Lecture 5

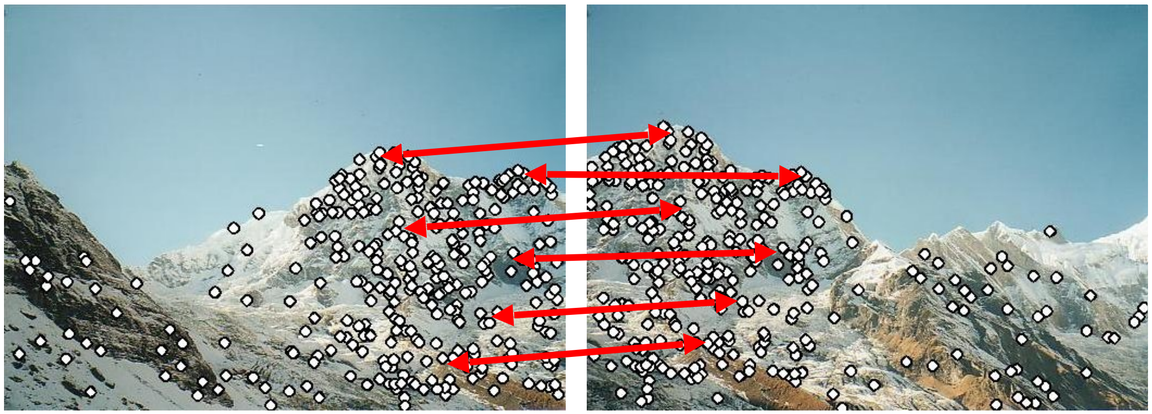
Wrap-up of SIFT
Then fitting, RANSAC, Hough transforms

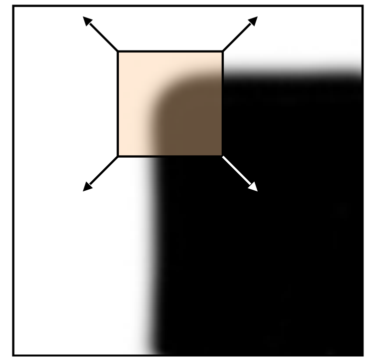
COS 429: Computer Vision

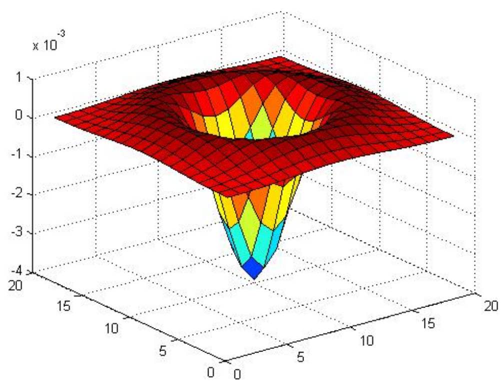
PRINCETON UNIVERSITY

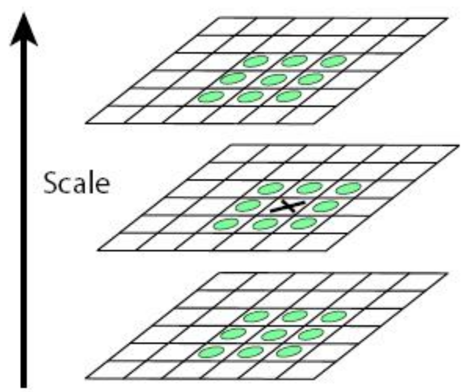
Slides adapted from: Szymon Rusinkiewicz, Jia Deng, Svetlana Lazebnik

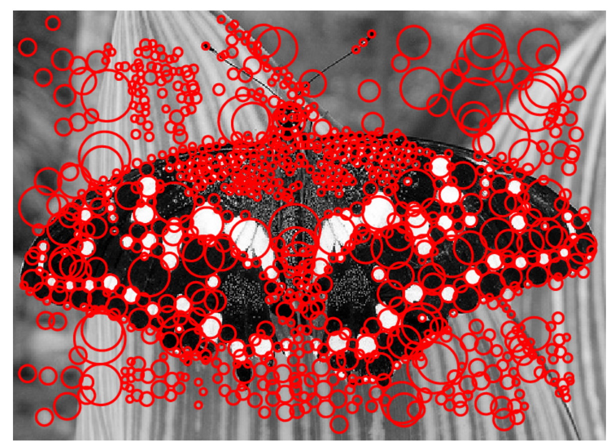
Last time: interest point detection





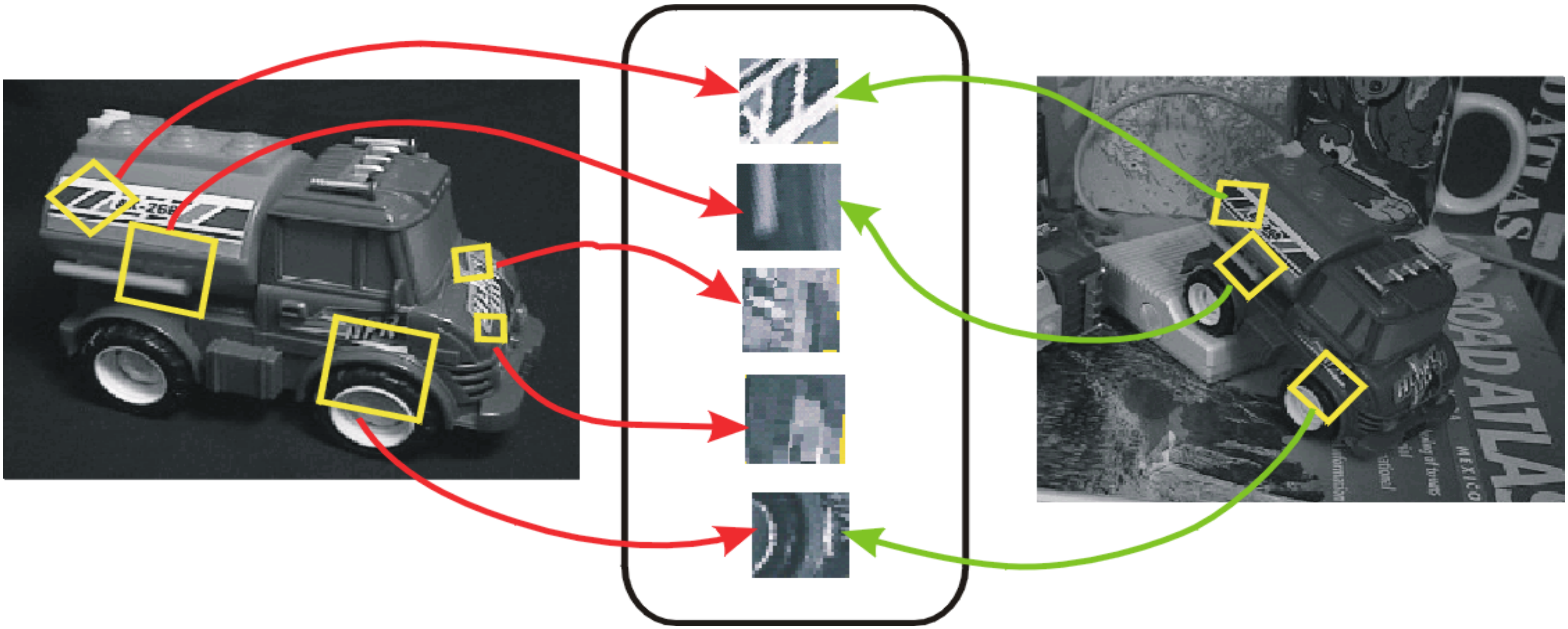






SIFT descriptors

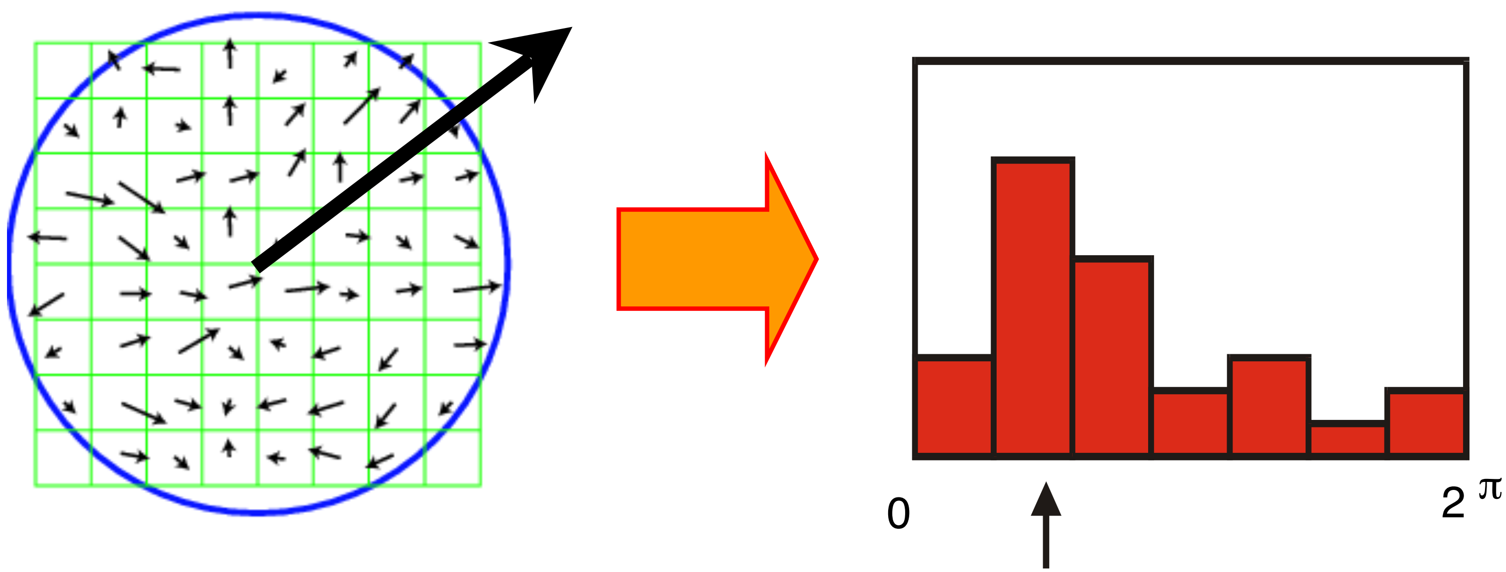
From feature detection to feature description



Slide: S. Lazebnik

Eliminating rotation ambiguity

- To assign a unique orientation to circular image windows:
  - Create histogram of local gradient directions in the patch
  - Assign canonical orientation at peak of smoothed histogram



Slide: S. Lazebnik

SIFT detected features

- Detected features with characteristic scales and orientations:

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

Slide: S. Lazebnik

SIFT Descriptor

- Divide 16×16 window into 4×4 grid of cells
- Compute an orientation histogram for each cell
  - 16 cells * 8 orientations = 128-dimensional descriptor

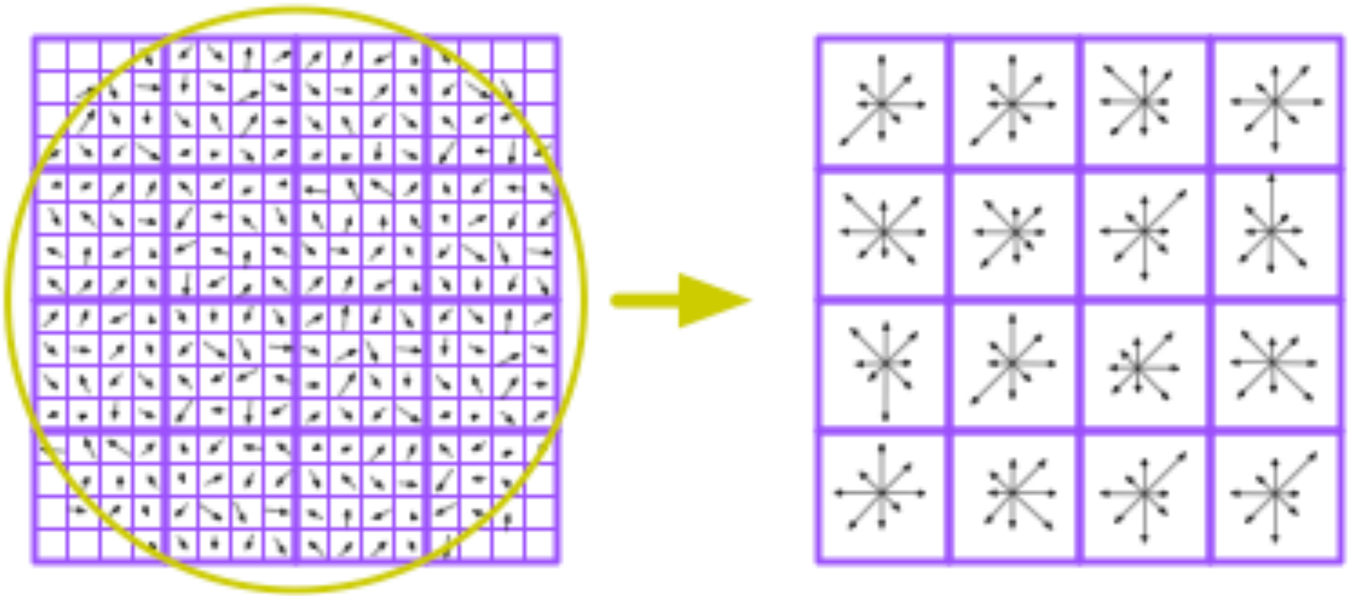


Image gradients

Keypoint descriptor

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

Properties of Feature Descriptors

- Easily compared (compact, fixed-dimensional)
- Easily computed
- Invariant
  - Translation
  - Rotation
  - Scale
  - Change in image brightness
  - Change in perspective?

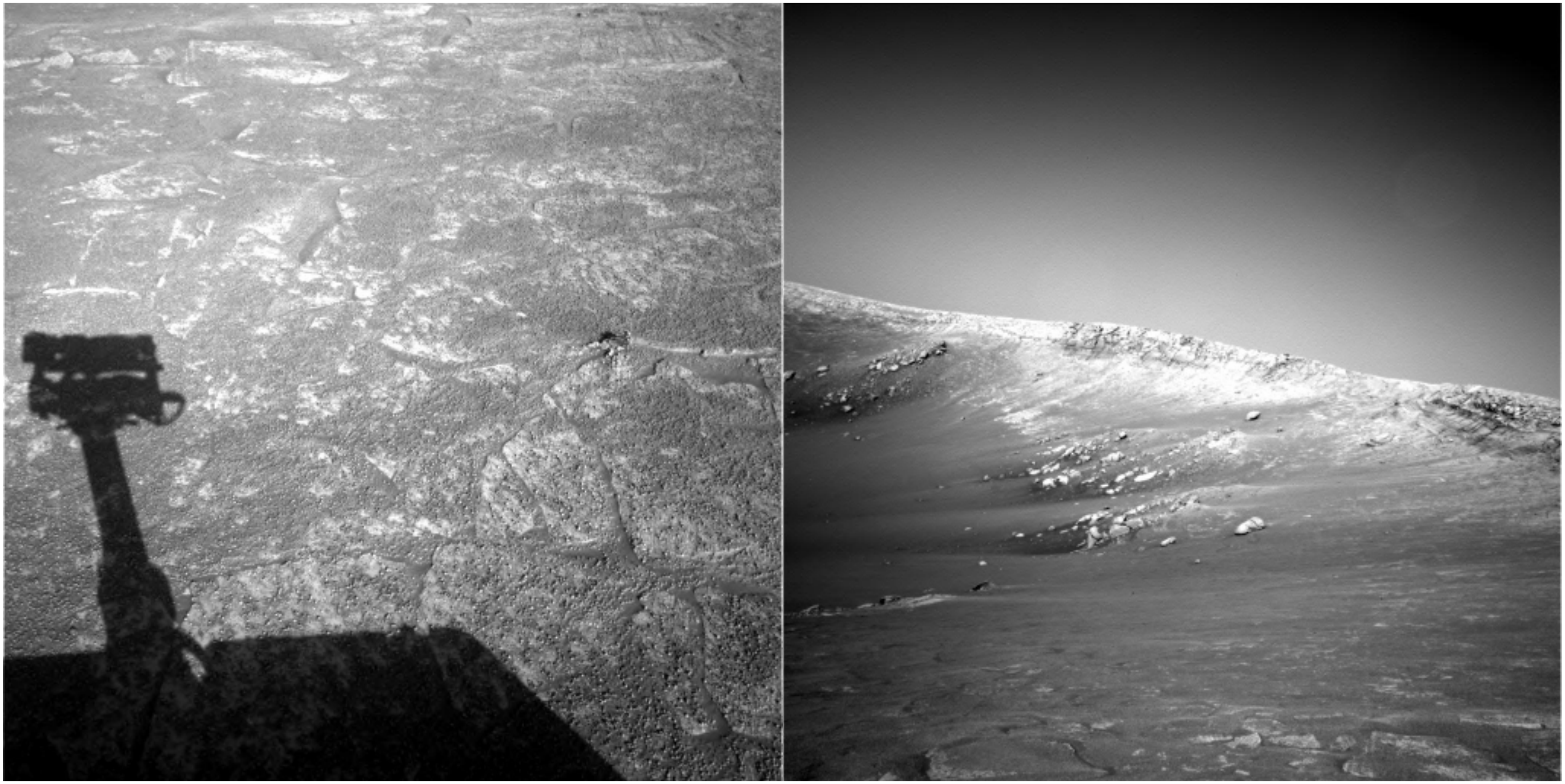
Properties of SIFT

Extraordinarily robust detection and description technique

- Handles changes in viewpoint (~ 60 degree out-of-plane rotation)
- Handles significant changes in illumination (sometimes even day vs night)
- Fast and efficient—can run in real time
- Lots of code available

Source: N. Snavely

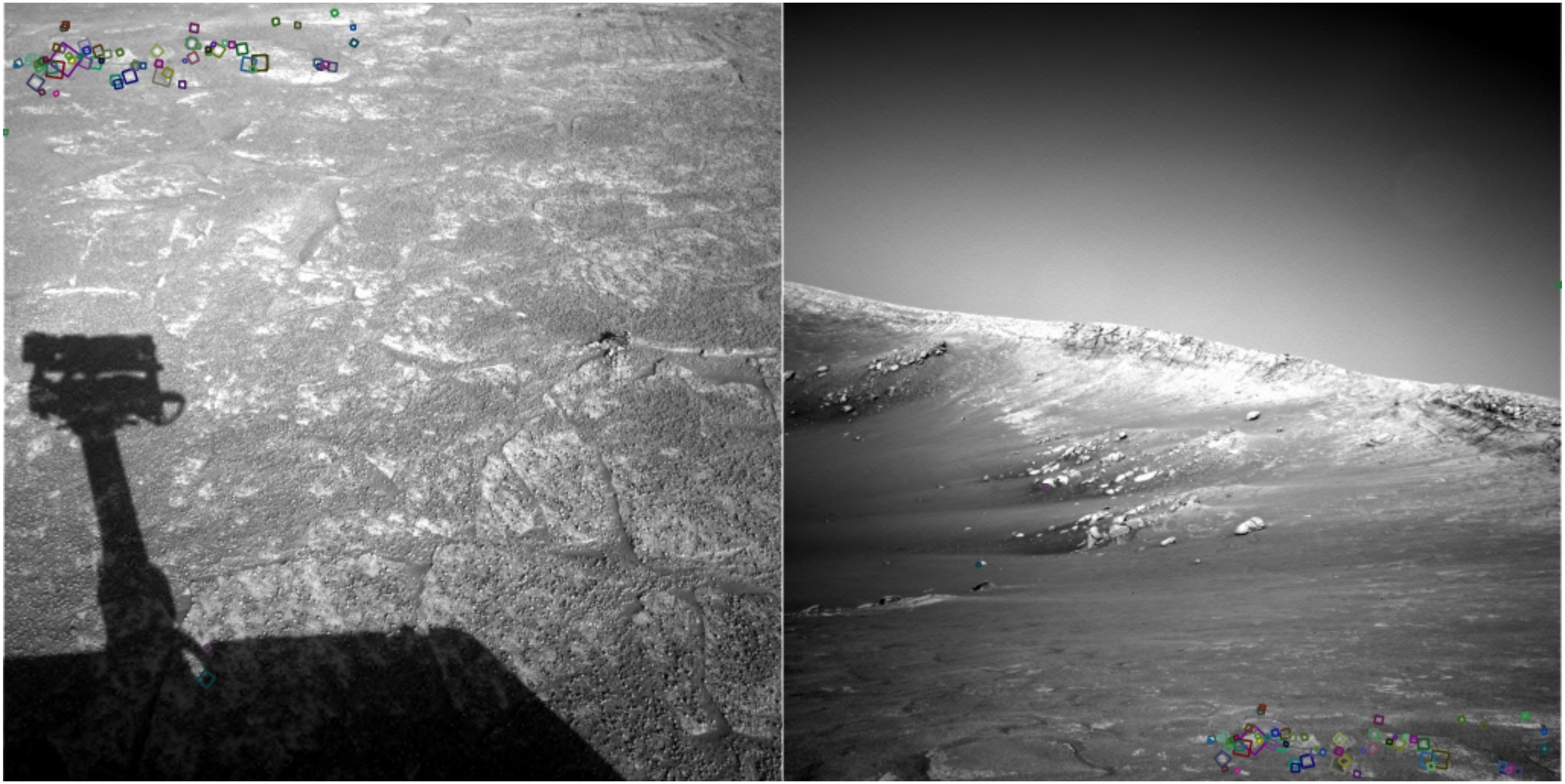
A hard feature matching problem



NASA Mars Rover images

Slide credit: S. Lazebnik

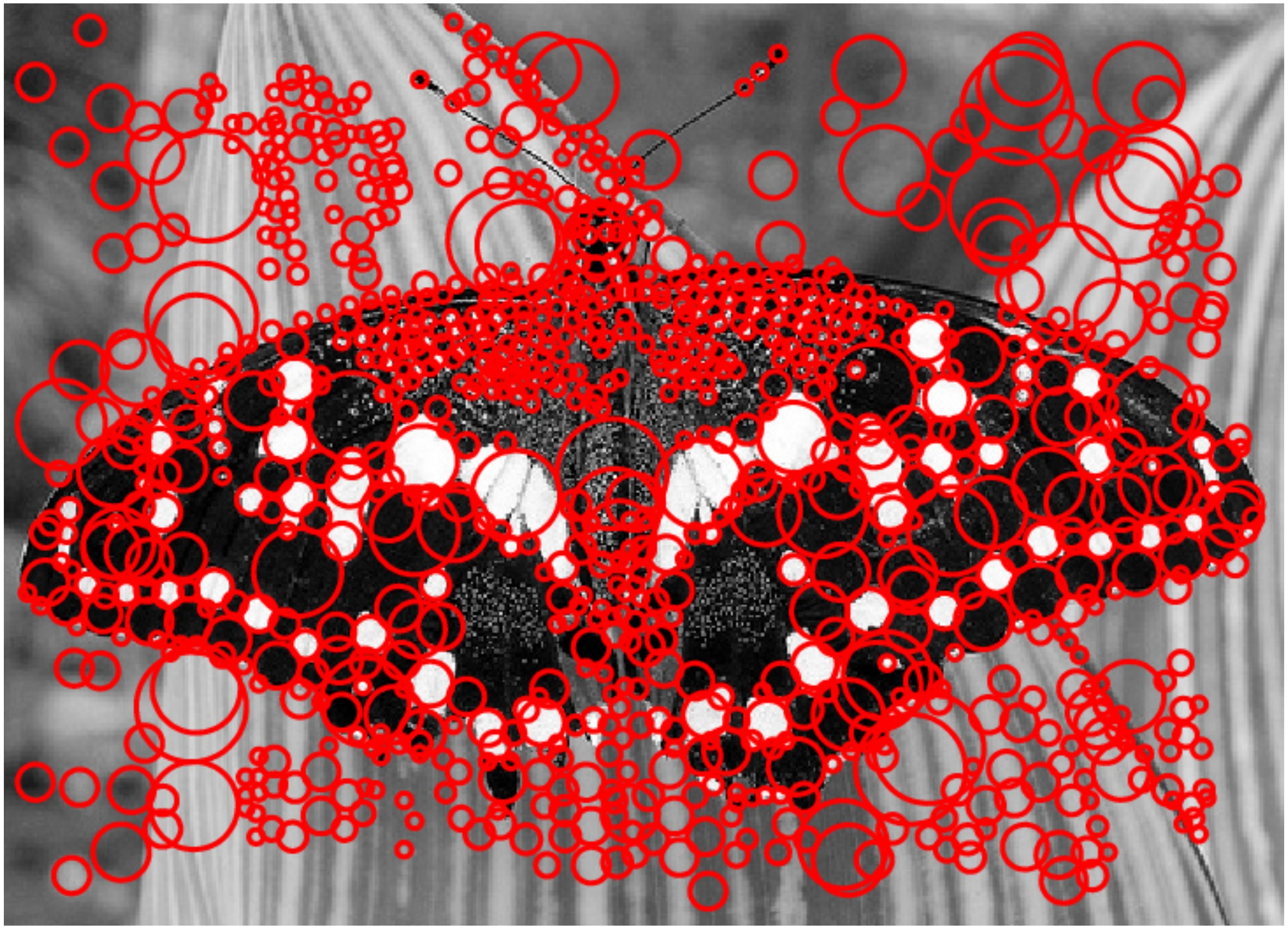
Answer below (look for tiny colored squares…)



NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

Slide credit: S. Lazebnik

Going deeper

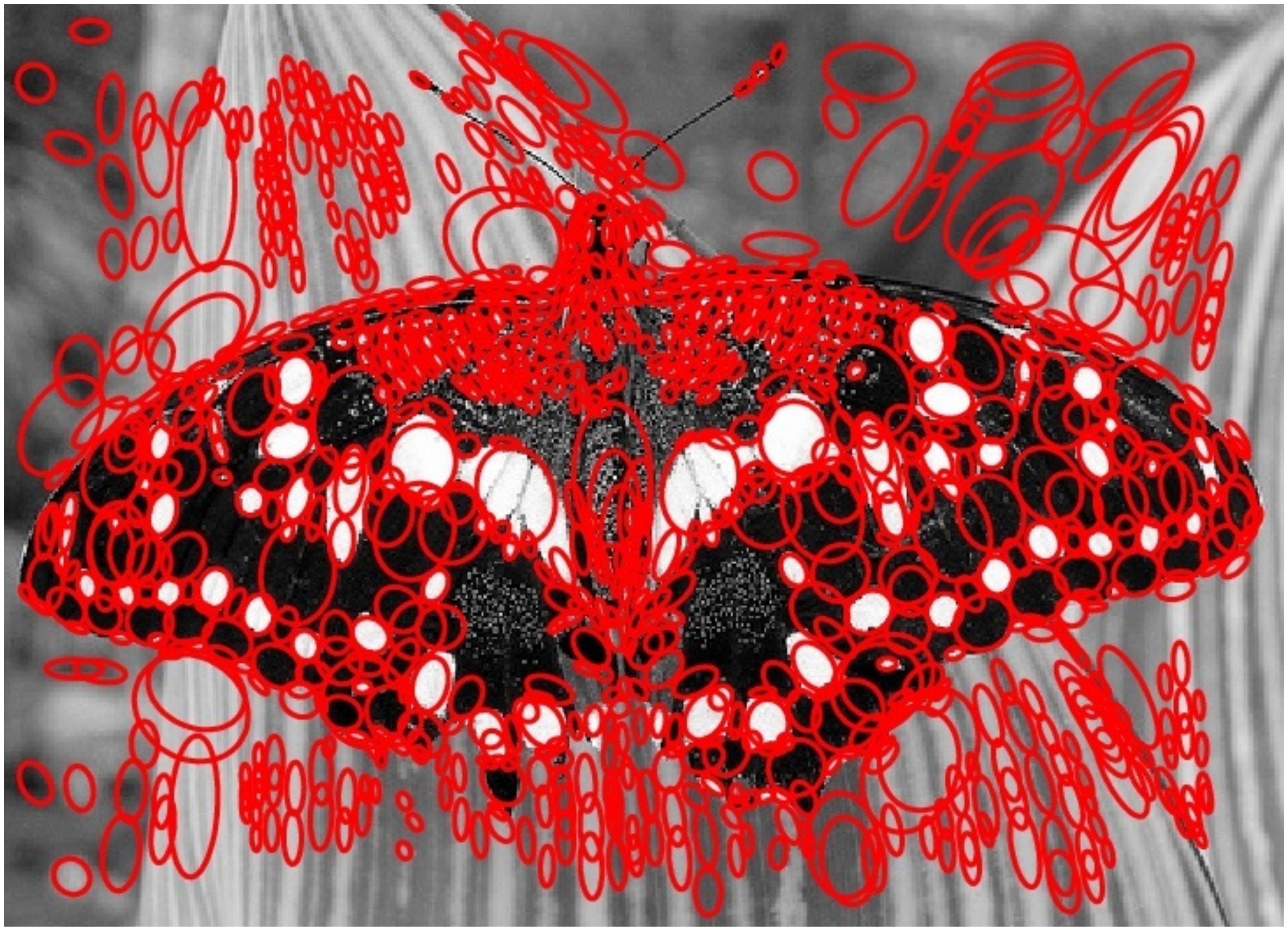


Scale-invariant regions (blobs)

K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors. IEEE PAMI 2005

Slide: S. Lazebnik

Going deeper

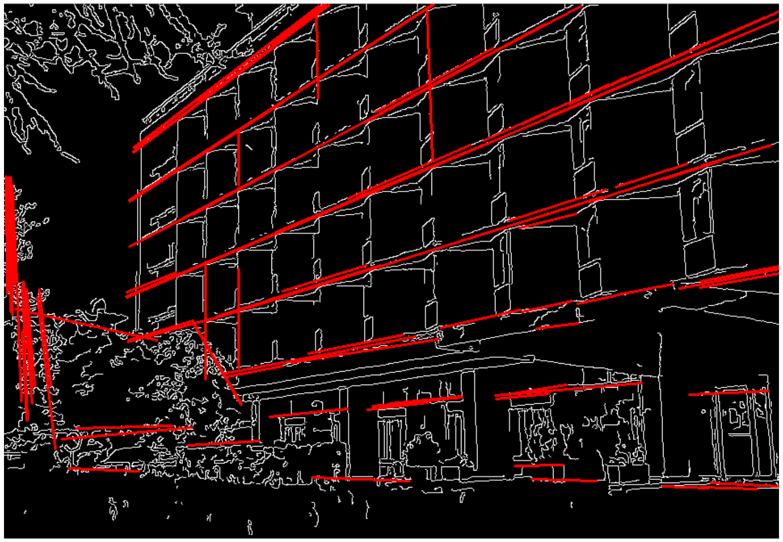


Affine-adapted blobs

K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors. IEEE PAMI 2005

Slide: S. Lazebnik

Fitting



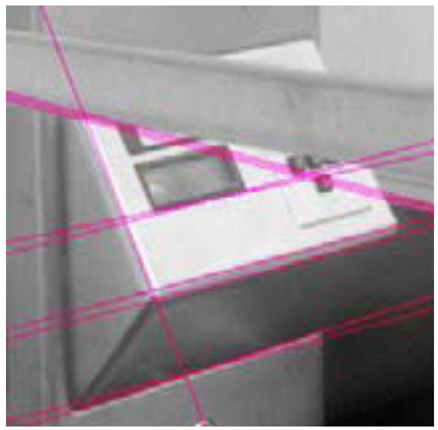
Fitting

- 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

Slide: S. Lazebnik

Fitting

- Choose a parametric model to represent a set of features



simple model: lines

simple model: circles

complicated model: car

Source: K. Grauman

Fitting: Issues

Case study: Line detection

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

http://vision.caltech.edu/malaa/software/research/caltech-lane-detection/

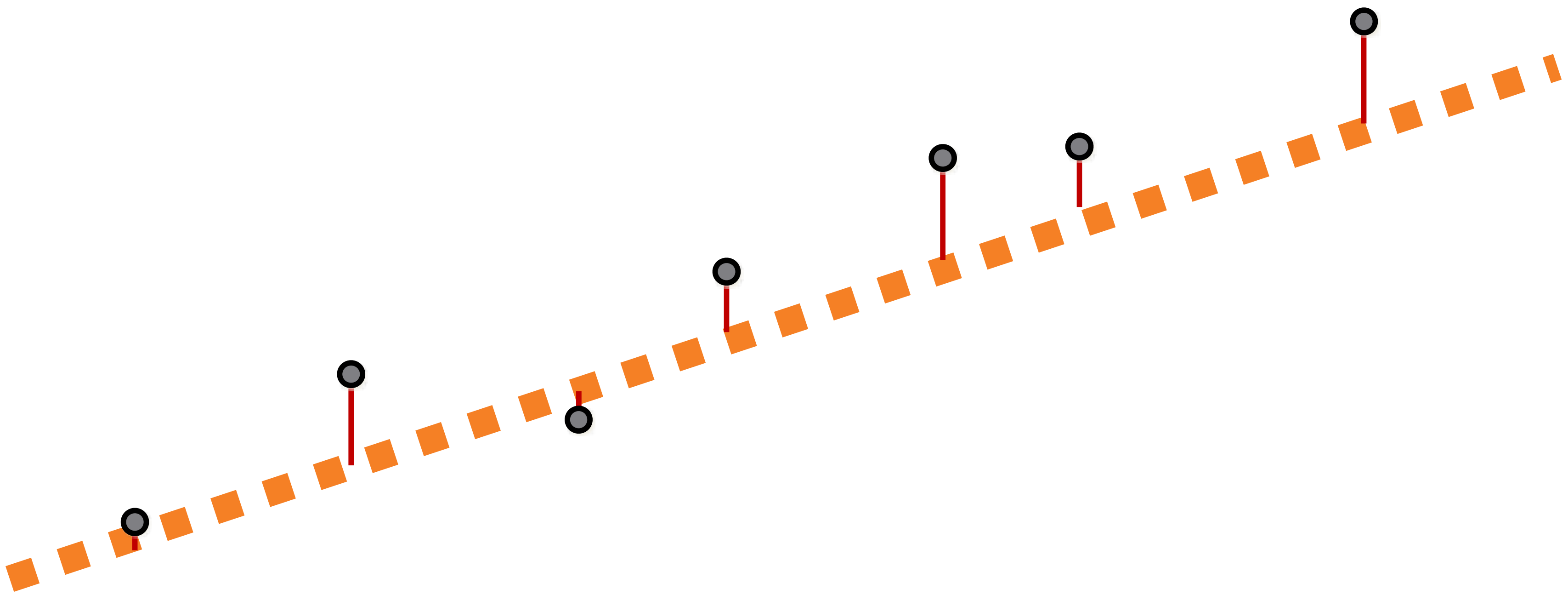
Source: S. Lazebnik

Fitting

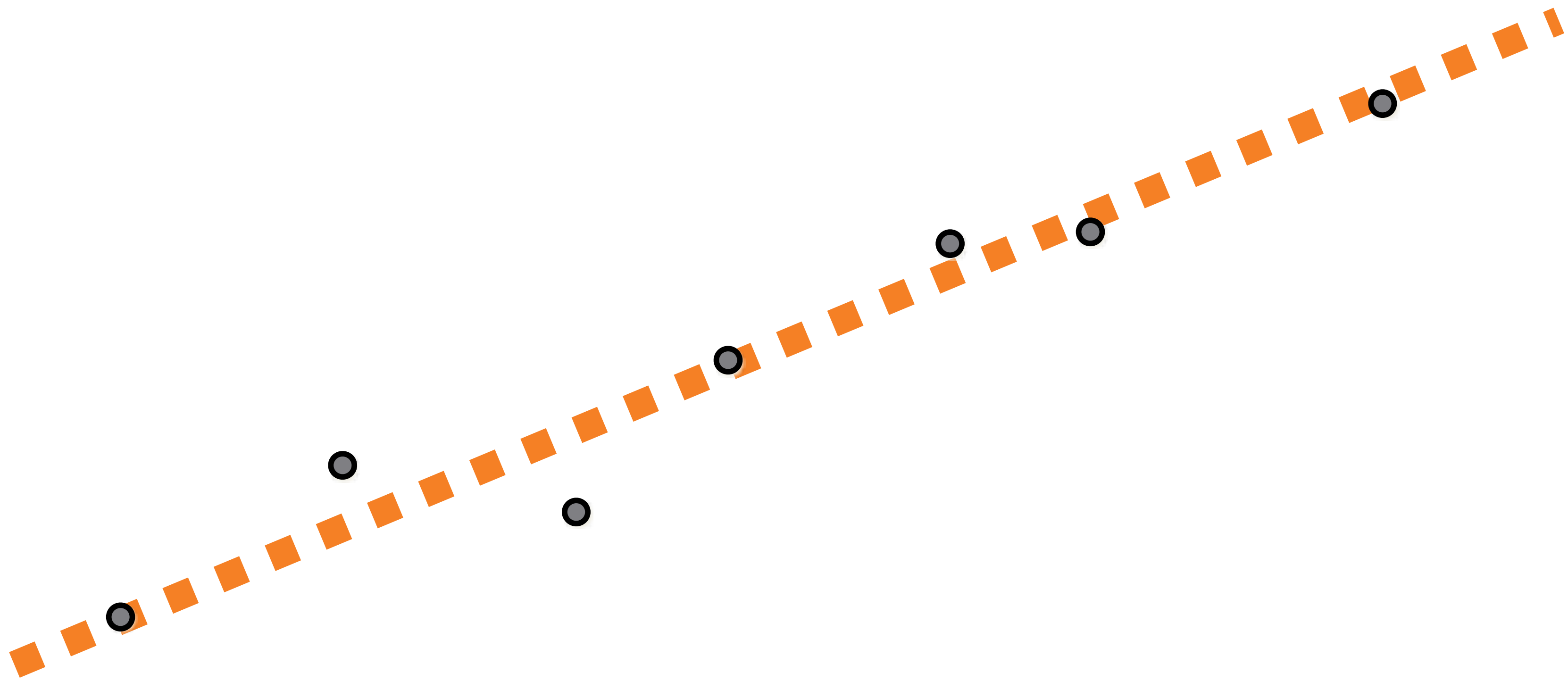
- If we know which points belong to the line, how do we find the "optimal" line parameters?
  - Least squares

Source: S. Lazebnik

Least squares minimization



Least squares minimization



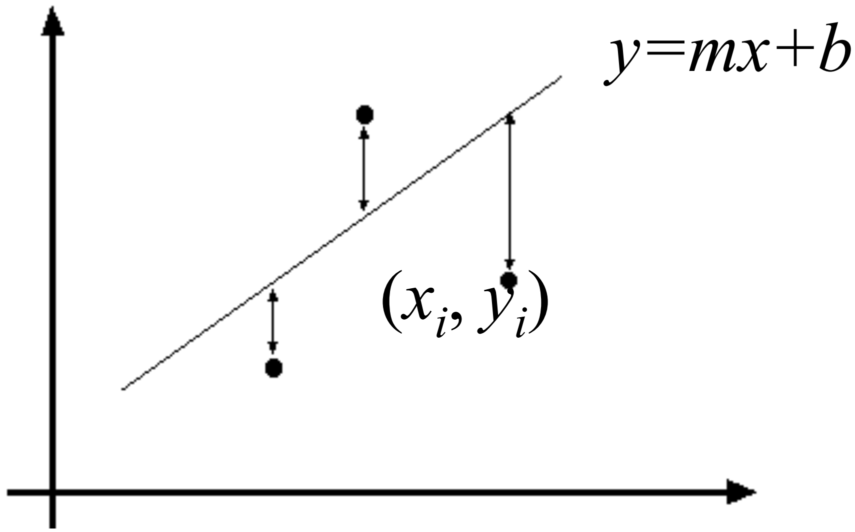
Least squares line fitting

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

Line equation: $y_i = m\,x_i + b$

Find (m, b) to minimize

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



Good: closed-form solution

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

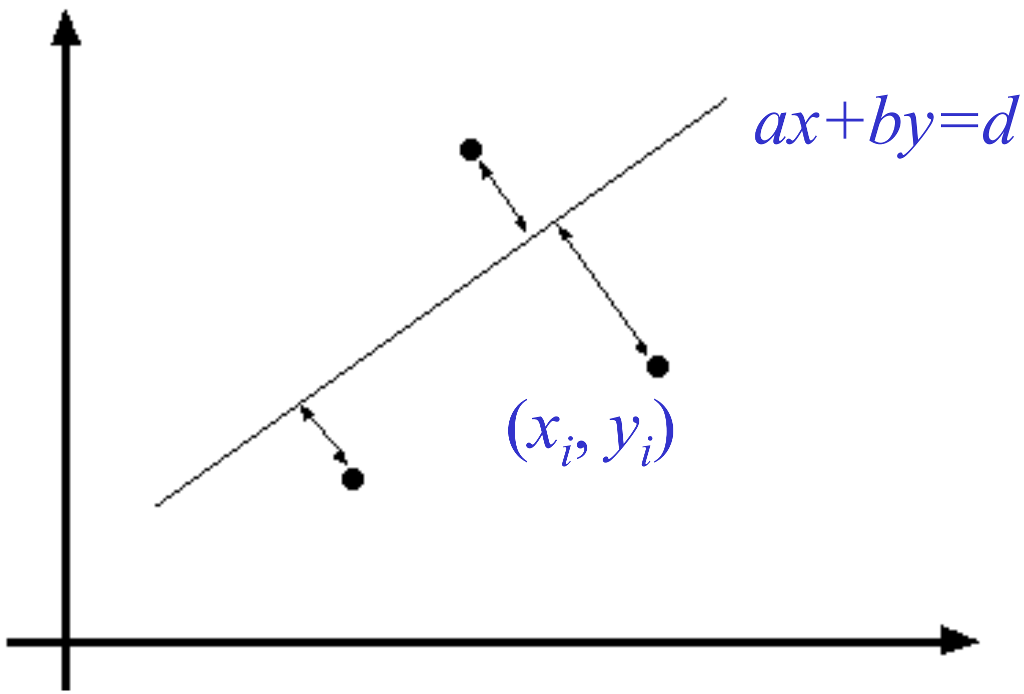
where

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

Bad:

1) Fails completely for vertical lines
2) Not rotation-invariant

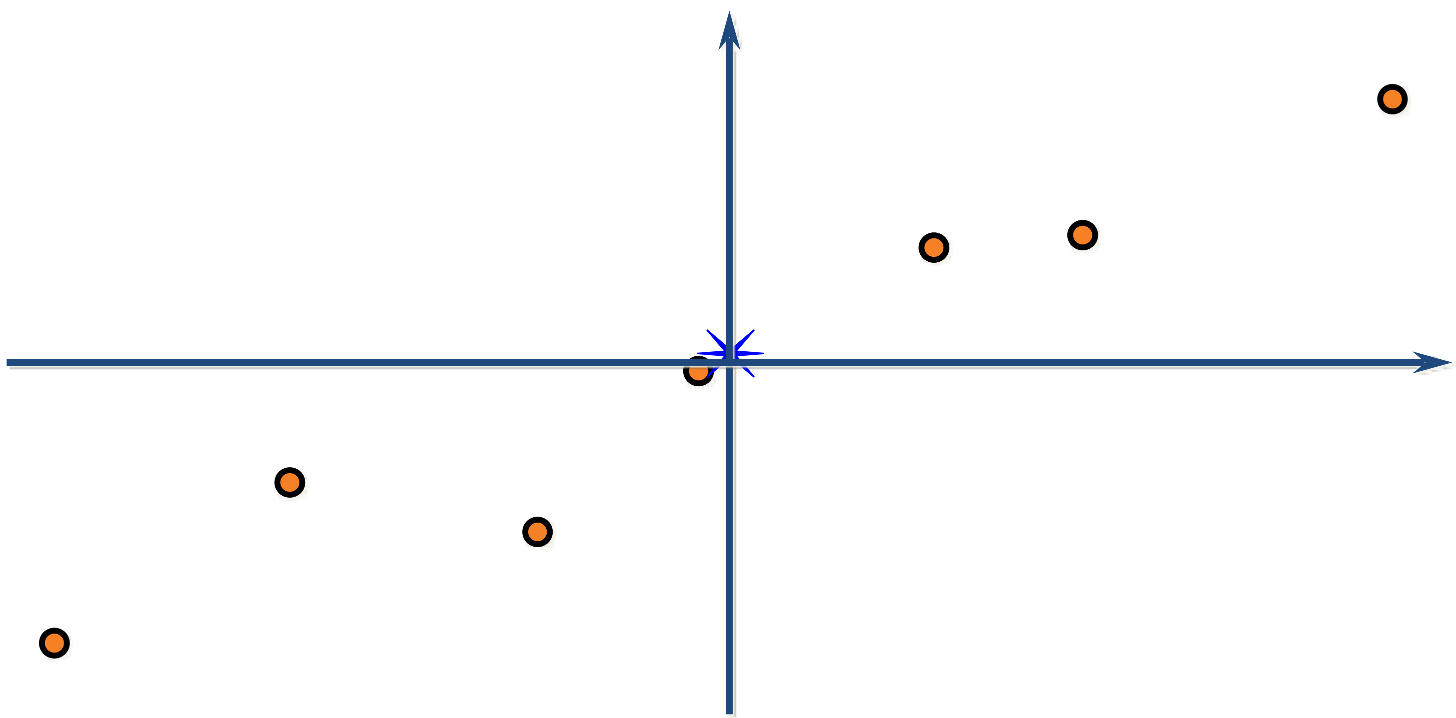
Total least squares



(a bit more detail at the end of the slide deck, posted online)

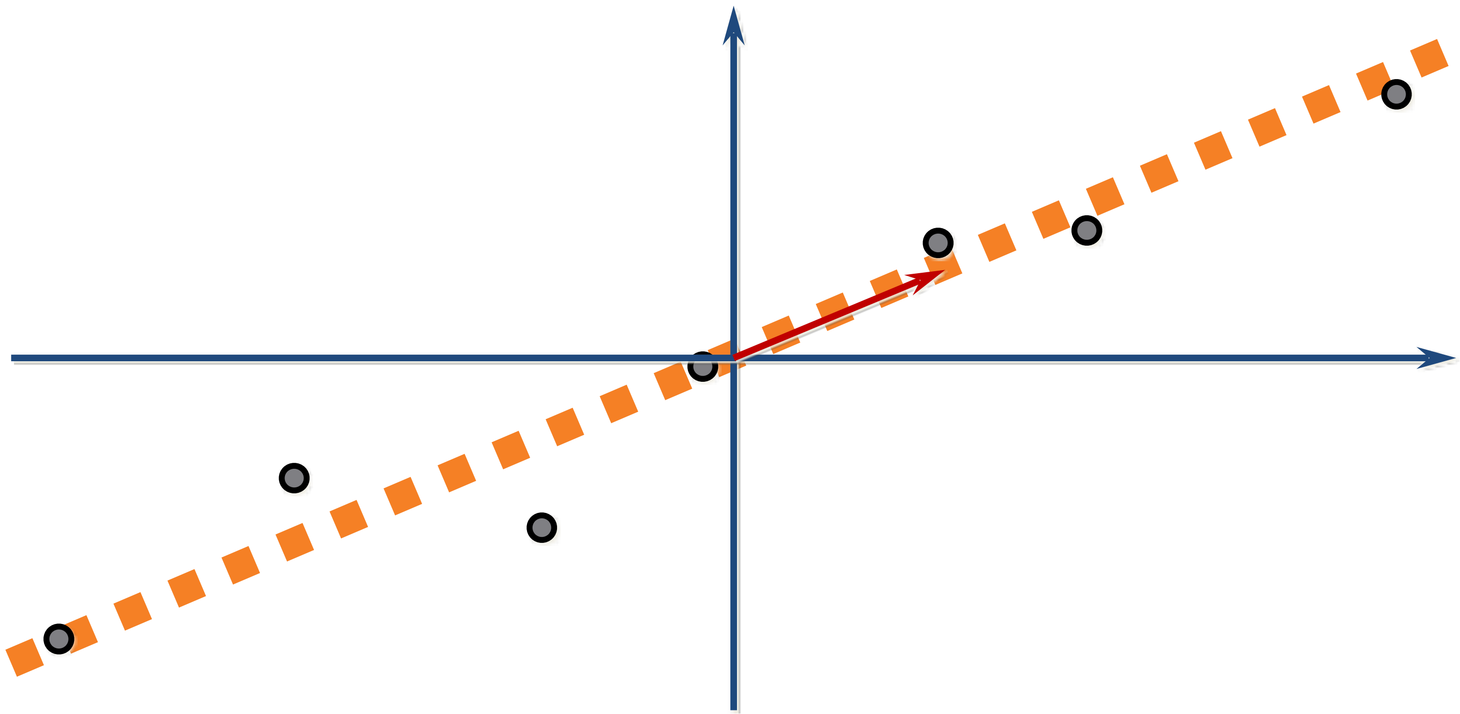
Total Least Squares

1. Translate center of mass to origin



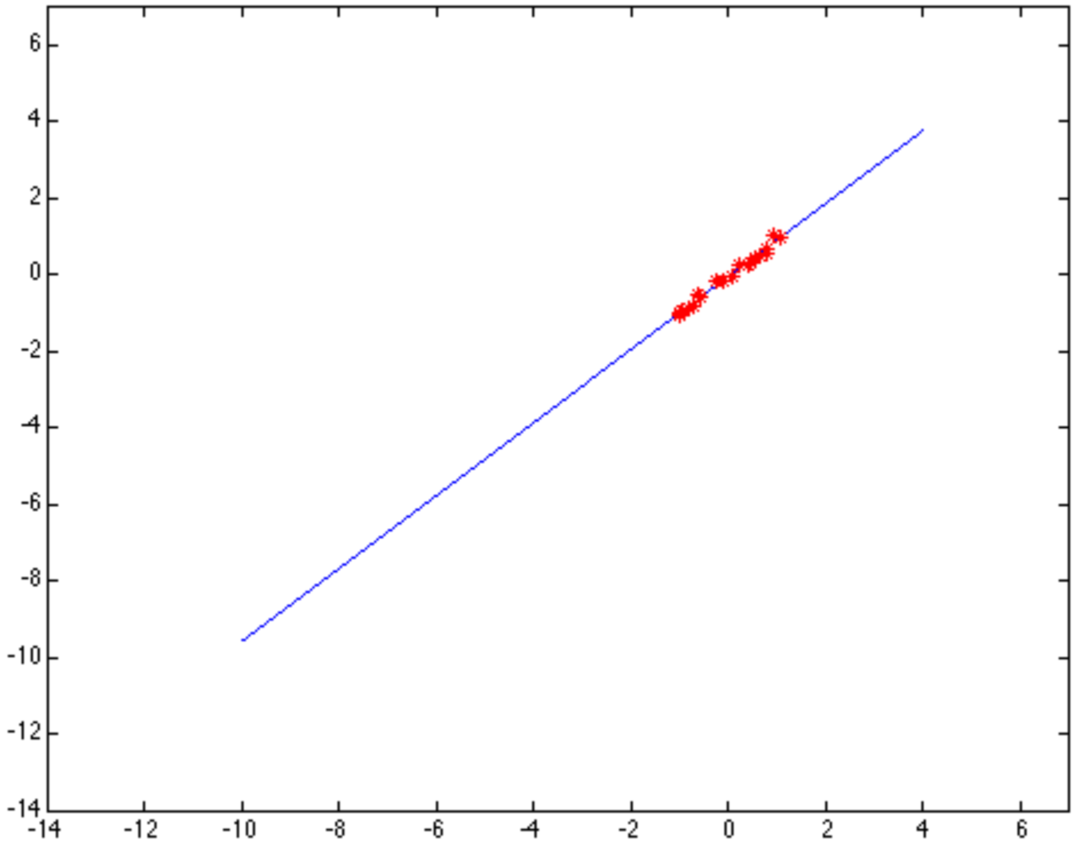
Total Least Squares

2. Compute covariance matrix,
   find eigenvector w. largest eigenvalue



Least squares: Robustness to noise

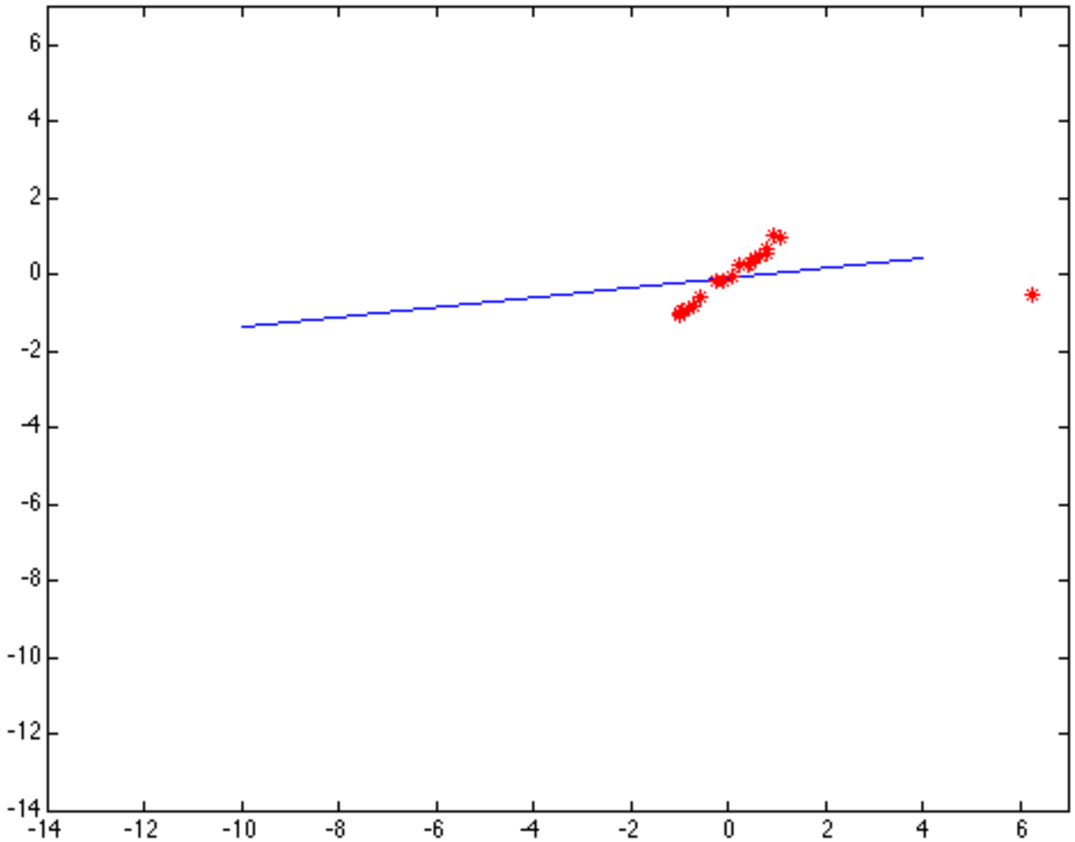
Least squares fit to the red points:



Source: S. Lazebnik

Least squares: Robustness to noise

Least squares fit with an outlier:



Problem: squared error heavily penalizes outliers

Source: S. Lazebnik

Outliers

- Least squares assumes Gaussian errors
- Outliers: points with extremely low probability of occurrence (according to Gaussian statistics)
  - Can be result of *data association* problems
- Can have strong influence on least squares

Robust Estimation

- Goal: develop parameter estimation methods insensitive to *small* numbers of *large* errors
- General approach: try to give large deviations less weight
- e.g., median is a robust measure, mean is not

Least Absolute Value Fitting

- Minimize $\sum_i |y_i - f(x_i, a, b, \ldots)|$ *(median)*

  instead of $\sum_i (y_i - f(x_i, a, b, \ldots))^2$ *(mean)*

- Points far away from trend get comparatively less influence

Outlier detection and rejection

- Lots of methods for fitting models in the presence of outliers
  - e.g., look up “iteratively reweighed least squares”
- Often not guaranteed to converge; require good starting point
  - (least squares estimator is often a good starting point)

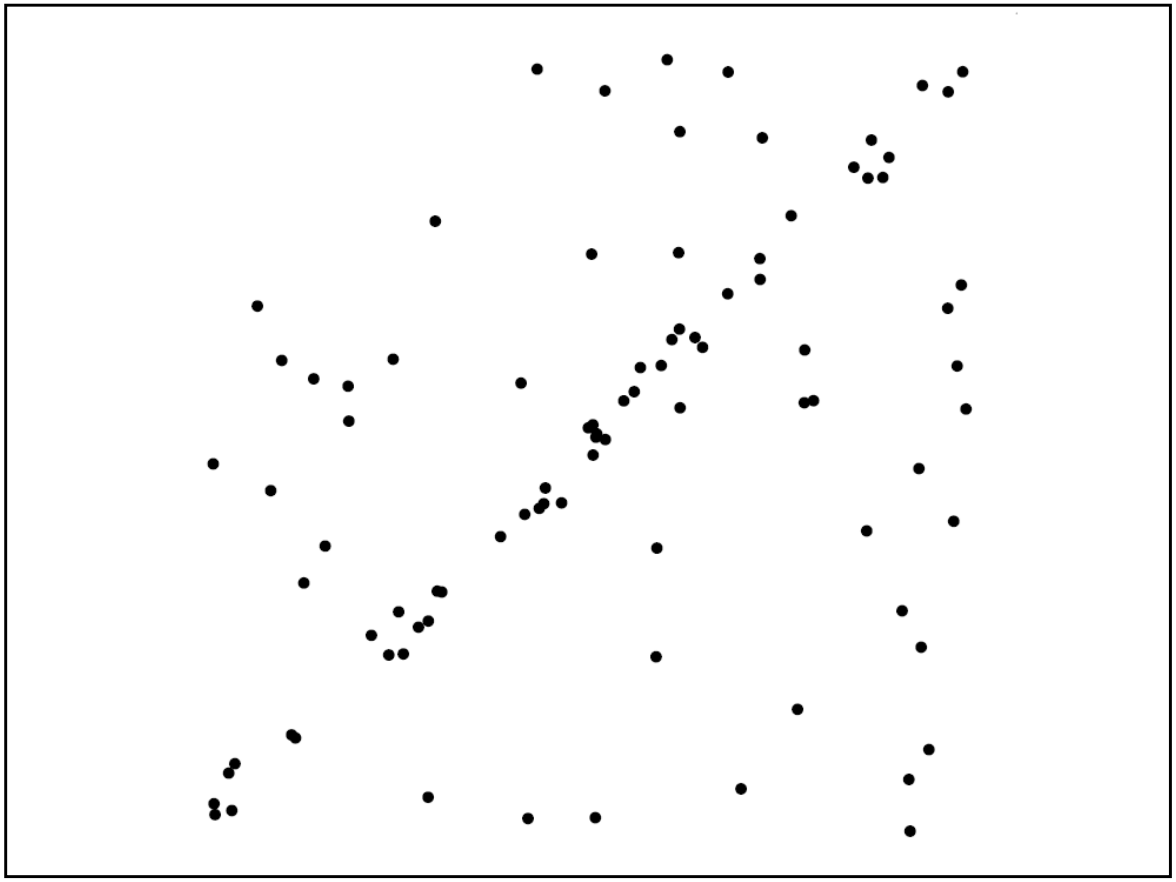
RANSAC

RANSAC

- RANdom SAmple Consensus: designed for bad data (in best case, up to 50% outliers)
- Take many random subsets of data
  - Choose a small subset uniformly at random
  - Fit a model to the data
  - Find all remaining points that are “close” to the model and reject the rest as outliers
- At the end, select model that agreed with most points

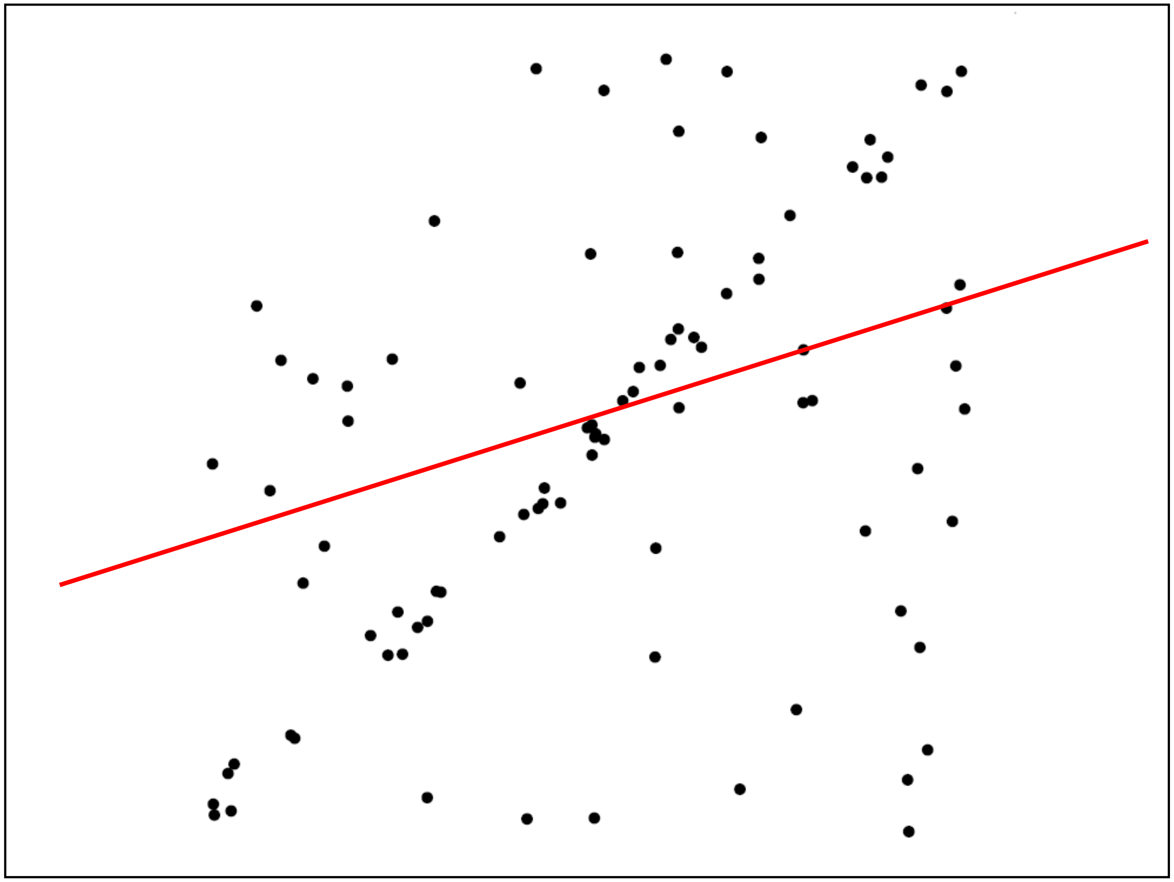
M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

RANSAC for line fitting example



Source: R. Raguram

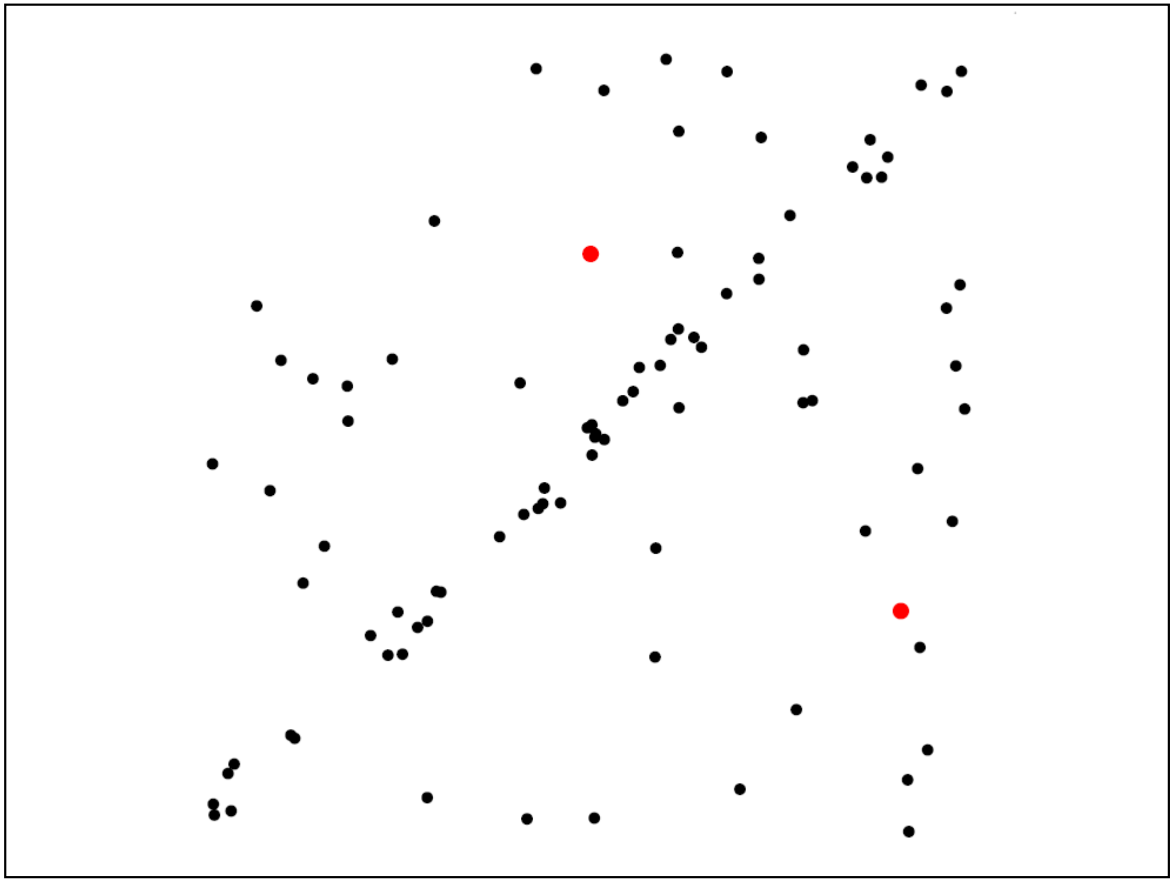
RANSAC for line fitting example



Least-squares fit

Source: R. Raguram

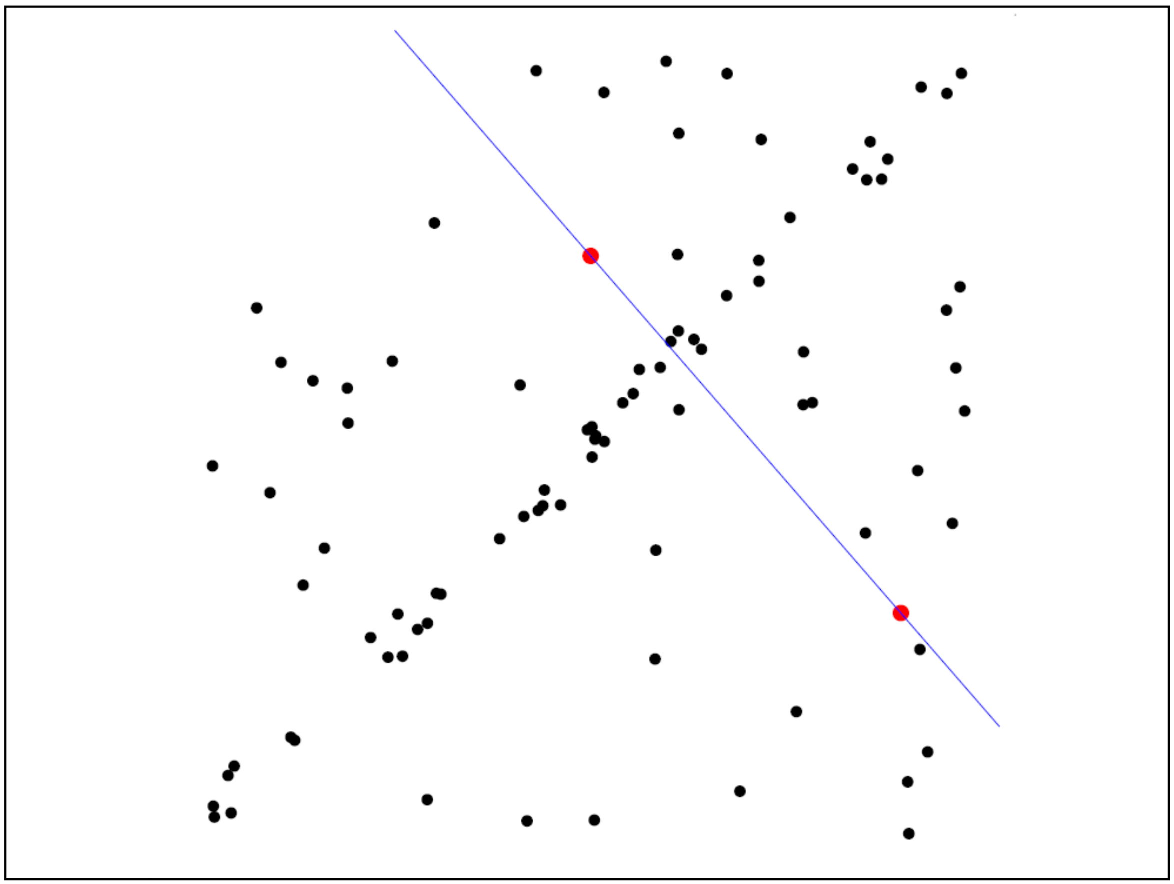
RANSAC for line fitting example



1. Randomly select minimal subset of points

Source: R. Raguram

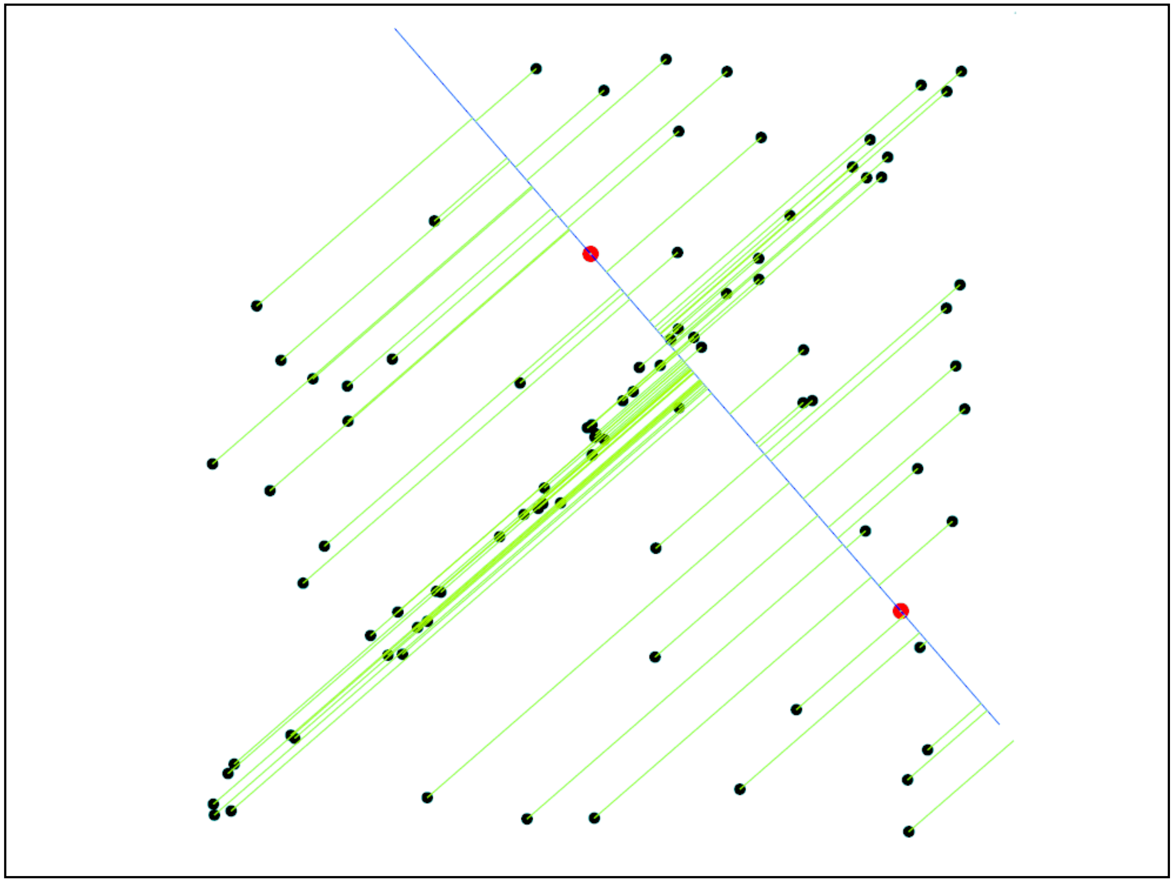
RANSAC for line fitting example



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

Source: R. Raguram

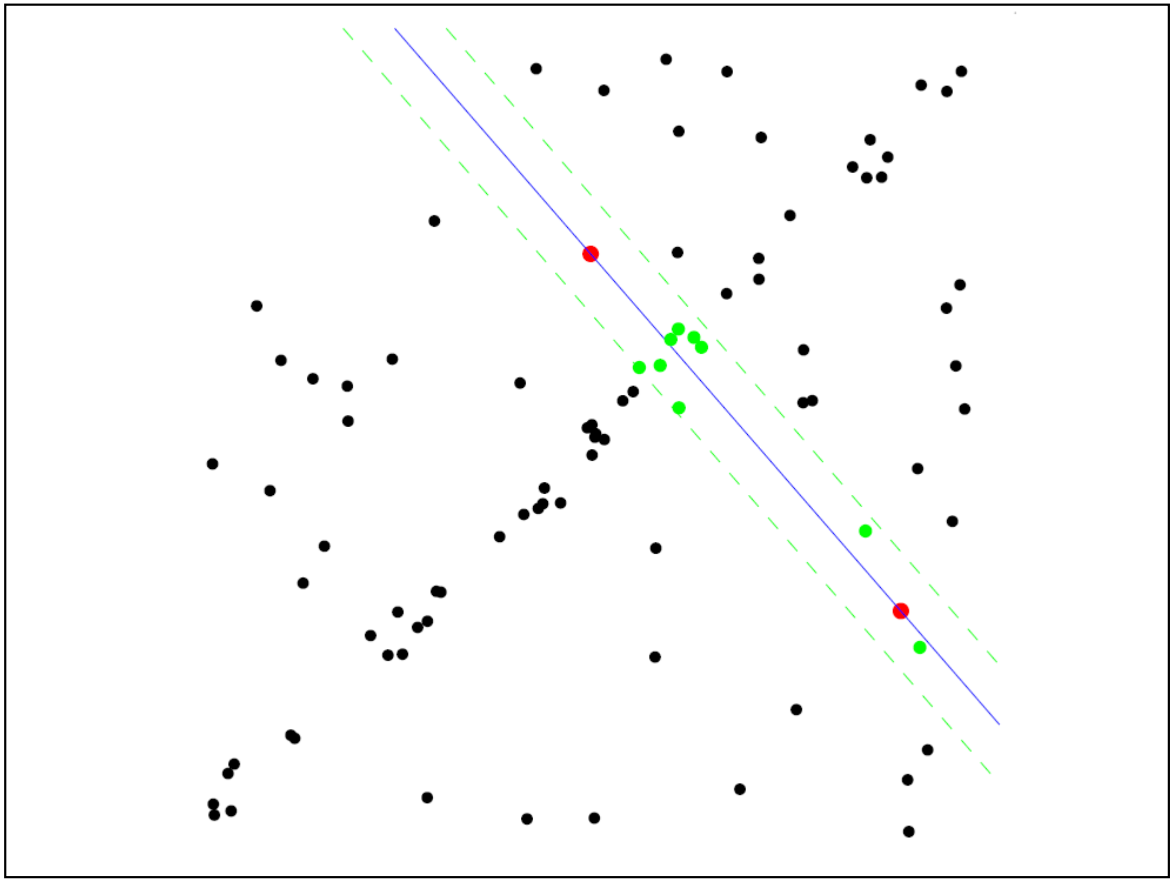
RANSAC for line fitting example



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

Source: R. Raguram

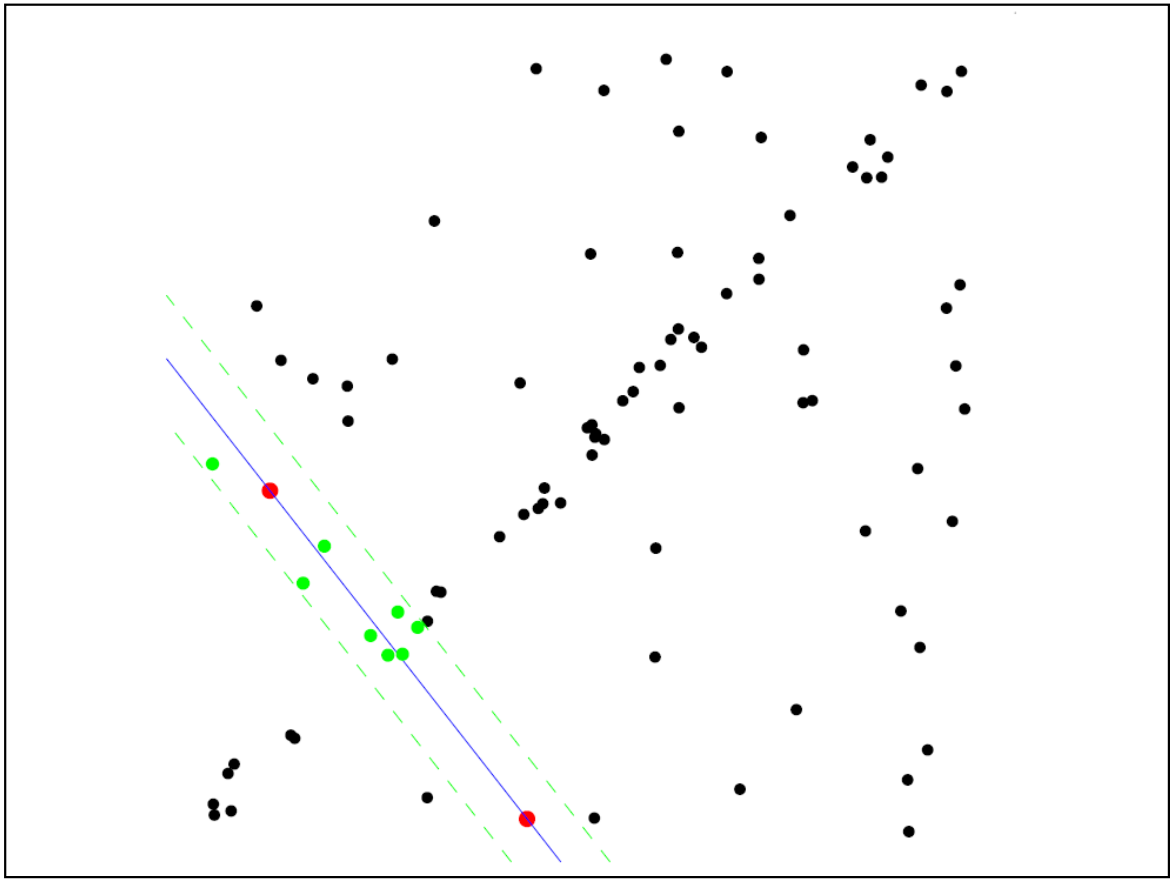
RANSAC for line fitting example



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

Source: R. Raguram

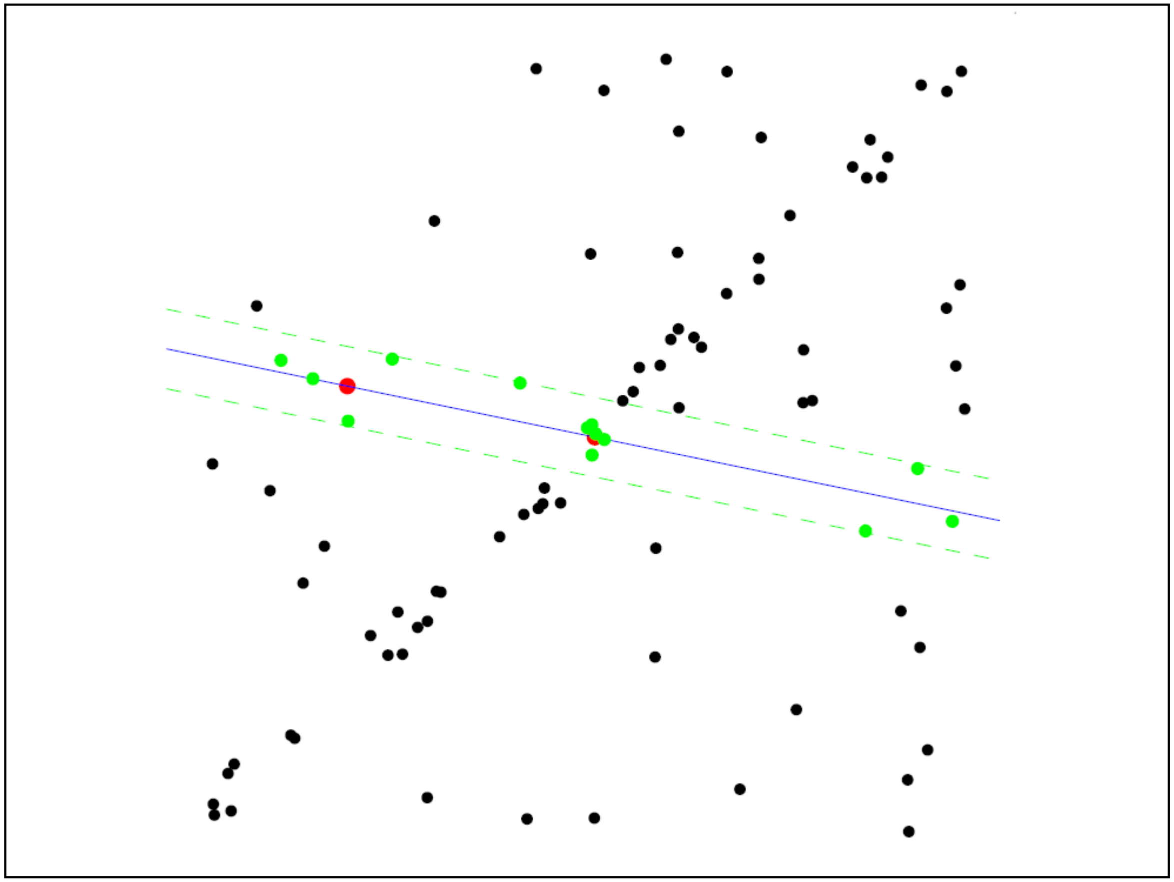
RANSAC for line fitting example



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

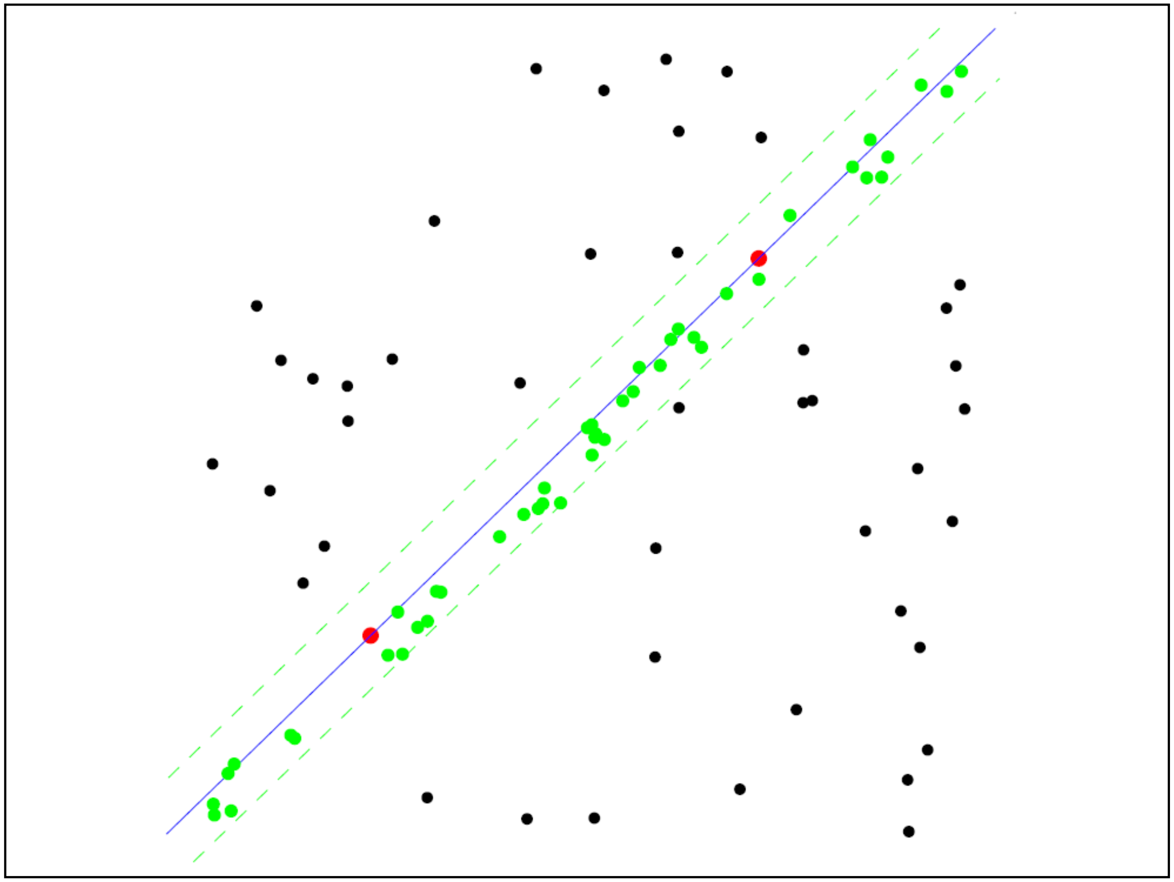


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example

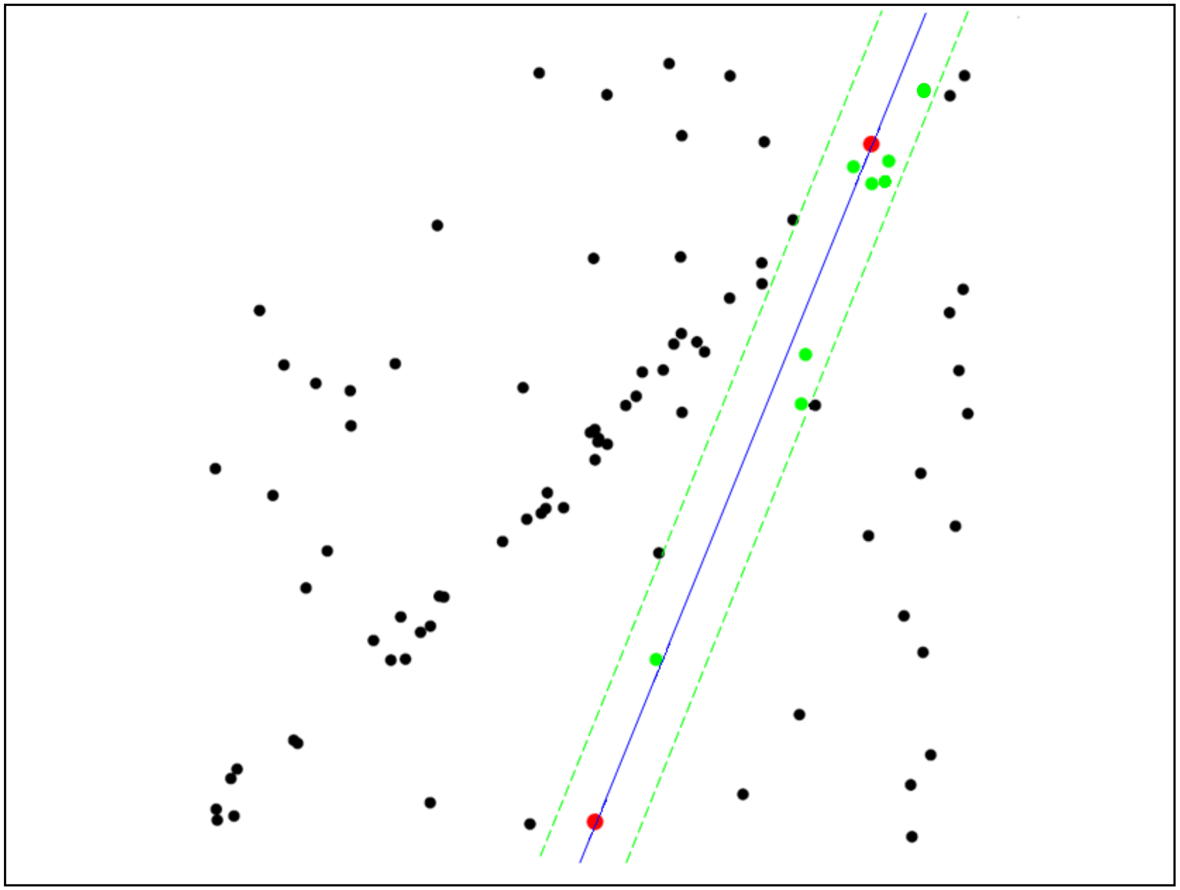
Uncontaminated sample



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

Source: S. Lazebnik

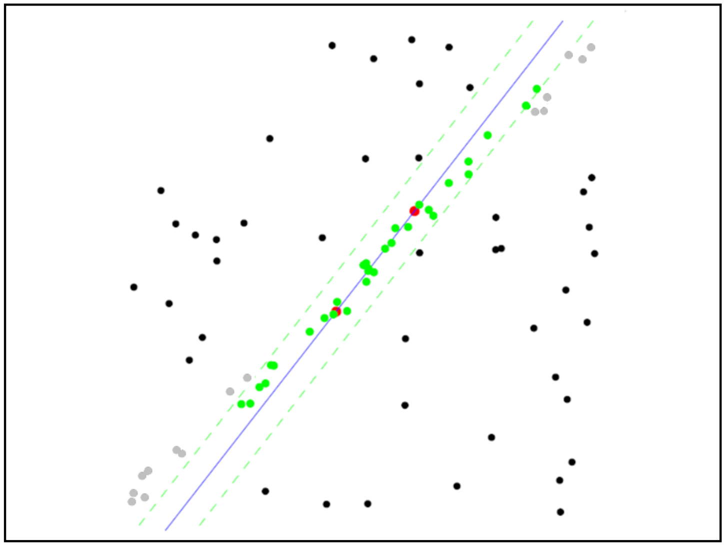
Choosing the parameters

- 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: $t = 1.96\sigma$
- 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

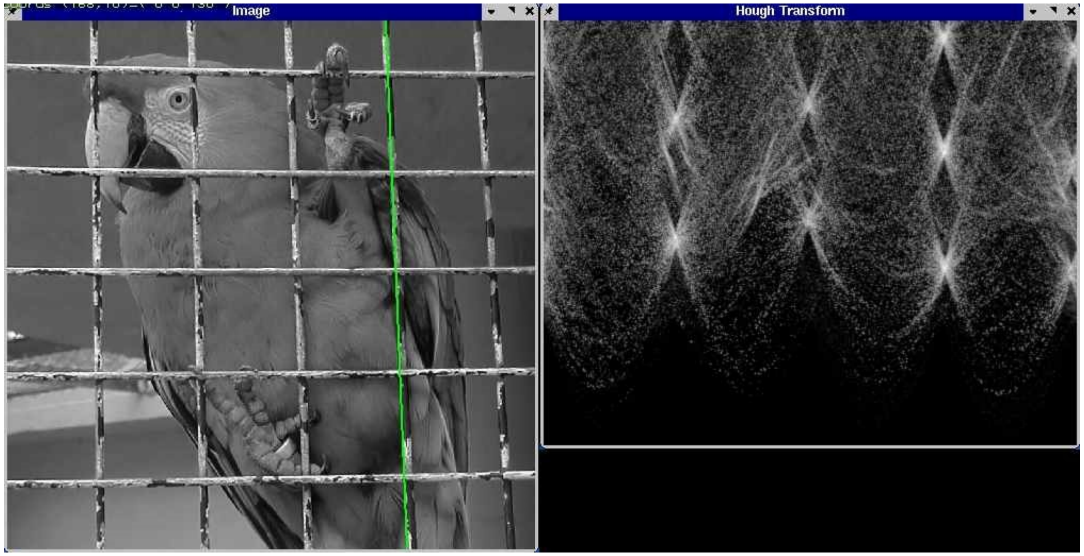
RANSAC pros and cons

- 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

Hough transform



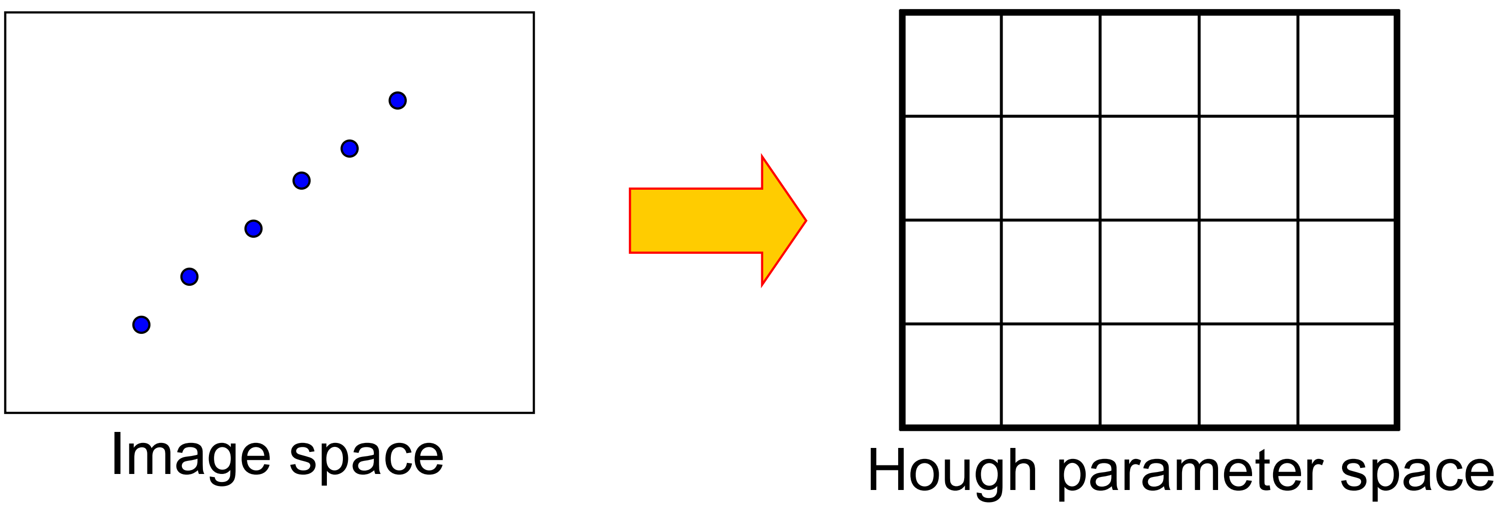
Voting schemes

- 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

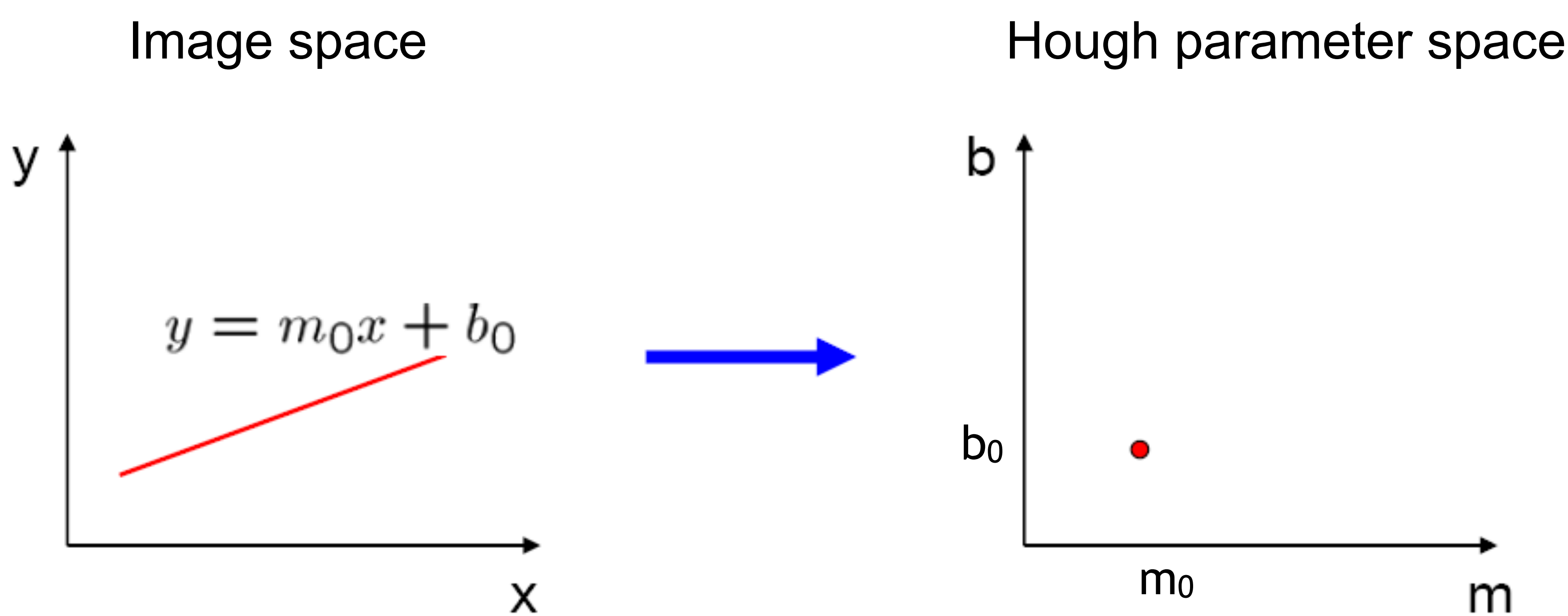


P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

Source: S. Lazebnik

Parameter space representation

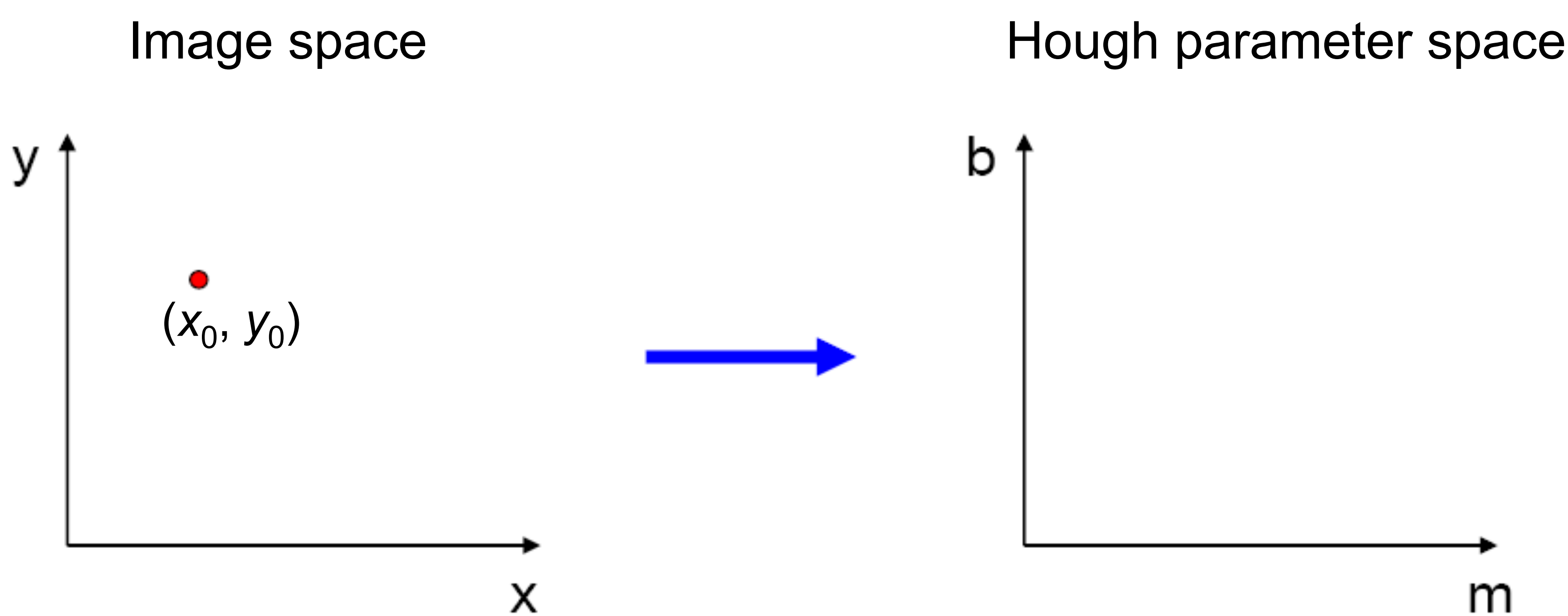
- A line in the image corresponds to a point in Hough space



Source: S. Seitz

Parameter space representation

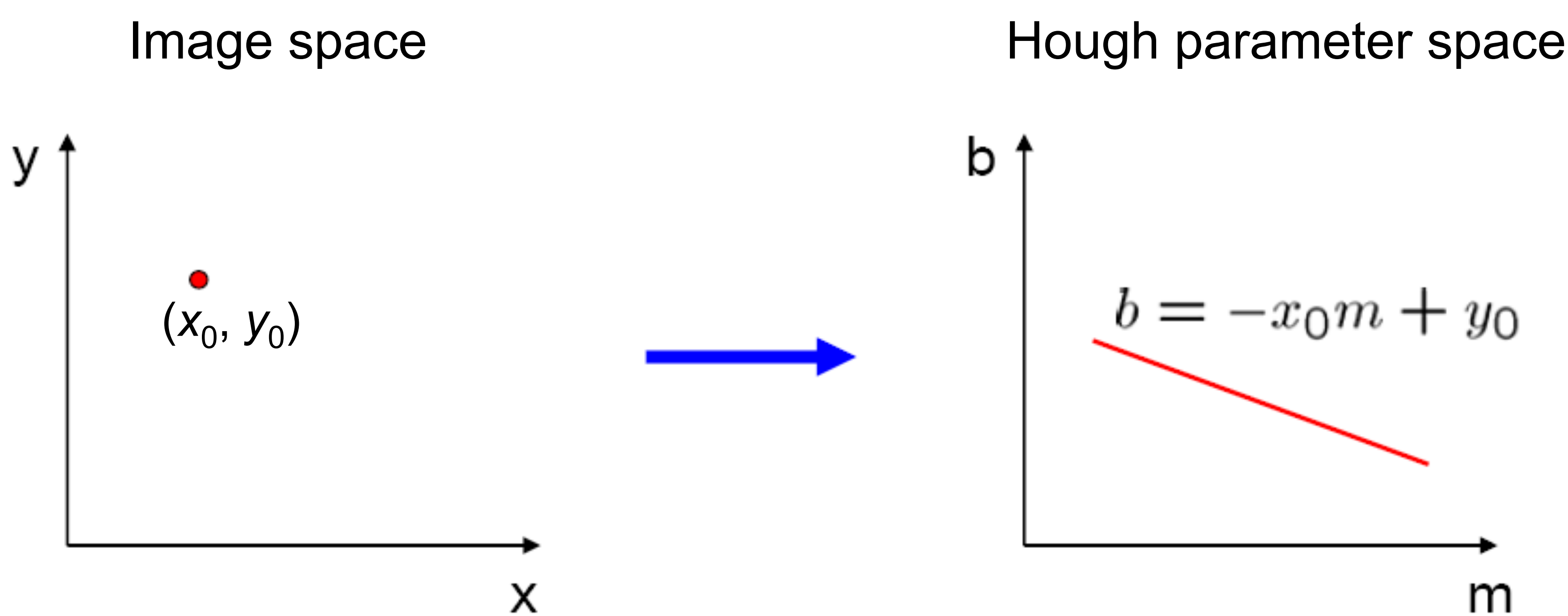
- What does a point (x_0, y_0) in the image space map to in the Hough space?



Source: S. Seitz

Parameter space representation

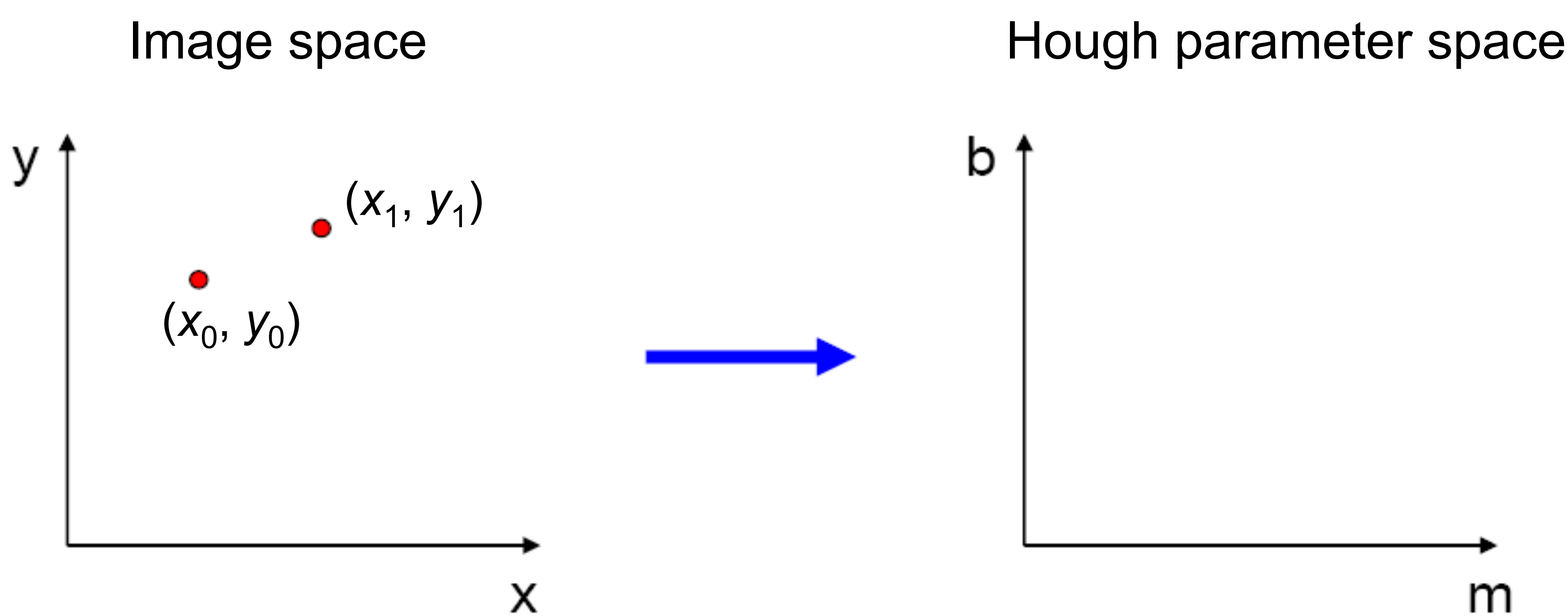
- What does a point (x_0, y_0) in the image space map to in the Hough space?
  - Answer: the solutions of $b = -x_0 m + y_0$, which is a line in Hough space



Source: S. Seitz

Parameter space representation

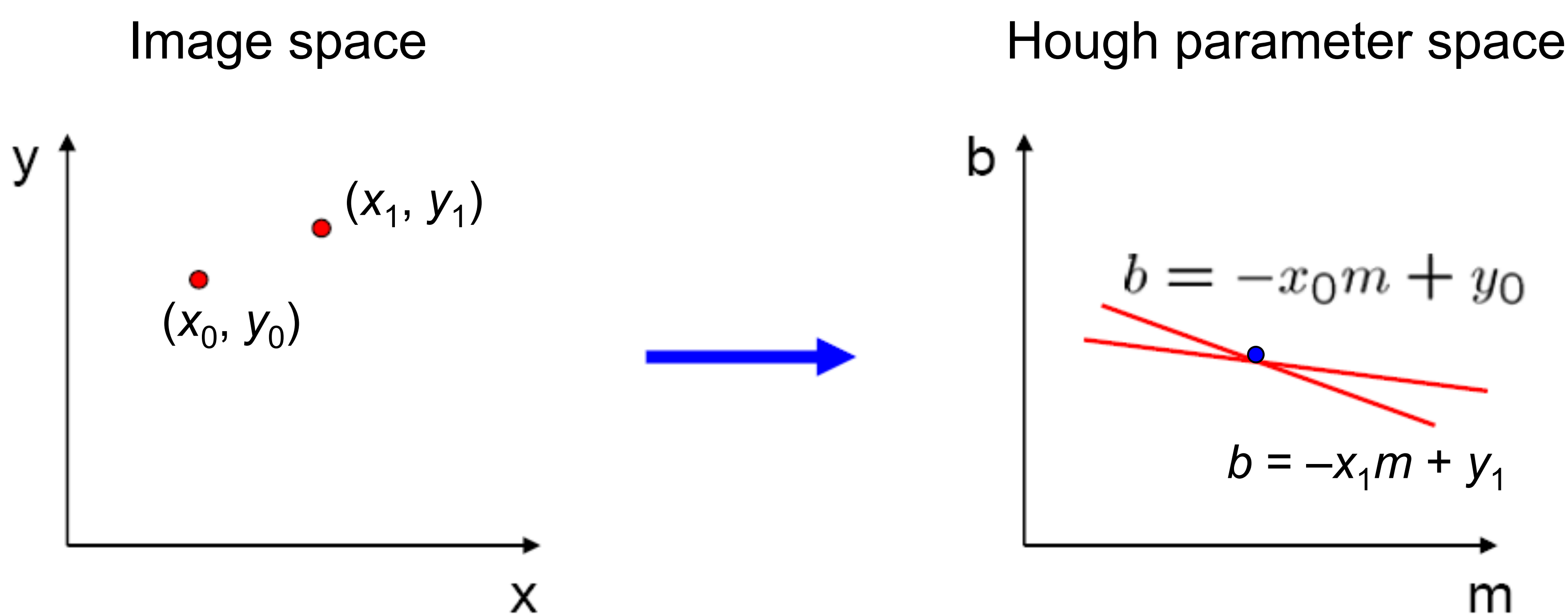
- Where does the line that contains both (x_0, y_0) and (x_1, y_1) map to?



Source: S. Seitz

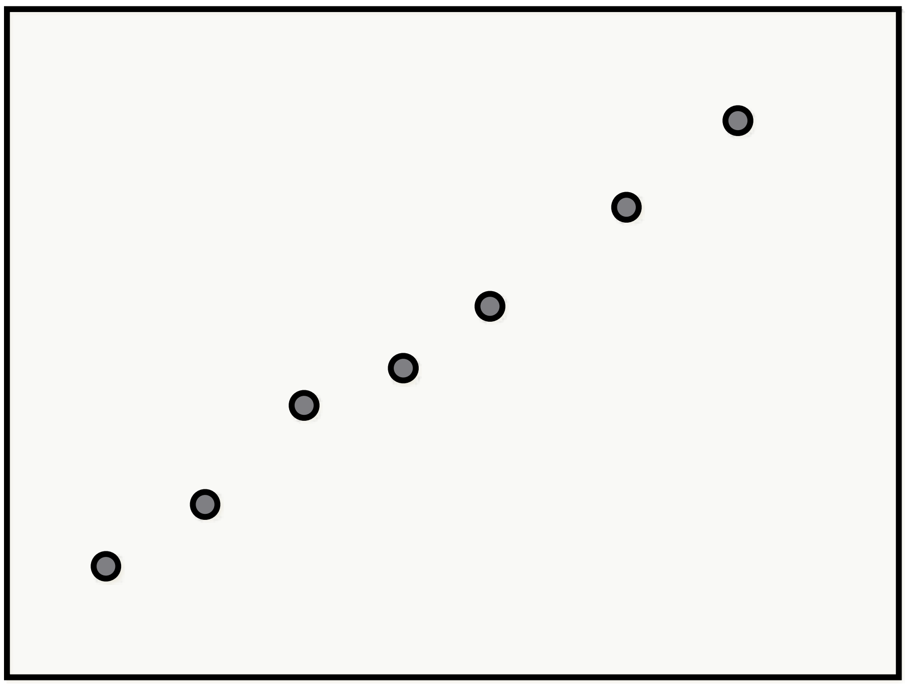
Parameter space representation

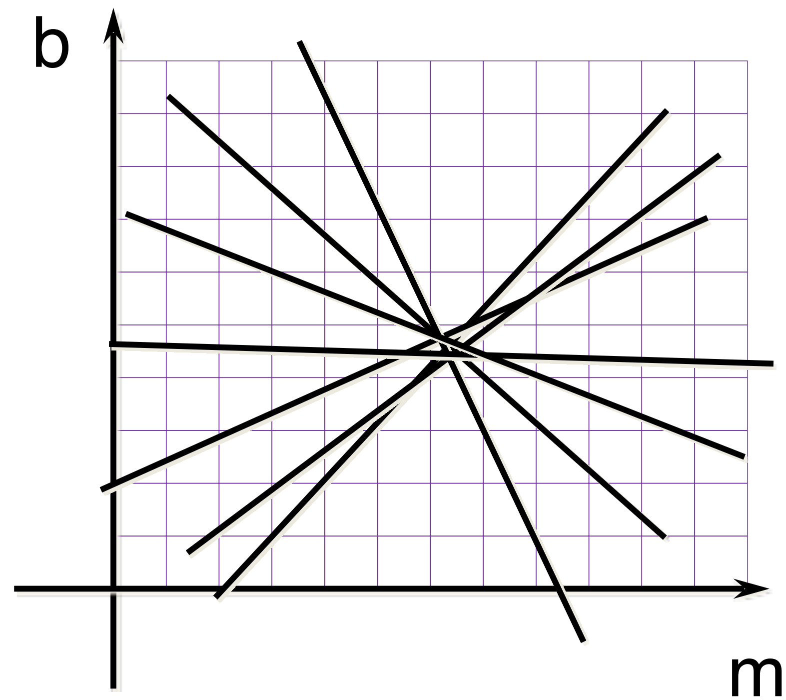
- Where does the line that contains both (x_0, y_0) and (x_1, y_1) map to?
  - It is the intersection of the lines $b = -x_0 m + y_0$ and $b = -x_1 m + y_1$



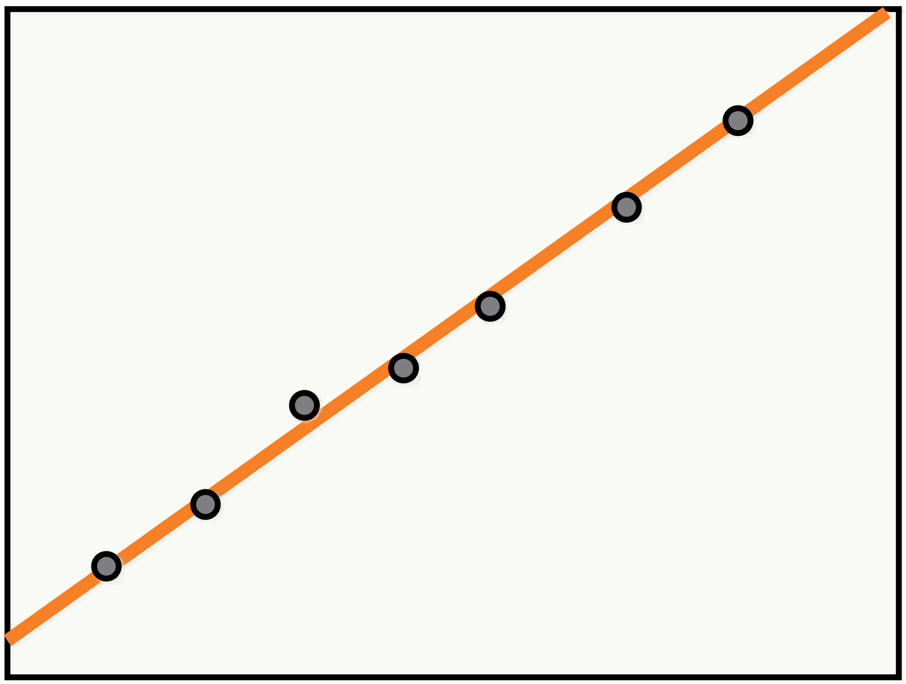
Source: S. Seitz

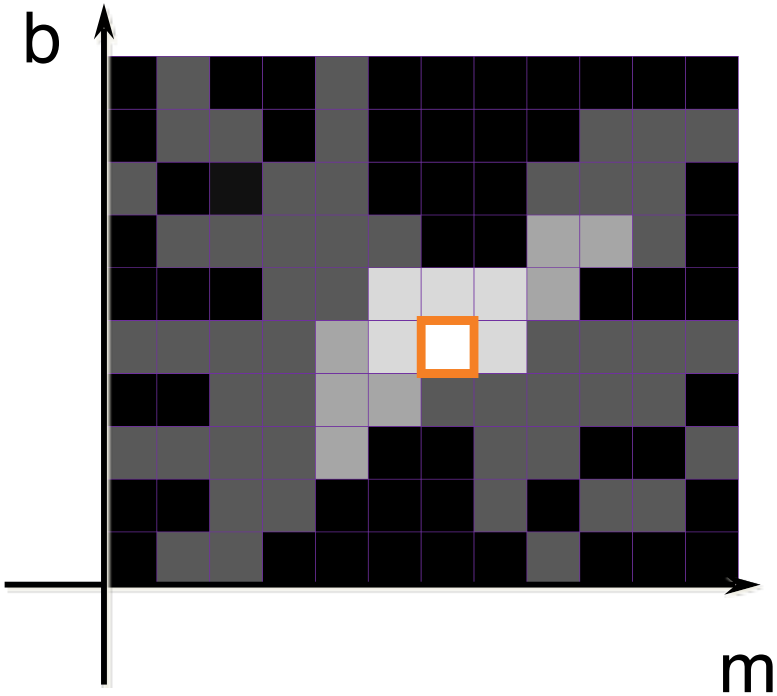
Hough Transform for Lines





Hough Transform for Lines





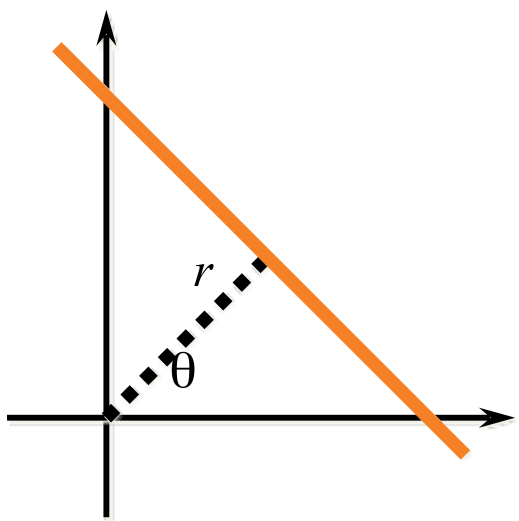
Bucket Selection

- How to select bucket size?
  - Too small: poor performance on noisy data
  - Too large: poor accuracy, possibility of false positives
- Large buckets + verification and refinement
  - Problems distinguishing nearby lines
- Be smarter at selecting buckets
  - Use gradient information to select subset of buckets
  - More sensitive to noise

Difficulties with Hough Transform for Lines

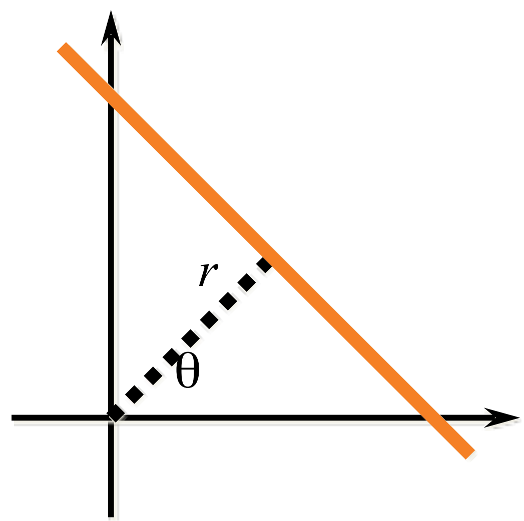
- Slope / intercept parameterization not ideal
  - Non-uniform sampling of directions
  - Can't represent vertical lines
- Angle / distance parameterization
  - Line represented as (r, θ) where

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

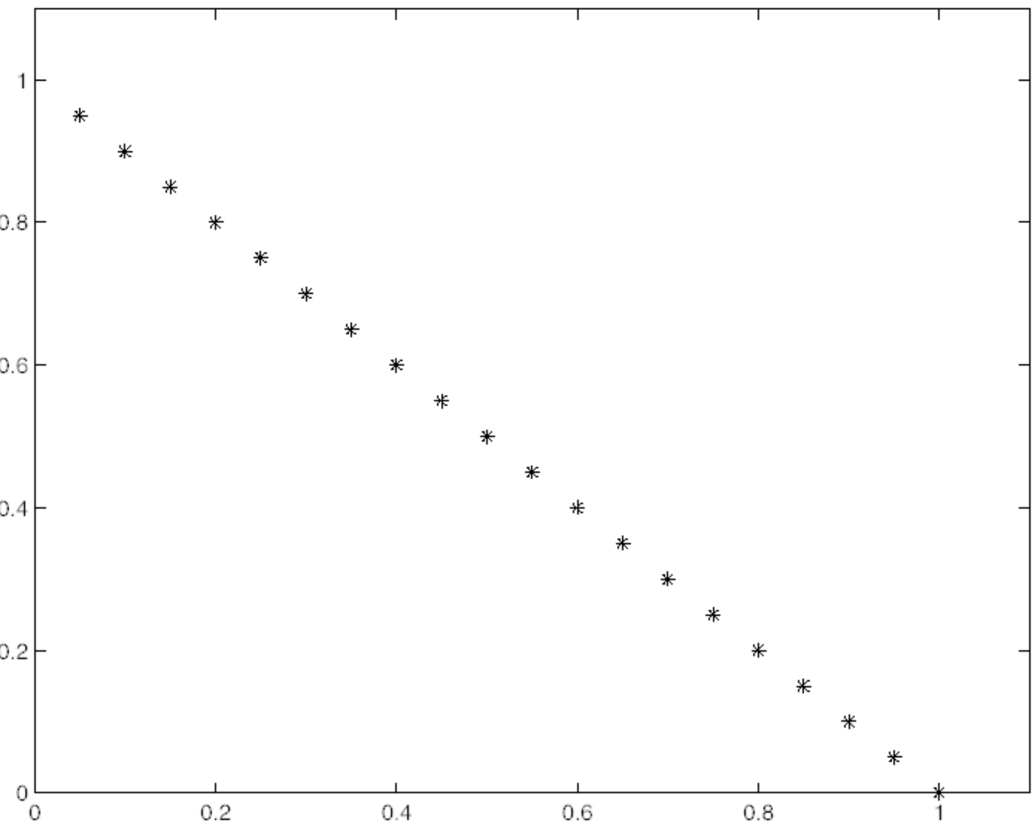


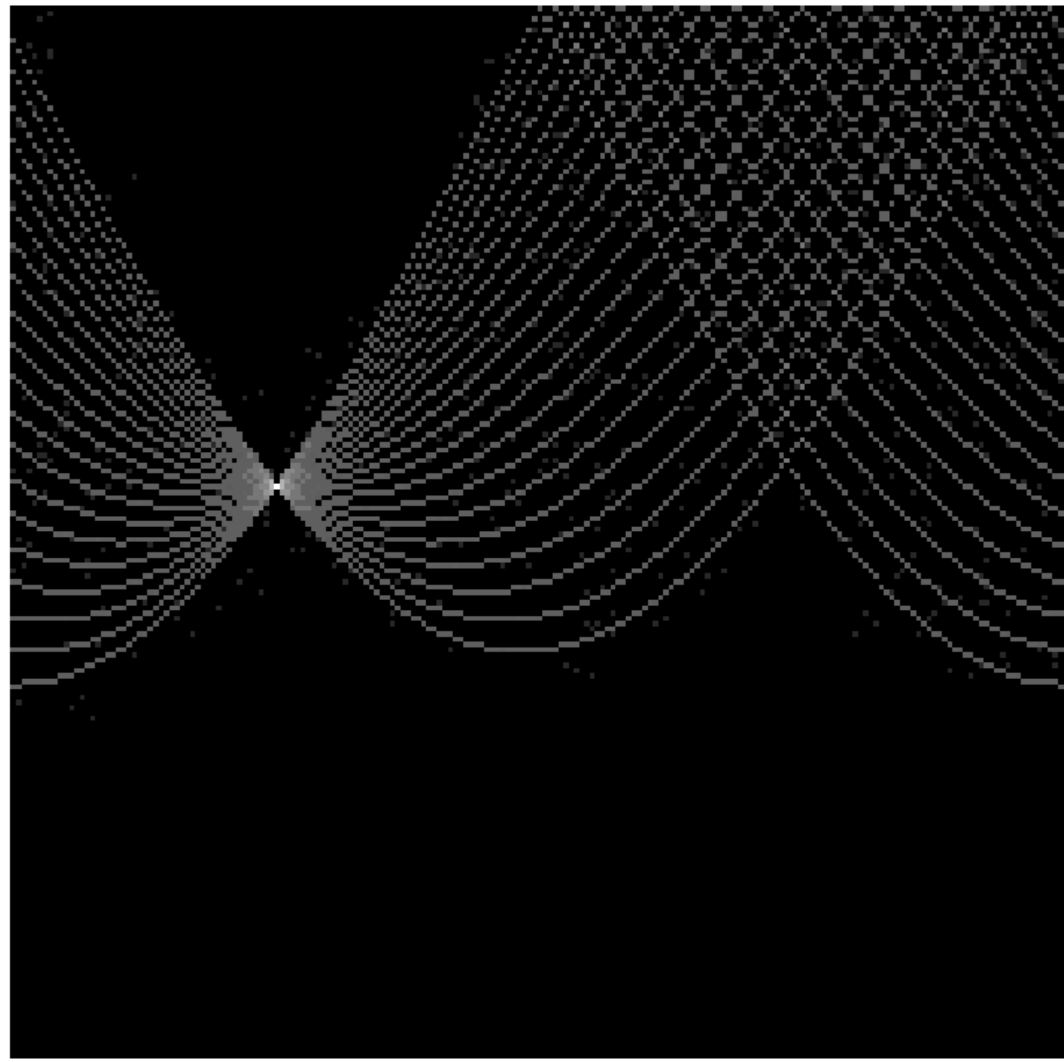
Angle / Distance Parameterization

- Advantage: uniform parameterization of directions
- Disadvantage: space of all lines passing through a point becomes a sinusoid in (r, θ) space



Hough Transform Results

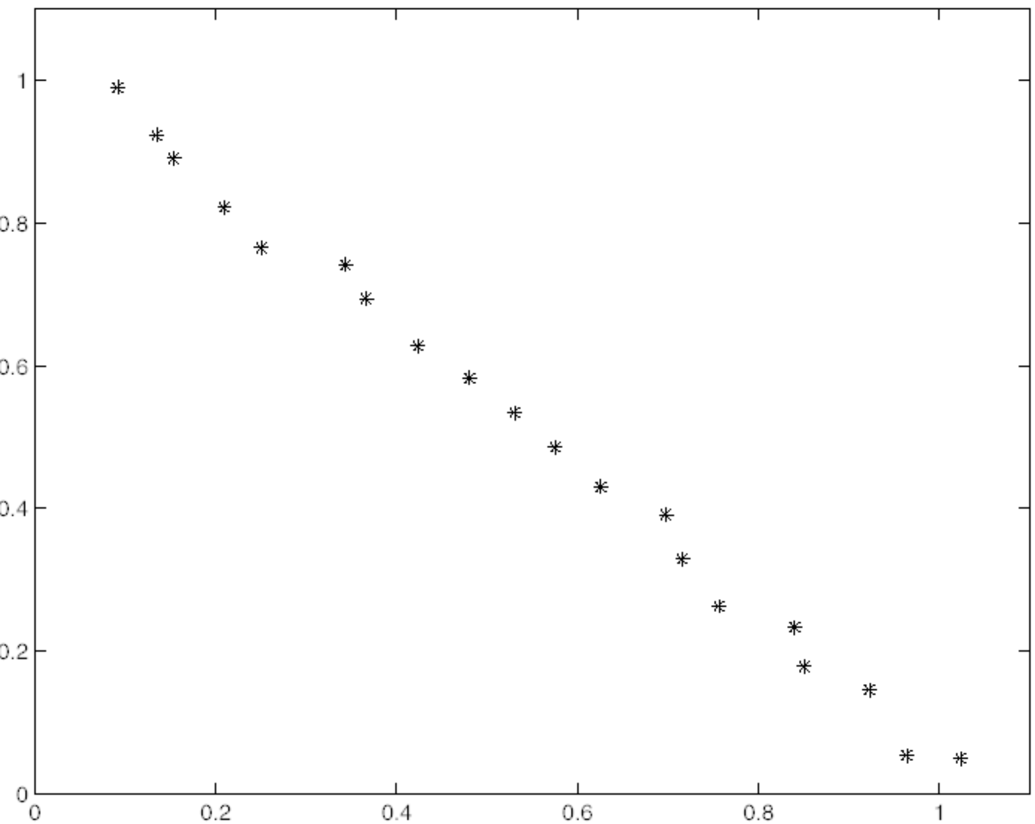


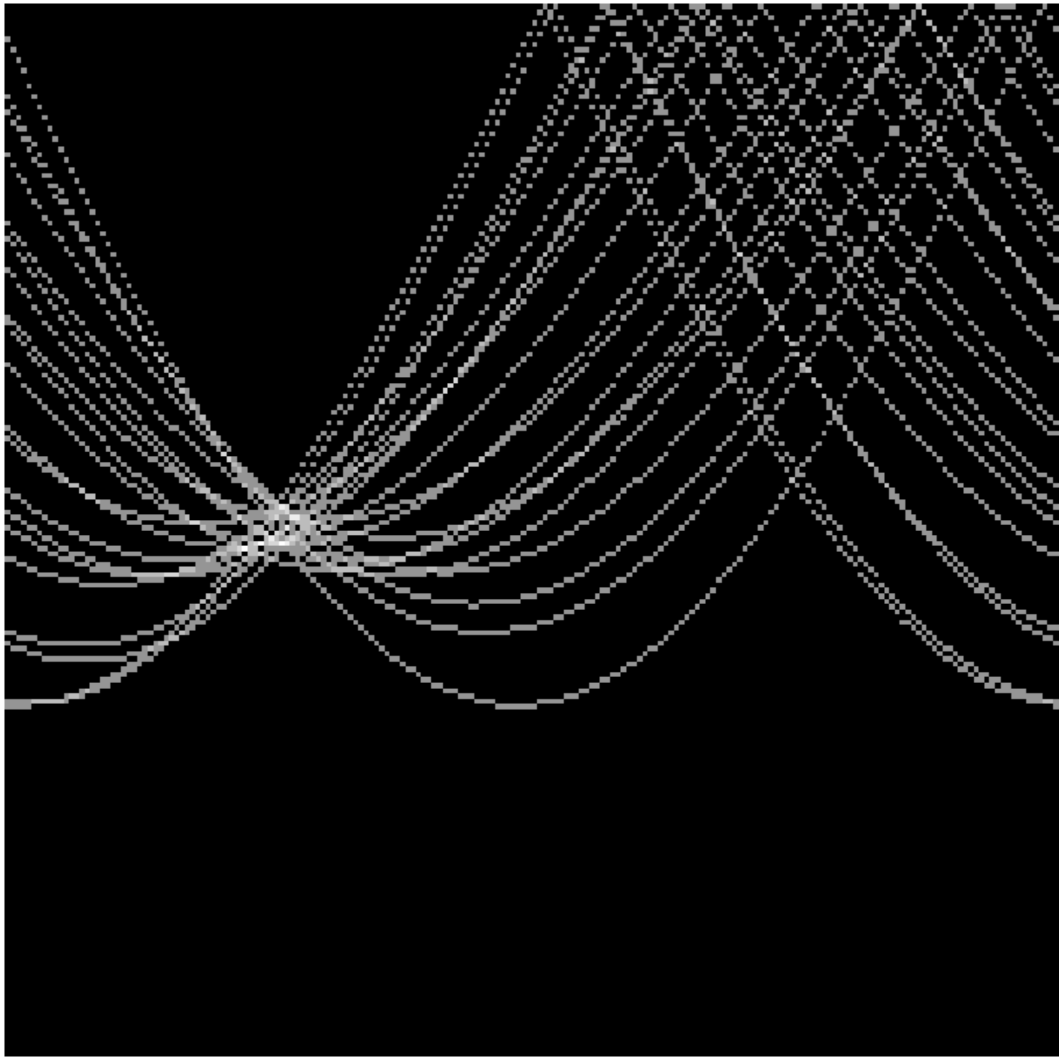


Forsyth & Ponce

Hough Transform with Noise

Peak gets fuzzy and hard to locate

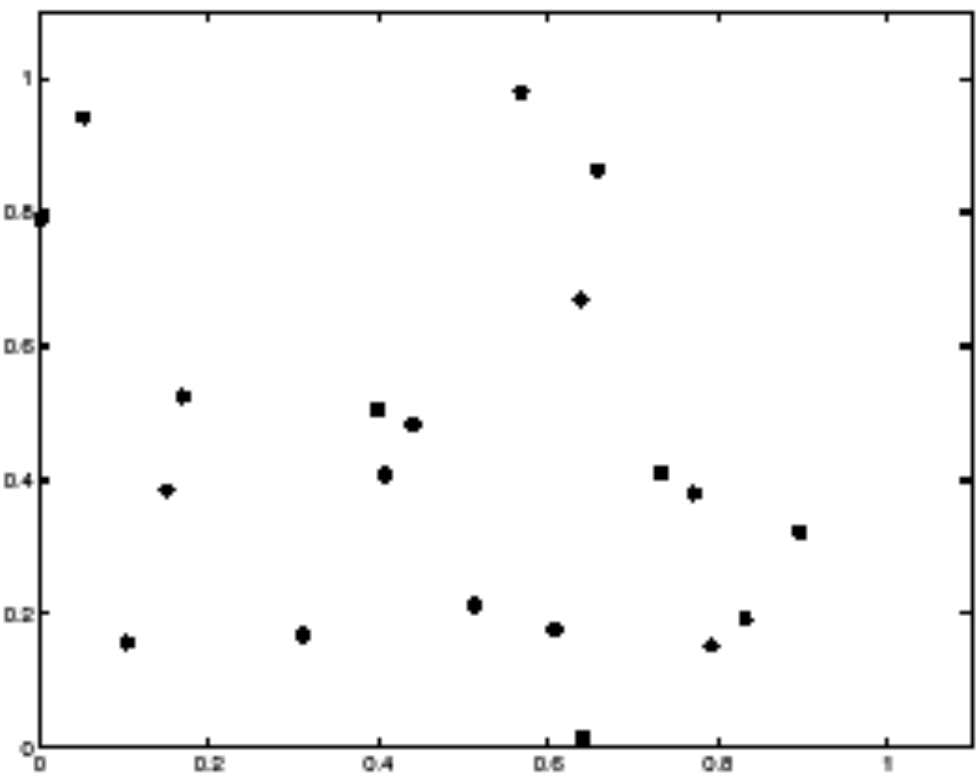


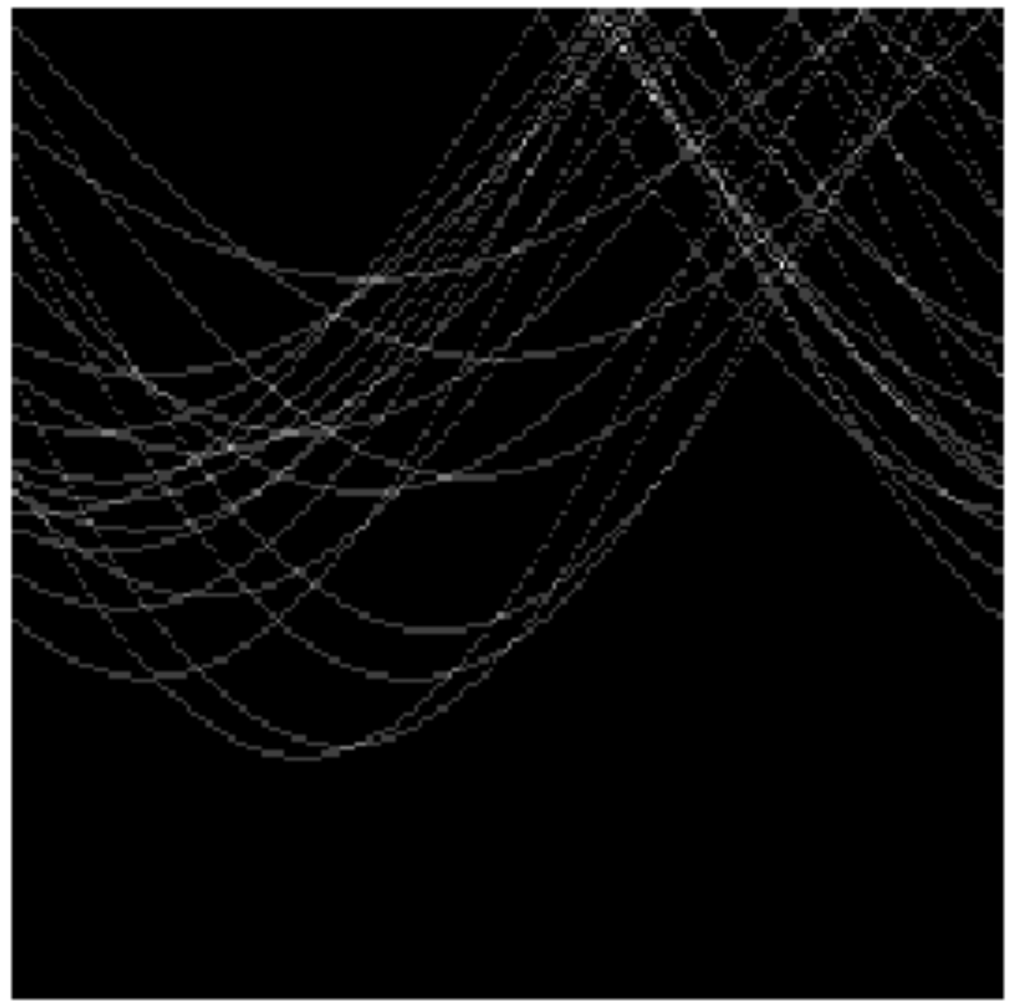


Forsyth & Ponce

Random points

Uniform noise can lead to spurious peaks in the array





Source: S. Lazebnik

Simplifying Hough Transforms

- Use local gradient information to reduce the search space
- Another trick: use prior information
  - For example, if looking for lines in a particular direction, can reduce the search space even further

Fitting lines: 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)

Source: S. Lazebnik

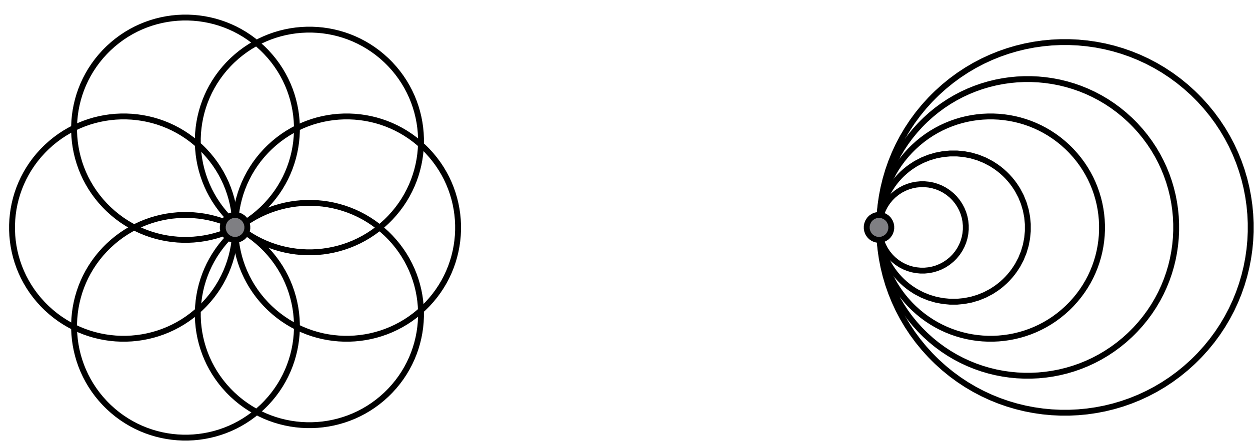
Hough transform beyond lines

Hough Transform

- What else can be detected using Hough transform?
- Anything, but *dimensionality* is key

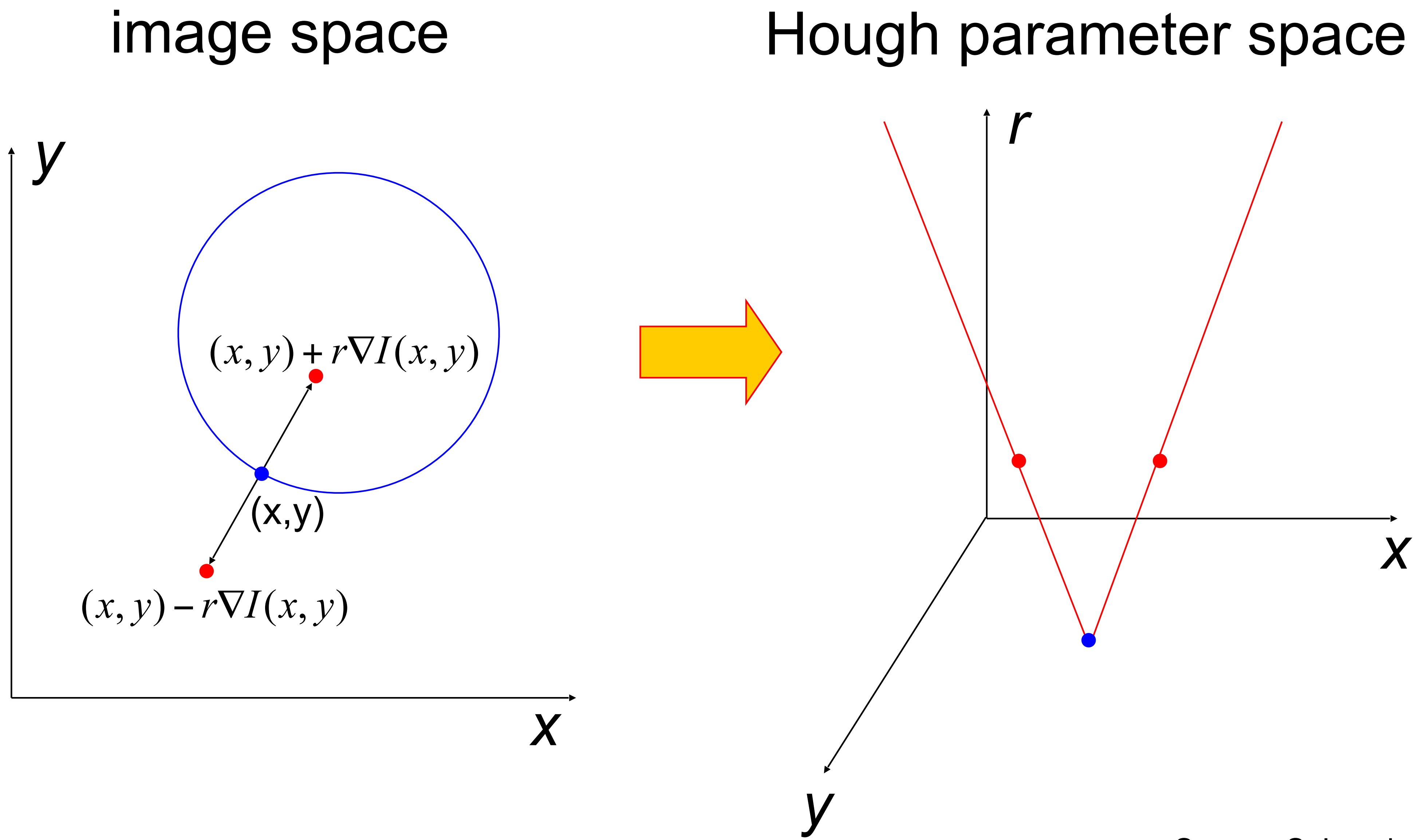
Hough transform for circles

- How many dimensions will the parameter space have?
- Given an 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

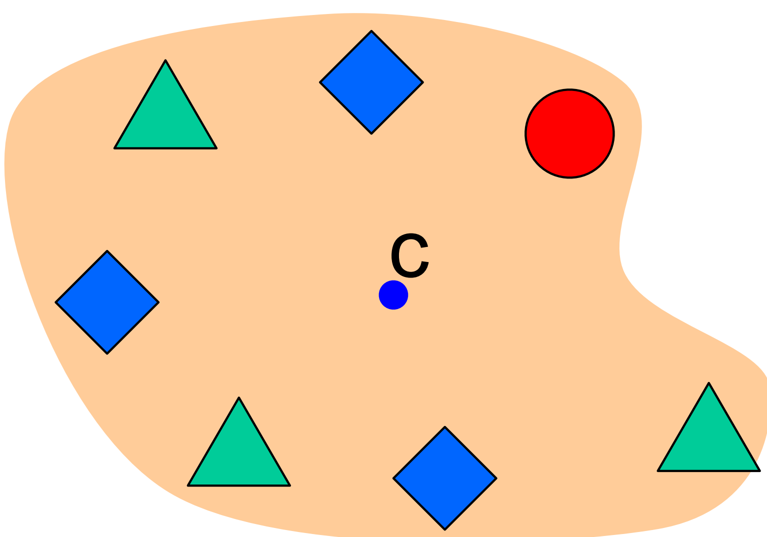


Source: S. Lazebnik

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

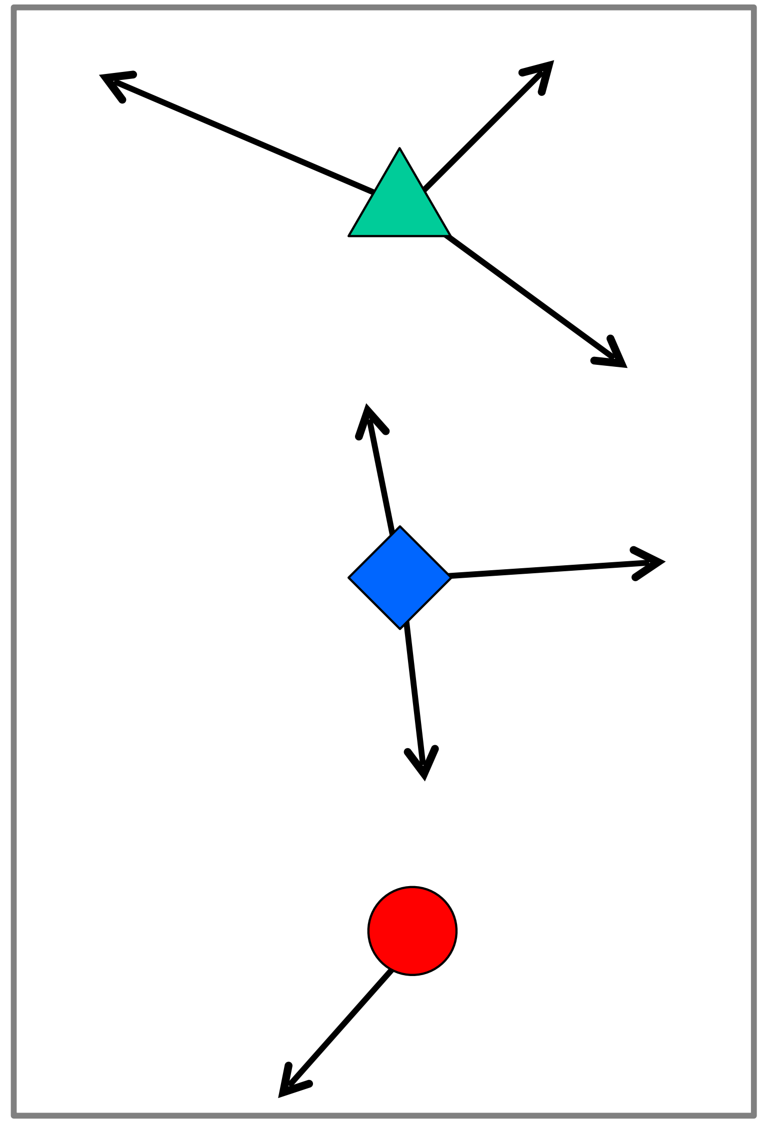


Source: S. Lazebnik

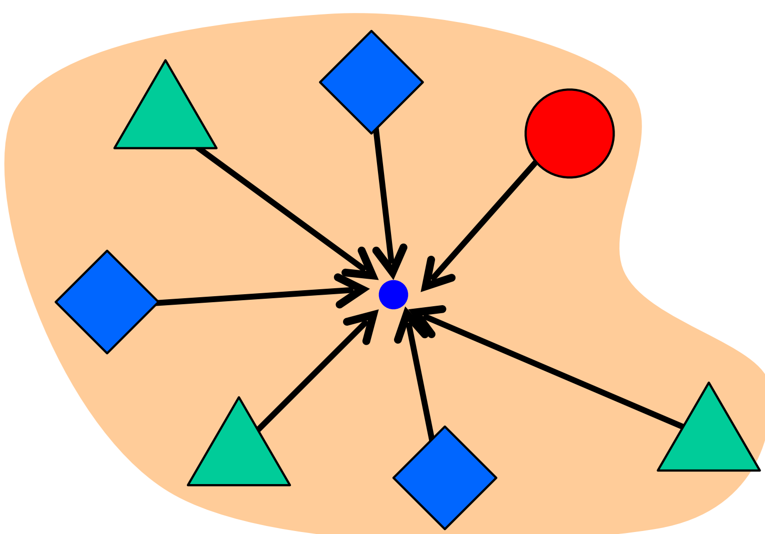
Generalized Hough transform

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

Model



Template

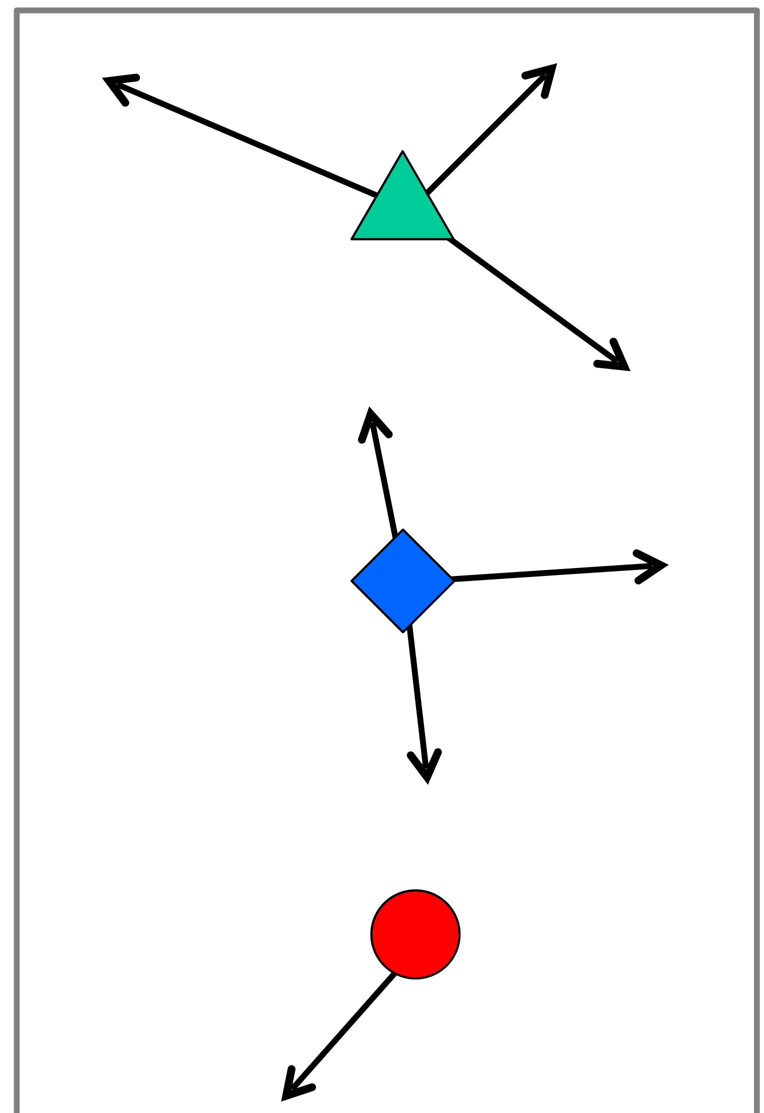


Source: S. Lazebnik

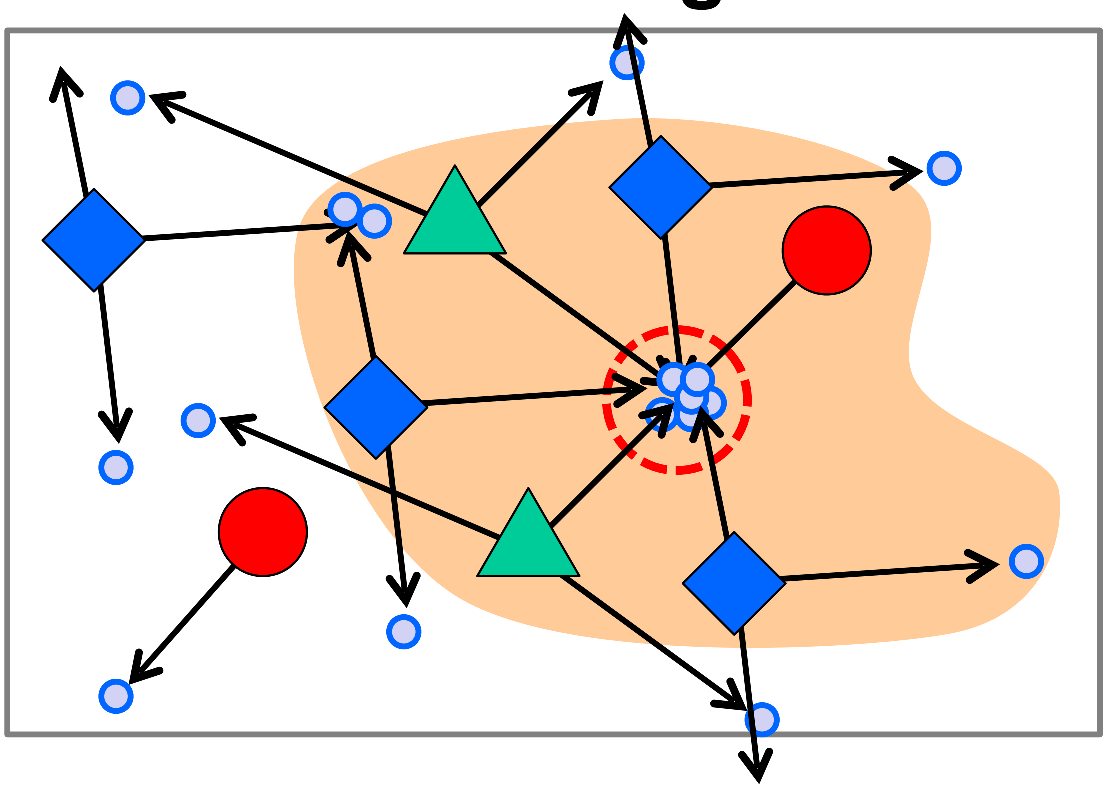
Generalized Hough transform

- 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



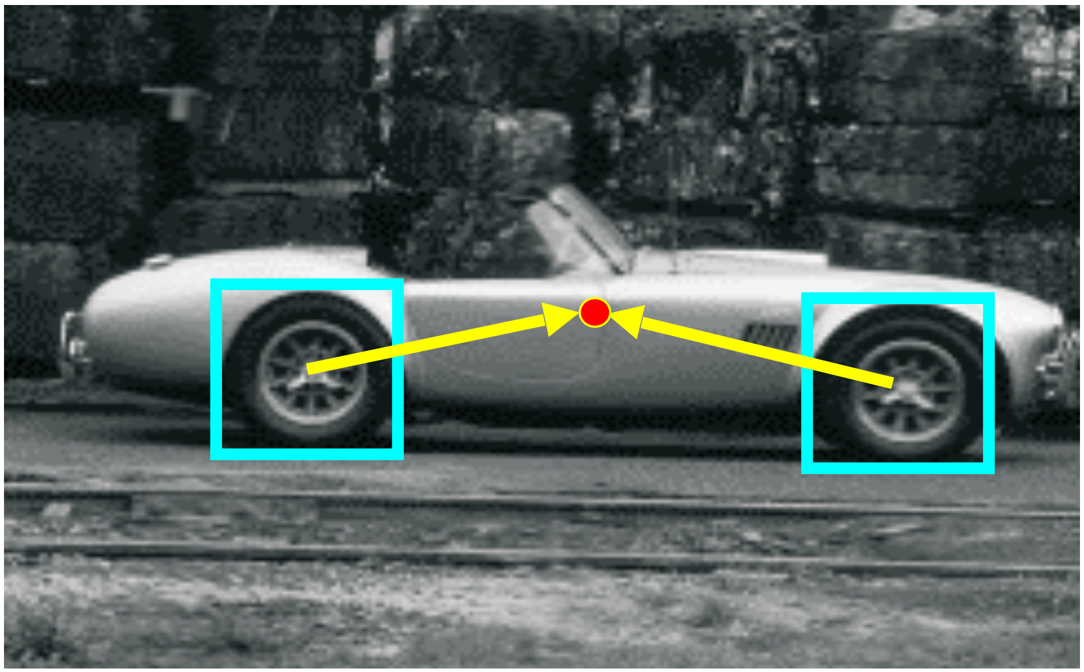
Test image



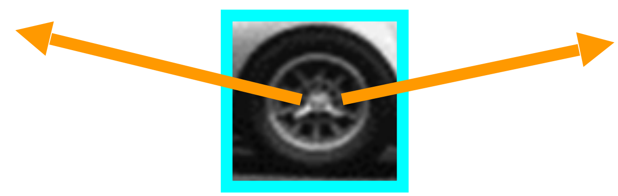
Source: S. Lazebnik

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

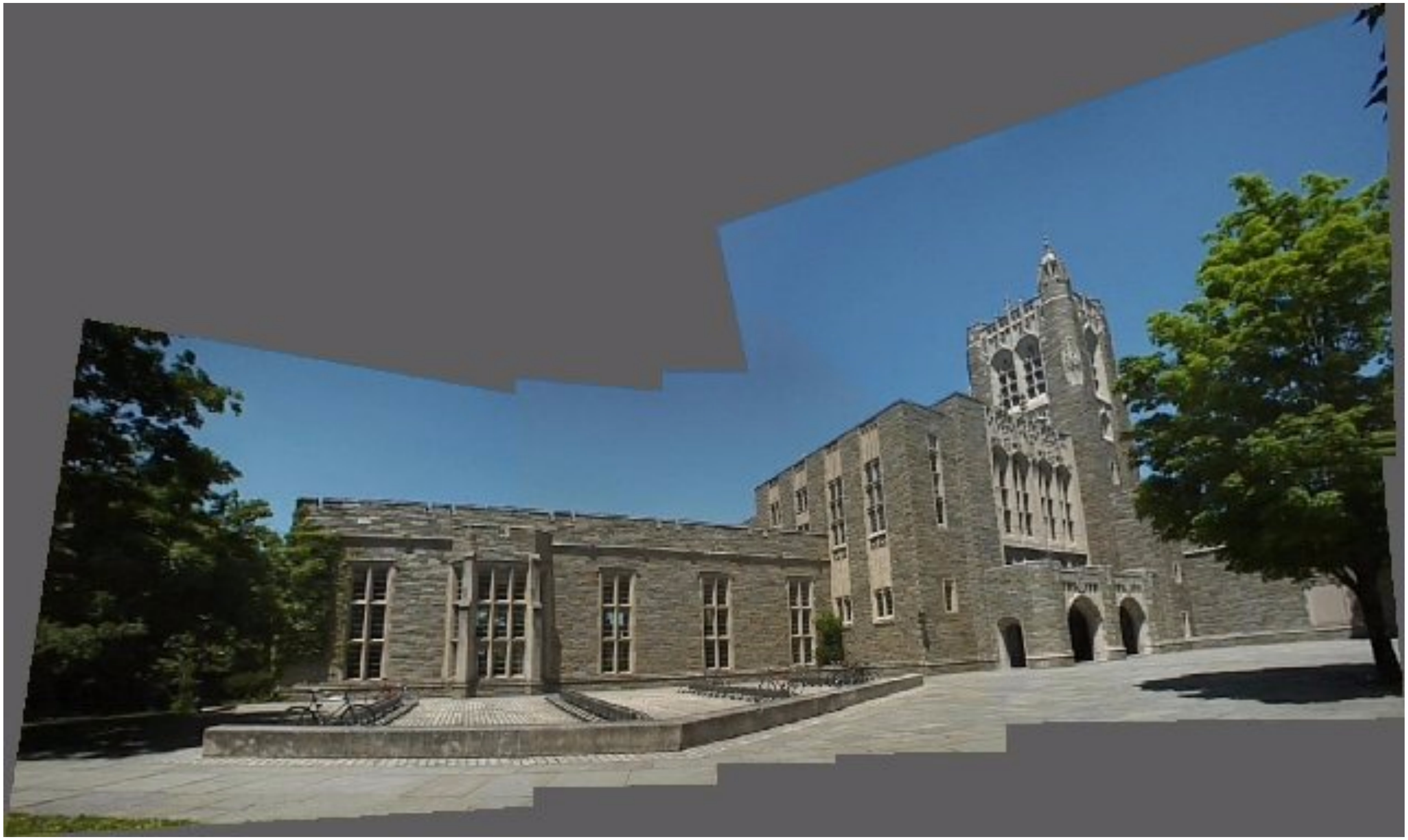
Source: S. Lazebnik

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

Source: S. Lazebnik

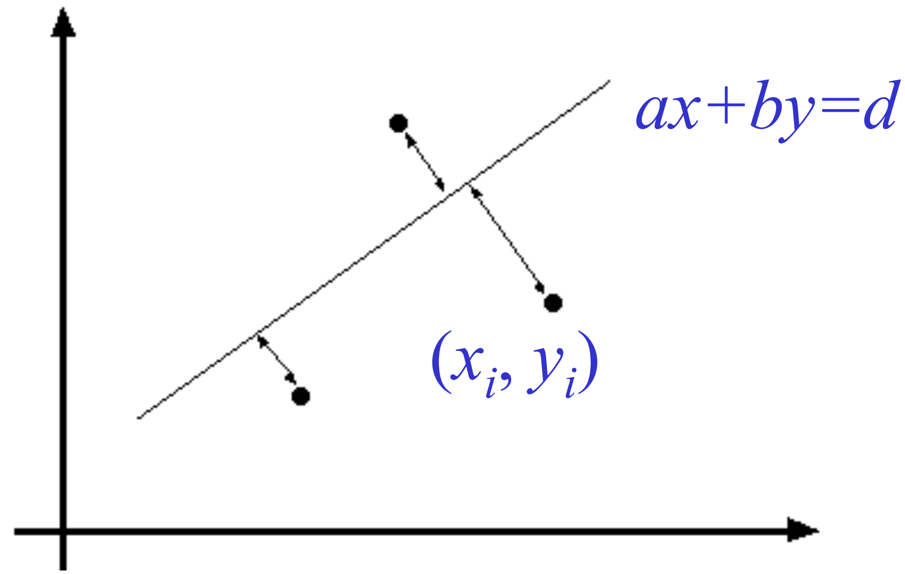
Next time: matching & alignment



Total least squares

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

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



With a bit of algebra, can show that the solution will amount to minimizing:

$$E = (UN)^T(UN)$$

where

$$U = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \quad N = \begin{bmatrix} a \\ b \end{bmatrix}$$

Solution to minimizing E, subject to $||N||^2 = 1$:

eigenvector of U^TU associated with the smallest eigenvalue