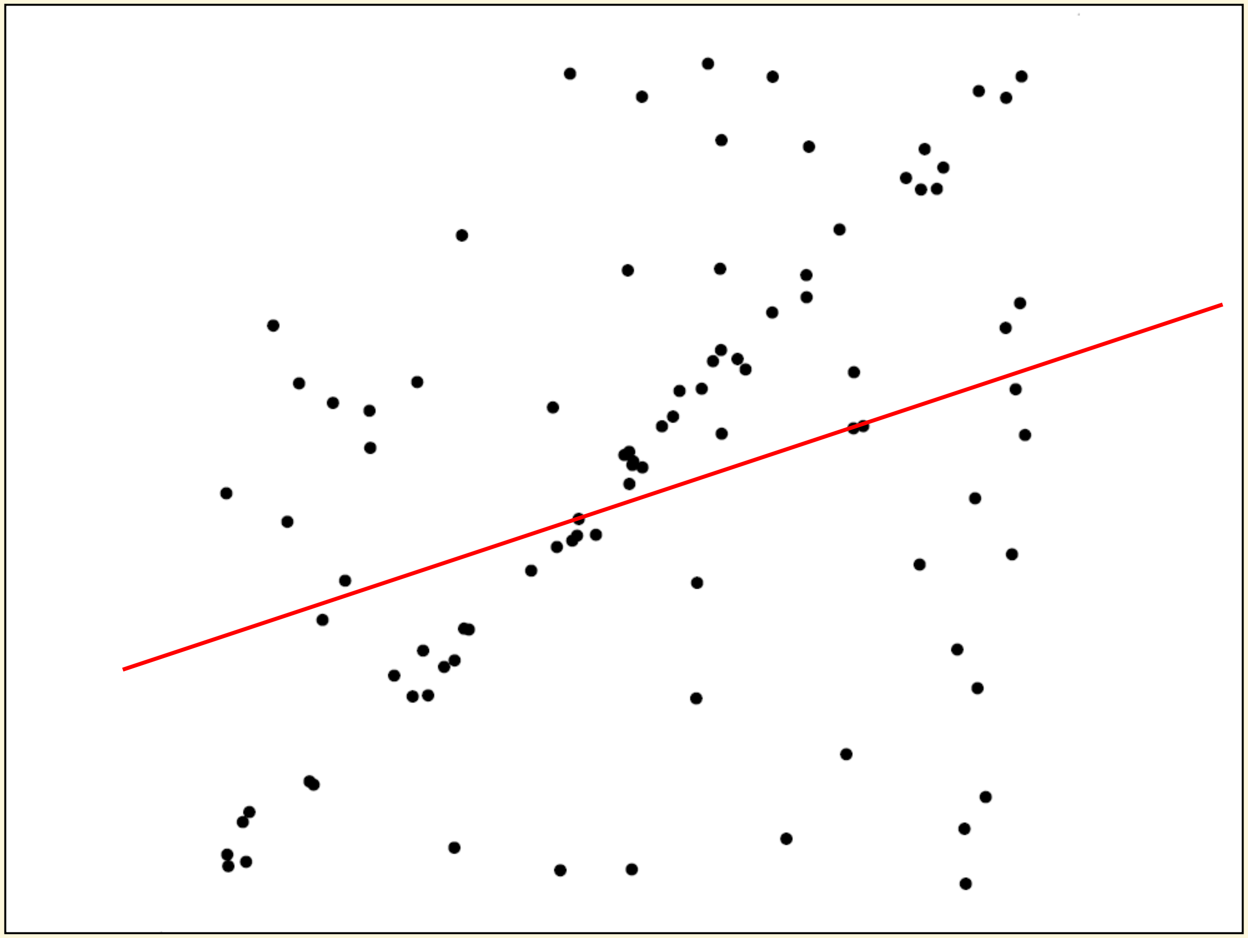


Problem: squared error heavily penalizes outliers

Line fitting problem



Line fitting – the least-square approach



Search / Least squares conclusions

Good

- Clearly specified objective
- Optimization is easy (for least squares)

Bad

- Not appropriate for non-convex objectives
  - May get stuck in local minima
- Sensitive to outliers
  - Bad matches, extra points
- Doesn't allow you to get multiple good fits
  - Detecting multiple objects, lines, etc.

Slide from Derek Hoiem

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

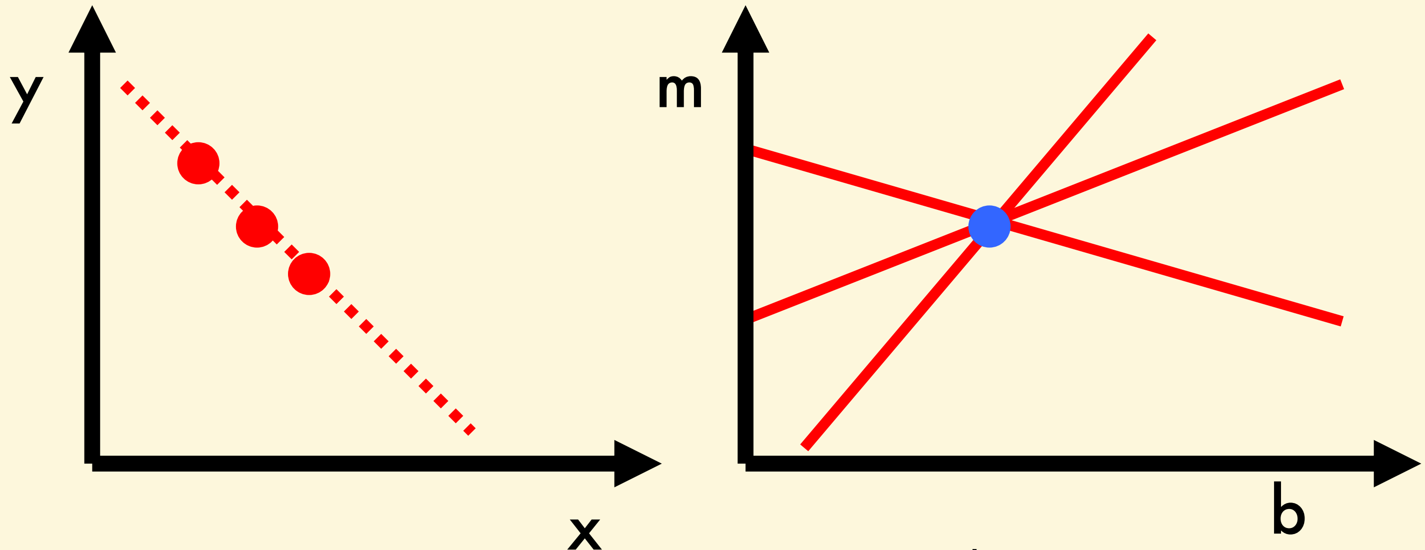
4. Possibly refine parameters using inliers

Hough transform

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

$y = m\,x + b$

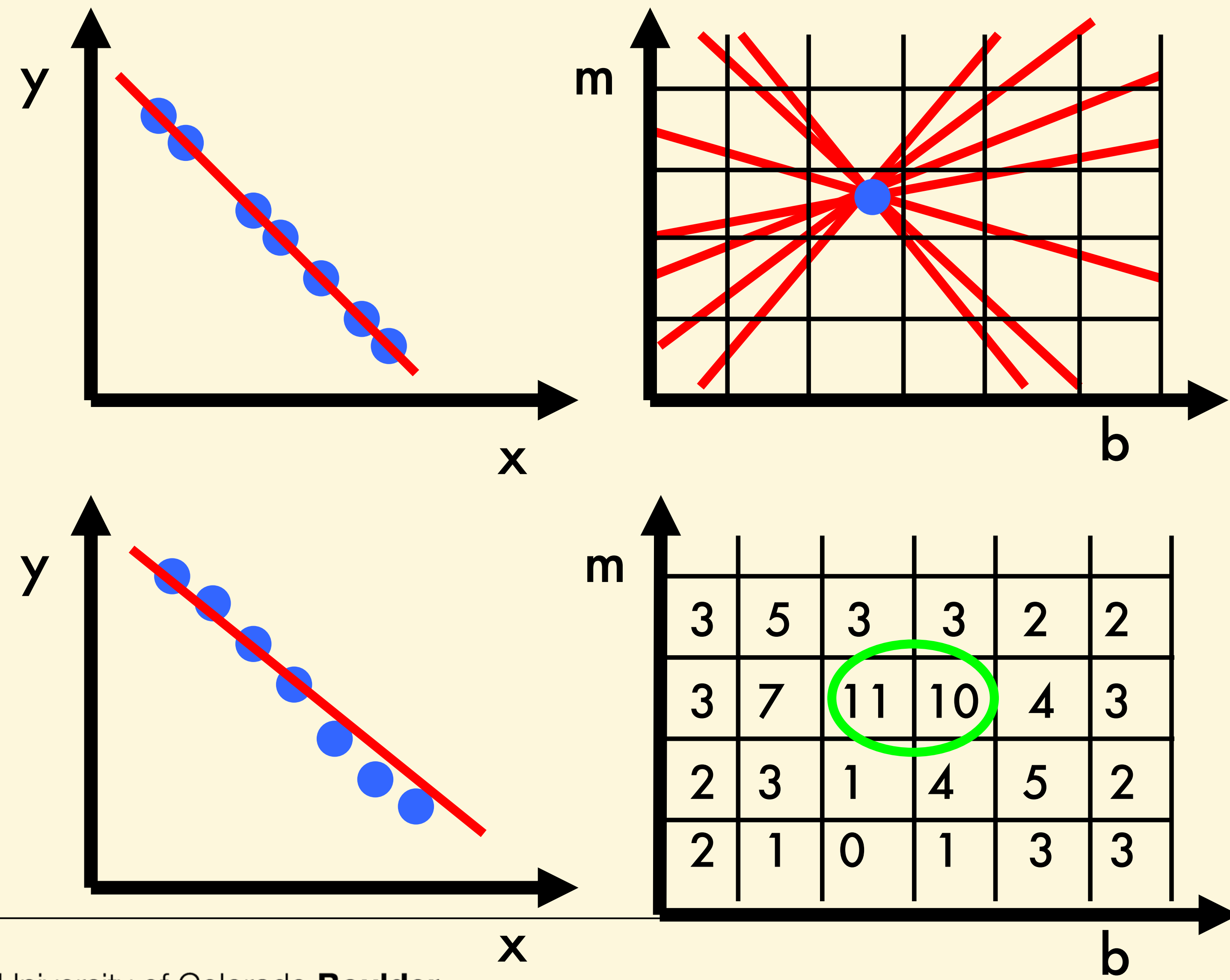


Hough space

University of Colorado **Boulder**

Slide from S. Savarese

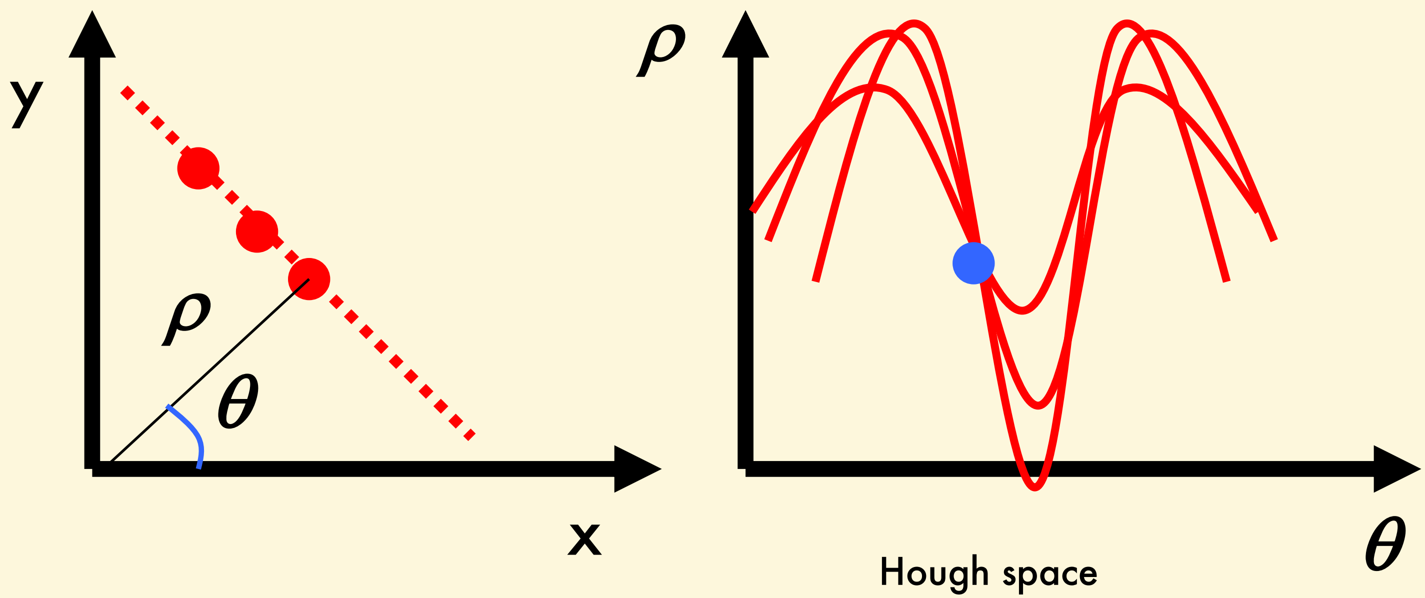
Hough transform



Hough transform

Issue : parameter space [m,b] is unbounded...

Use a polar representation for the parameter space



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

Hough transform conclusions

Good

- Robust to outliers: each point votes separately
- Fairly efficient (often faster than trying all sets of parameters)
- Provides multiple good fits

Bad

- Some sensitivity to noise
- Bin size trades off between noise tolerance, precision, and speed/memory
  - Can be hard to find sweet spot
- Not suitable for more than a few parameters
  - grid size grows exponentially

Common applications

- Line fitting (also circles, ellipses, etc.)
- Object instance recognition
- Object category recognition

RANSAC

Robust fitting can deal with a few outliers – what if we have very many?

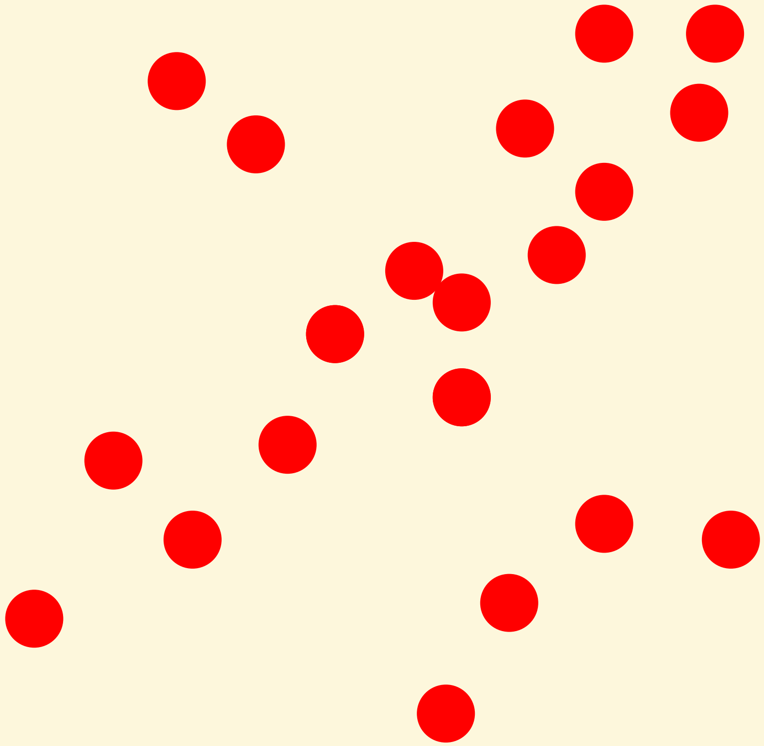
- Random sample consensus (RANSAC):
  Very general framework for model fitting in the presence of outliers
- Outline
  - Choose a small subset of points uniformly at random
  - Fit a model to that subset
  - Find all remaining points that are "close" to the model and reject the rest as outliers
  - Do 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

(RANdom SAmple Consensus) :

Fischler & Bolles in '81.



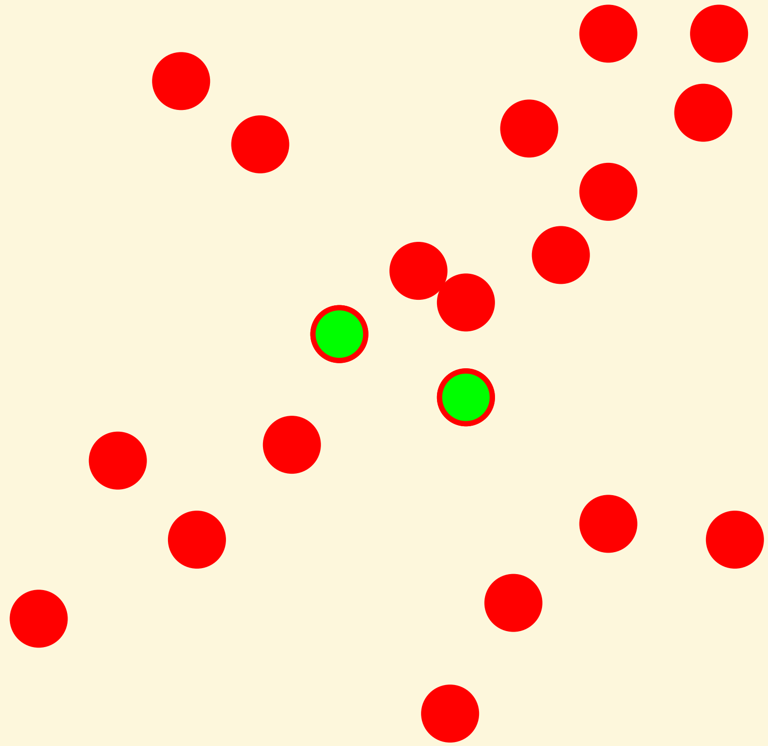
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



Algorithm:

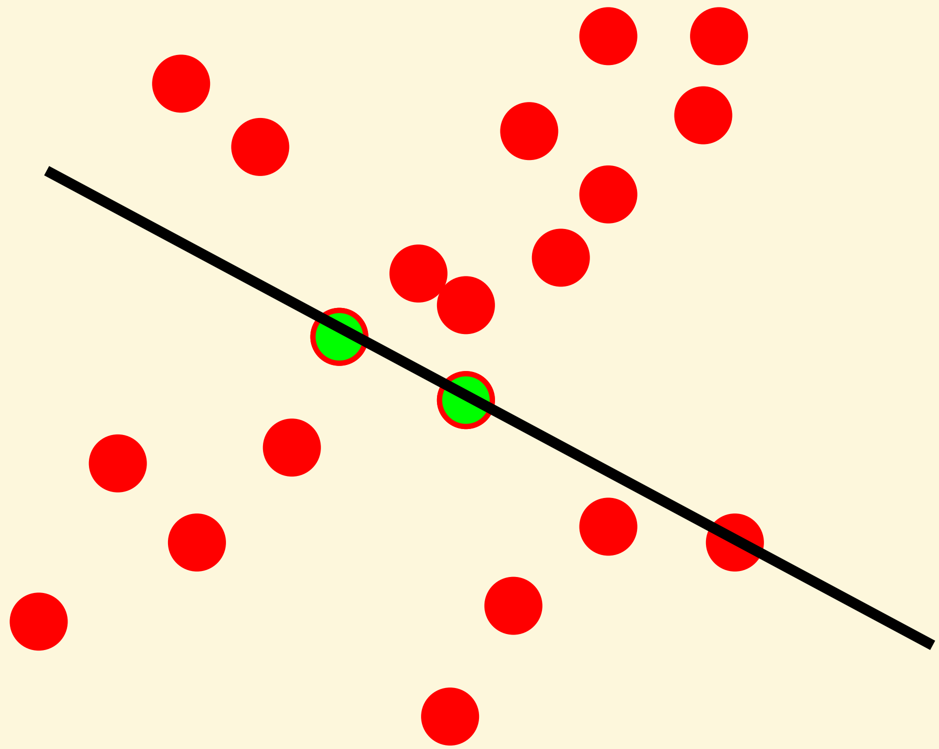
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Illustration by Savarese

RANSAC

Line fitting example



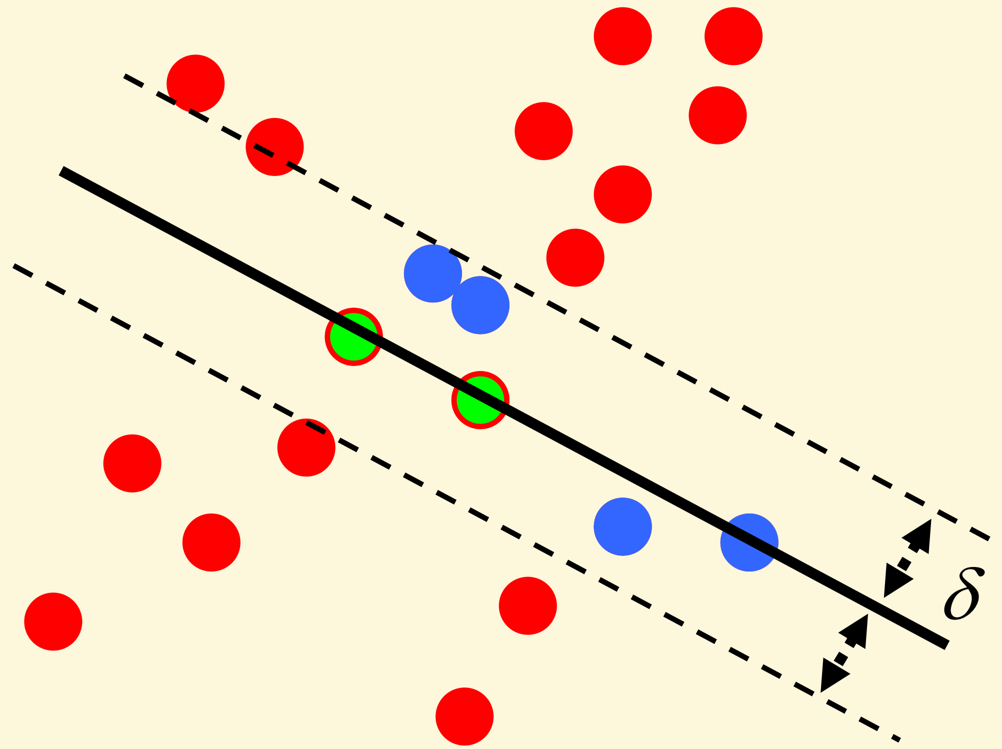
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

Line fitting example



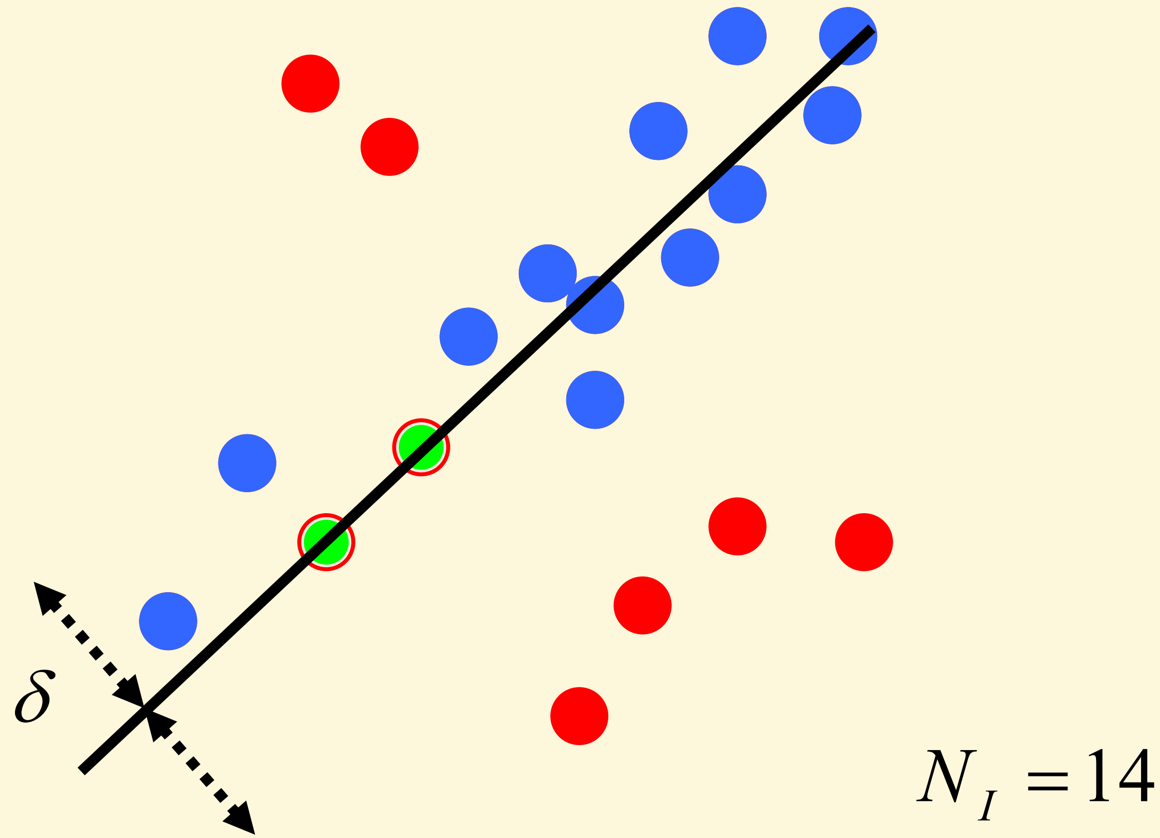
$$N_I = 6$$

Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

RANSAC

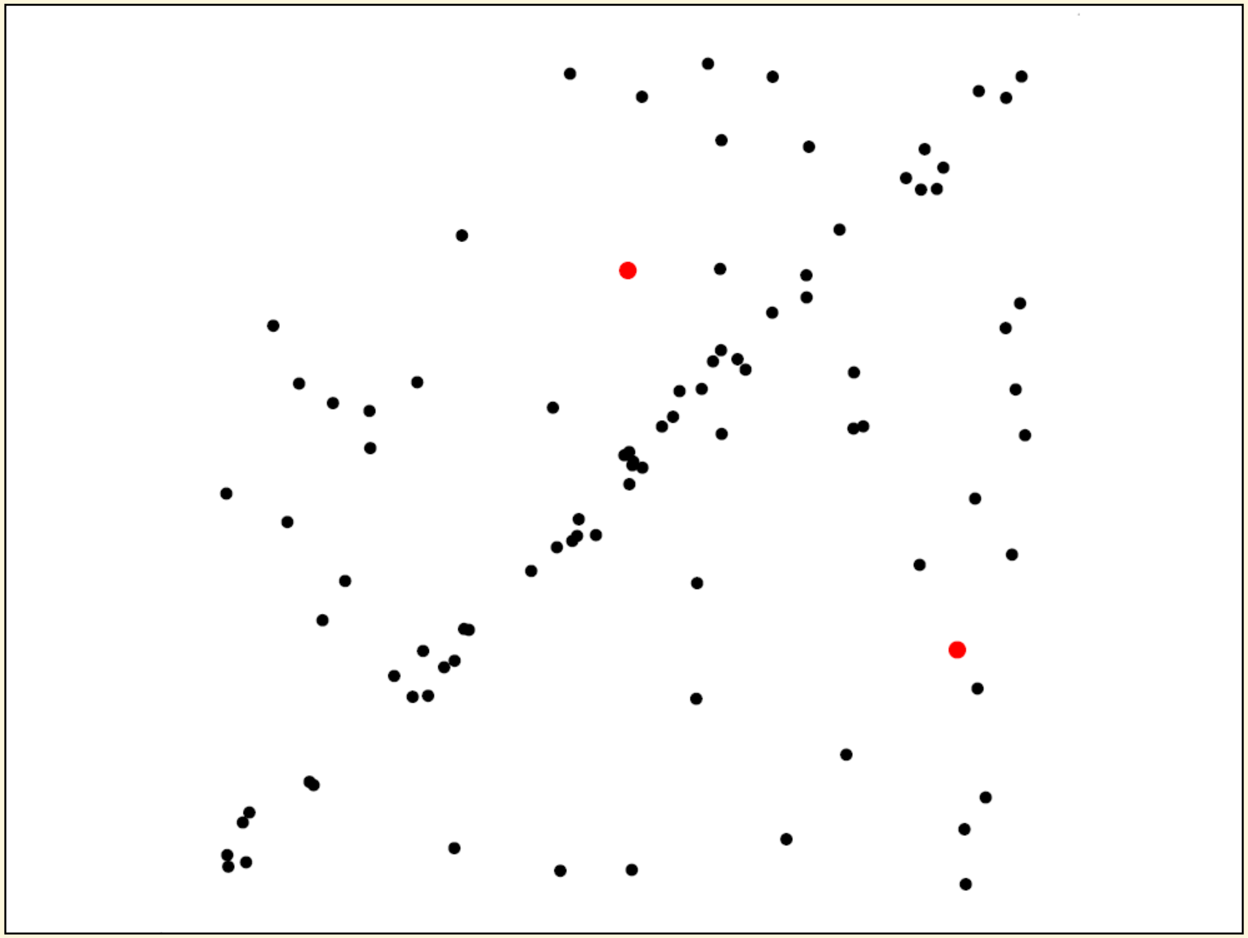


Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

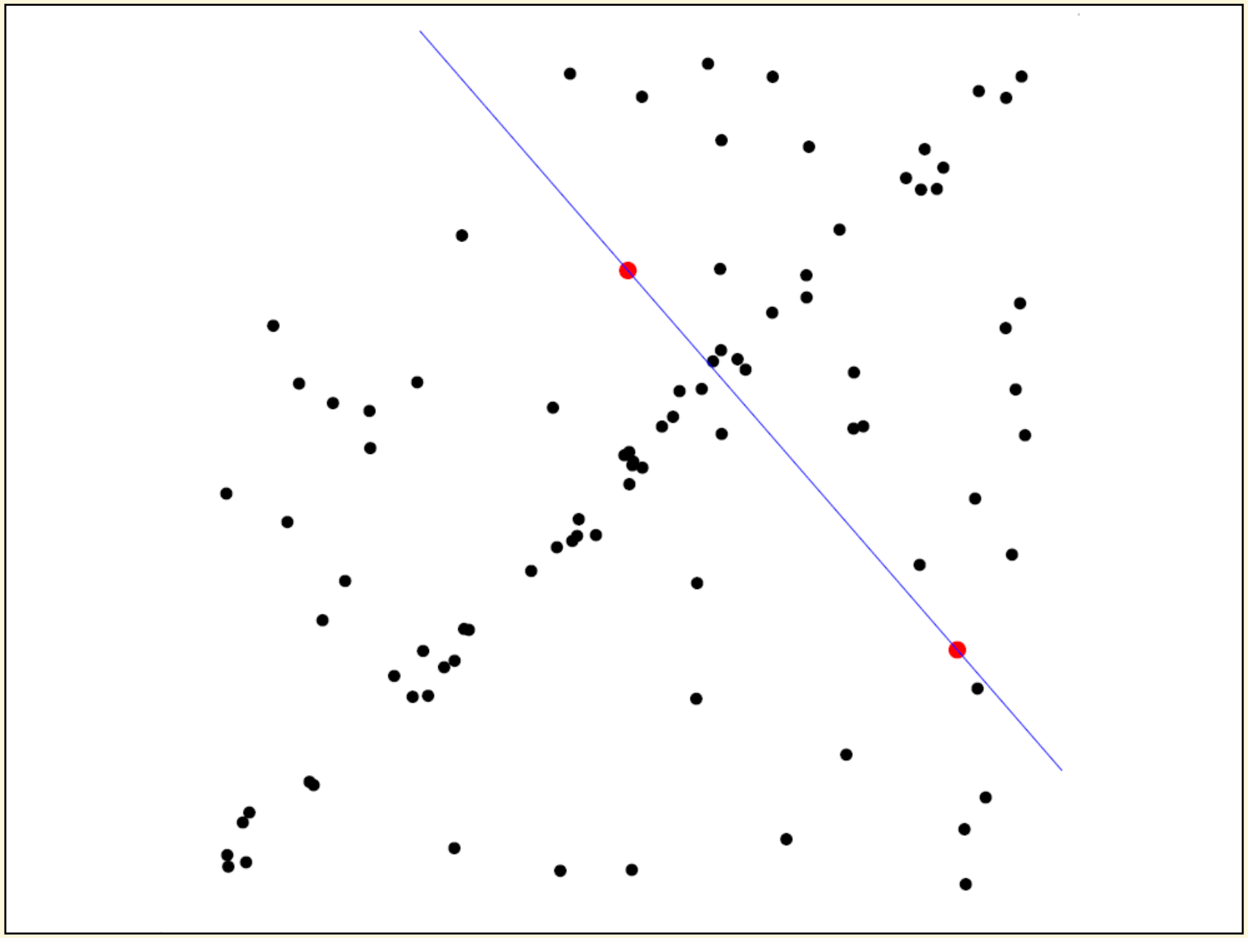
Repeat 1-3 until the best model is found with high confidence

RANSAC for line fitting example



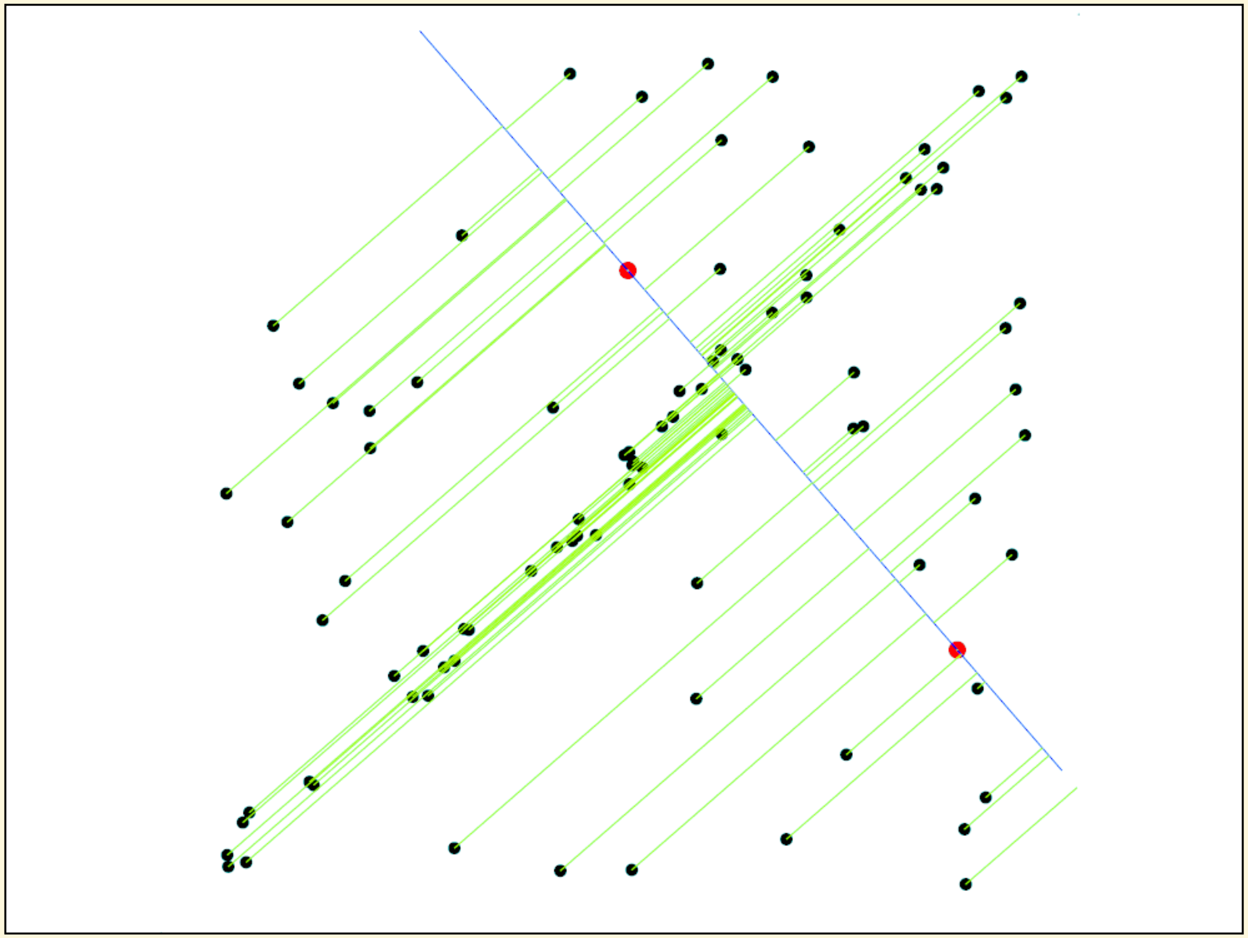
1. Randomly select minimal subset of points

RANSAC for line fitting example



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

RANSAC for line fitting example

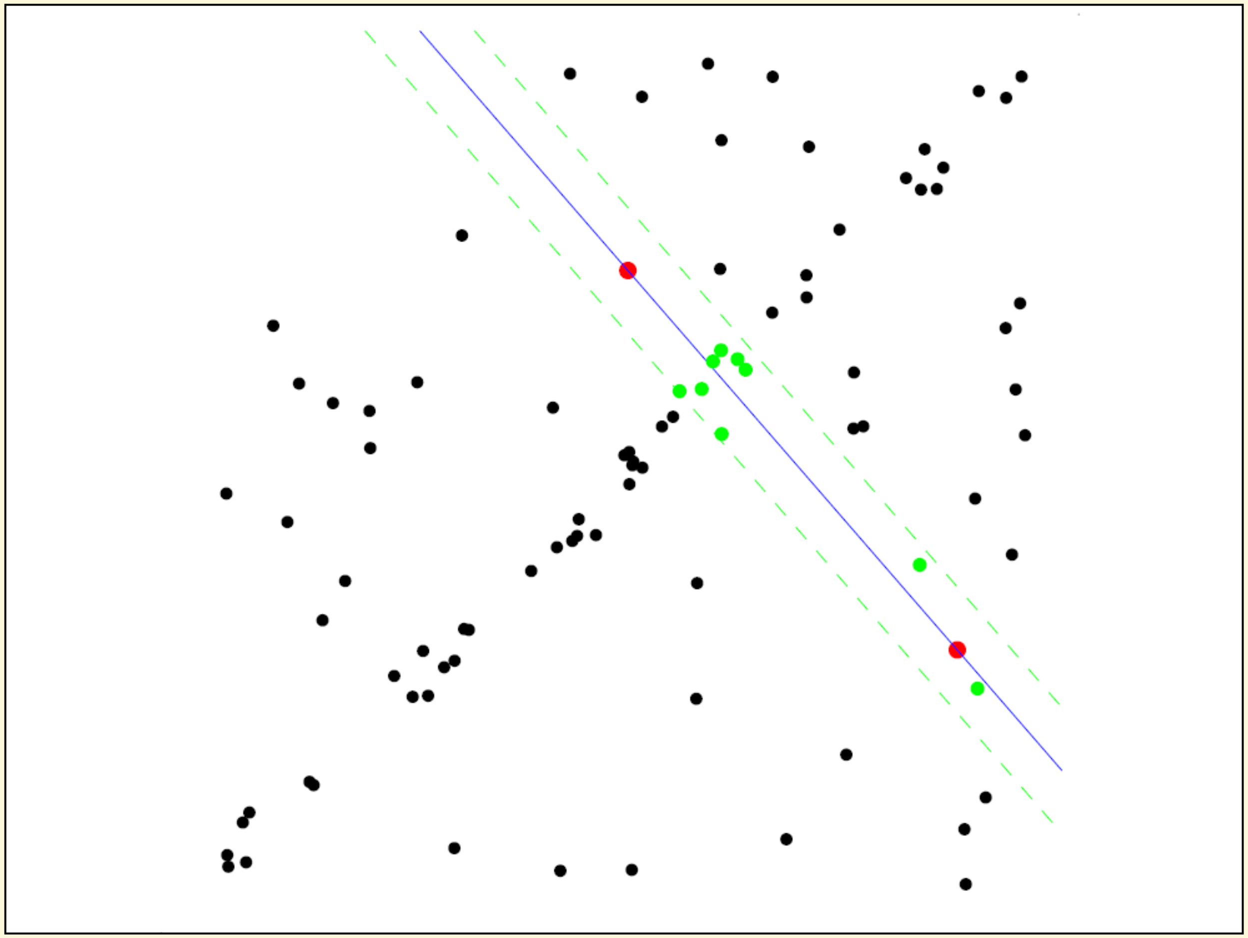


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

Source: R. Raguram

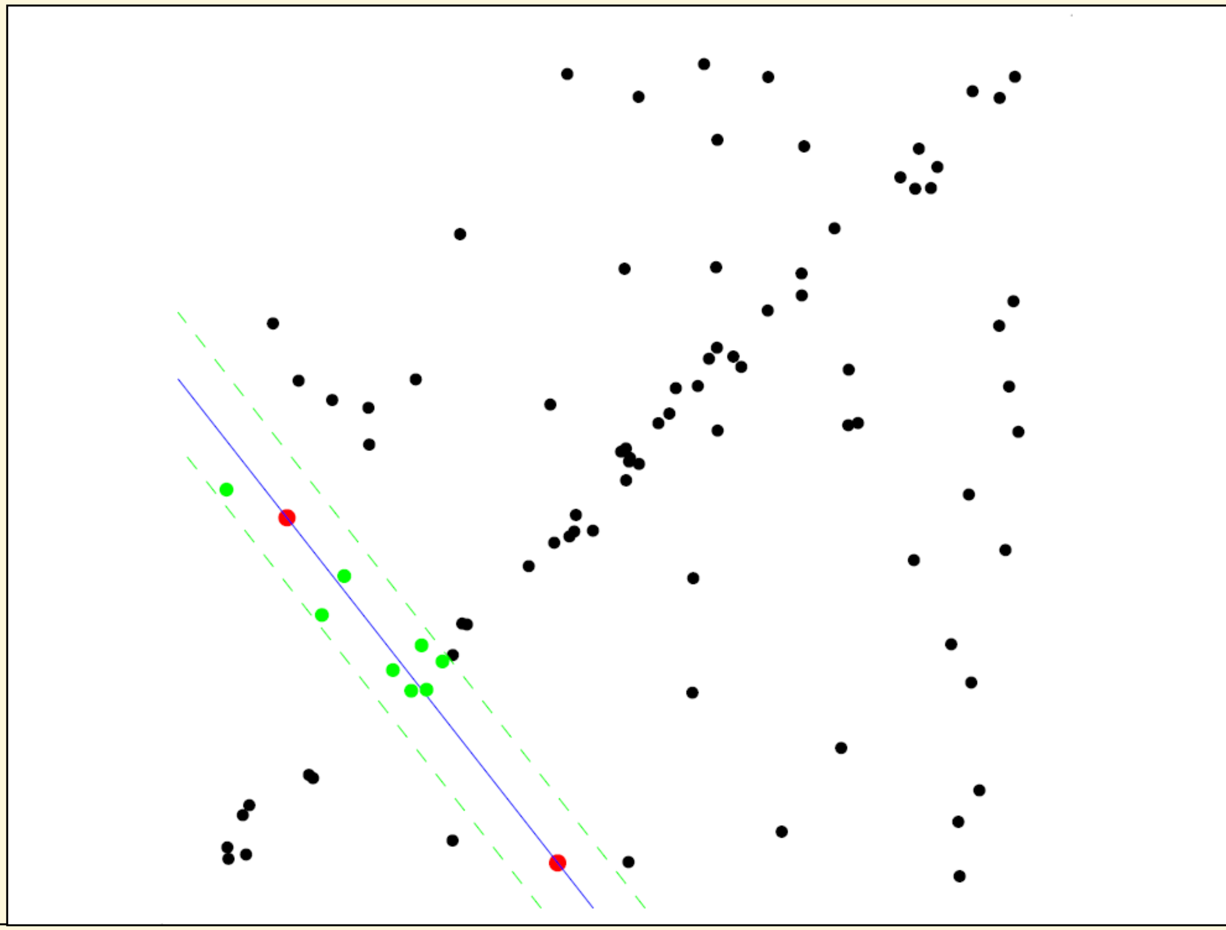
University of Colorado **Boulder**

RANSAC for line fitting example



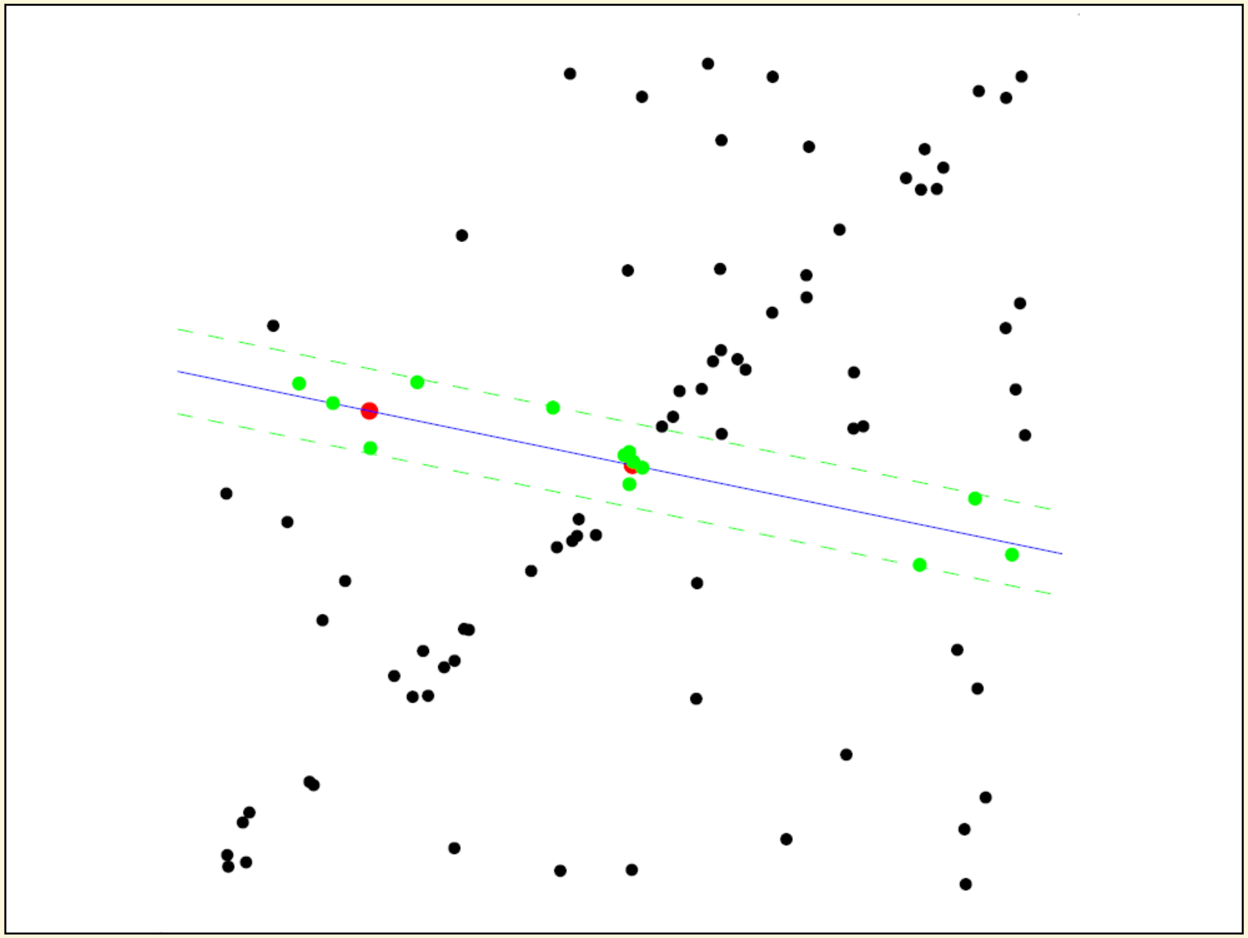
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

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

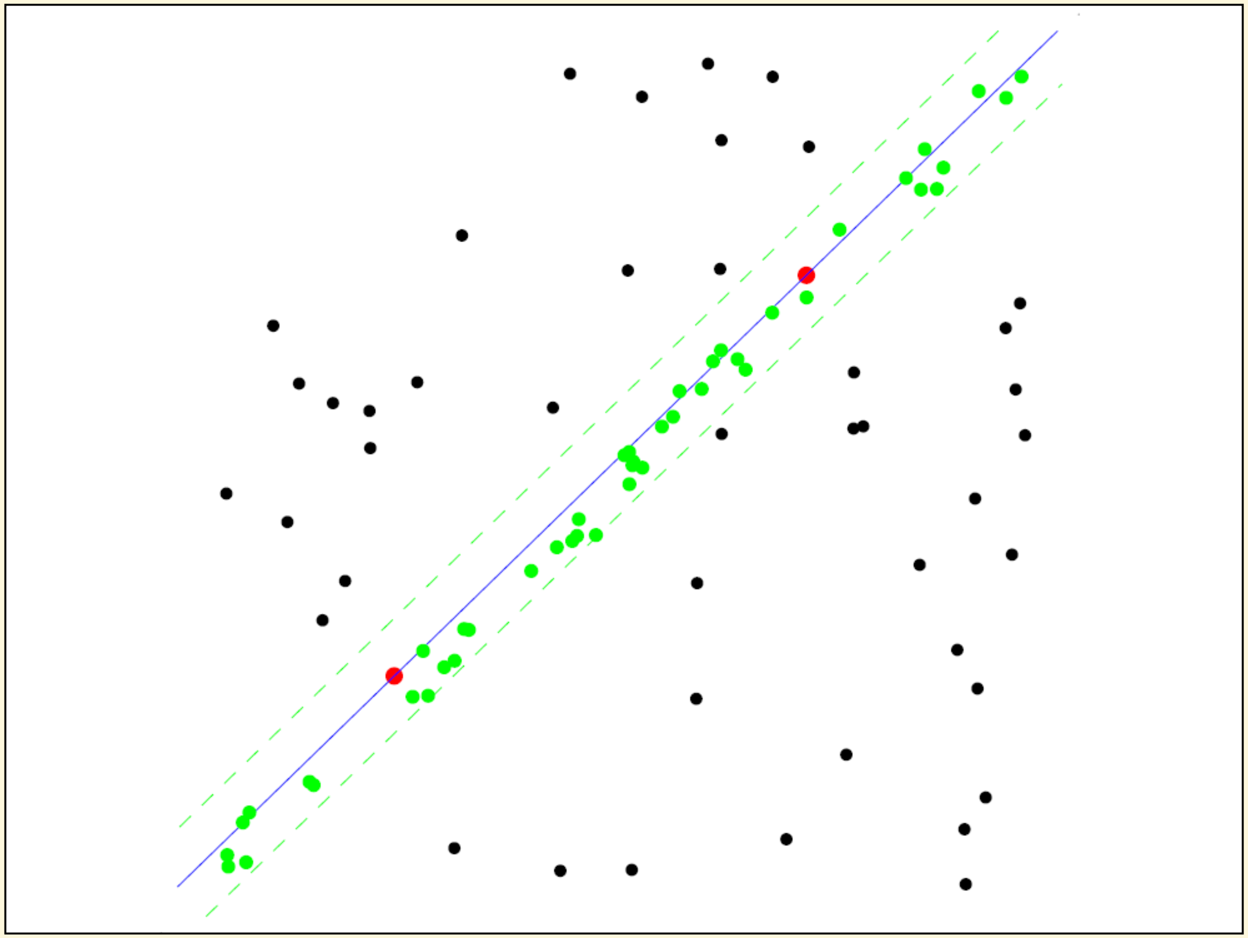
RANSAC for line fitting example



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

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

Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:
- n — the smallest number of points required
- k — the number of iterations required
- t — the threshold used to identify a point that fits well
- d — the number of nearby points required to assert a model fits well

Until k iterations have occurred
    Draw a sample of n points from the data uniformly and at random
    Fit to that set of n points
    For each data point outside the sample
        Test the distance from the point to the line against t; if the distance from the point to the line is less than t, the point is close
    end
    If there are d or more points close to the line then there is a good fit. Refit the line using all these points.
end
Use the best fit from this collection, using the fitting error as a criterion

Testing goodness

- Algorithm is called RANSAC (RANdom SAmple Consensus) – example of a randomized algorithm
- Used in an amazing number of computer vision algorithms
- Requires two parameters:
  - The agreement threshold (how close does an inlier have to be?)
  - The number of rounds (how many do we need?)

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*)

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

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

	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

RANSAC conclusions

Good

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

Bad

- Computational time grows quickly with fraction of outliers and number of parameters
- Not good for getting multiple fits

Common applications

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

Randomized algorithms

- Very common in computer science
  - In this case, we avoid testing an infinite set of possible lines, or all $O(n^2)$ lines generated by pairs of points

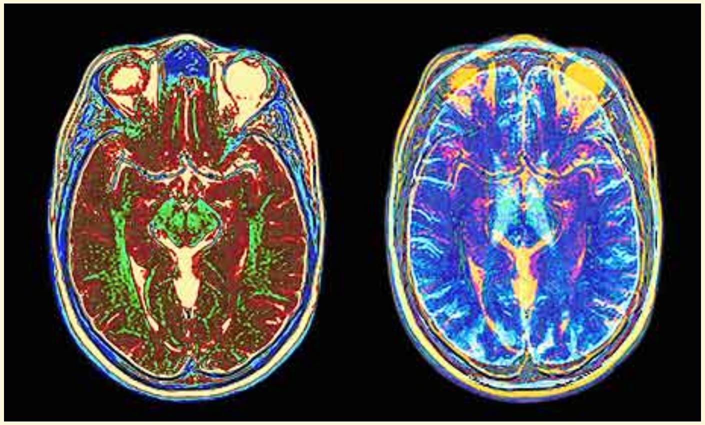
- These algorithms find the right answer with some probability

- Often work very well in practice

What if you want to align but have no prior matched pairs?

- Hough transform and RANSAC not applicable

- Important applications



Medical imaging: match brain scans or contours

Robotics: match point clouds

Slide from Derek Hoiem

Iterative Closest Points (ICP) Algorithm

Goal: estimate transform between two dense sets of points

1. **Assign** each point in {Set 1} to its nearest neighbor in {Set 2}
2. **Estimate** transformation parameters
   – e.g., least squares or robust least squares
3. **Transform** the points in {Set 1} using estimated parameters
4. **Repeat** steps 1-3 until change is very small

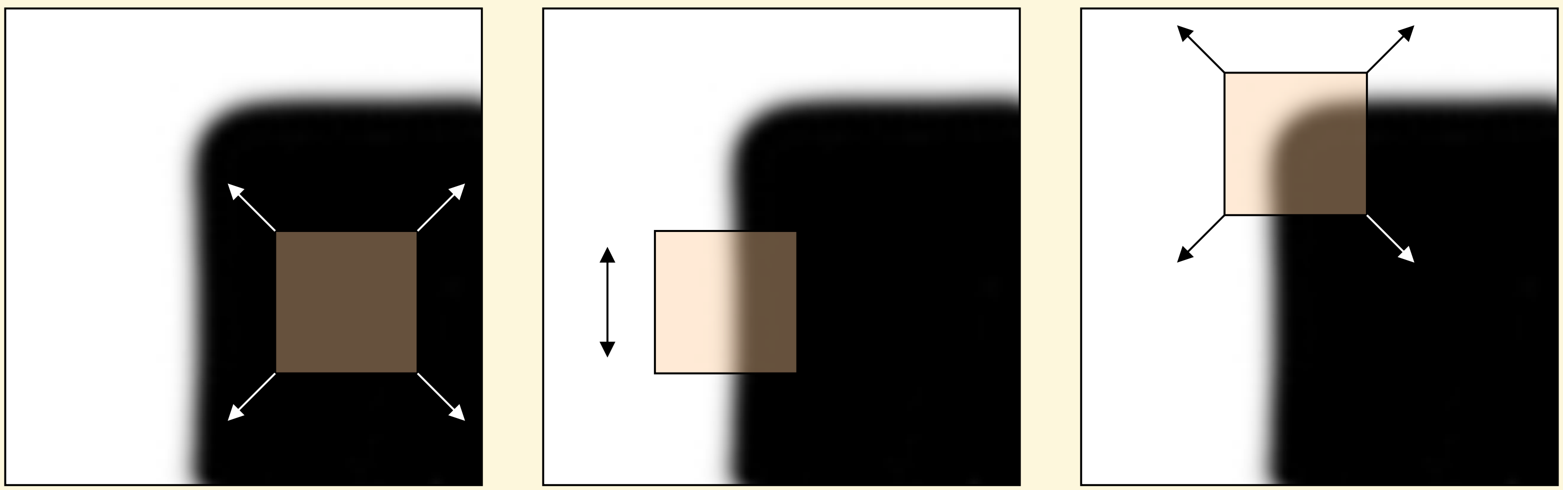
Slide from Derek Hoiem

Fitting curves other than lines?

- **In principle, an easy generalisation**
  - The probability of obtaining a point, given a curve, is given by a negative exponential of distance squared
- **In practice, rather hard**
  - It is generally difficult to compute the distance between a point and a curve

Corners as distinctive interest points

- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region: no change in all directions

"edge": no change along the edge direction

"corner": significant change in all directions

Slide credit: Alyosha Efros, Darya Frolova, Denis Simakov

Harris Detector: Mathematics

Window-averaged change of intensity induced by shifting the image data by [*u,v*]:

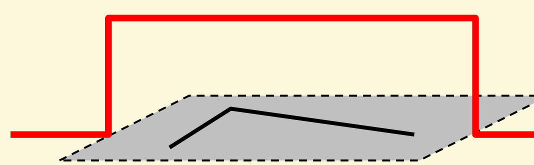
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

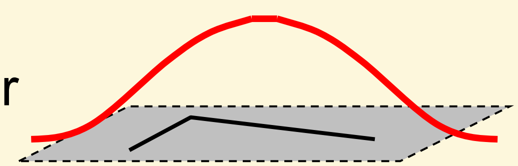
Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

Corner Detection: Mathematics

Change in appearance of window $w(x,y)$
for the shift $[u,v]$:

$$E(u,v)=\sum_{x,y} w(x,y)\left[I(x+u,y+v)-I(x,y)\right]^2$$

We want to find out how this function behaves for small shifts

Local quadratic approximation of $E(u,v)$ in the neighborhood of (0,0) is given by the *second-order Taylor expansion*:

$$E(u,v)\approx E(0,0)+[u\ \ v]\begin{bmatrix}E_u(0,0)\\E_v(0,0)\end{bmatrix}+\frac{1}{2}[u\ \ v]\begin{bmatrix}E_{uu}(0,0) & E_{uv}(0,0)\\E_{uv}(0,0) & E_{vv}(0,0)\end{bmatrix}\begin{bmatrix}u\\v\end{bmatrix}$$

Taylor series approx to shifted image

$$E(u,v) \approx \sum_{x,y} w(x,y)[I(x,y) + uI_x + vI_y - I(x,y)]^2$$

$$= \sum_{x,y} w(x,y)[uI_x + vI_y]^2$$

$$= \sum_{x,y} w(x,y)\begin{pmatrix} u & v \end{pmatrix} \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

Harris Detector: Mathematics

Expanding I(x,y) in a Taylor series expansion, we have, for small shifts [*u,v*], a *bilinear* approximation:

$$E(u,v) \cong [u,v] \; M \; \begin{bmatrix} u \\ v \end{bmatrix}$$

where *M* is a 2×2 matrix computed from image derivatives:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

M is also called "structure tensor"

Harris Detector: Mathematics

Intensity change in shifting window: eigenvalue analysis

$$E(u,v) \cong [u,v] \; M \begin{bmatrix} u \\ v \end{bmatrix}$$

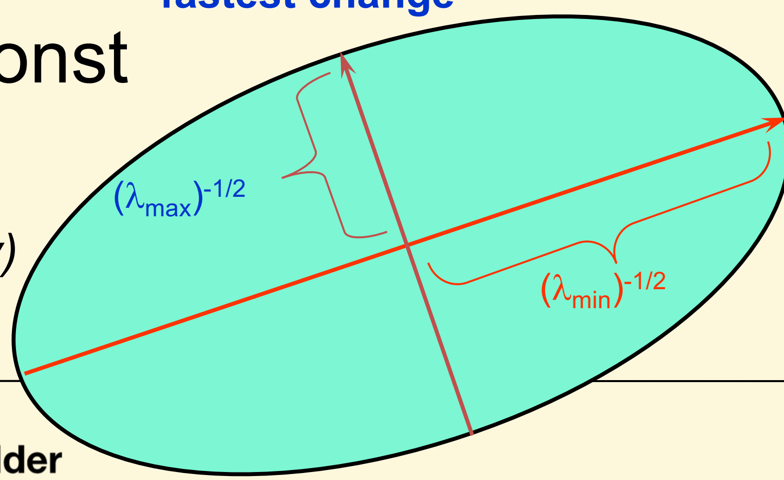
λ_1, λ_2 – eigenvalues of M

Ellipse $E(u,v)$ = const

Iso-intensity contour of $E(u,v)$

direction of the fastest change

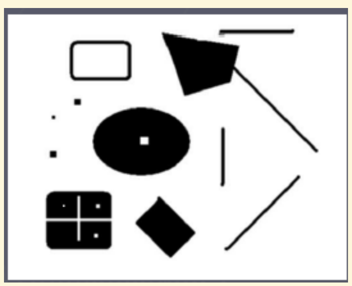
direction of the slowest change

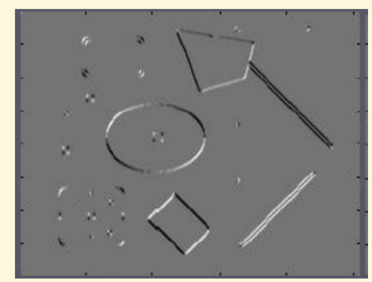


Corners as distinctive interest points

$$M = \sum w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

2 x 2 matrix of image derivatives (averaged in neighborhood of a point).



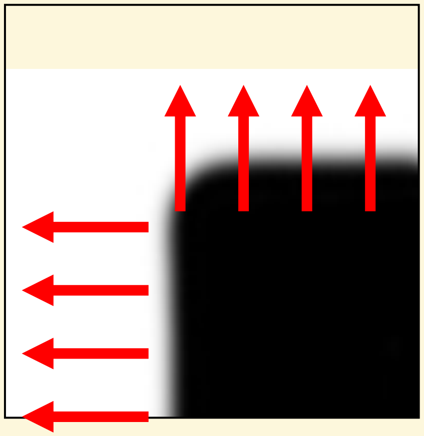



Notation:

$$I_x \Leftrightarrow \frac{\partial I}{\partial x} \qquad I_y \Leftrightarrow \frac{\partial I}{\partial y} \qquad I_x I_y \Leftrightarrow \frac{\partial I}{\partial x}\frac{\partial I}{\partial y}$$

What does this matrix reveal?

First, consider an axis-aligned corner:



What does this matrix reveal?

First, consider an axis-aligned corner:

$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \Rightarrow \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means dominant gradient directions align with x or y axis

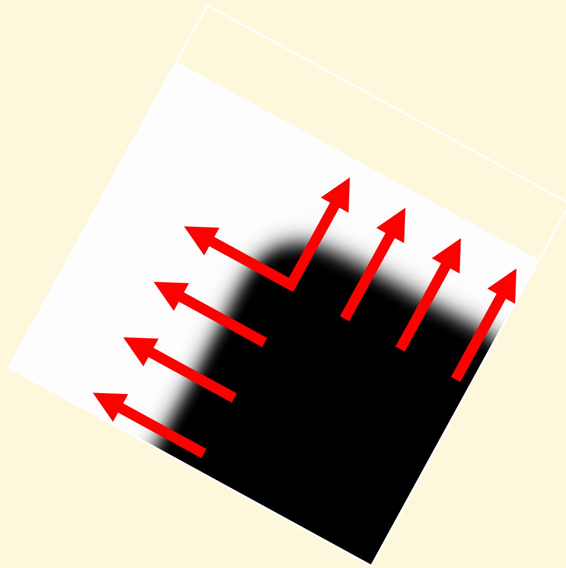
Look for locations where **both** λ's are large.

If either λ is close to 0, then this is **not** corner-like.

What if we have a corner that is not aligned with the image axes?

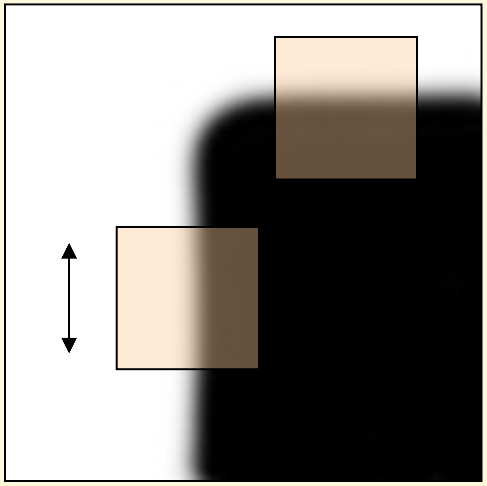
What does this matrix reveal?

Since *M* is symmetric, we have $M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$



The *eigenvalues* of *M* reveal the amount of intensity change in the two principal orthogonal gradient directions in the window.

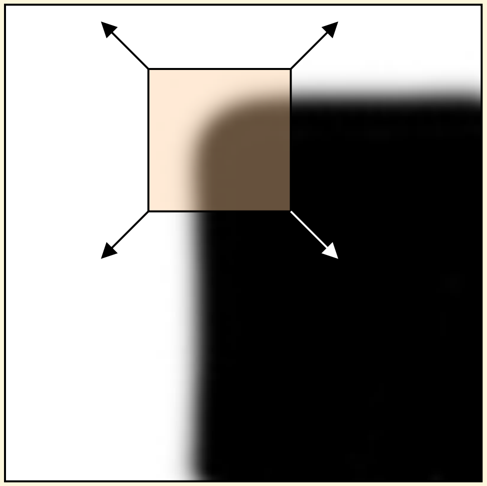
Corner response function



"edge":

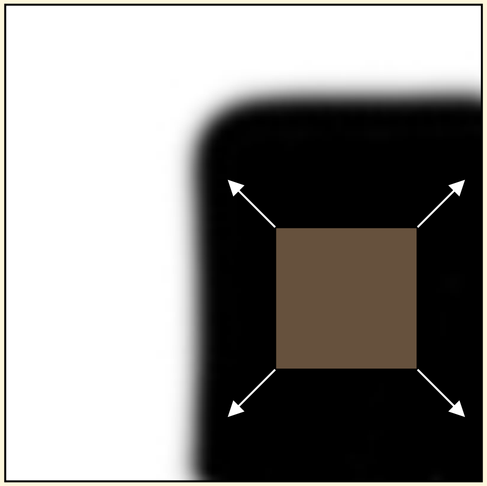
$\lambda_1 >> \lambda_2$

$\lambda_2 >> \lambda_1$



"corner":

λ_1 and λ_2 are large,
$\lambda_1 \sim \lambda_2$;



"flat" region

λ_1 and λ_2 are small;

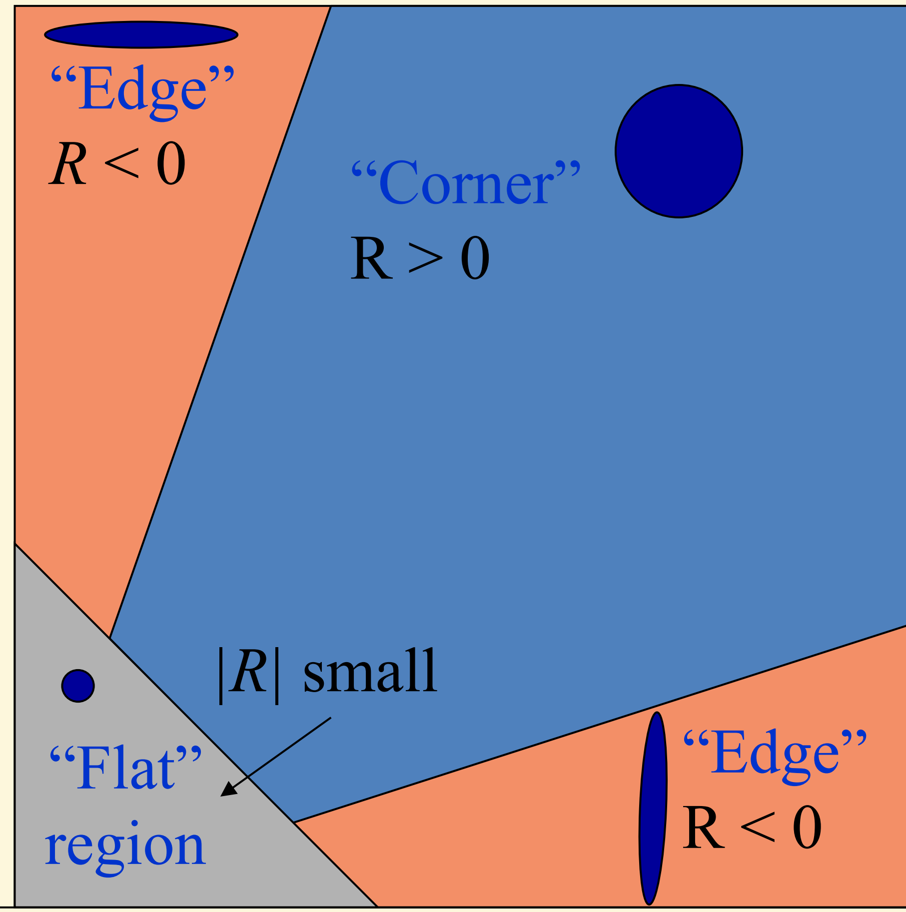
$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

Corner response function

$$R = \det(M) - \alpha\,\mathrm{trace}(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

α: constant (0.04 to 0.06)

R = “cornerness”



Harris corner detector

1) Filter image.
2) Compute magnitude of the gradient everywhere.
3) Compute M matrix for each image window and their *cornerness* scores.
4) Use Linear Algebra to find λ1 and λ2. If they are both big, we have a corner.
5) Find points whose surrounding window gave large corner response (f > threshold)
6) Take the points of local maxima

Key property: in the region around a corner, image gradient has two or more dominant directions.

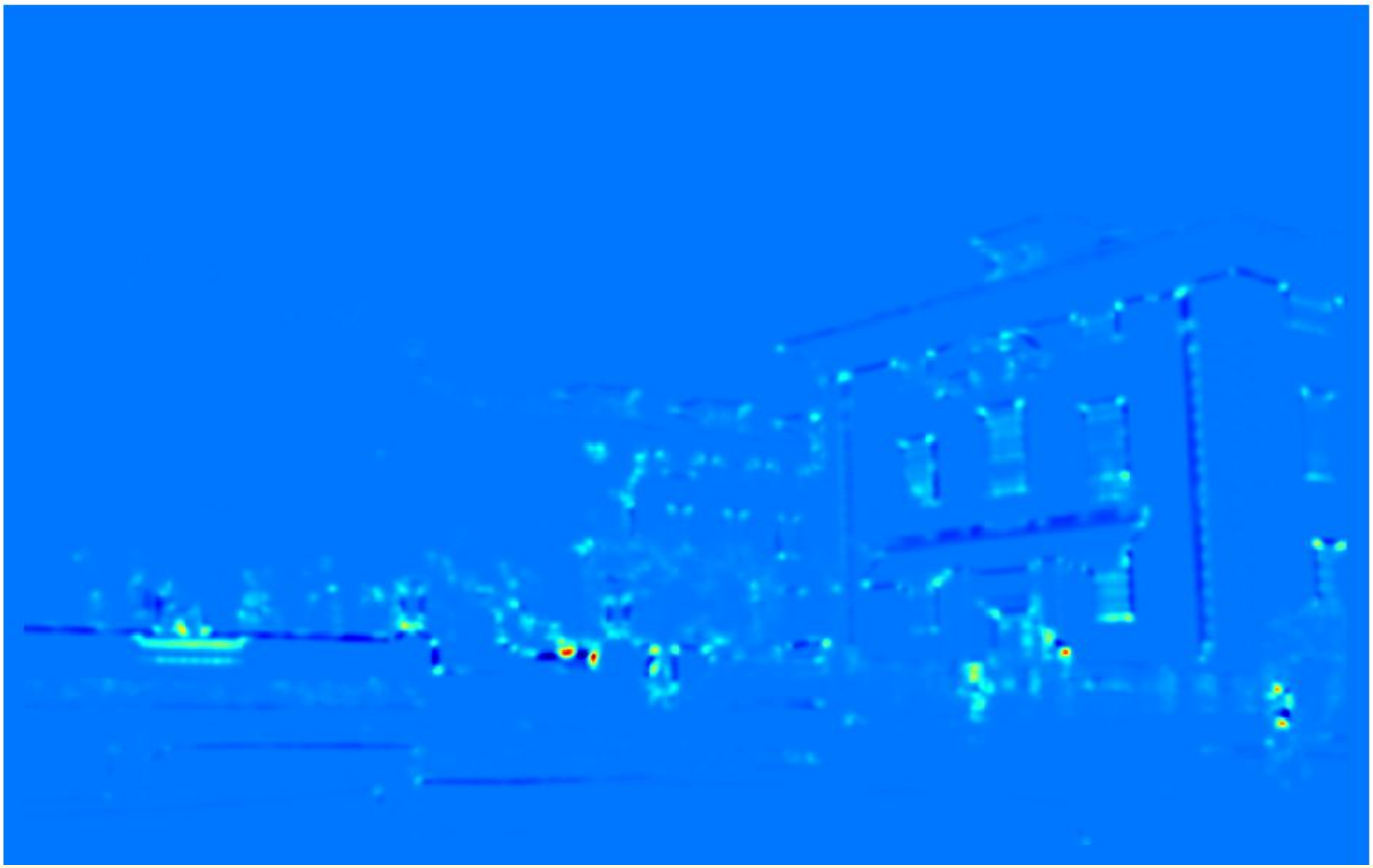
Corners are repeatable and distinctive

Example of Harris application

University of Colorado **Boulder**

Kristen Grauman

Example of Harris application



Compute corner response at every pixel.

Kristen Grauman

Example of Harris application

Kristen Grauman

Properties of the Harris corner detector

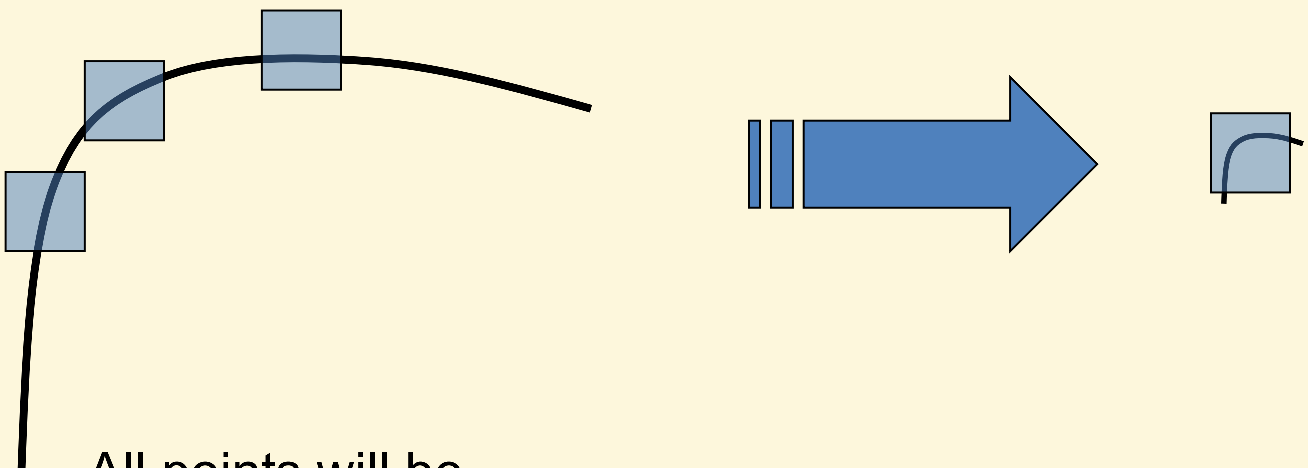
- Rotation invariant? Yes

$$M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T$$

- Scale invariant?

Properties of the Harris corner detector

- Rotation invariant? Yes
- Scale invariant? No

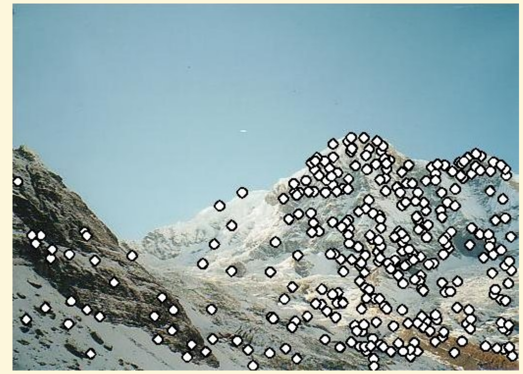


All points will be classified as edges

Corner !

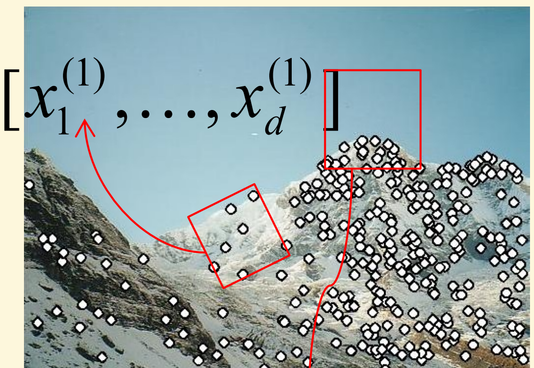
Local features: main components

1) Detection: Identify the interest points



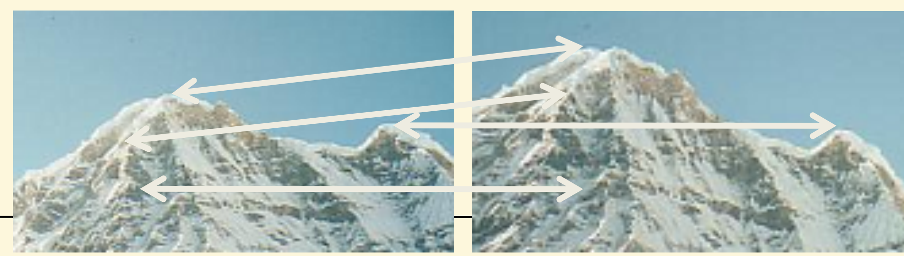
1) Description: Extract vector feature descriptor surrounding each interest point.

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

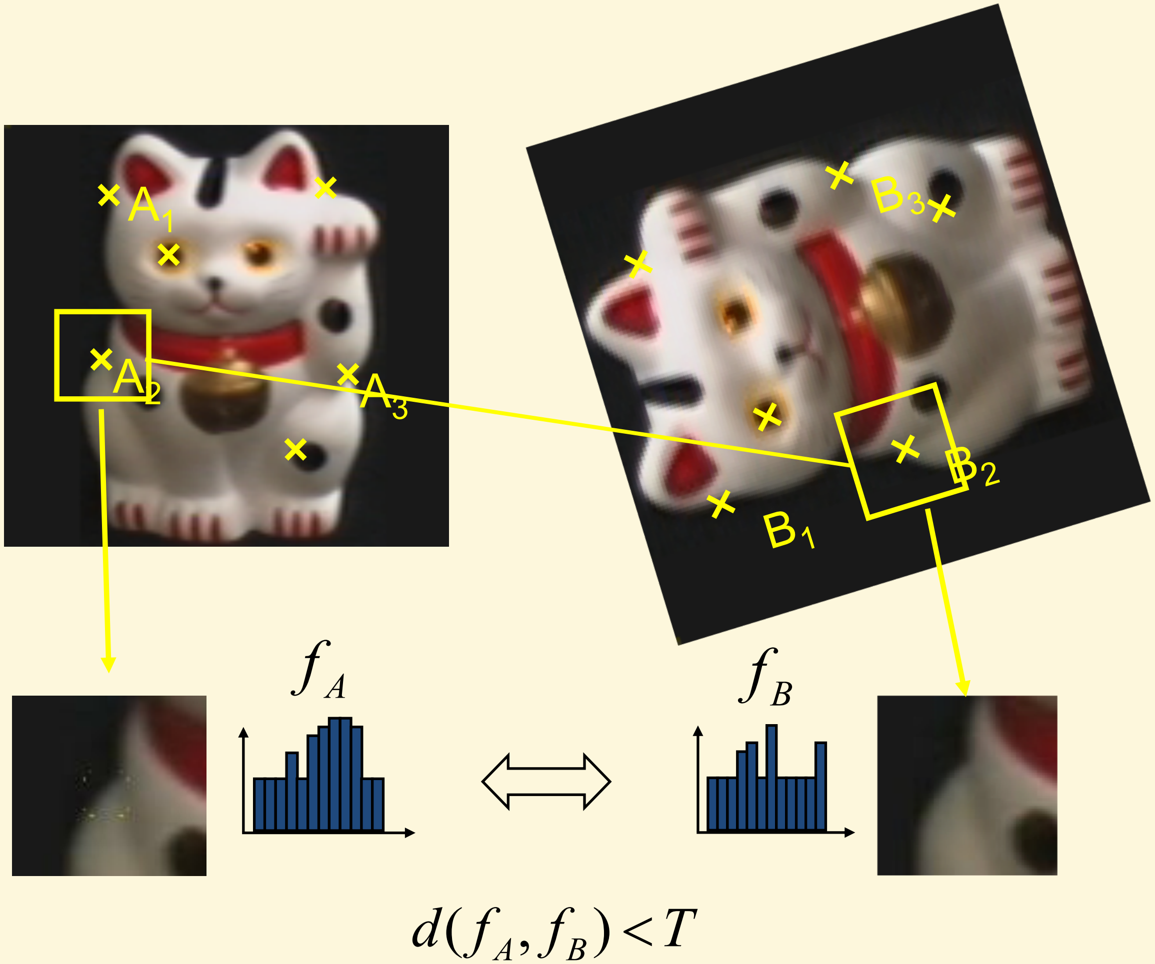
1) Matching: Determine correspondence between descriptors in two views



Kristen Grauman

K. Graumann, B. Leibe

Overview of 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**