Computer Vision

CS-E4850, 5 study credits

Lecturer: Juho Kannala

Lecture 4: Model estimation (fitting)

- Least-squares
- Robust fitting
- RANSAC
- Hough transform

These topics are covered in Szeliski's book briefly, but more thoroughly in Chapter 17 of Forsyth & Ponce:

http://courses.cs.washington.edu/courses/cse455/02wi/readings/book-7-revised-a-indx.pdf

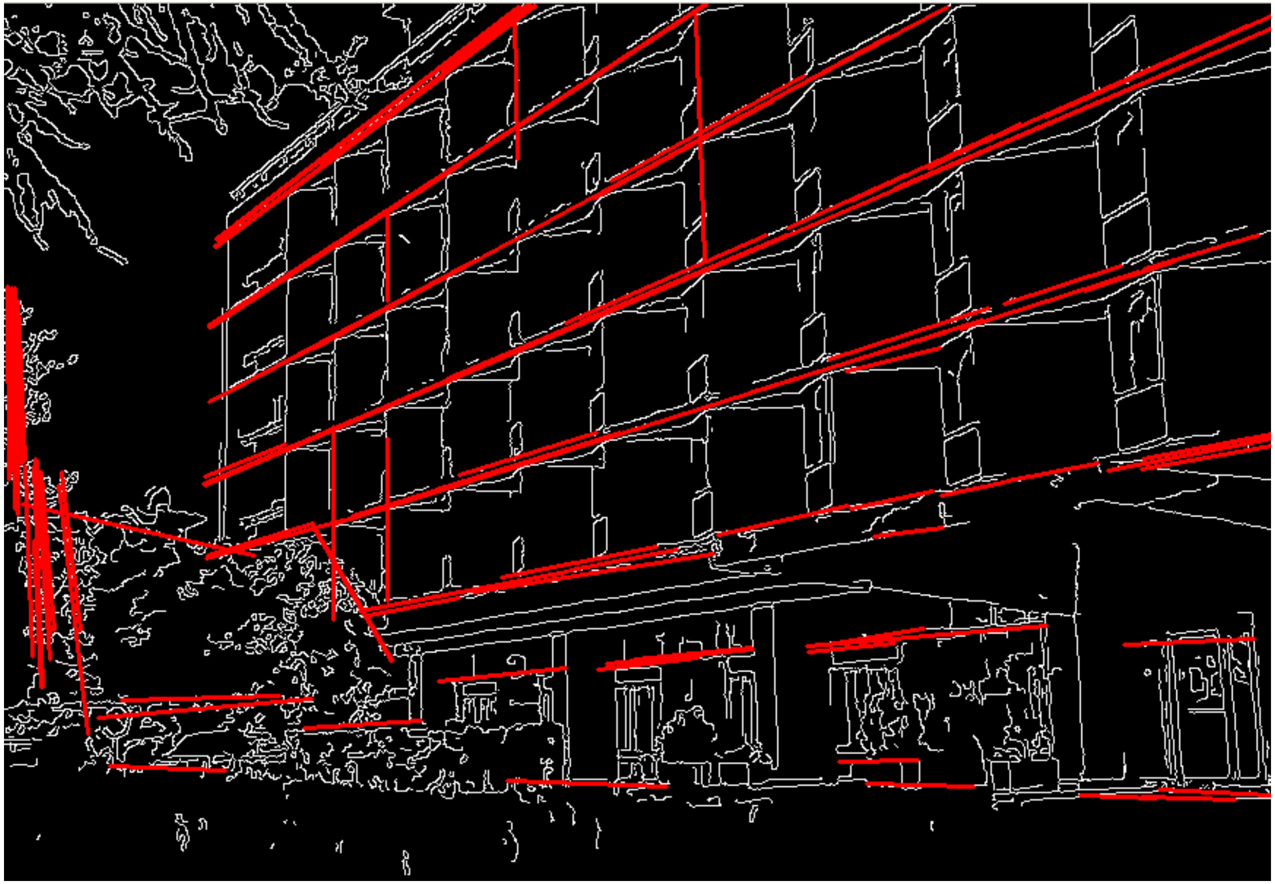
Acknowledgement: many slides from Svetlana Lazebnik, Derek Hoiem, Kristen Grauman, David Forsyth, Marc Pollefeys, and others (detailed credits on individual slides)

Relevant reading

- These topics are covered in Szeliski's book briefly, but more thoroughly in the following books:
  - Chapter 17 of Forsyth & Ponce:
  http://cmuems.com/excap/readings/forsyth-ponce-computer-vision-a-modern-approach.pdf
  - Chapter 4 of Hartley & Zisserman:
  http://cvrs.whu.edu.cn/downloads/ebooks/Multiple%20View%20Geometry%20in%20Computer%20Vision%20(Second%20Edition).pdf

Source: S. Lazebnik

Fitting

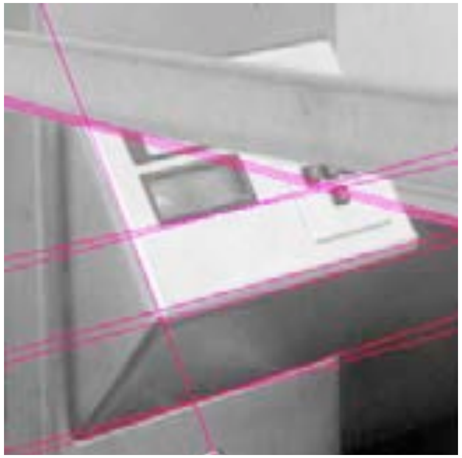


Fitting

- 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

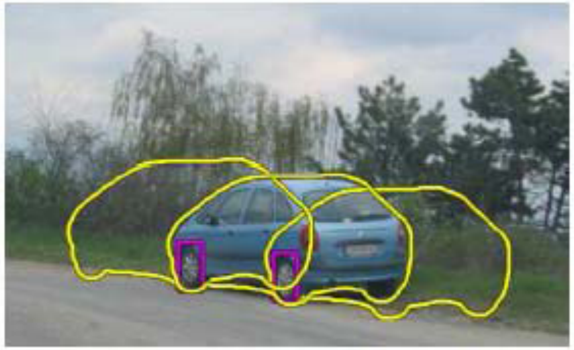
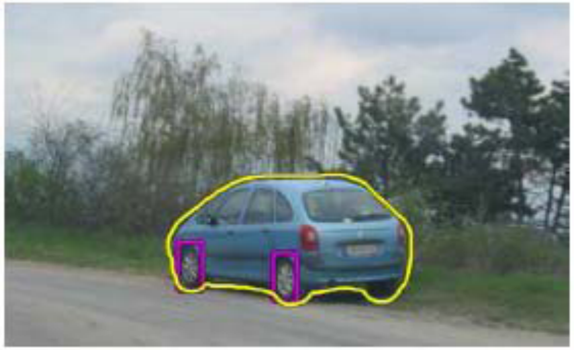
Fitting

- Choose a *parametric model* to represent a set of features



simple model: lines

simple model: circles

complicated model: car

Source: K. Grauman

Source: S. Lazebnik

Fitting: Issues

Case study: Line detection

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

Source: S. Lazebnik

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 (not covered)

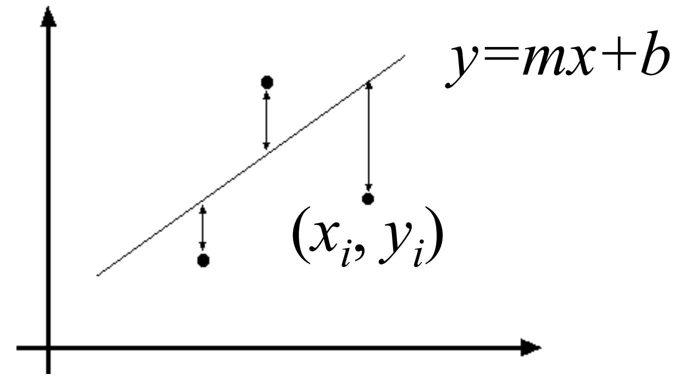
Least squares line fitting

Data: $(x_1, y_1), \ldots, (x_n, y_n)$

Line equation: $y_i = m x_i + b$

Find (m, b) to minimize

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



Source: S. Lazebnik

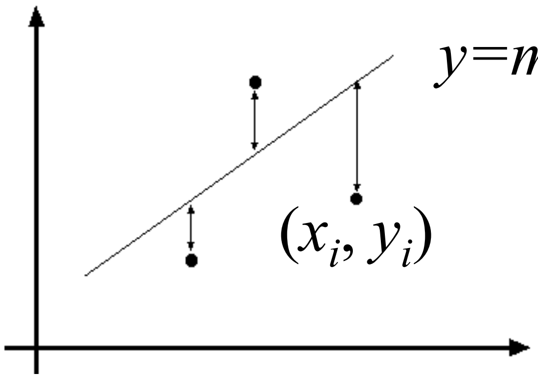
Least squares line fitting

Data: $(x_1, y_1), \ldots, (x_n, y_n)$

Line equation: $y_i = m x_i + b$

Find (m, b) to minimize

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



$$E = \|Y - XB\|^2 \quad \text{where} \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \qquad B = \begin{bmatrix} m \\ b \end{bmatrix}$$

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

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

$$\boxed{X^T XB = X^T Y}$$

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

Source: S. Lazebnik

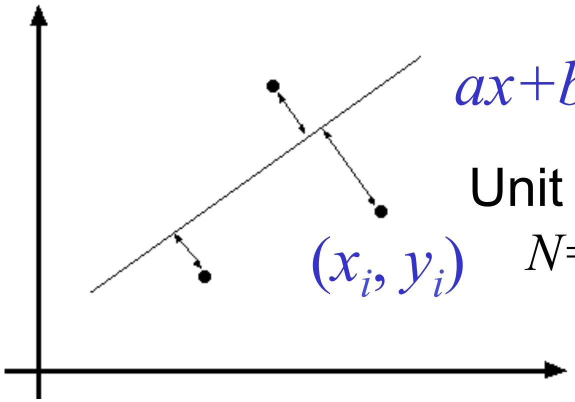
Problem with “vertical” least squares

- Not rotation-invariant
- Fails completely for vertical lines

Source: S. Lazebnik

Total least squares

Distance between point (x_i, y_i) and line $ax+by=d$ $(a^2+b^2=1)$: $|ax_i + by_i - d|$

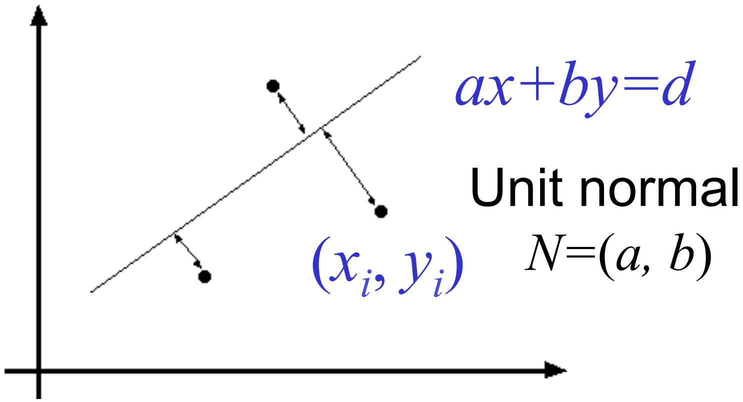


Total least squares

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

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

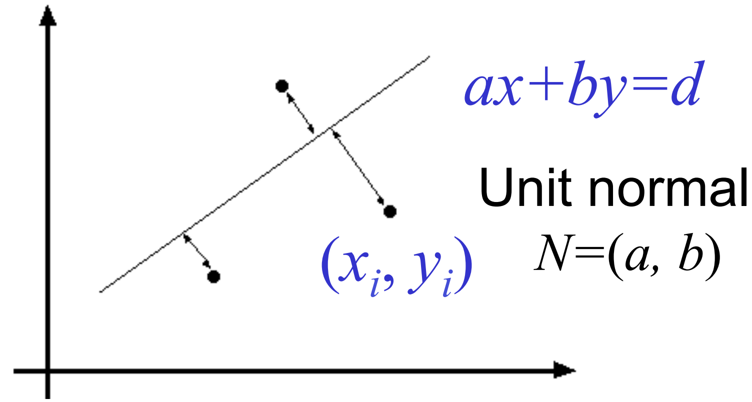


Source: S. Lazebnik

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



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

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

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

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

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

Source: S. Lazebnik

Total least squares

$$U = \begin{bmatrix} x_1 - \overline{x} & y_1 - \overline{y} \\ \vdots & \vdots \\ x_n - \overline{x} & y_n - \overline{y} \end{bmatrix} \quad U^T U = \begin{bmatrix} \sum_{i=1}^{n} (x_i - \overline{x})^2 & \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y}) \\ \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y}) & \sum_{i=1}^{n} (y_i - \overline{y})^2 \end{bmatrix}$$

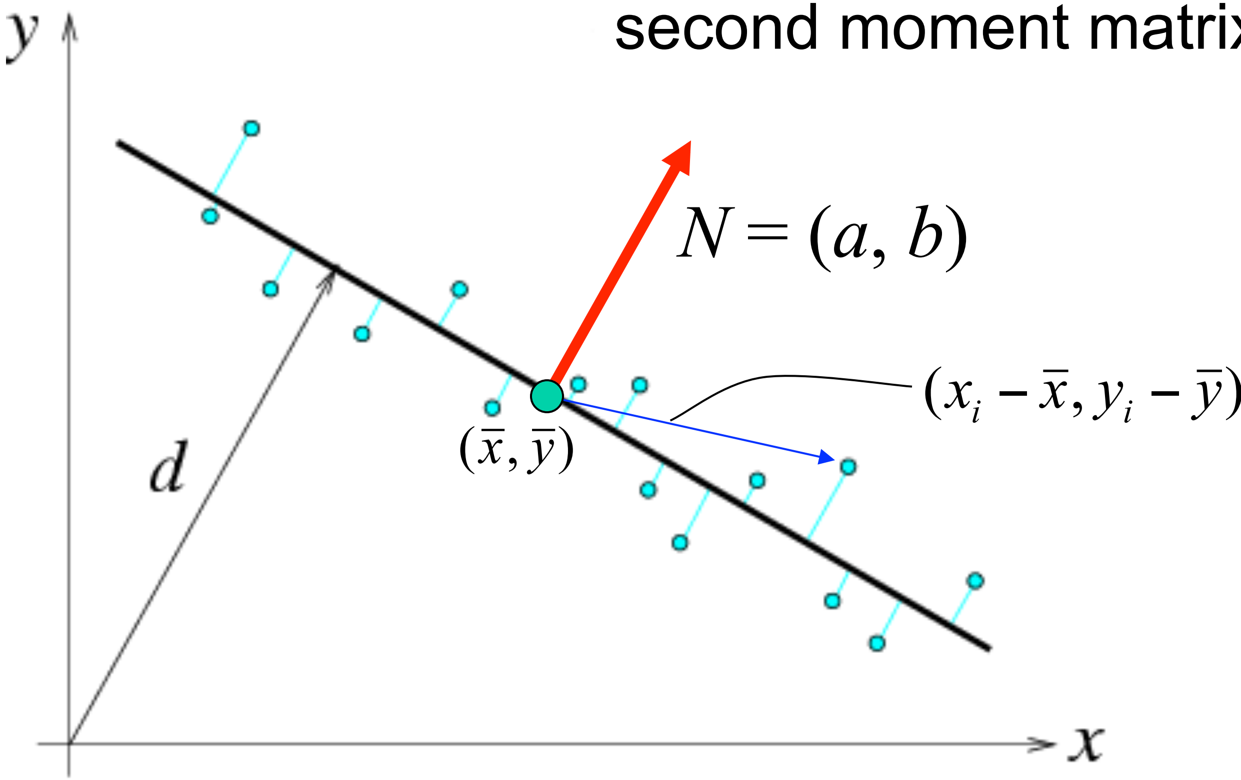
second moment matrix

Source: S. Lazebnik

Total least squares

$$U = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \qquad U^T U = \begin{bmatrix} \sum_{i=1}^{n} (x_i - \bar{x})^2 & \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \\ \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^{n} (y_i - \bar{y})^2 \end{bmatrix}$$

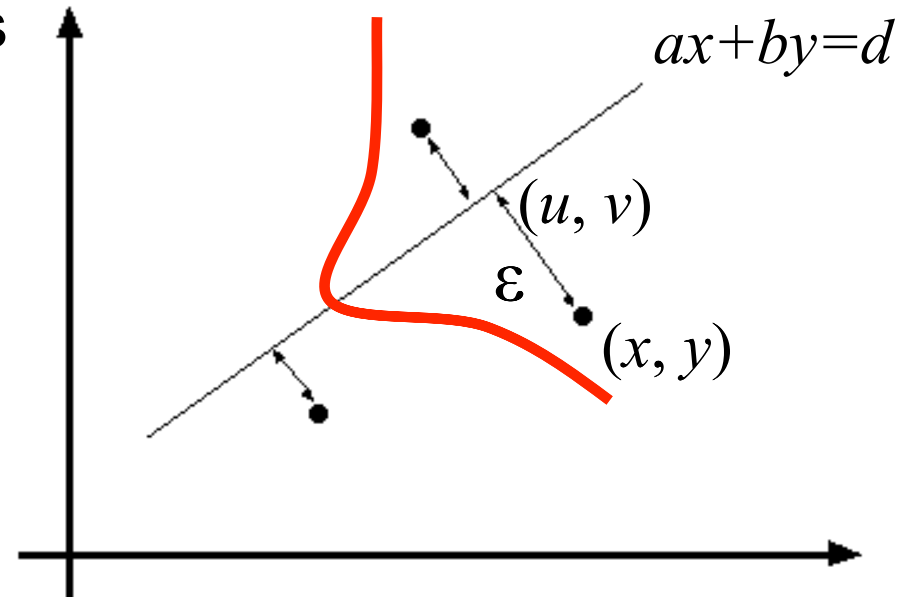
second moment matrix



F&P (2nd ed.) sec. 22.1

Least squares as likelihood maximization

- **Generative model**: line points are sampled independently and corrupted by Gaussian noise in the direction perpendicular to the line



$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + \varepsilon \begin{pmatrix} a \\ b \end{pmatrix}$$

point on the line

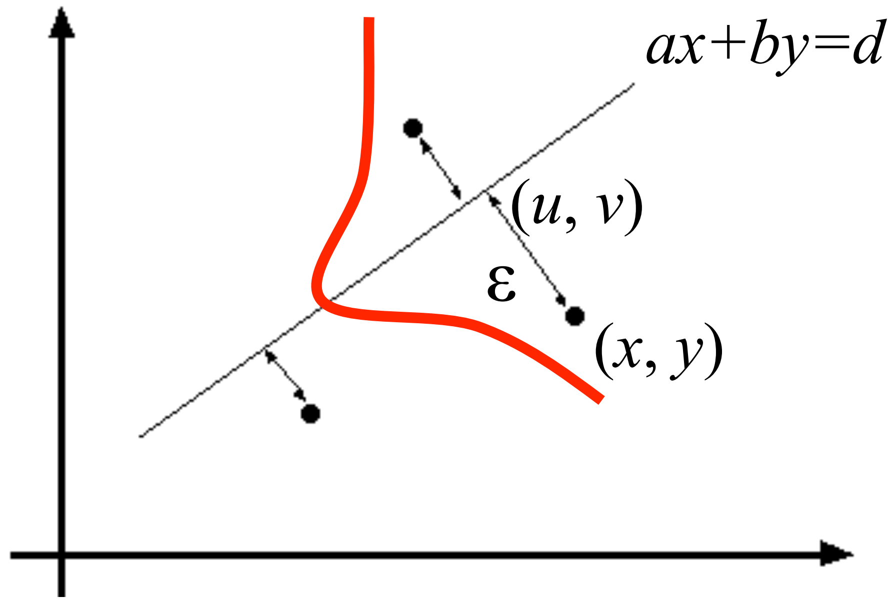
noise: sampled from zero-mean Gaussian with std. dev. σ

normal direction

Source: S. Lazebnik

Least squares as likelihood maximization

- **Generative model**: line points are sampled independently and corrupted by Gaussian noise in the direction perpendicular to the line



$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + \varepsilon \begin{pmatrix} a \\ b \end{pmatrix}$$

Likelihood of points given line parameters (*a*, *b*, *d*):

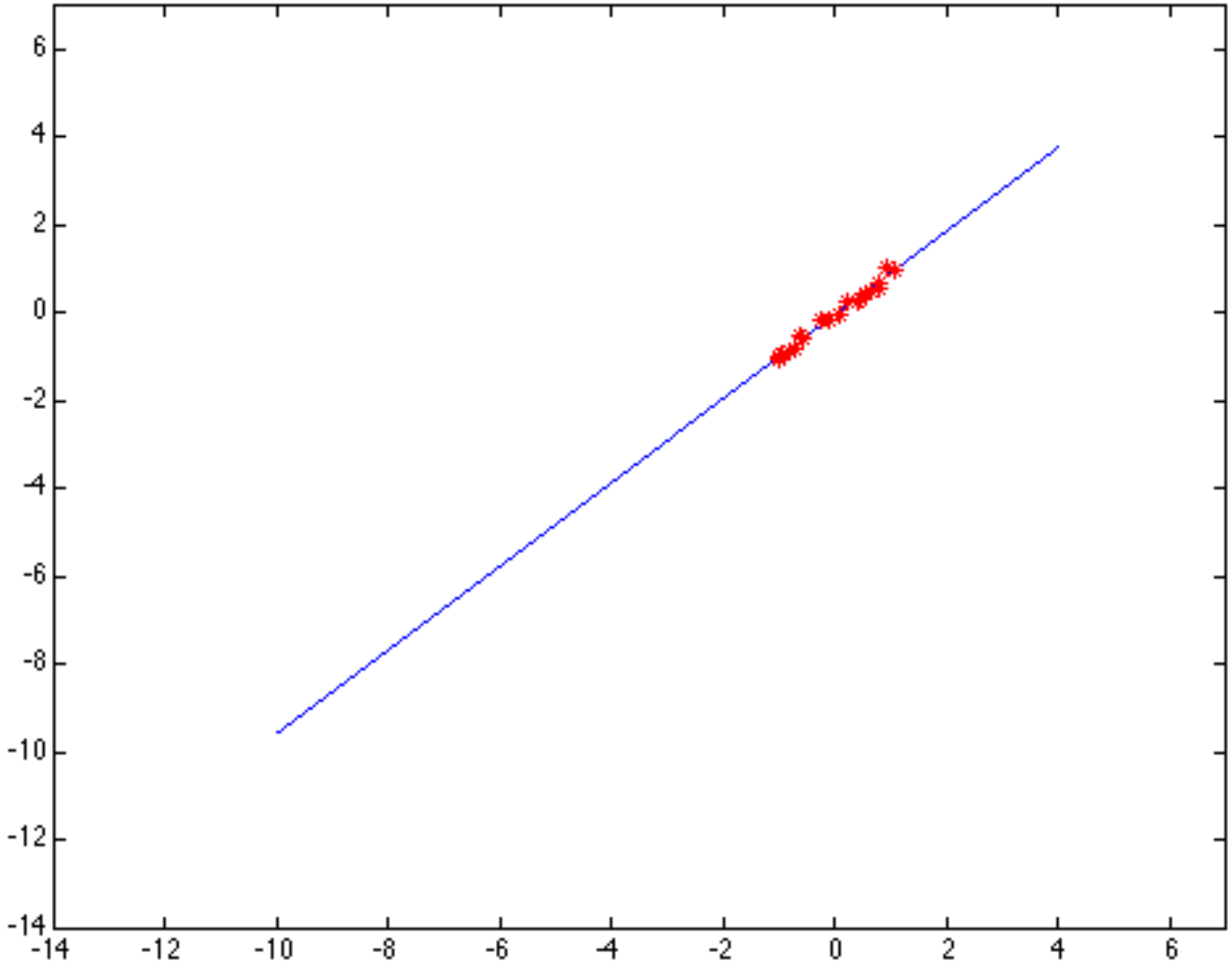
$$P(x_1, y_1, \ldots, x_n, y_n \mid a, b, d) = \prod_{i=1}^{n} P(x_i, y_i \mid a, b, d) \propto \prod_{i=1}^{n} \exp\left(-\frac{(ax_i + by_i - d)^2}{2\sigma^2} \right)$$

Log-likelihood: $L(x_1, y_1, \ldots, x_n, y_n \mid a, b, d) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (ax_i + by_i - d)^2$

Source: S. Lazebnik

Least squares: Robustness to noise

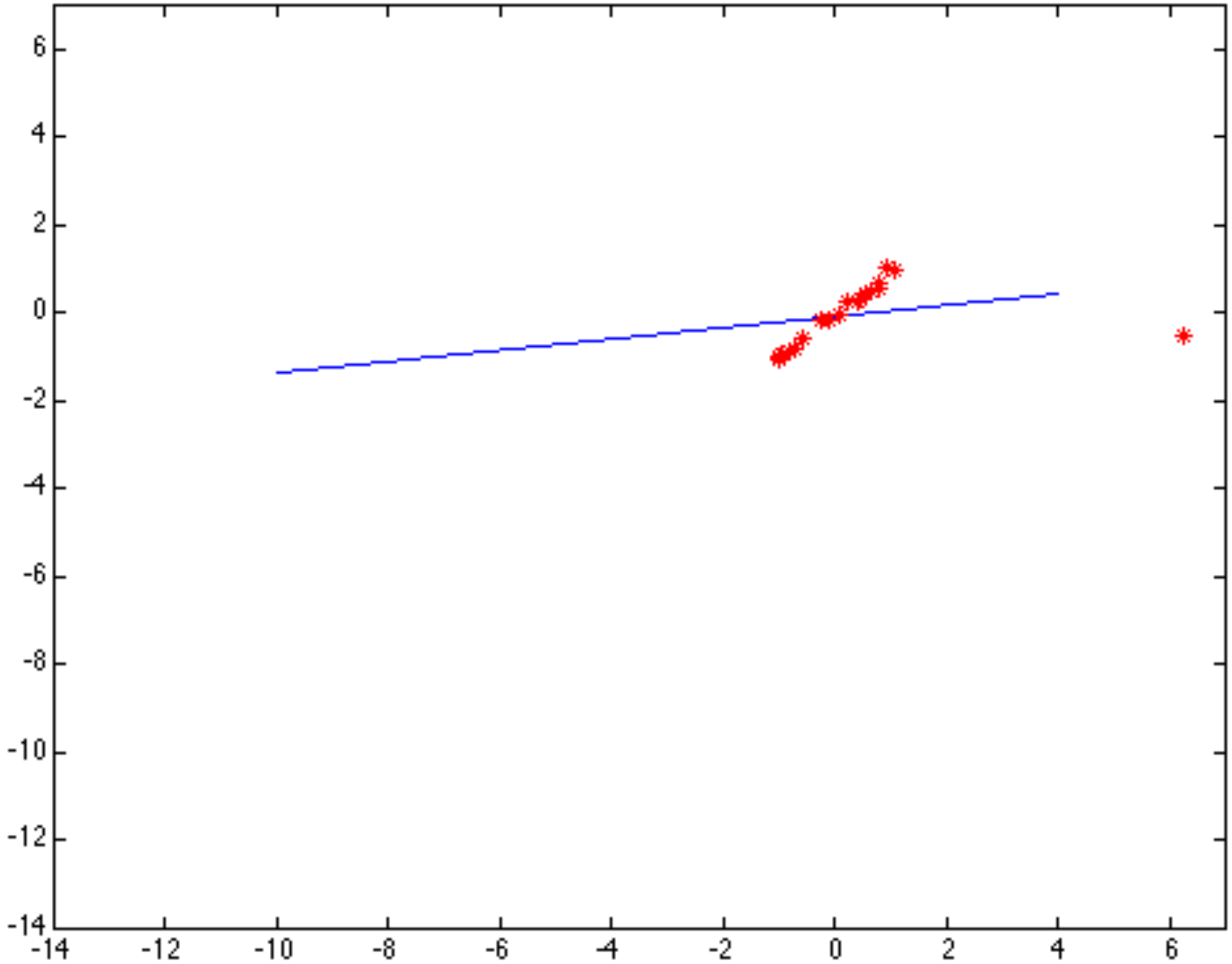
Least squares fit to the red points:



Source: S. Lazebnik

Least squares: Robustness to noise

Least squares fit with an outlier:



Problem: squared error heavily penalizes outliers

Source: S. Lazebnik

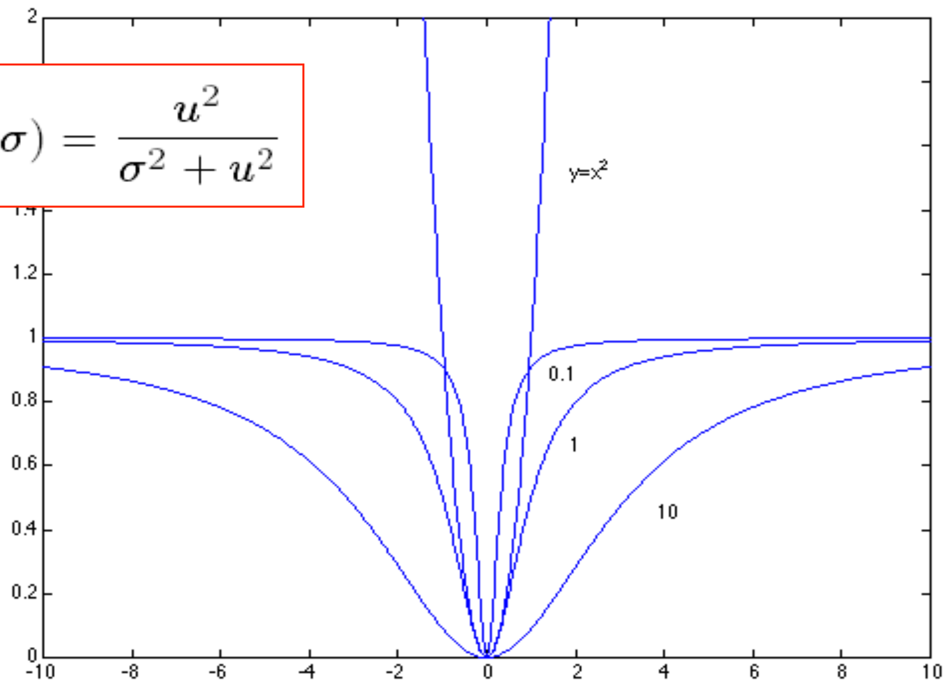
Robust estimators

- General approach: find model parameters θ that minimize

$$\sum_i \rho\left(r_i\left(x_i, \theta\right); \sigma\right)$$

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

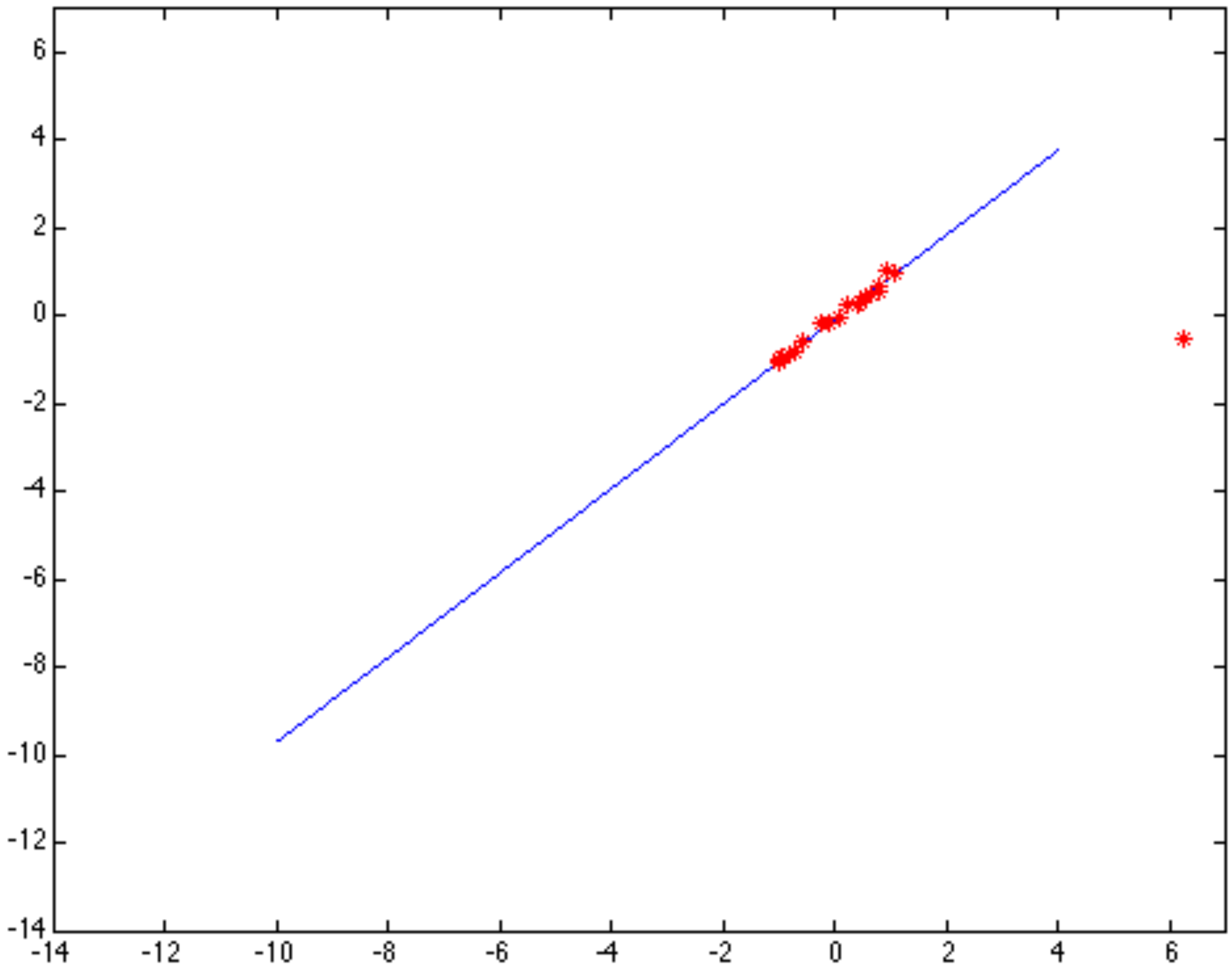
$$\rho(u;\sigma) = \frac{u^2}{\sigma^2 + u^2}$$



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

Source: S. Lazebnik

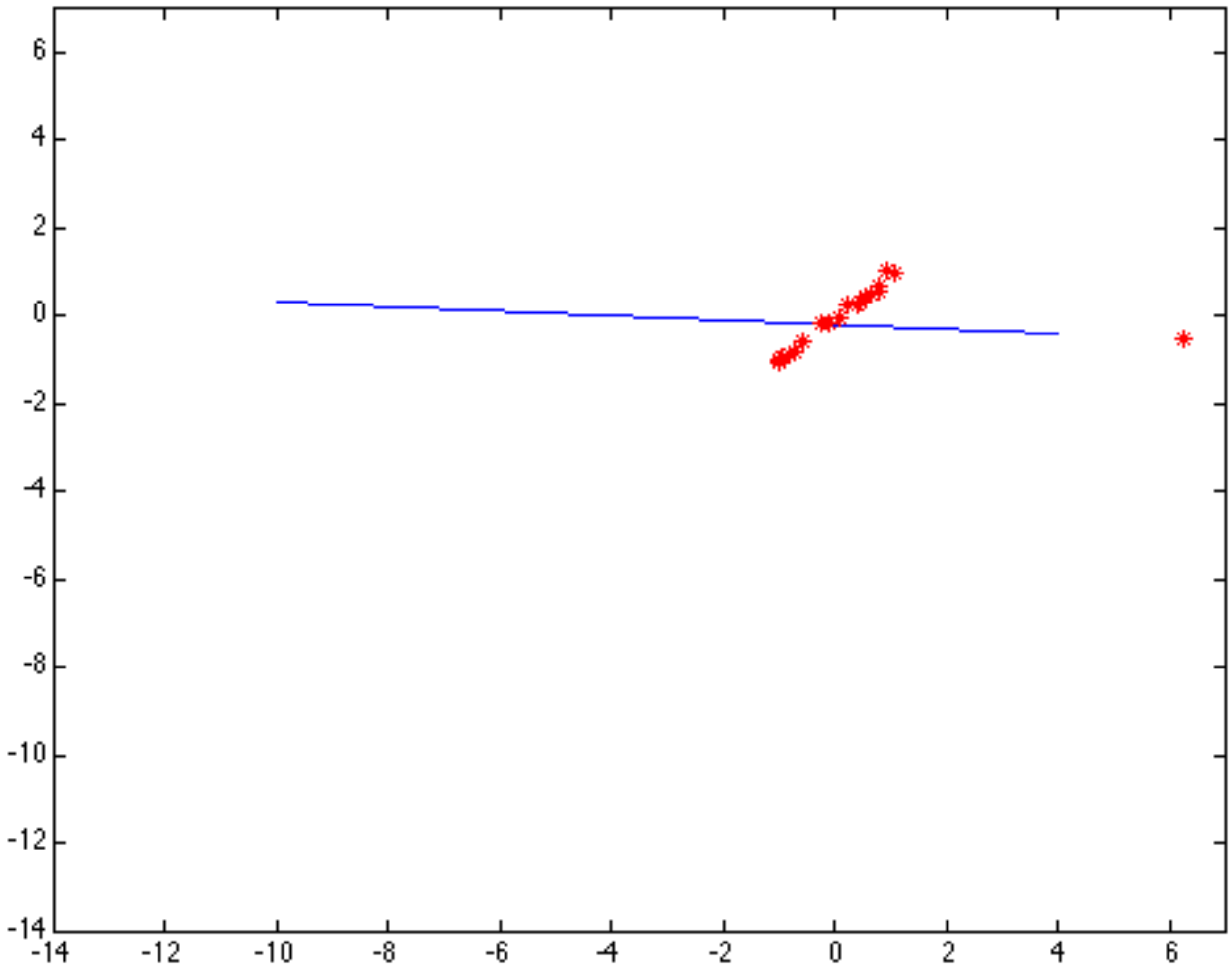
Choosing the scale: Just right



The effect of the outlier is minimized

Source: S. Lazebnik

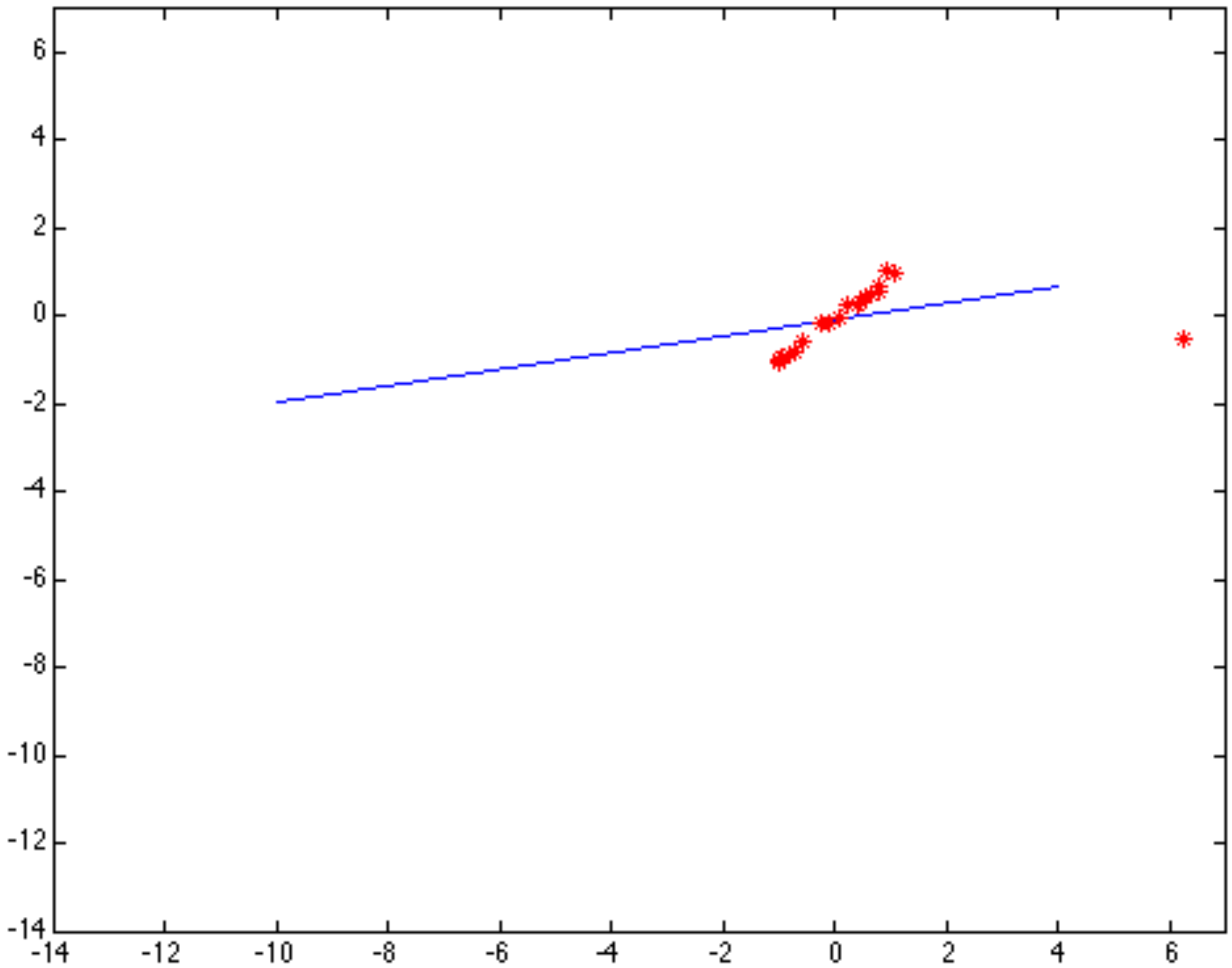
Choosing the scale: Too small



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

Source: S. Lazebnik

Choosing the scale: Too large



Behaves much the same as least squares

Source: S. Lazebnik

Robust estimation: Details

- Robust fitting is a nonlinear optimization problem that must be solved iteratively
- Least squares solution can be used for initialization
- Scale of robust function should be chosen adaptively based on median residual

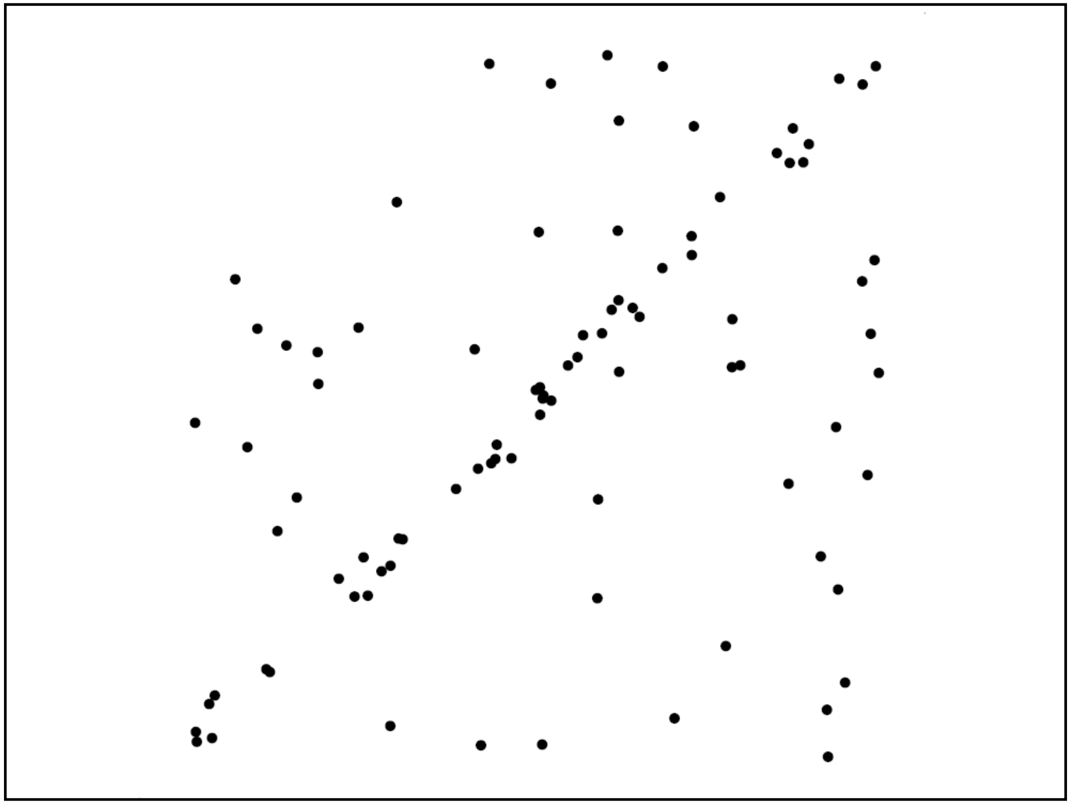
Source: S. Lazebnik

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

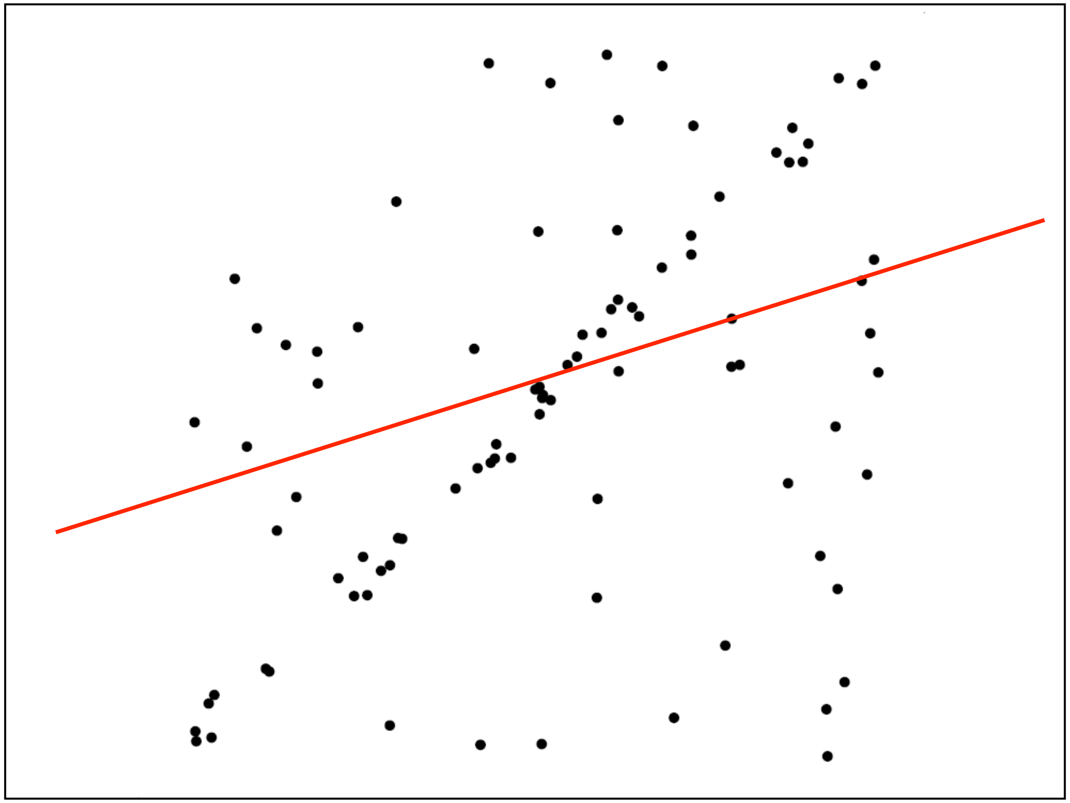
M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

RANSAC for line fitting example



Source: R. Raguram

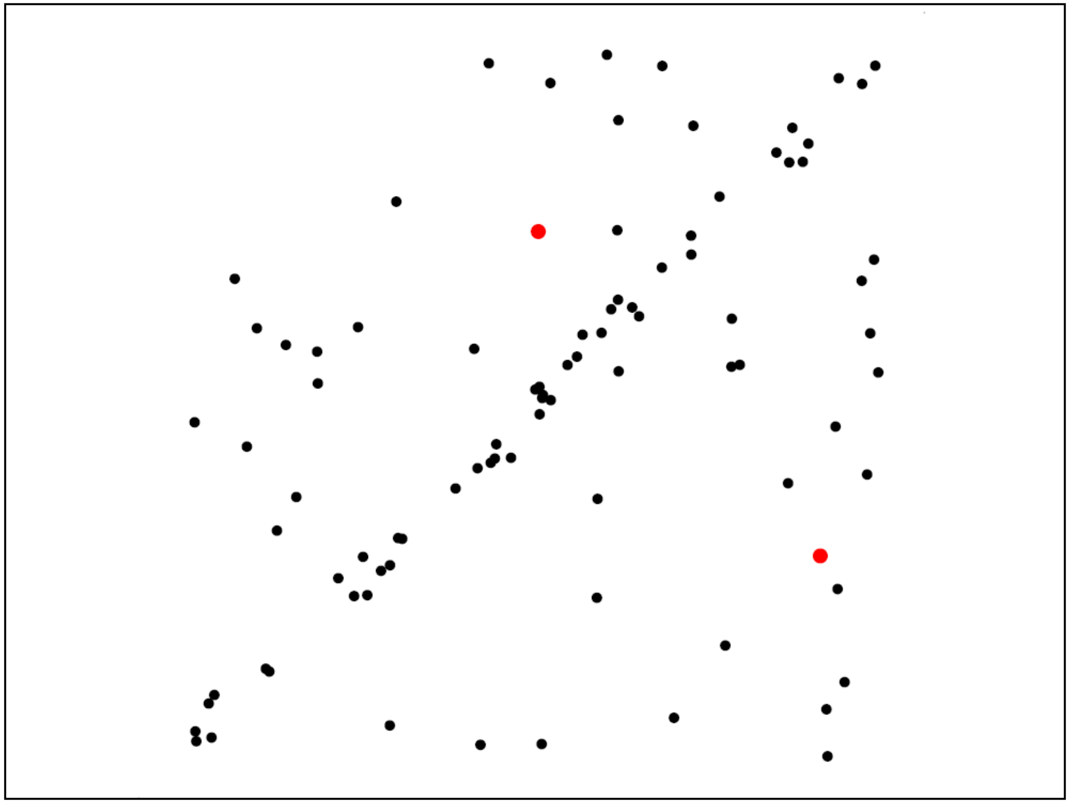
RANSAC for line fitting example



Least-squares fit

Source: R. Raguram

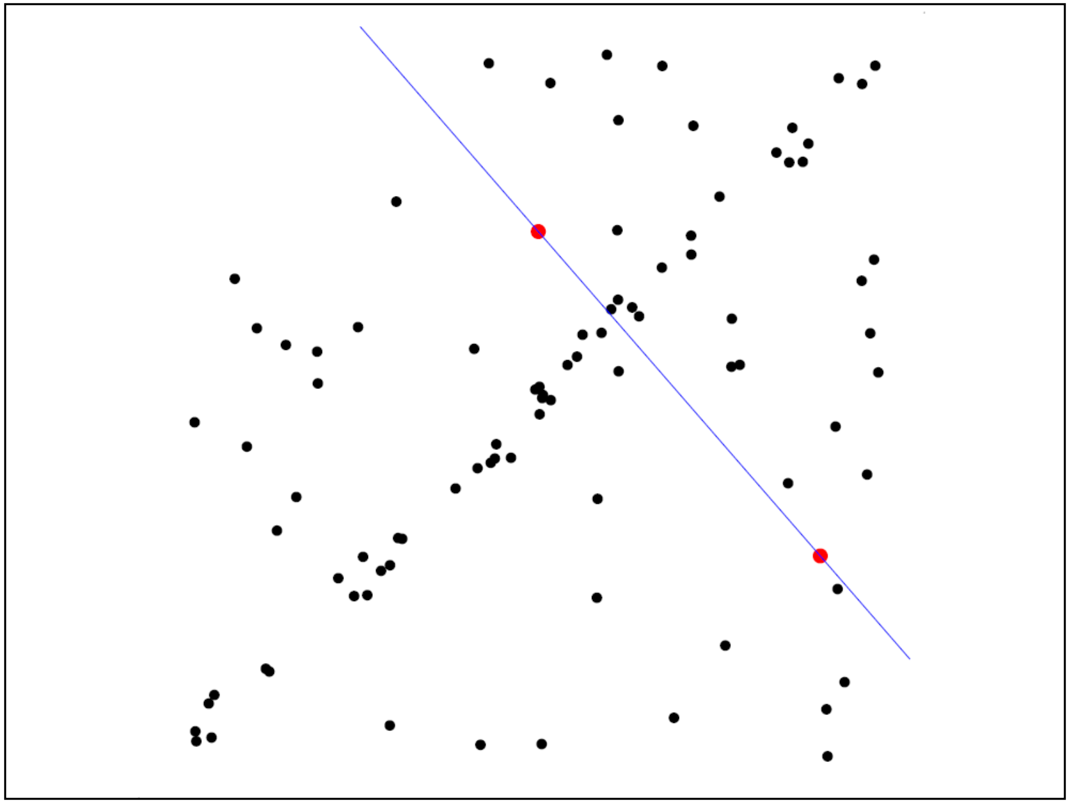
RANSAC for line fitting example



1. Randomly select minimal subset of points

Source: R. Raguram

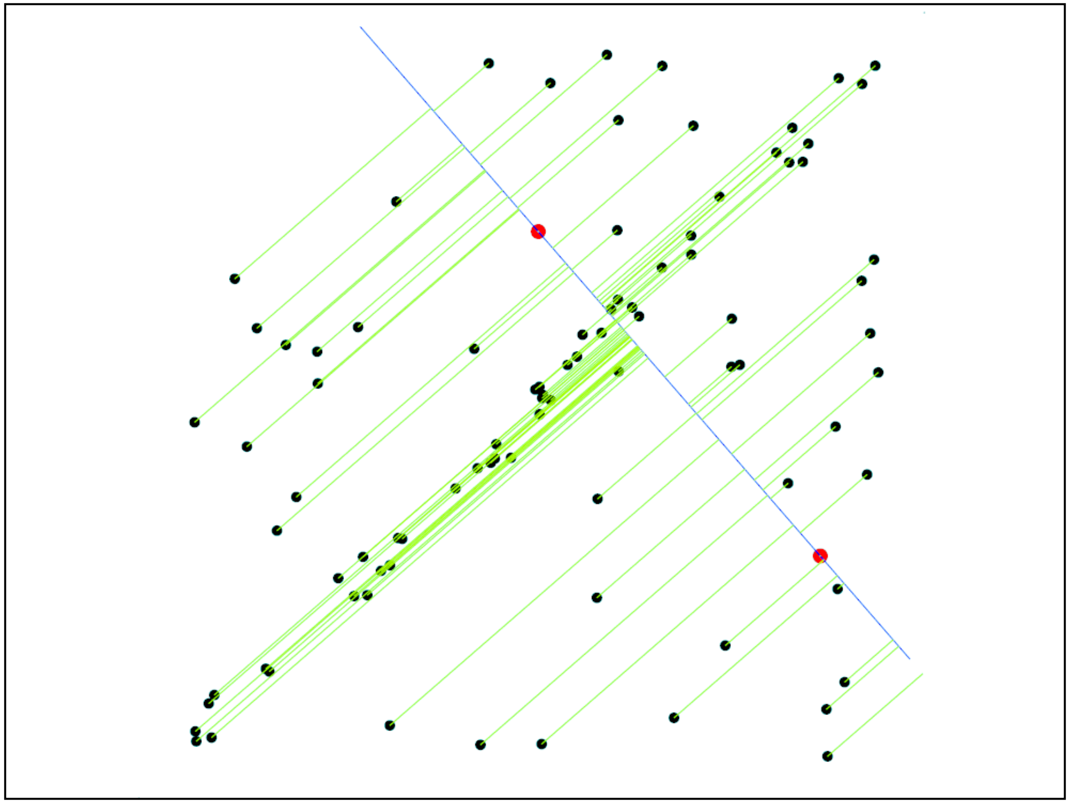
RANSAC for line fitting example



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

Source: R. Raguram

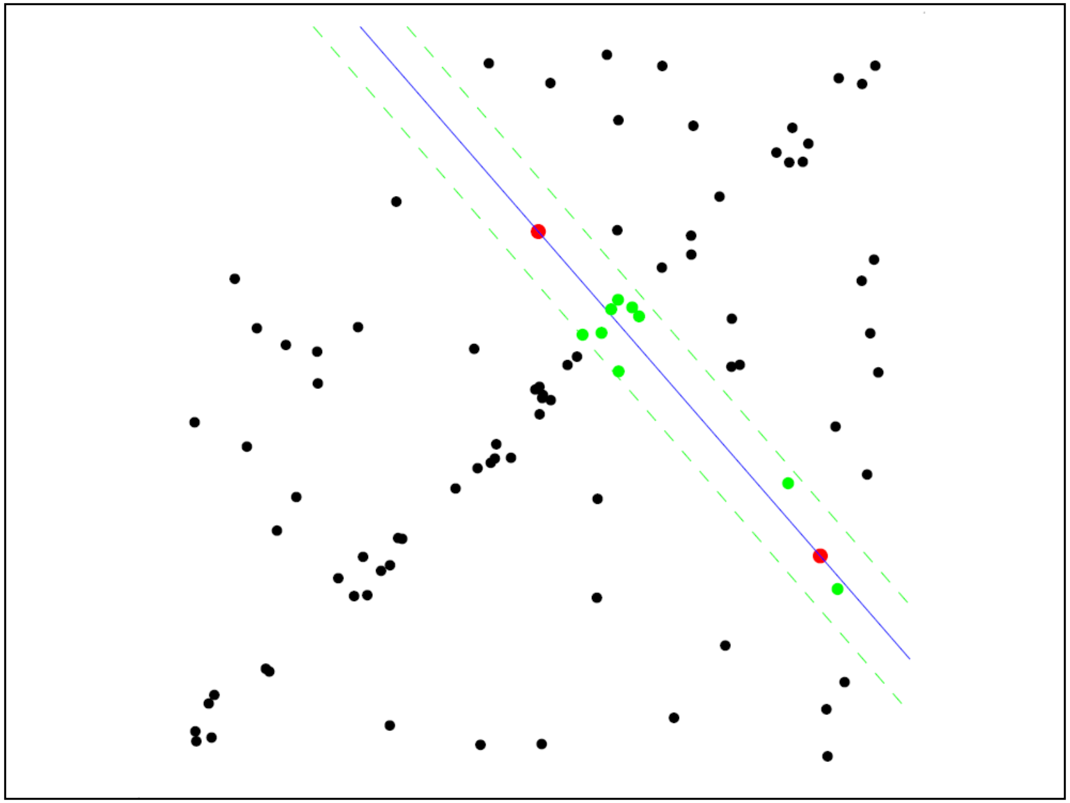
RANSAC for line fitting example



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

Source: R. Raguram

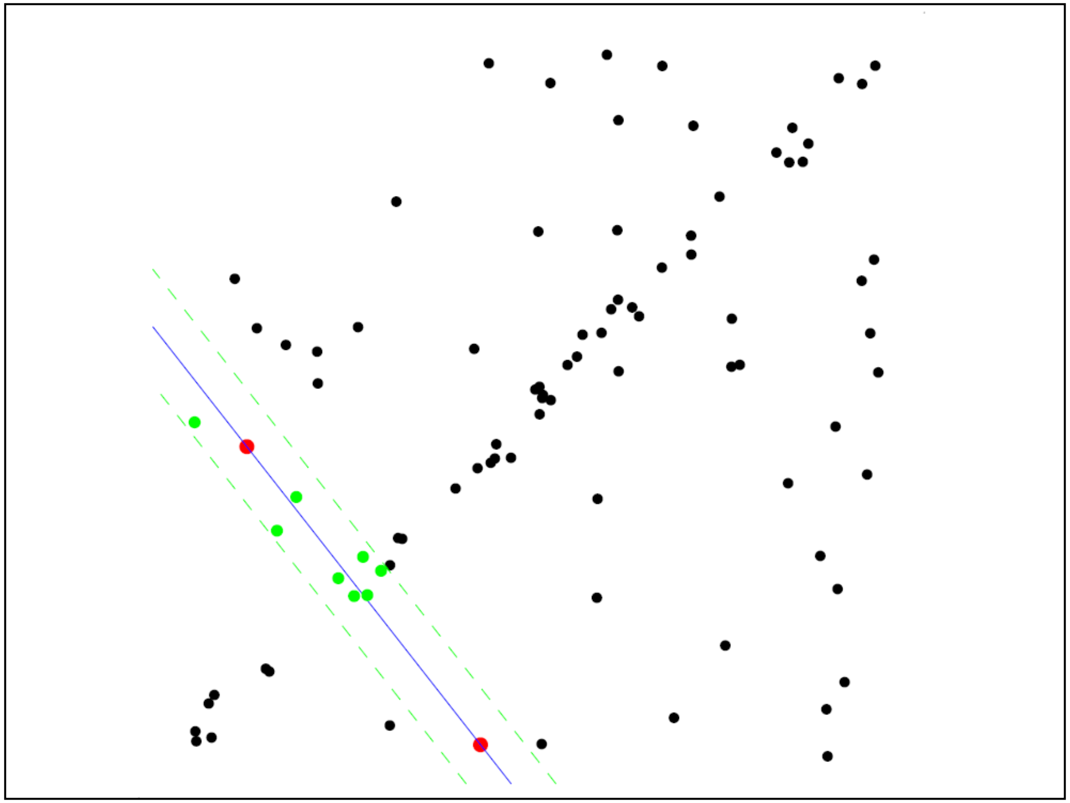
RANSAC for line fitting example



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

Source: R. Raguram

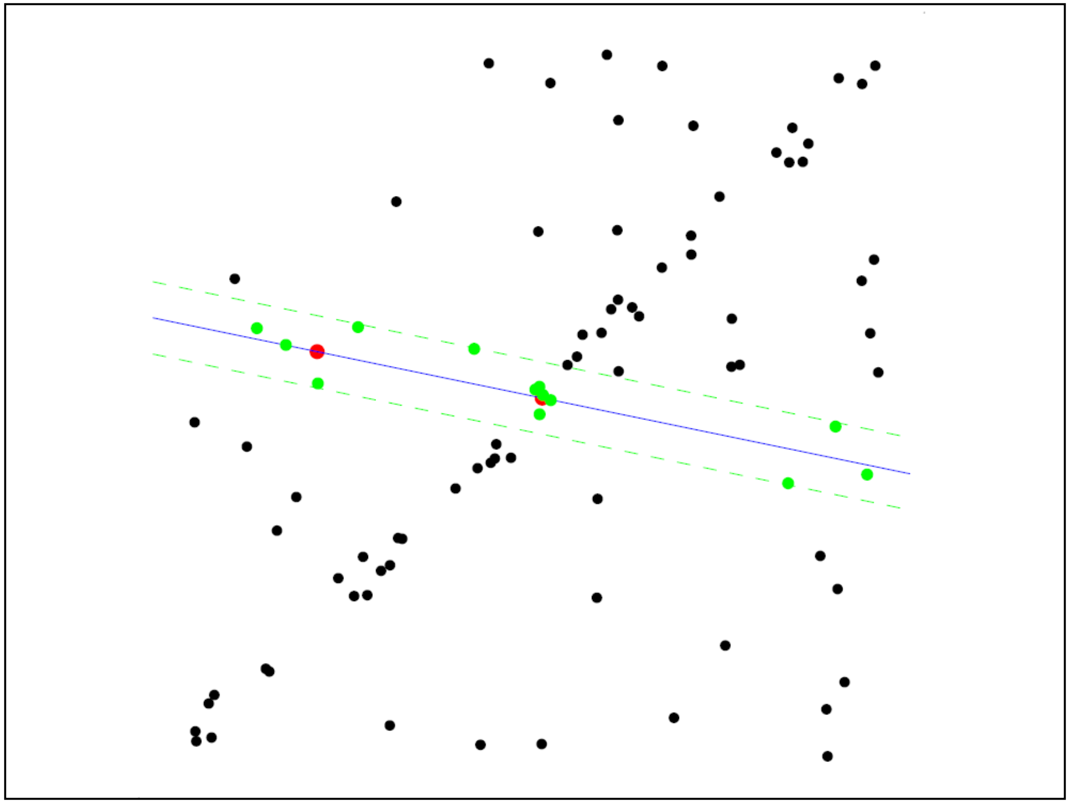
RANSAC for line fitting example



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

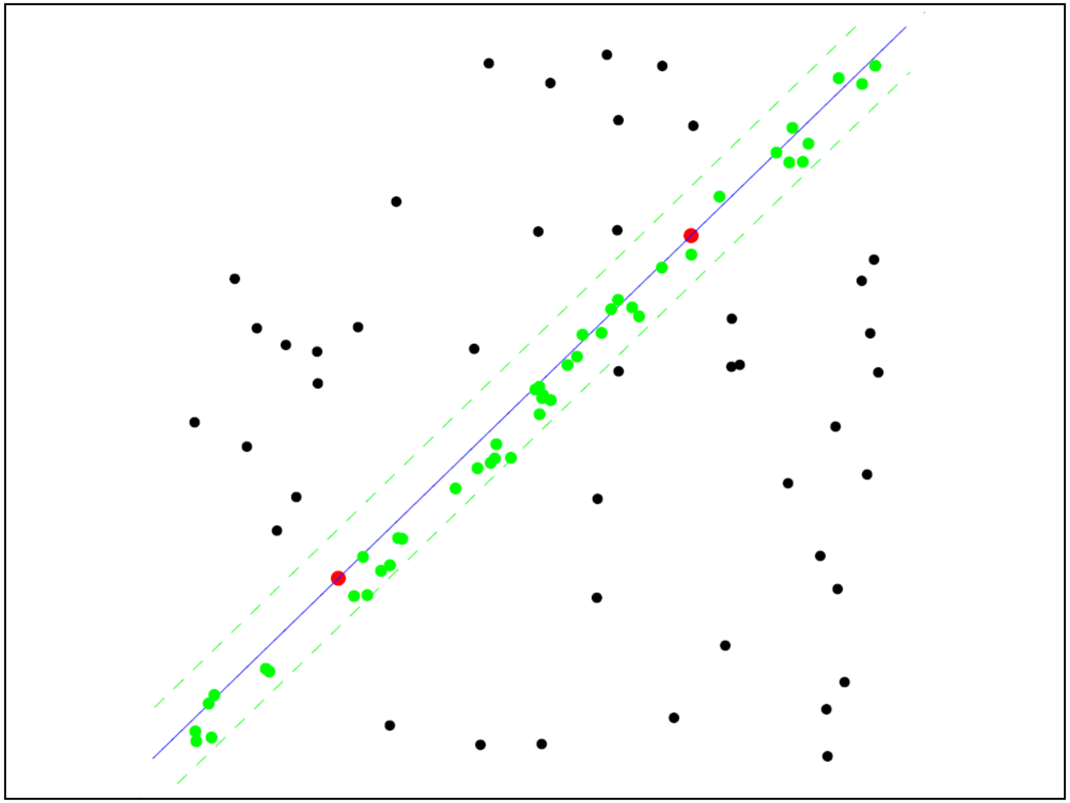


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example

Uncontaminated sample

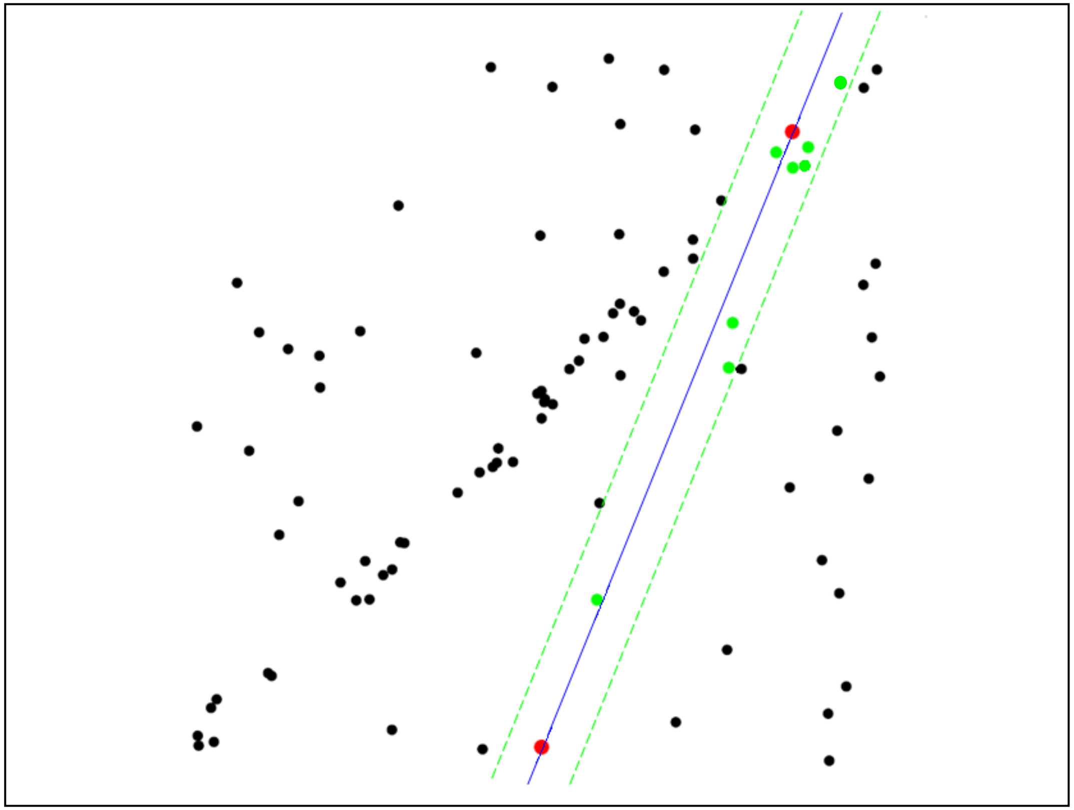


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

Source: S. Lazebnik

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

Source: S. Lazebnik

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*)

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

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

	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

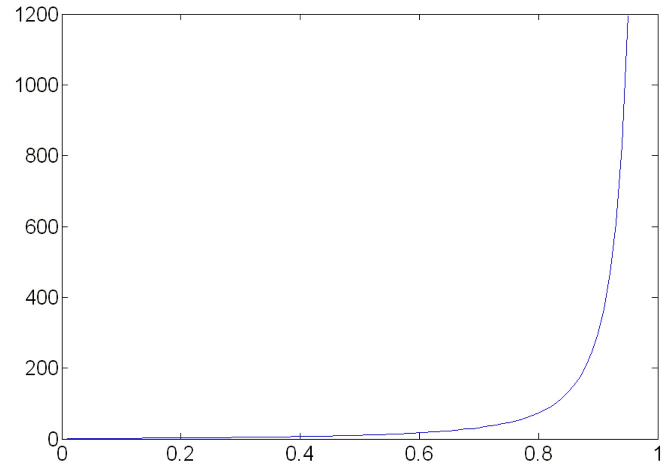
Source: M. Pollefeys

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*)

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

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



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

- Outlier 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
    - If inlier ratio is highest of any found so far, set e = 1 – (number of inliers)/(total number of points)
    - Recompute *N* from *e:*

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

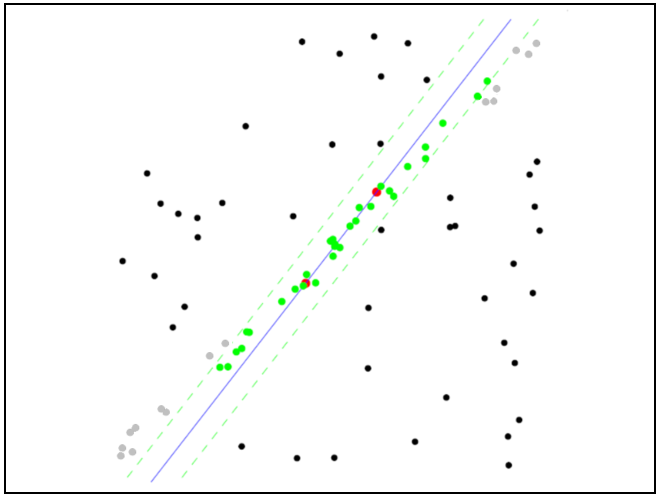
    - Increment the *sample_count* by 1

Source: M. Pollefeys

Source: S. Lazebnik

RANSAC pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- 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



Source: S. Lazebnik

Fitting: Review

- Least squares
- Robust fitting
- RANSAC

Source: S. Lazebnik

Fitting: Review

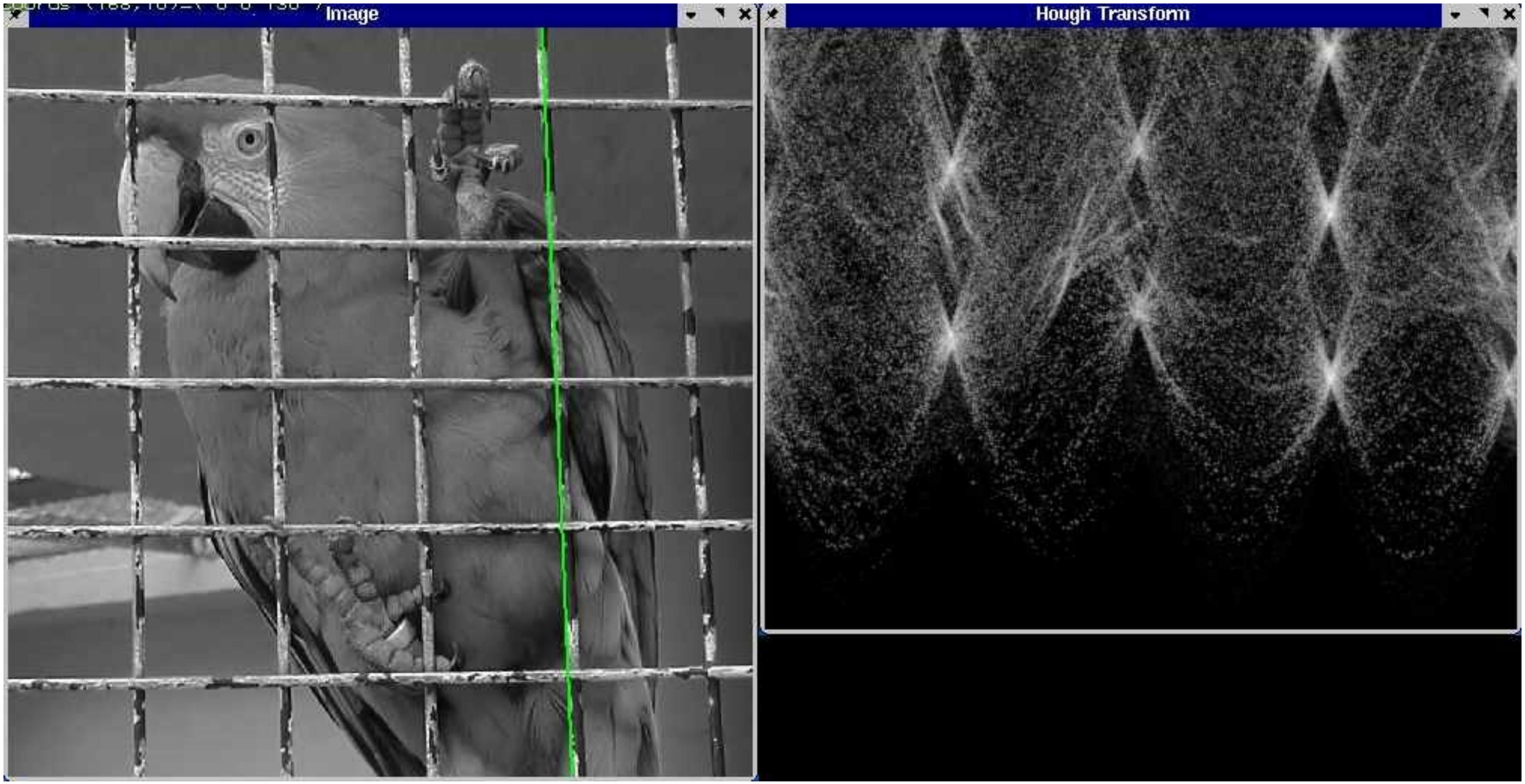
- ✓ If we know which points belong to the line, how do we find the "optimal" line parameters?
  - ✓ Least squares

- ✓ What if there are outliers?
  - ✓ Robust fitting, RANSAC

- What if there are many lines?
  - Voting methods: RANSAC, Hough transform

Source: S. Lazebnik

Fitting: 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

Source: S. Lazebnik

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

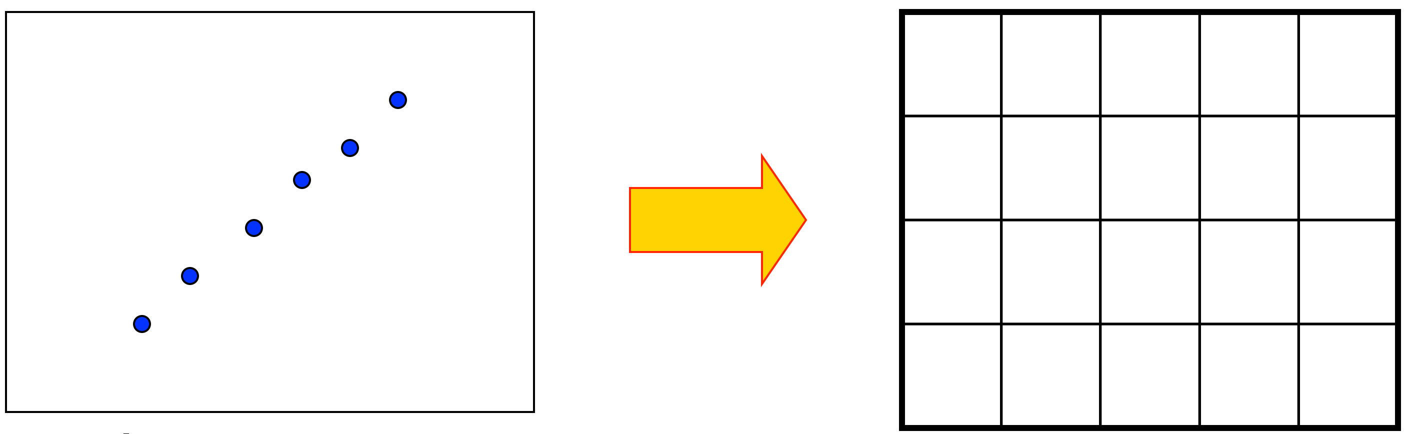


Image space

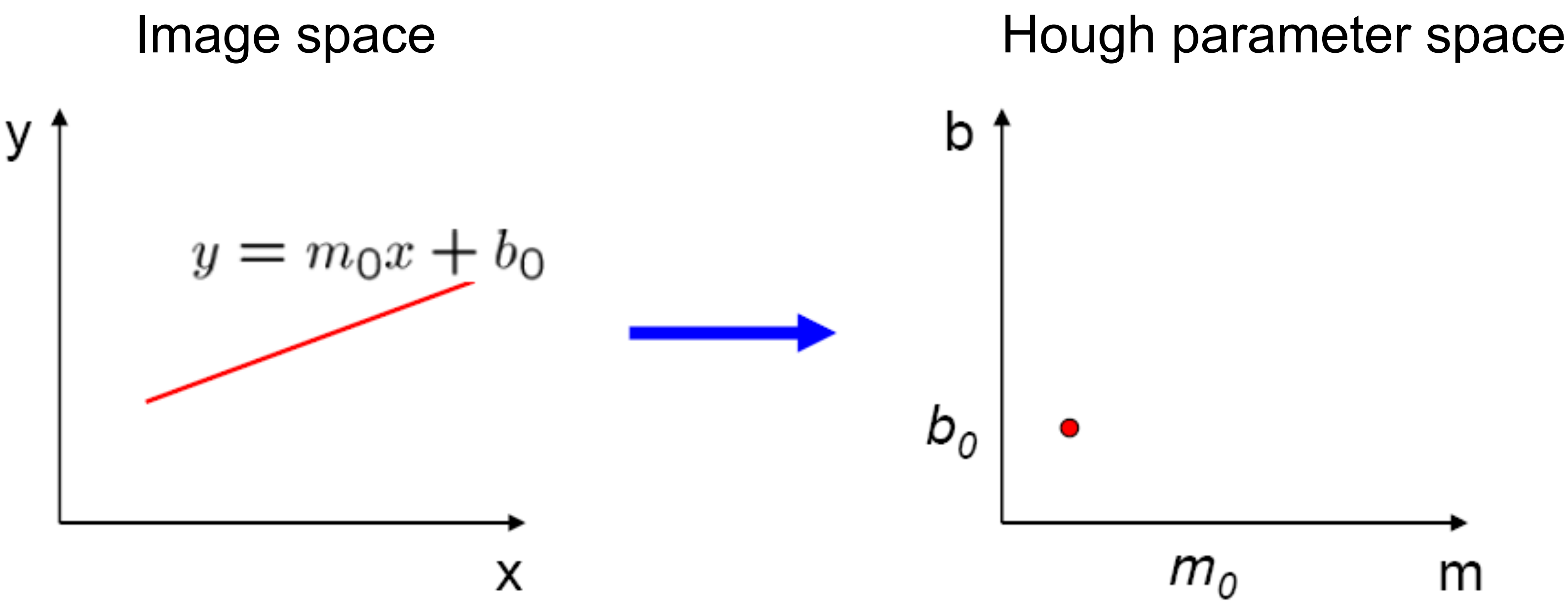
Hough parameter space

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

Source: S. Lazebnik

Parameter space representation

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

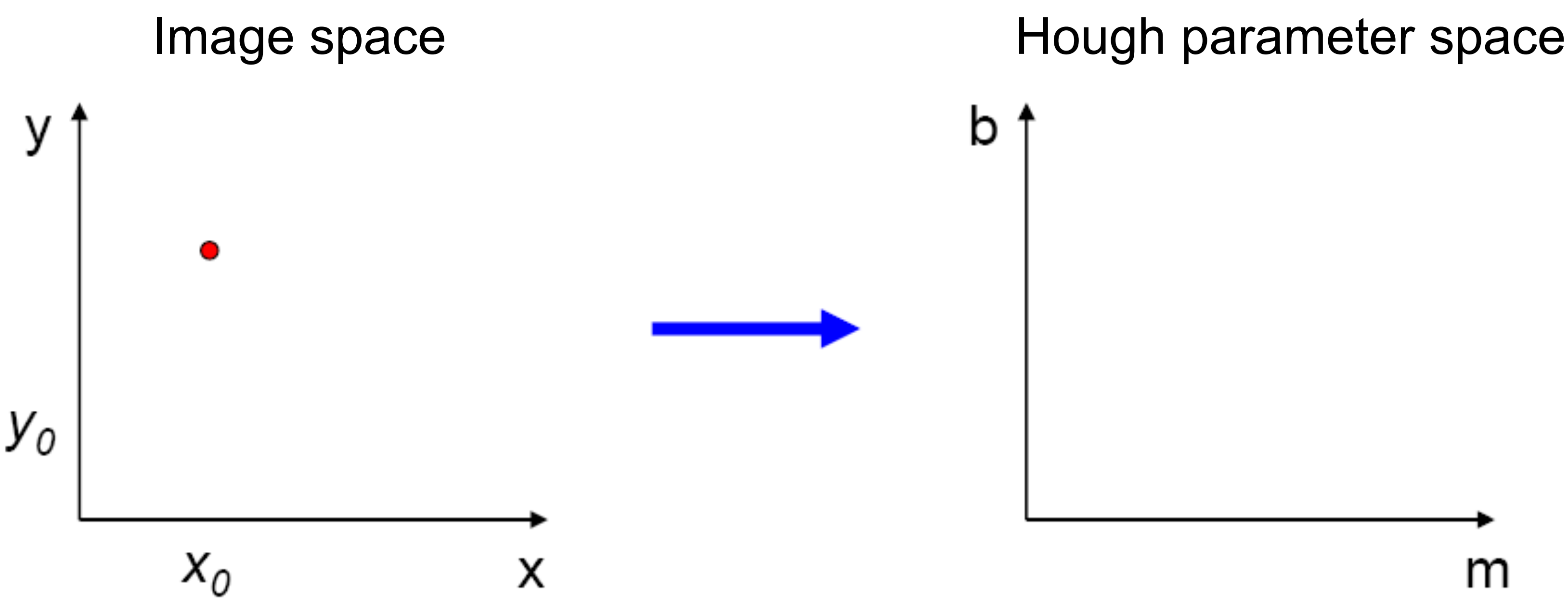


Source: S. Seitz

Source: S. Lazebnik

Parameter space representation

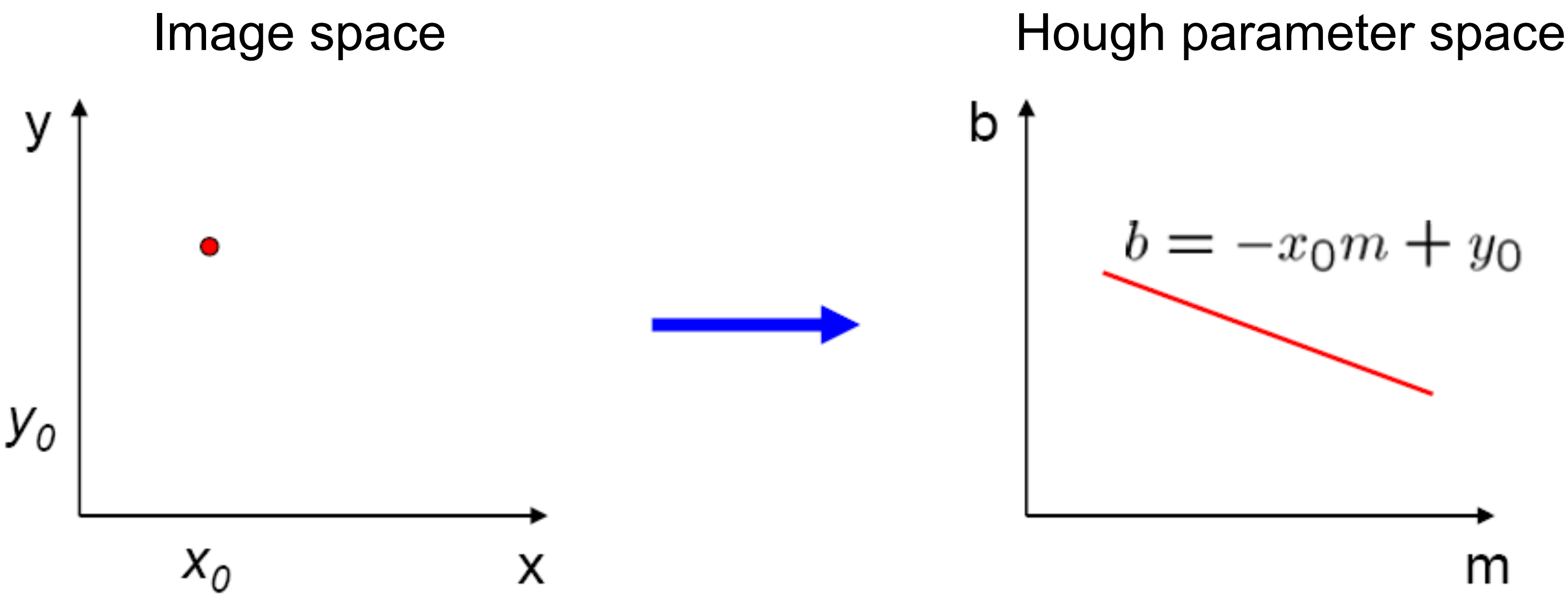
- What does a point (x_0, y_0) in the image space map to in the Hough space?



Source: S. Lazebnik

Parameter space representation

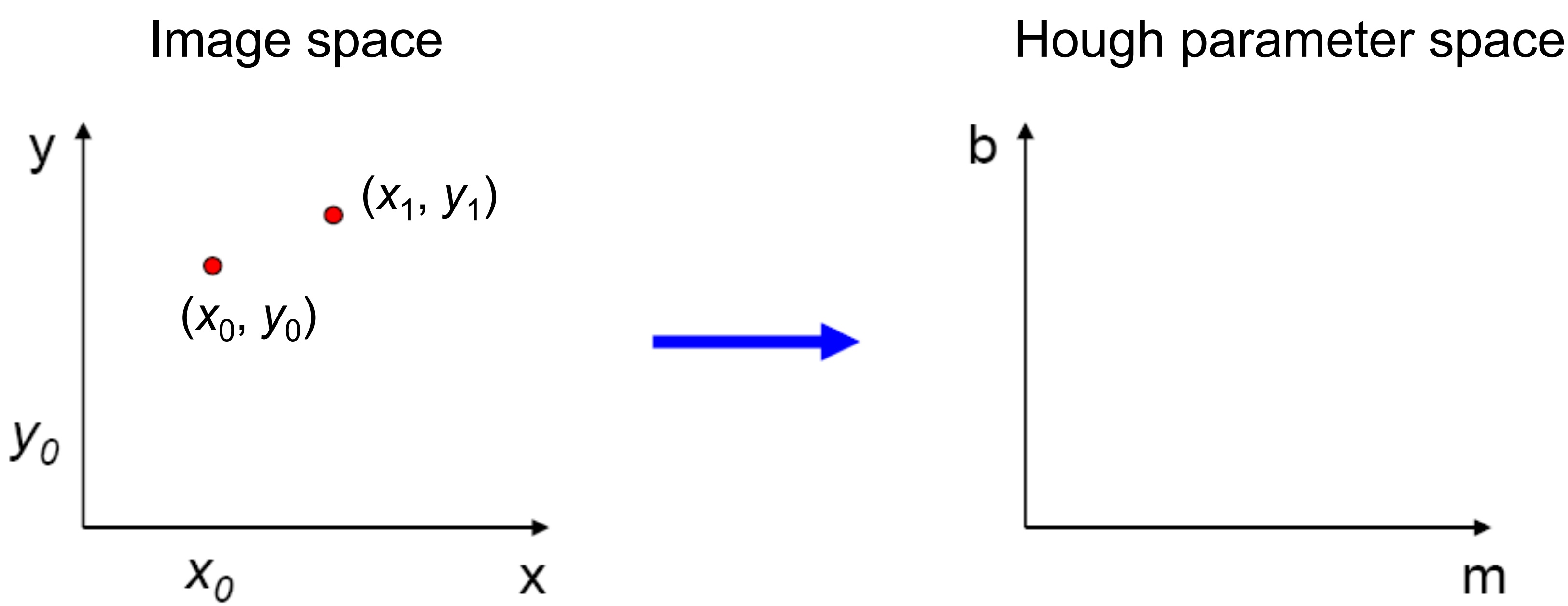
- What does a point (x_0, y_0) in the image space map to in the Hough space?
  - Answer: the solutions of $b = -x_0 m + y_0$
  - This is a line in Hough space



Source: S. Lazebnik

Parameter space representation

- Where is the line that contains both (x_0, y_0) and (x_1, y_1)?



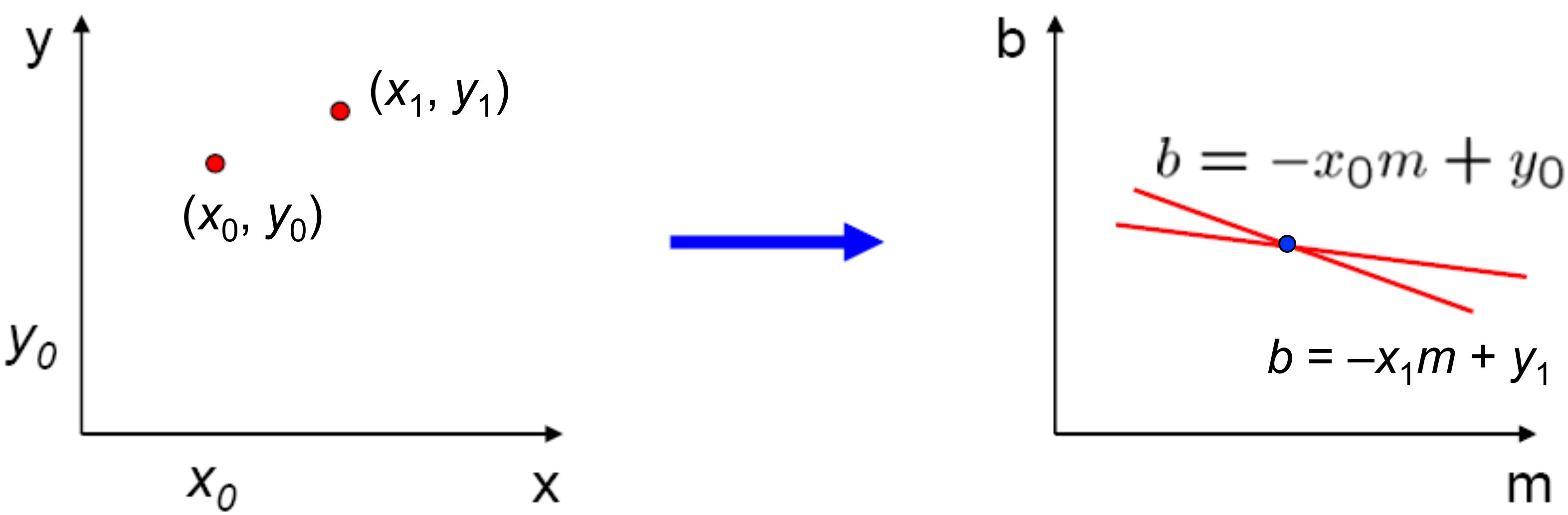
Source: S. Lazebnik

Parameter space representation

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

Image space

Hough parameter space



Source: S. Lazebnik

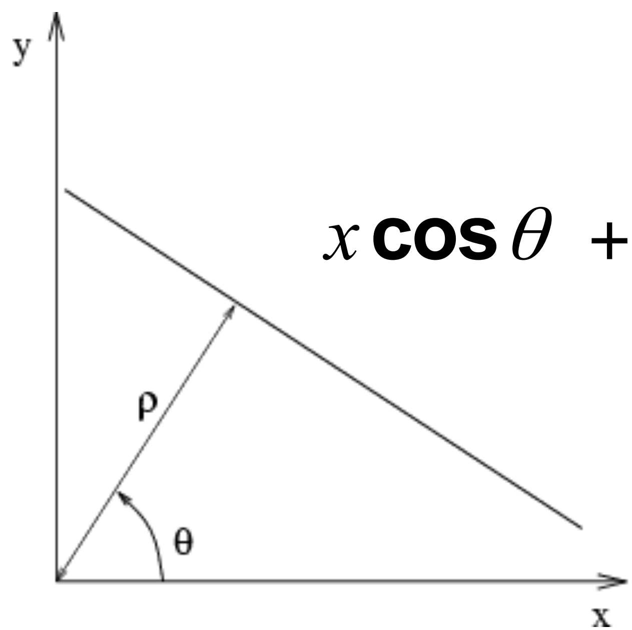
Parameter space representation

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

Source: S. Lazebnik

Parameter space representation

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



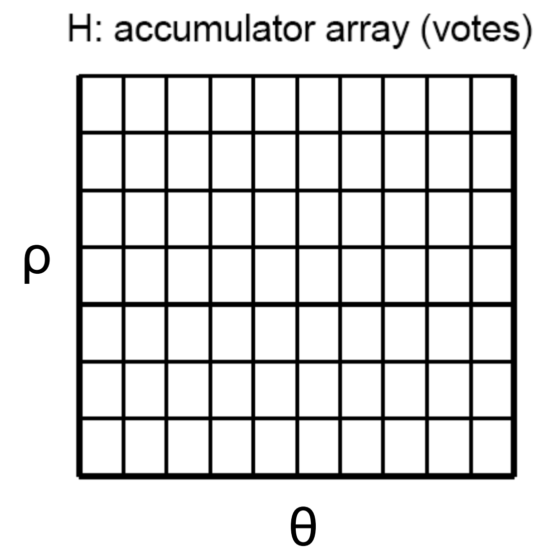
$$x\cos\theta + y\sin\theta = \rho$$

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

Source: S. Lazebnik

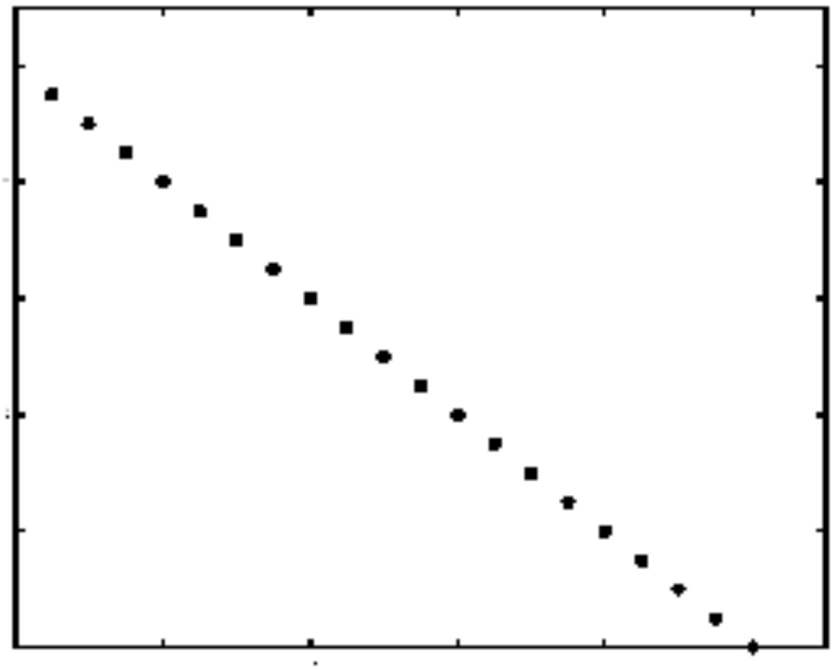
Algorithm outline

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

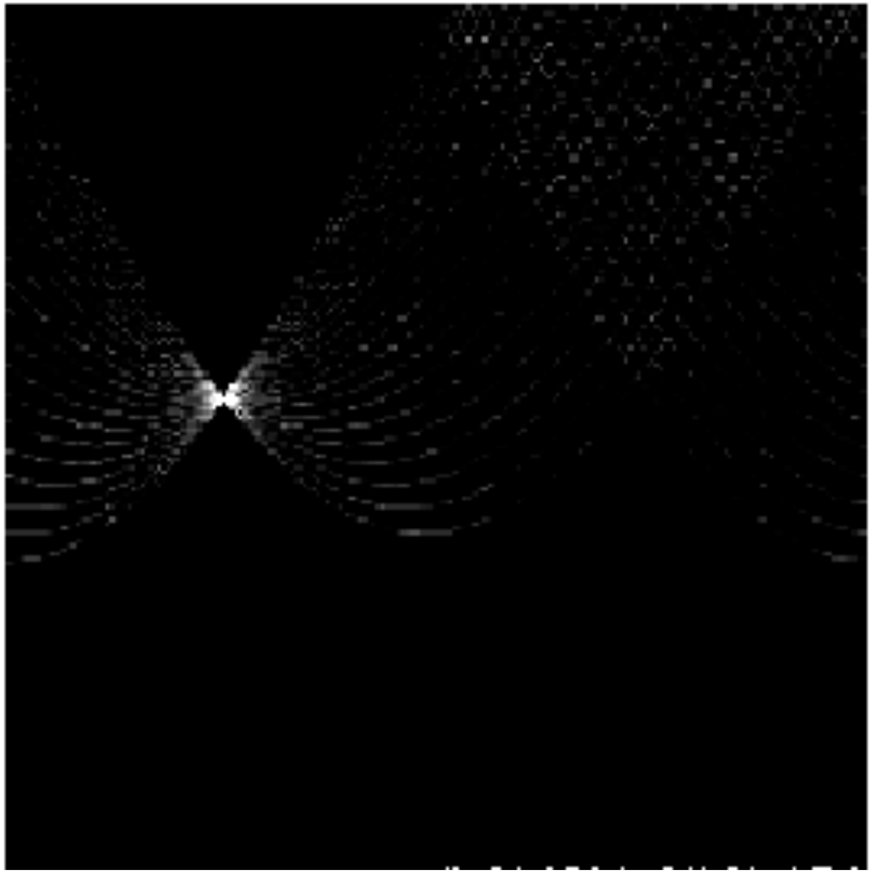


Source: S. Lazebnik

Basic illustration



features



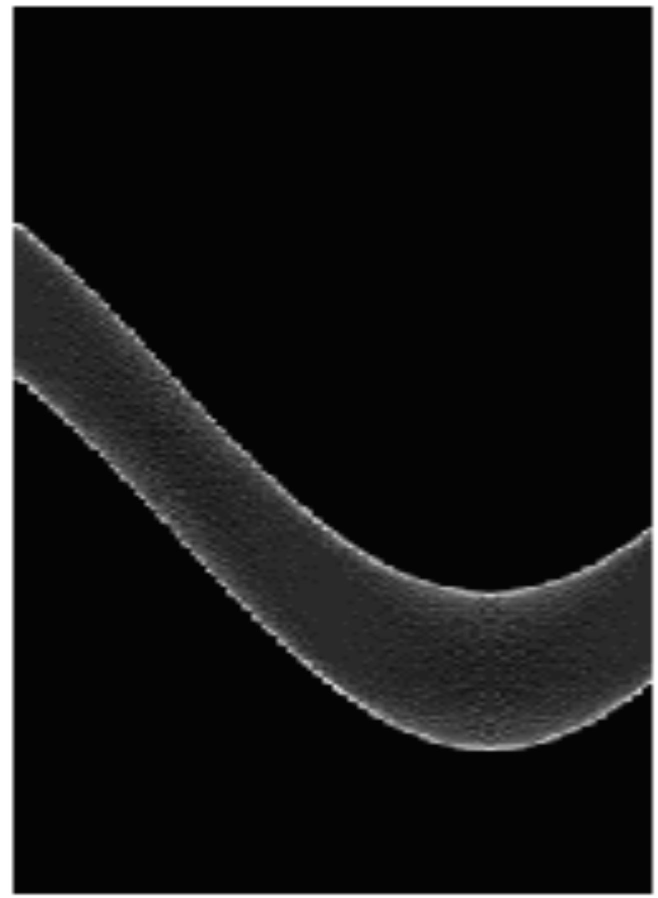
votes

[Hough transform demo]

Other shapes

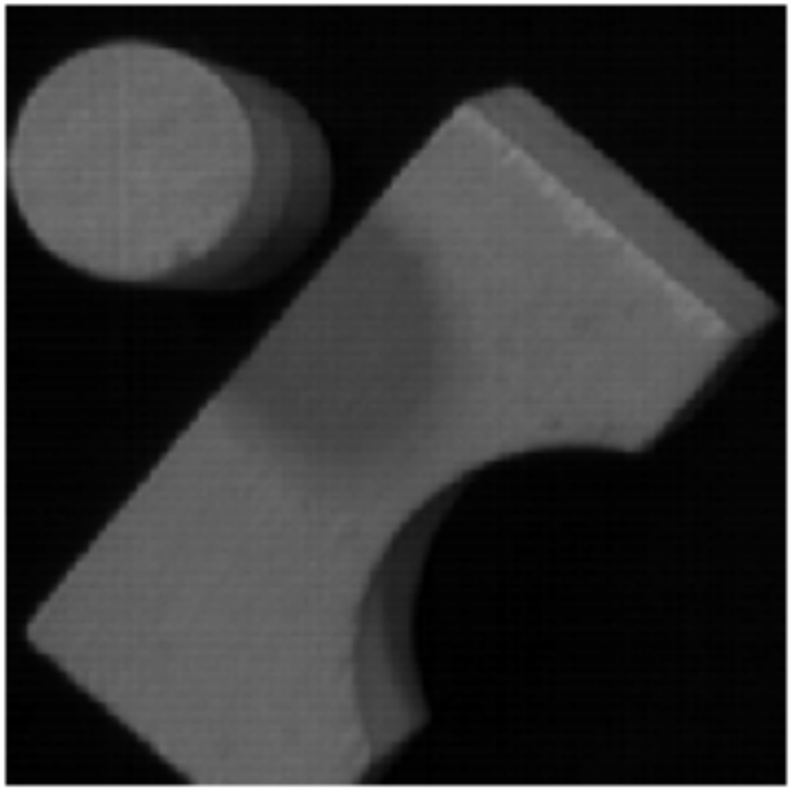
Square

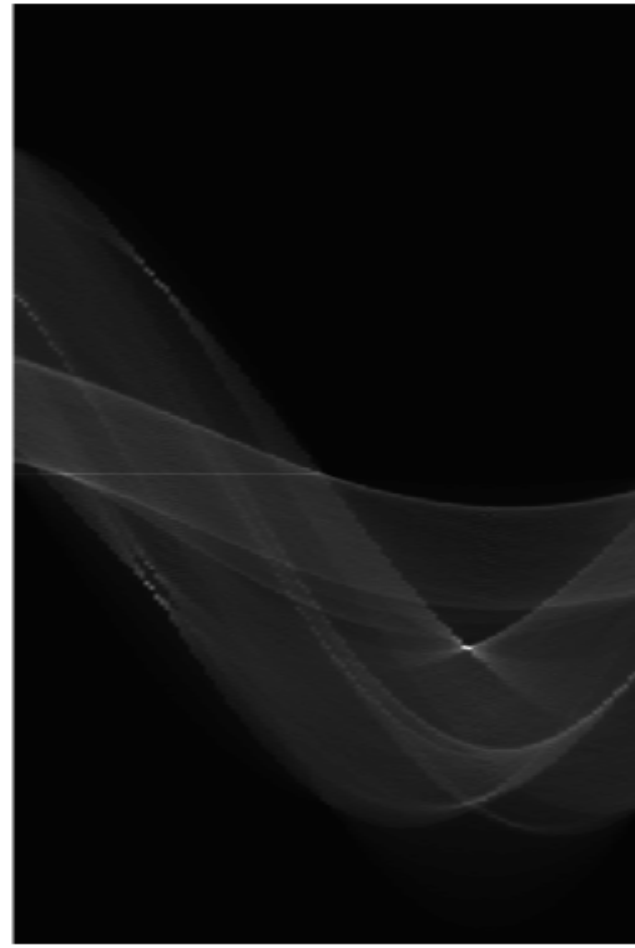
Circle



Source: S. Lazebnik

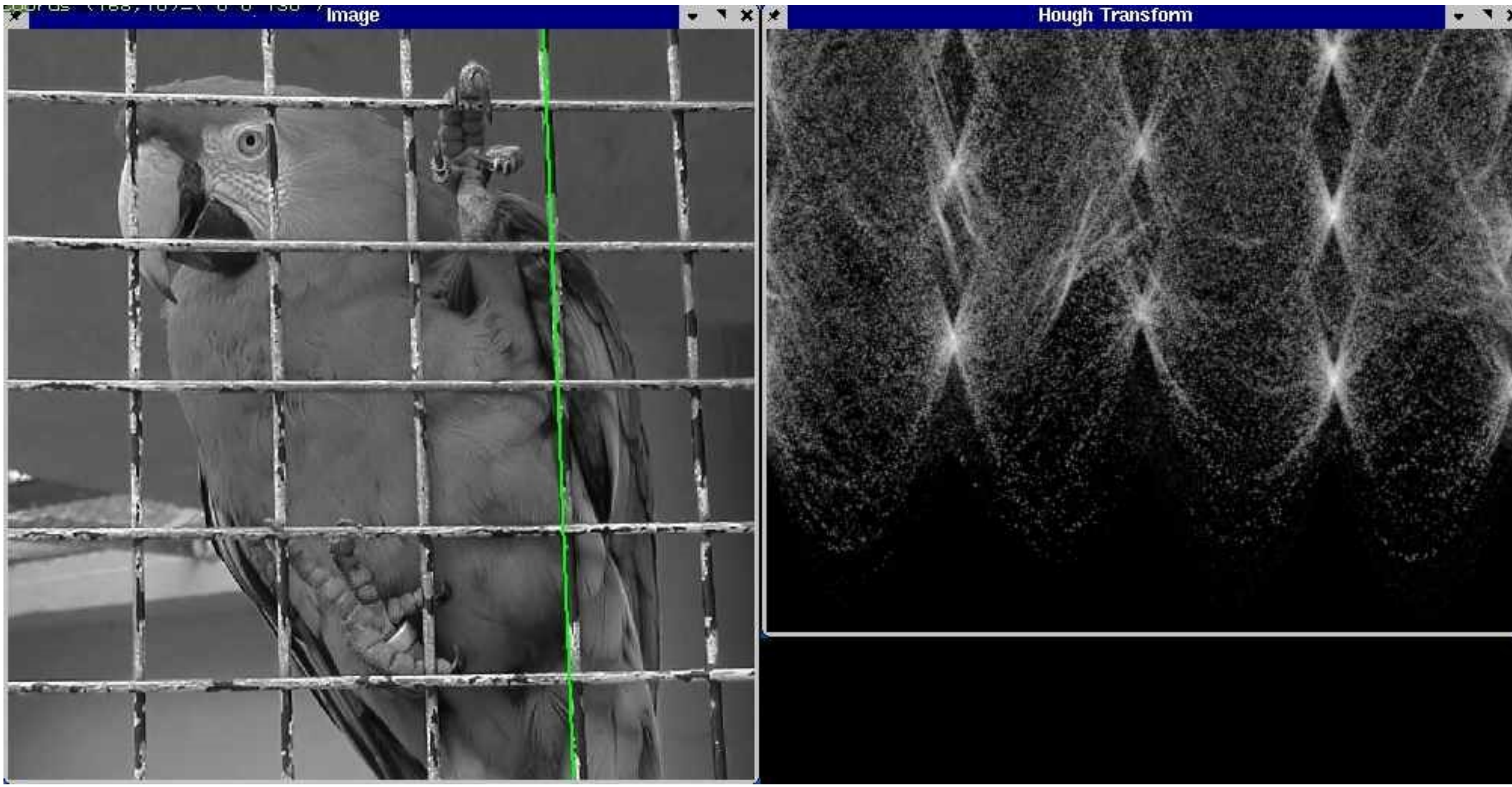
Several lines





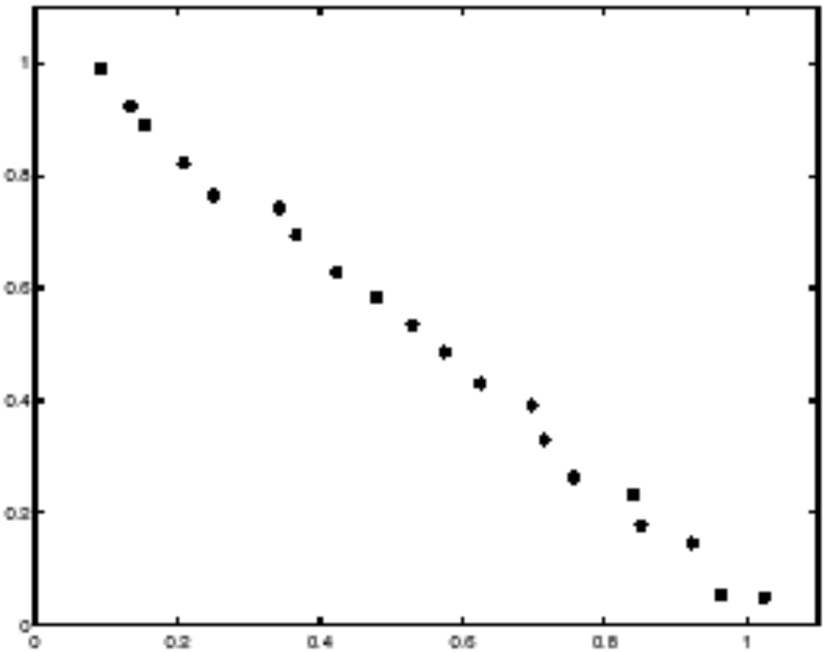
Source: S. Lazebnik

A more complicated image



http://ostatic.com/files/images/ss_hough.jpg

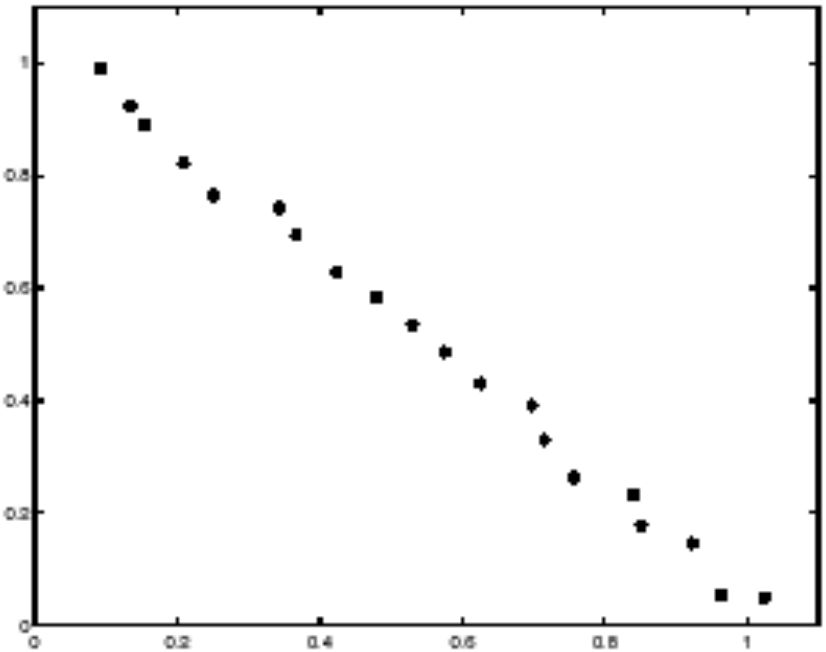
Effect of noise



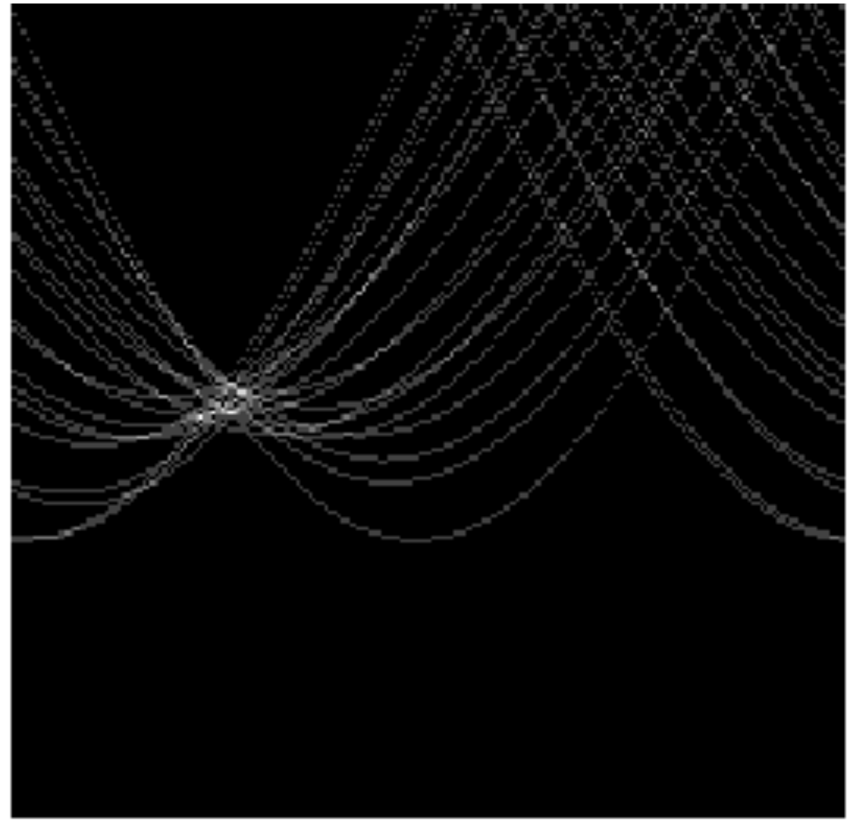
features

Source: S. Lazebnik

Effect of noise



features



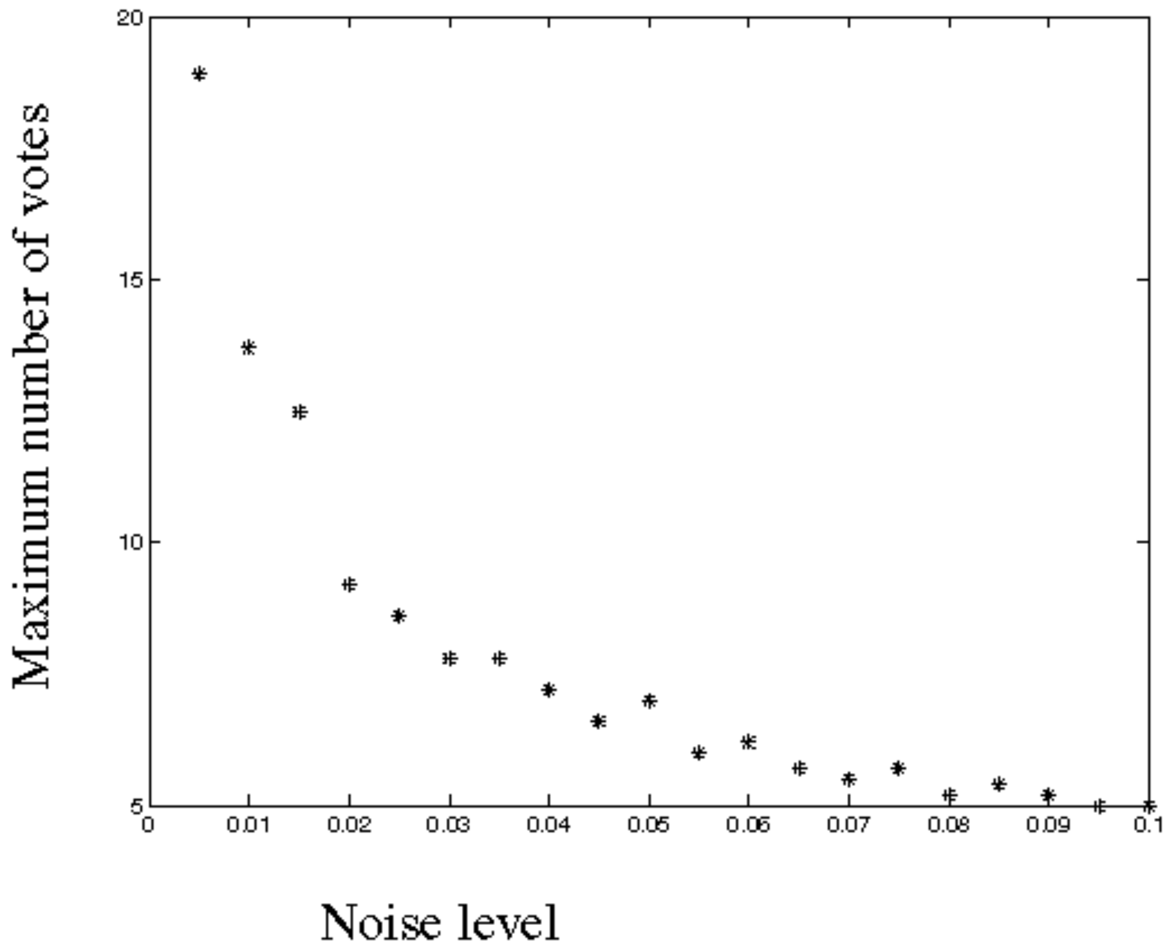
votes

Peak gets fuzzy and hard to locate

Source: S. Lazebnik

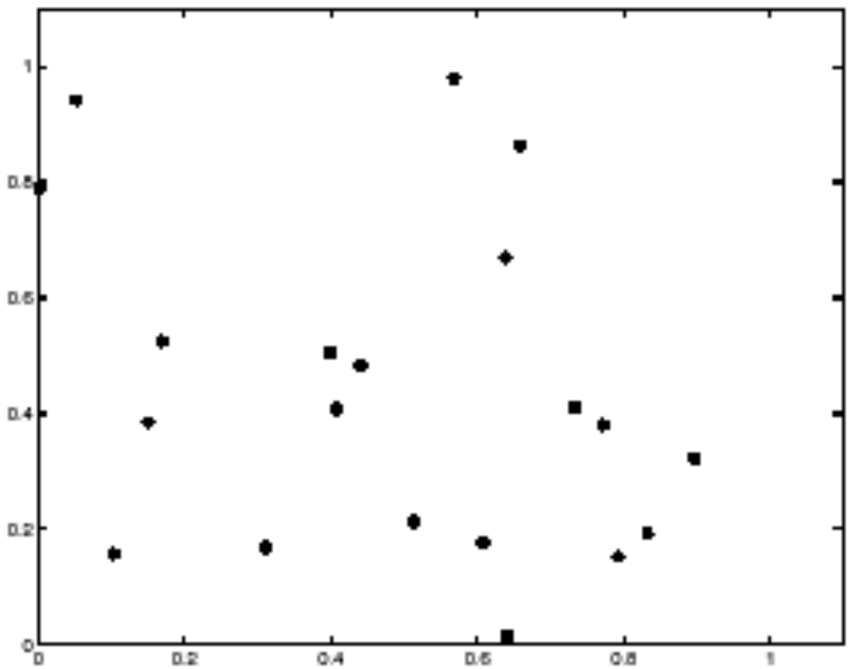
Effect of noise

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

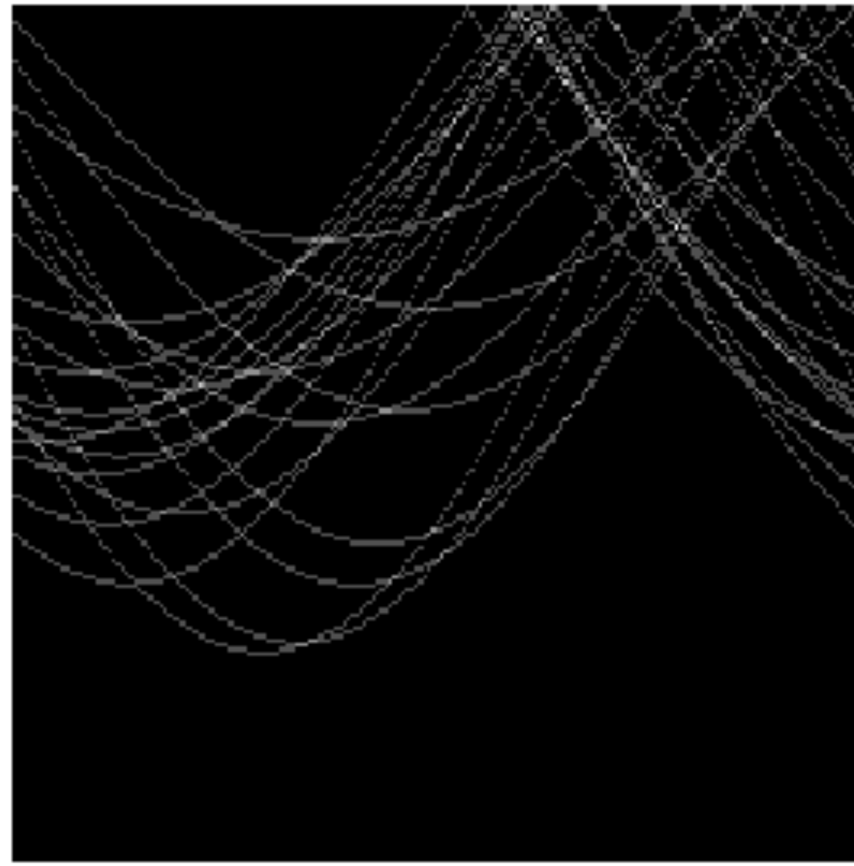


Source: S. Lazebnik

Random points



features

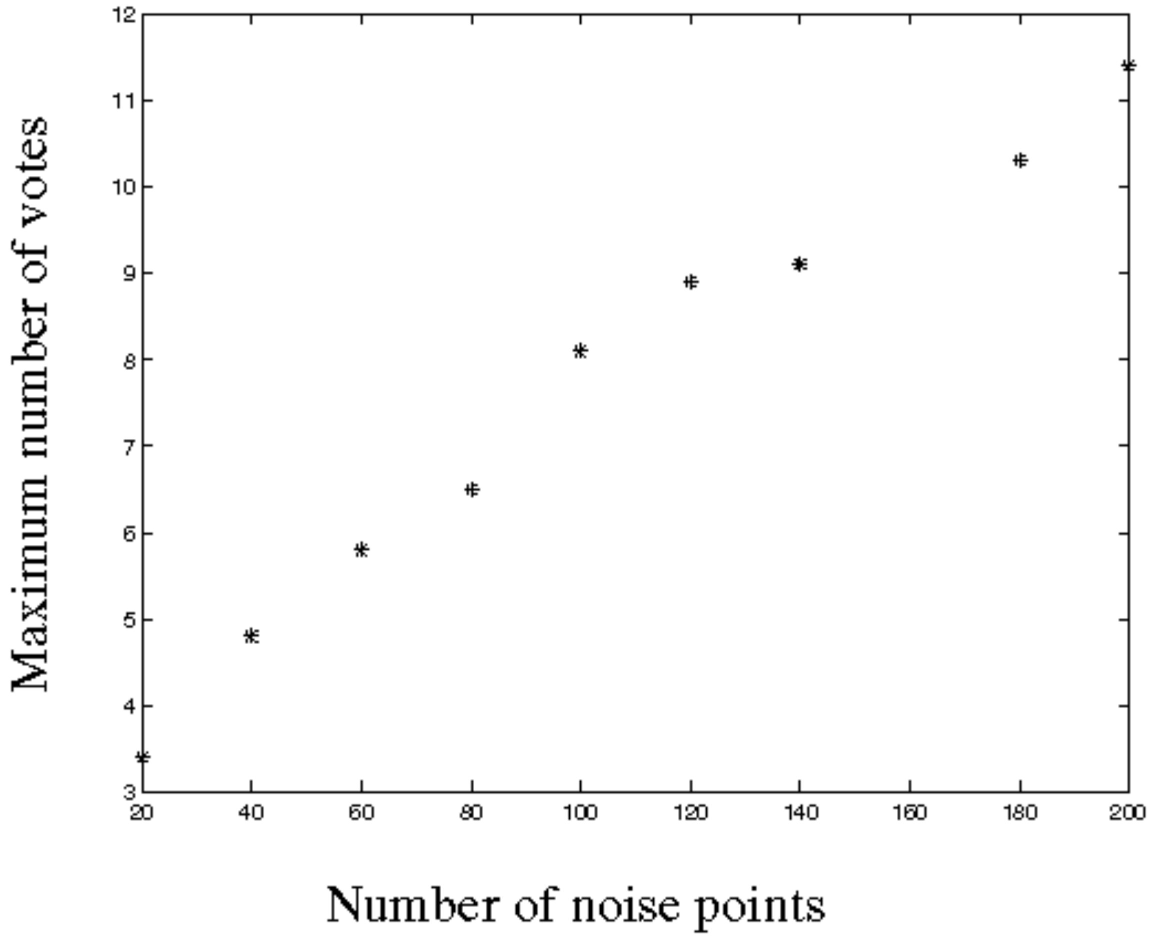


votes

Uniform noise can lead to spurious peaks in the array

Random points

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



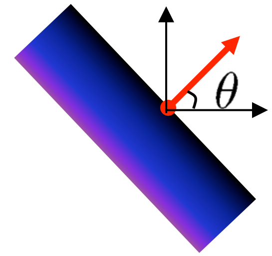
Dealing with noise

- Choose a good grid / discretization
  - **Too coarse:** large 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)
- Try to get rid of irrelevant features
  - E.g., take only edge points with significant gradient magnitude

Source: S. Lazebnik

Incorporating image gradients

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



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

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

- Modified Hough transform:

For each edge point (x,y)
    θ = gradient orientation at (x,y)
    ρ = x cos θ + y sin θ
    H(θ, ρ) = H(θ, ρ) + 1
end

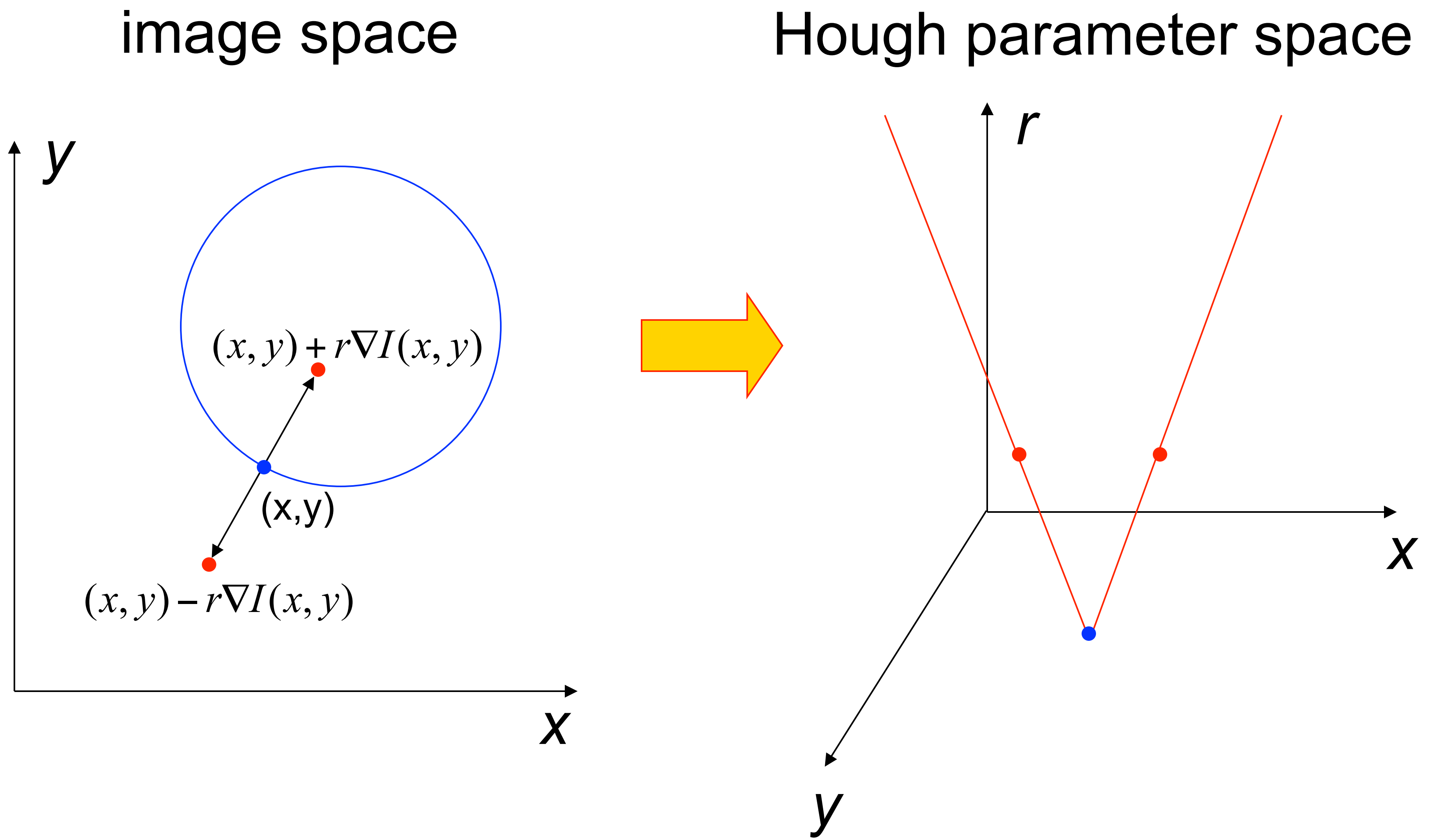
Source: S. Lazebnik

Hough transform for circles

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

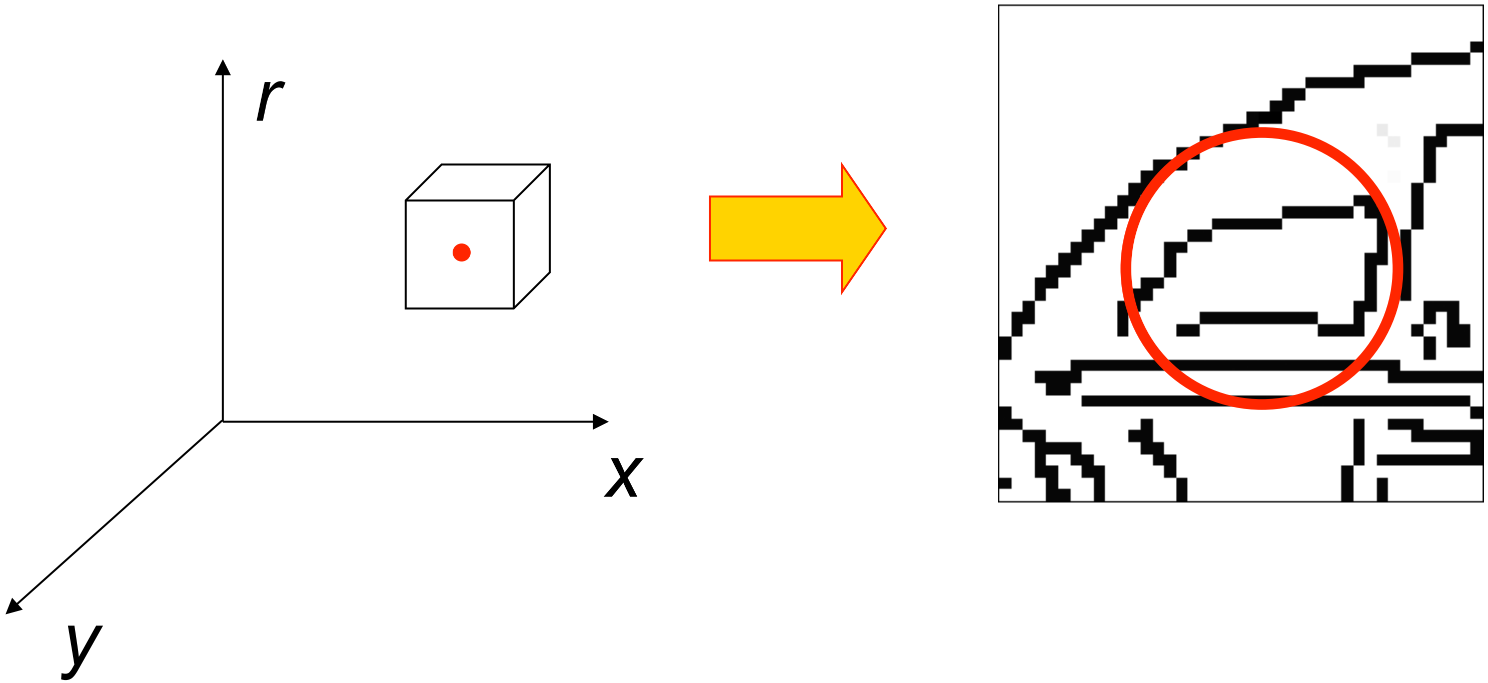
Source: S. Lazebnik

Hough transform for circles



Hough transform for circles

- Conceptually equivalent procedure: for each (x,y,r), draw the corresponding circle in the image and compute its "support"



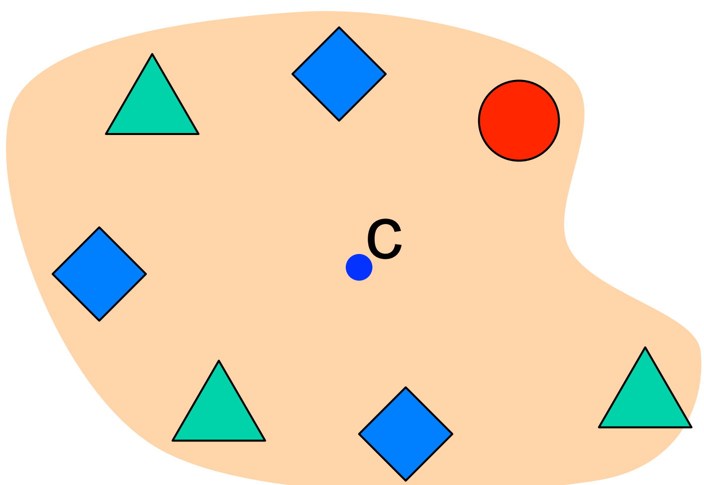
Is this more or less efficient than voting with features?

Source: S. Lazebnik

Generalized Hough transform

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

Template

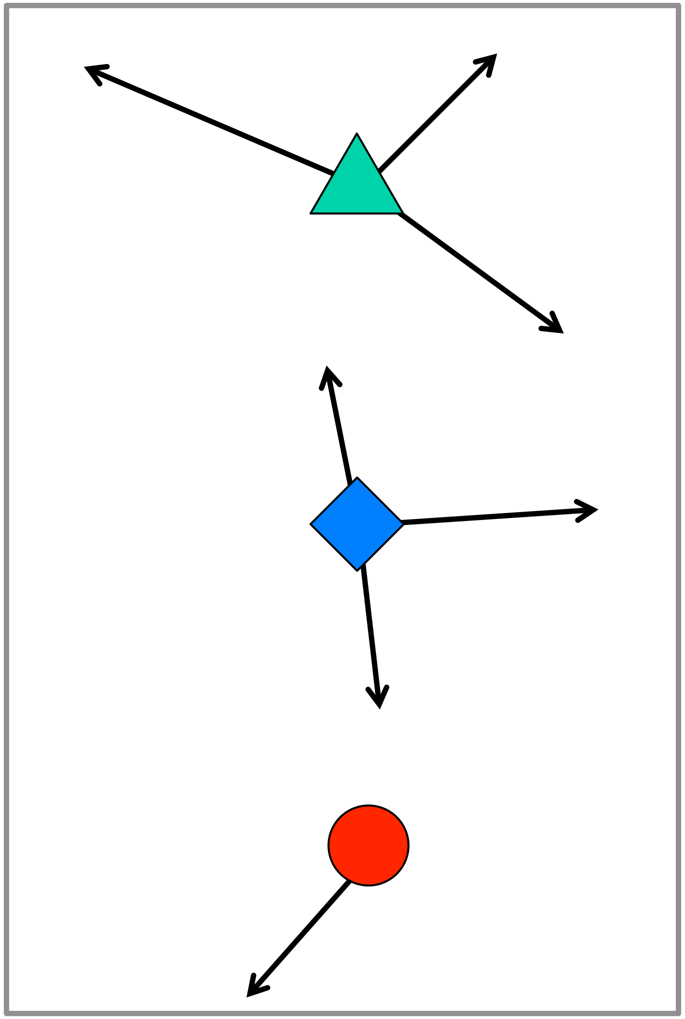


Source: S. Lazebnik

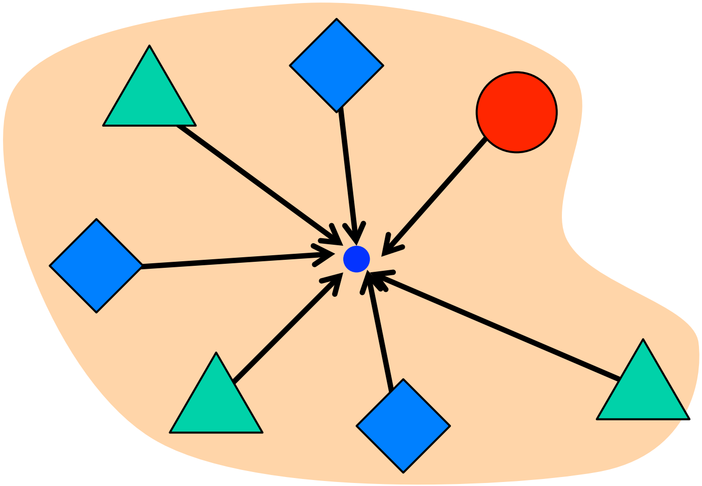
Generalized Hough transform

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

Model



Template

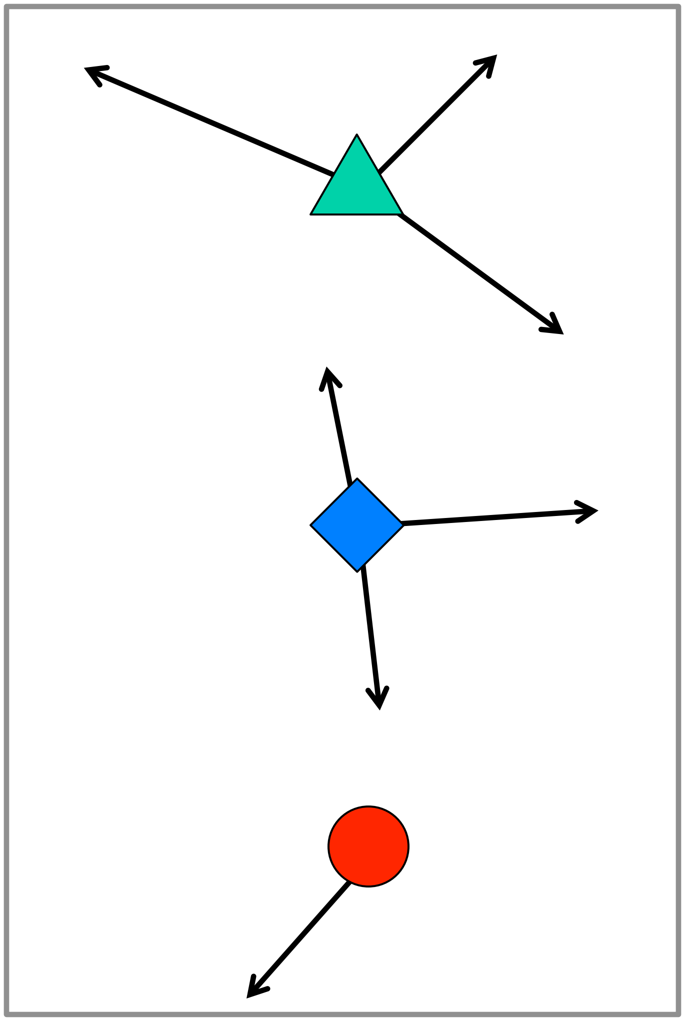


Source: S. Lazebnik

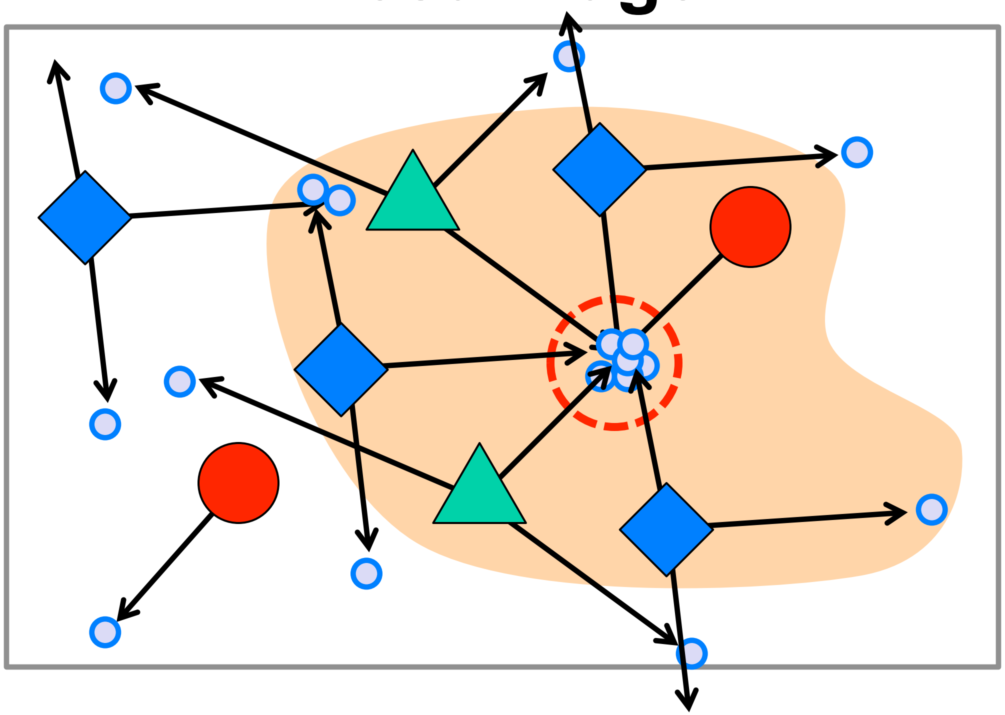
Generalized Hough transform

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

Model



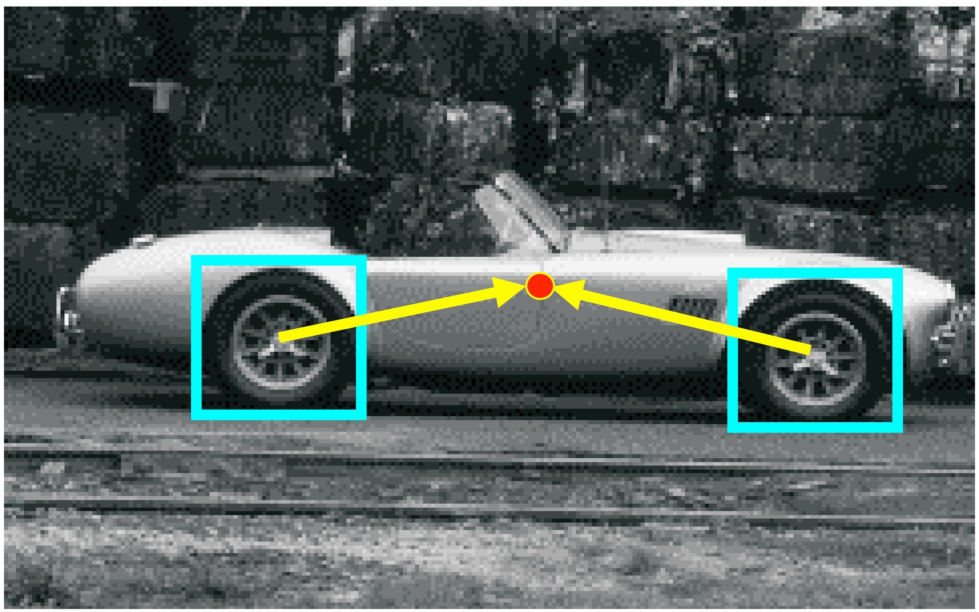
Test image



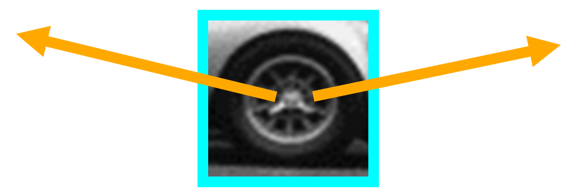
Source: S. Lazebnik

Application in recognition

- Index displacements by “visual codeword”



training image



visual codeword with displacement vectors

B. Leibe, A. Leonardis, and B. Schiele,
Combined Object Categorization and Segmentation with an Implicit Shape Model,
ECCV Workshop on Statistical Learning in Computer Vision 2004

Source: S. Lazebnik

Application in recognition

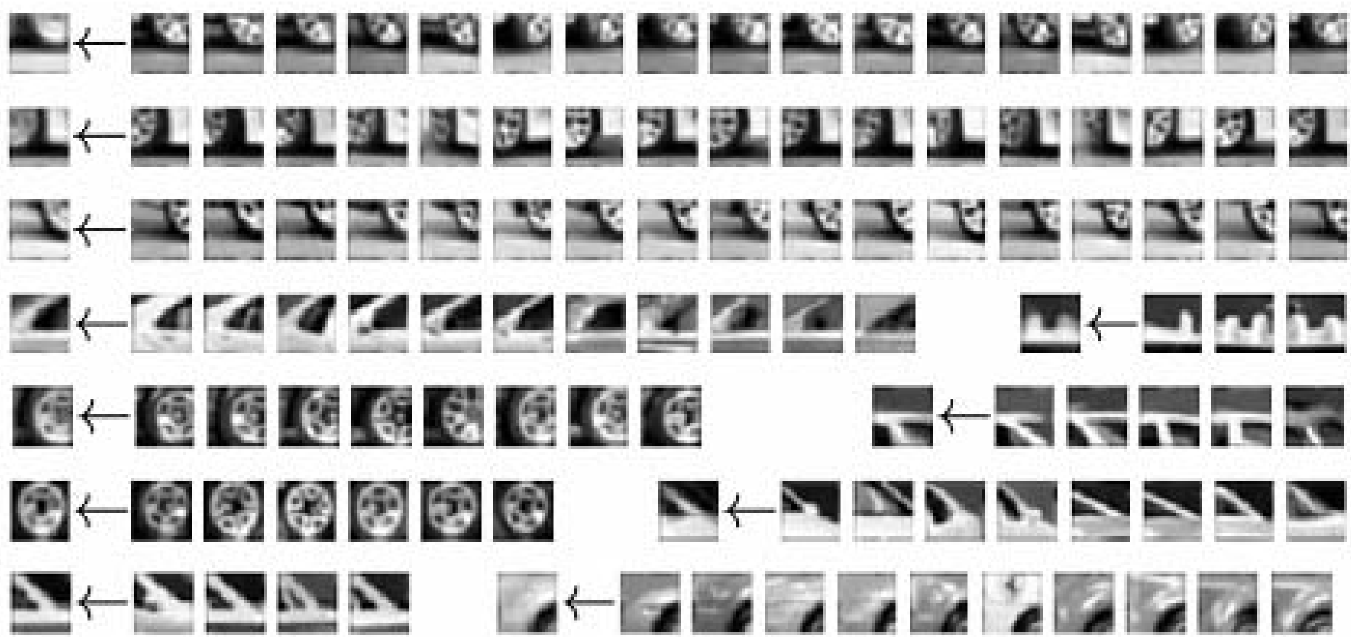
- Index displacements by “visual codeword”

test image

B. Leibe, A. Leonardis, and B. Schiele,
[Combined Object Categorization and Segmentation with an Implicit Shape Model](),
ECCV Workshop on Statistical Learning in Computer Vision 2004

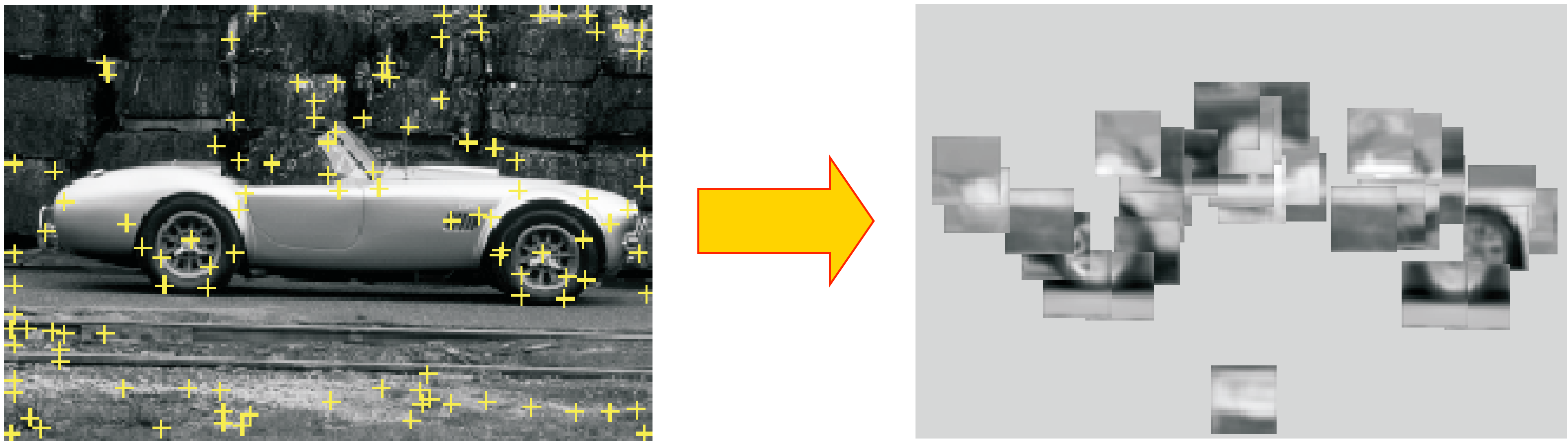
Implicit shape models: Training

1. Build *codebook* of patches around extracted interest points using clustering (more on this later in the course)



Implicit shape models: Training

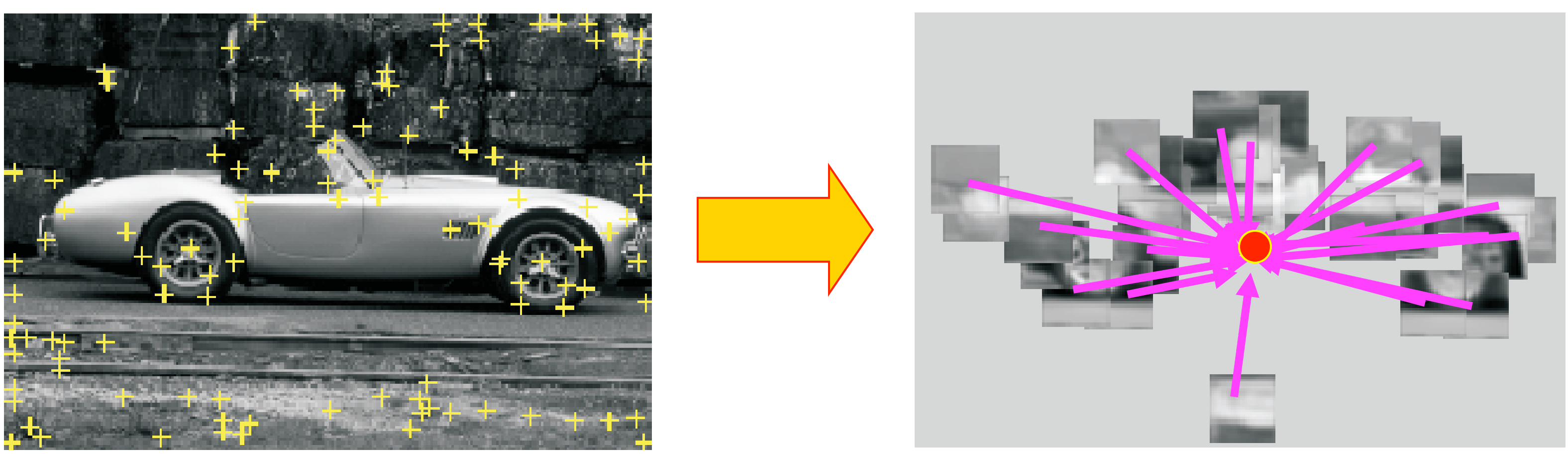
1. Build *codebook* of patches around extracted interest points using clustering
2. Map the patch around each interest point to closest codebook entry



Source: S. Lazebnik

Implicit shape models: Training

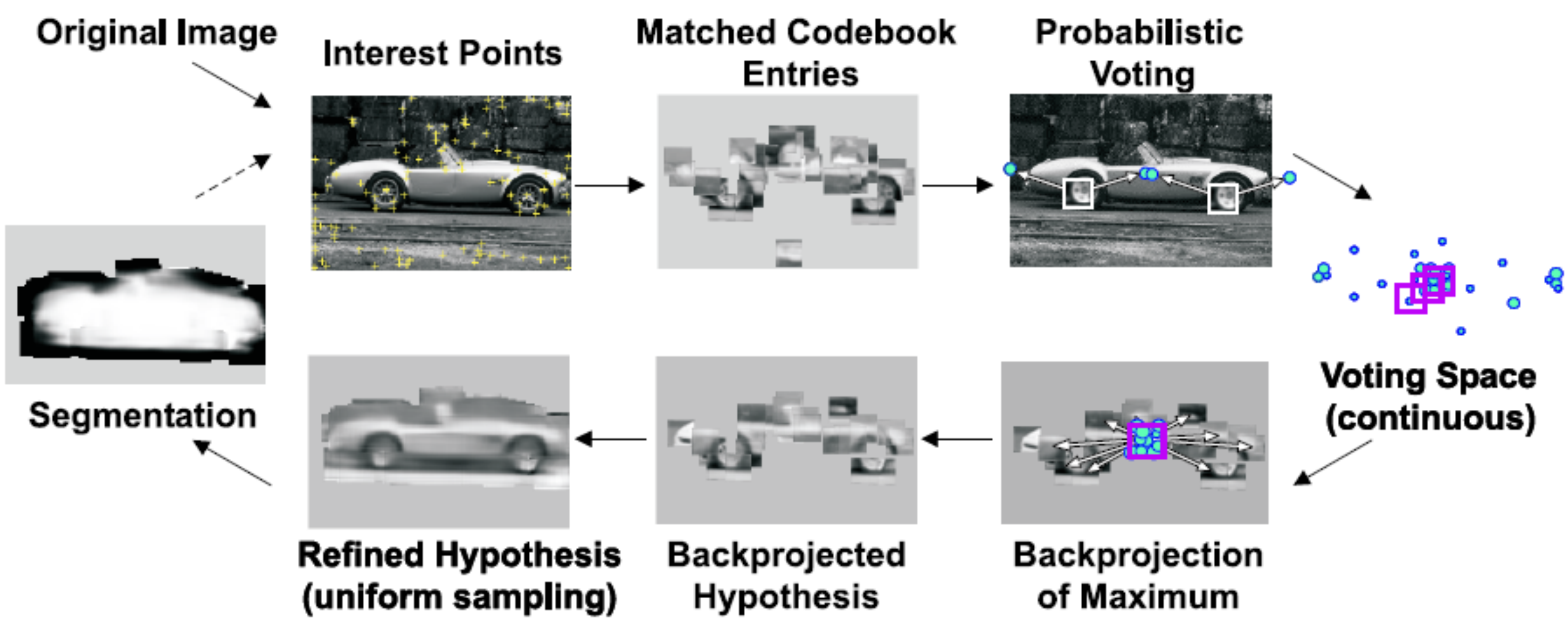
1. Build *codebook* of patches around extracted interest points using clustering
2. Map the patch around each interest point to closest codebook entry
3. For each codebook entry, store all positions it was found, relative to object center



Source: S. Lazebnik

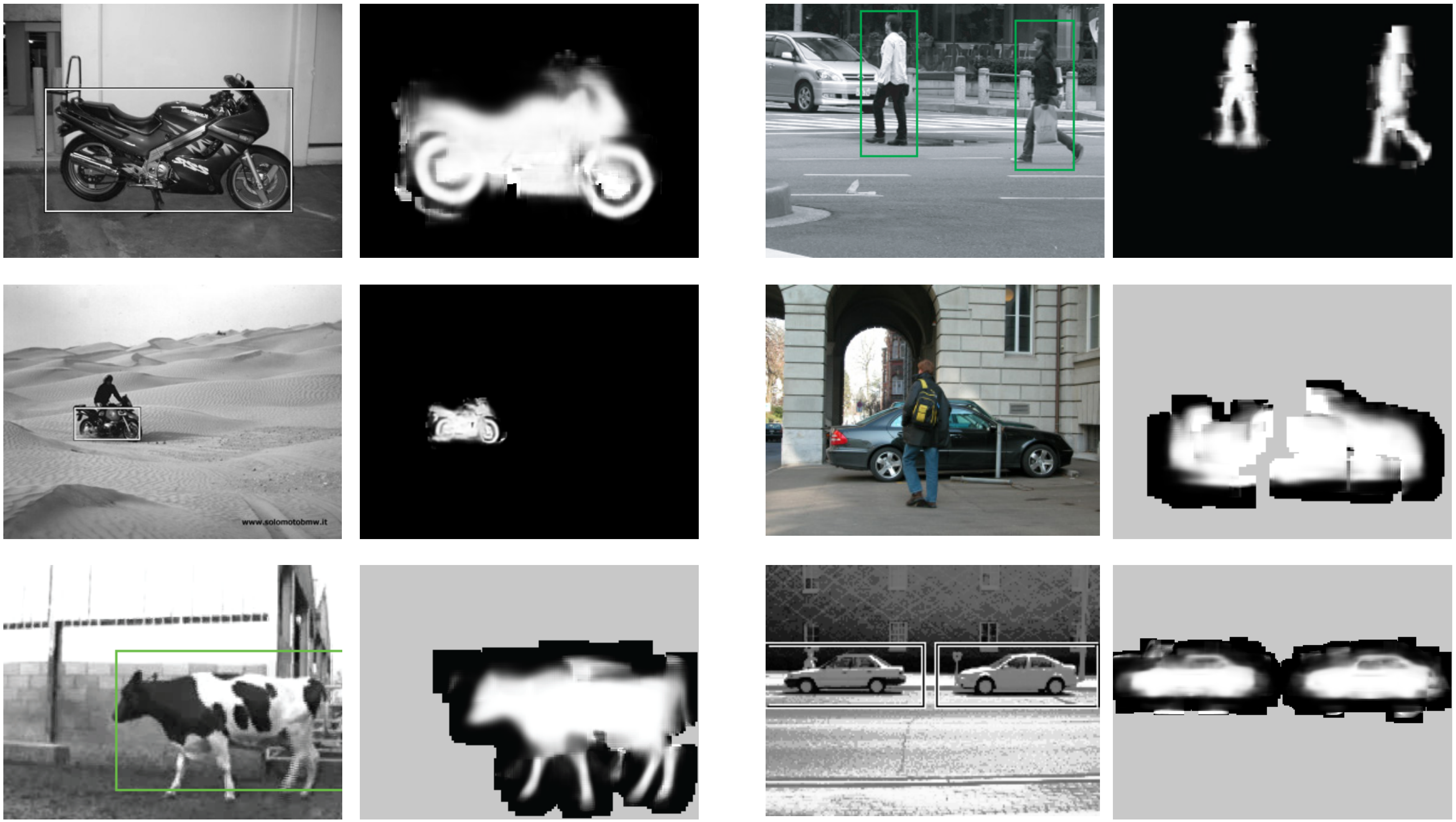
Implicit shape models: Testing

1. Given test image, extract patches, match to codebook entry
2. Cast votes for possible positions of object center
3. Search for maxima in voting space
4. Extract weighted segmentation mask based on stored masks for the codebook occurrences



Source: S. Lazebnik

Additional examples



B. Leibe, A. Leonardis, and B. Schiele, [Robust Object Detection with Interleaved Categorization and Segmentation](), IJCV 77 (1-3), pp. 259-289, 2008.

Source: S. Lazebnik

Implicit shape models: Details

- Supervised training
  - Need reference location and segmentation mask for each training car
- Voting space is continuous, not discrete
  - Clustering algorithm needed to find maxima
- How about dealing with scale changes?
  - Option 1: search a range of scales, as in Hough transform for circles
  - Option 2: use interest points with characteristic scale
- Verification stage is very important
  - Once we have a location hypothesis, we can overlay a more detailed template over the image and compare pixel-by-pixel, transfer segmentation masks, etc.

Source: S. Lazebnik

Review: Hough transform

- Hough transform for lines
- Hough transform for circles
- Hough transform pros and cons

Source: S. Lazebnik

Hough transform: Pros and cons

- 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